LEARNING MACHINE TRANSLATION

edited by Cyril Goutte, Nicola Cancedda, Marc Dymetman, and George Foster

Learning Machine Translation

Neural Information Processing Series
Michael I. Jordan and Thomas Dietterich, editors

Advances in Large Margin Classifiers, Alexander J. Smola, Peter L. Bartlett, Bernhard Schölkopf, and Dale Schuurmans, eds., 2000

Advanced Mean Field Methods: Theory and Practice, Manfred Opper and David Saad, eds., 2001

Probabilistic Models of the Brain: Perception and Neural Function, Rajesh P. N. Rao, Bruno A. Olshausen, and Michael S. Lewicki, eds., 2002

Exploratory Analysis and Data Modeling in Functional Neuroimaging, Friedrich T. Sommer and Andrzej Wichert, eds., 2003

Advances in Minimum Description Length: Theory and Applications, Peter D. Grünwald, In Jae Myung, and Mark A. Pitt, eds., 2005

Nearest-Neighbor Methods in Learning and Vision: Theory and Practice, Gregory Shakhnarovich, Piotr Indyk, and Trevor Darrell, eds., 2006

New Directions in Statistical Signal Processing: From Systems to Brains, Simon Haykin, José C. Príncipe, Terrence J. Sejnowski, and John McWhirter, eds., 2007

Predicting Structured Data, Gökhan Bakır, Thomas Hofmann, Bernhard Schölkopf, Alexander J. Smola, Ben Taskar, S. V. N. Vishwanathan, eds., 2007

Toward Brain-Computer Interfacing, Guido Dornhege, José del R. Millán, Thilo Hinterberger, Dennis J. McFarland, Klaus-Robert Müller, eds., 2007

Large–Scale Kernel Machines, Léon Bottou, Olivier Chapelle, Denis DeCoste, Jason Weston, eds., 2007

Learning Machine Translation, Cyril Goutte, Nicola Cancedda, Marc Dymetman, George Foster, eds., 2009

Learning Machine Translation

Cyril Goutte
Nicola Cancedda
Marc Dymetman
George Foster

The MIT Press
Cambridge, Massachusetts
London, England

This book was set in LaTex by the authors and was printed and bound in the United States of America

Library of Congress Cataloging-in-Publication Data

Learning machine translation / edited by Cyril Goutte... [et al.].
p. cm. — (Neural information processing)
Includes bibliographical references and index.
ISBN 978-0-262-07297-7 (hardcover : alk. paper) 1. Machine translating—Statistical methods. I. Goutte, Cyril.
P309.L43 2009
418'.020285—dc22

2008029324

10 9 8 7 6 5 4 3 2 1

Contents

Series Foreword

The yearly Neural Information Processing Systems (NIPS) workshops bring together scientists with broadly varying backgrounds in statistics, mathematics, computer science, physics, electrical engineering, neuroscience, and cognitive science, unified by a common desire to develop novel computational and statistical strategies for information processing and to understand the mechanisms for information processing in the brain. In contrast to conferences, these workshops maintain a flexible format that both allows and encourages the presentation and discussion of work in progress. They thus serve as an incubator for the development of important new ideas in this rapidly evolving field. The series editors, in consultation with workshop organizers and members of the NIPS Foundation Board, select specific workshop topics on the basis of scientific excellence, intellectual breadth, and technical impact. Collections of papers chosen and edited by the organizers of specific workshops are built around pedagogical introductory chapters, while research monographs provide comprehensive descriptions of workshop-related topics, to create a series of books that provides a timely, authoritative account of the latest developments in the exciting field of neural computation.

Michael I. Jordan and Thomas G. Dietterich

Preface

Foreign languages are all around us. Modern communication technologies give us access to a wealth of information in languages we do not fully understand. Millions of internet users are online at any time with whom we cannot communicate, essentially because of a language barrier. The dream of automatic translation has fueled a sustained interest in automated or semiautomated approaches to translation. Despite some success in limited domains and applications, and the fact that millions of webpages are translated automatically on a daily basis, machine translation (MT) is often met with skepticism, usually by people who do not appreciate the challenges of producing fully automated translations. It is, however, a very dynamic and fertile field where statistical approaches seem to have taken the mainstream, at least at the moment.

This book is a follow-up to the workshop on *Machine Learning for Multilingual Information Access* organized at the NIPS 19 conference in December 2006. Several of the contributors to this book also presented their work there in 2006. However, a number of contributions were also submitted to the book only, which means that roughly half of the content of the final book is newer material that was not presented at the workshop.

Compared to the original workshop, this book is also firmly focused on statistical machine translation (SMT). Its aim is to investigate how machine learning techniques can improve various aspects of SMT. This book is split into two roughly equal parts. The first part deals with enabling technologies, i.e., technologies that solve problems that are not machine translation proper, but are closely linked to the development of a machine translation system. For example, chapter 2 deals with the acquisition of bilingual sentence–aligned data from comparable corpora, a crucial task for domains or language pairs where no parallel corpus is available. Chapters 3 and 4 address the problem of identifying multilingual equivalents of various named entities. One application for such a technology would be to improve the translation of named entities across languages. Chapter 5 deals with word alignment, an essential enabling technology for most SMT systems. It shows how to leverage multiple preprocessing schemes to improve the quality of the alignment. Finally, chapter 6 shows how word-sequence kernels may be used to combine various types of linguistic information, and suggests that this can improve discriminative language modeling.

The second part of the book presents either new statistical MT techniques, or improvements over existing techniques, relying on statistics or machine learning. Chapter 7 addresses the problem of including syntactic information in the trans-

lation model, which is estimated in a discriminative training framework. Chapter 8 proposes a new approach to reranking the hypotheses output by an SMT system trained on a very large corpus. Chapter 9 presents a novel approach to MT that eschews the traditional log-linear combination of feature functions in favor of a kernel-based approach (to our knowledge the first of its kind in the context of MT). Chapters 10 and 11 focus on improving the selection of words or phrases in the translation models. In chapter 10, a discriminative procedure decides whether a word of the target vocabulary is present in the target sentence based on global information on the source sentence, and uses a weighted transducer incorporating a language model to correctly order the selected target words. In chapter 11, a discriminative phrase selection model is integrated with a phrase-based SMT system. The chapter also provides an interesting analysis and comparison of a large number of automatic MT evaluation metrics. Chapter 12 explores the use of semisupervised learning to improve MT output by leveraging large amounts of untranslated material in the source language. Finally, chapter 13 shows how outputs of multiple MT systems may be combined in order to improve the overall translation quality. This approach allows collaborative projects where several partners contribute different MT systems. System combination currently yields the best results in international evaluation campaigns.

We intend this book to be useful both to the machine learning and to the statistical machine translation communities. We hope that the machine learning researcher will get a good overview of various advanced aspects of SMT, and get an idea of the different ways in which learning approaches can directly influence and have an impact on a very challenging and important problem. We also hope that the statistical MT researcher will be interested in this book, in part as a presentation of some advanced topics that may not be covered in introductory SMT textbooks, and in part as a presentation of some novel machine learning–inspired techniques which have the potential to yield new directions of research in this fertile field.

We wish to thank The MIT Press who gave us the opportunity to publish this book, and in particular Susan Buckley and Robert Prior for their support in preparing the manuscript. All the contributed chapters of this book were reviewed, and we are grateful to the reviewers who took the time to provide comments and criticism which contributed to improving the overall quality of the book: Thanks to Caroline Brun, Marine Carpuat, Mauro Cettolo, Hal Daumé III, Hervé Déjean, Andreas Eisele, Alex Fraser, Patrick Haffner, Xiaodong He, Pierre Isabelle, Roland Kuhn, Dragos Munteanu, Miles Osborne, Jean-Michel Renders, Antti-Veikko Rosti, Craig Saunders, Libin Shen, Michel Simard, Sandor Szedmak, and Dan Tufis.

Cyril Goutte
Nicola Cancedda
Marc Dymetman
George Foster

1 A Statistical Machine Translation Primer

Nicola Cancedda
Marc Dymetman
George Foster
Cyril Goutte

This first chapter is a short introduction to the main aspects of statistical machine translation (SMT). In particular, we cover the issues of automatic evaluation of machine translation output, language modeling, word-based and phrase-based translation models, and the use of syntax in machine translation. We will also do a quick roundup of some more recent directions that we believe may gain importance in the future. We situate statistical machine translation in the general context of machine learning research, and put the emphasis on similarities and differences with standard machine learning problems and practice.

1.1 Background

Machine translation (MT) has a long history of ambitious goals and unfulfilled promises. Early work in automatic, or "mechanical" translation, as it was known at the time, goes back at least to the 1940s. Its progress has, in many ways, followed and been fueled by advances in computer science and artificial intelligence, despite a few stumbling blocks like the ALPAC report in the United States (Hutchins, 2003).

Availability of greater computing power has made access to and usage of MT more straightforward. Machine translation has also gained wider exposure to the public through several dedicated services, typically available through search engine services. Most internet users will be familiar with at least one of Babel Fish,[1] Google Language Tools,[2] or Windows Live Translator.[3] Most of these services used to be

1. `http://babelfish.yahoo.com/` or `http://babelfish.altavista.com/`
2. `http://www.google.com/language_tools`
3. `http://translator.live.com/`

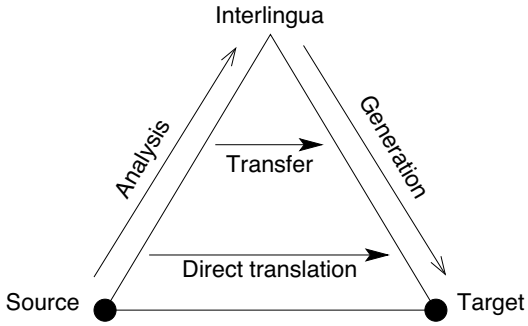

Figure 1.1 The machine translation pyramid. Approaches vary depending on how much analysis and generation is needed. The *interlingua* approach does full analysis and generation, whereas the *direct translation* approach does a minimum of analysis and generation. The *transfer* approach is somewhere in between.

powered by the rule-based system developped by Systran.[4] However, some of them (e.g., Google and Microsoft) now use statistical approaches, at least in part.

In this introduction and the rest of this book, translation is defined as the task of transforming an existing text written in a *source language*, into an equivalent text in a different language, the *target language*. Traditional MT (which in the context of this primer, we take as meaning "prestatistical") relied on various levels of linguistic analysis on the source side and language generation on the target side (see figure 1.1).

The first statistical approach to MT was pioneered by a group of researchers from IBM in the late 1980s (Brown et al., 1990). This may in fact be seen as part of a general move in computational linguistics: Within about a decade, statistical approaches became overwhelmingly dominant in the field, as shown, for example, in the proceedings of the annual conference of the Association for Computational Linguistics (ACL).

The general setting of statistical machine translation is to learn how to translate from a large corpus of pairs of equivalent source and target sentences. This is typically a machine learning framework: we have an *input* (the source sentence), an *output* (the target sentence), and a model trying to produce the correct output for each given input.

There are a number of key issues, however, some of them specific to the MT application. One crucial issue is the evaluation of translation quality. Machine learning techniques typically rely on some kind of cost optimization in order to learn relationships between the input and output data. However, evaluating automatically the quality of a translation, or the cost associated with a given MT output, is a very hard problem. It may be subsumed in the broader issue of language understanding, and will therefore, in all likelihood, stay unresolved for some time. The difficulties

4. `http://www.systransoft.com/`

associated with defining and automatically calculating an evaluation cost for MT will be addressed in section 1.2.

The early approach to SMT advocated by the IBM group relies on the *source-channel* approach. This is essentially a framework for combining two models: a *word-based* translation model (section 1.3) and a *language model* (section 1.4). The translation model ensures that the system produces target hypotheses that correspond to the source sentence, while the language model ensures that the output is as grammatical and fluent as possible.

Some progress was made with word-based translation models. However, a significant breakthrough was obtained by switching to *log-linear models* and *phrase-based* translation. This is described in more detail in section 1.5.

Although the early SMT models essentially ignored linguistic aspects, a number of efforts have attempted to reintroduce linguistic considerations into either the translation or the language models. This will be covered in section 1.6 and in some of the contributed chapters later on. In addition, we do a quick overview of some of the current trends in statistical machine translation in section 1.7, some of which are also addressed in later chapters.

Finally, we close this introductory chapter with a discussion of the relationships between machine translation and machine learning (section 1.8). We will address the issue of positioning translation as a learning problem, but also issues related to optimization and the problem of learning from an imprecise or unavailable loss.

1.2 Evaluation of Machine Translation

Entire books have been devoted to discussing what makes a translation a good translation. Relevant factors range from whether translation should convey emotion as well and above meaning, to more down-to-earth questions like the intended use of the translation itself.

Restricting our attention to machine translation, there are at least three different tasks which require a quantitative measure of quality:

1. assessing whether the output of an MT system can be useful for a specific application (*absolute evaluation*);

2. (a) comparing systems with one another, or similarly (b) assessing the impact of changes inside a system (*relative evaluation*);

3. in the case of systems based on learning, providing a loss function to guide parameter tuning.

Depending on the task, it can be more or less useful or practical to require a human intervention in the evaluation process. On the one hand, humans can rely on extensive language and world knowledge, and their judgment of quality tends to be more accurate than any automatic measure. On the other hand, human judgments tend to be highly subjective, and have been shown to vary considerably between

different judges, and even between different evaluations produced by the same judge at different times.

Whatever one's position is concerning the relative merits of human and automatic measures, there are contexts—such as (2(b)) and (3)—where requiring human evaluation is simply impractical because too expensive or time-consuming. In such contexts fully automatic measures are necessary.

A good automatic measure should above all correlate well with the quality of a translation as it is perceived by human readers. The ranking of different systems given by such a measure (on a given sample from a given distribution) can then be reliably used as a proxy for the ranking humans would produce. Additionally, a good measure should also display low intrasystem variance (similar scores for the same system when, e.g., changing samples from the same dataset, or changing human reference translations for the same sample) and high intersystem variance (to reliably discriminate between systems with similar performance). If those criteria are met, then it becomes meaningful to compare scores of different systems on different samples from the same distribution.

Correlation with human judgment is often assessed based on collections of (human and automatic) translations manually scored with *adequacy* and *fluency* marks on a scale from 1 to 5. Adequacy indicates the extent to which the information contained in one or more reference translations is also present in the translation under examination, whereas fluency measures how grammatical and natural the translation is. An alternate metric is the direct test of a user's comprehension of the source text, based on its translation (Jones et al., 2006).

A fairly large number of automatic measures have been proposed, as we will see, and automatic evaluation has become an active research topic in itself. In many cases new measures are justified in terms of correlation with human judgment. Many of the measures that we will briefly describe below can reach Pearson correlation coefficients in the 90% region on the task of ranking systems using a few hundred translated sentences. Such a high correlation led to the adoption of some such measures (e.g., BLEU and NIST scores) by government bodies running comparative technology evaluations, which in turn explains their broad diffusion in the research community. The dominant approach to perform model parameter tuning in current SMT systems is "minimum error–rate training" (MERT; see section 1.5.3), where an automatic measure is explicitly optimized.

It is important to notice at this point that high correlation was demonstrated for existing measures only *at the system level*: when it comes to the score given to individual translated sentences, the Pearson correlation coefficient between automatic measures and human assessments of adequacy and fluency drops to 0.3 to 0.5 (see, e.g., Banerjee and Lavie, 2005; Leusch et al., 2005). As a matter of fact, even the correlation between human judgments decreases drastically. This is an important observation in the context of this book, because many machine learning algorithms require the loss function to decompose over individual inferences/translations. Unlike in many other applications, when dealing with machine translation loss functions that decompose over inferences are only pale indicators of quality as perceived

by users. While in document categorization it is totally reasonable to penalize the number of misclassified documents, and the agreement between the system decision on a single document and its manually assigned label is a very good indicator of the perceived performance of the system on that document, an automatic score computed on an individual sentence translation is a much less reliable indicator of what a human would think of it.

Assessing the quality of a translation is a very difficult task even for humans, as witnessed by the relatively low interannotator agreement even when *quality* is decomposed into *adequacy* and *fluency*. For this reason most automatic measures actually evaluate something different, sometimes called *human likeness*. For each source sentence in a test set a reference translation produced by a human is made available, and the measure assesses how similar the translation proposed by a system is to the reference translation. Ideally, one would like to measure how similar the meaning of the proposed translation is to the meaning of the reference translation: an ideal measure should be invariant with respect to sentence transformations that leave meaning unchanged (paraphrases). One source sentence can have many perfectly valid translations. However, most measures compare sentences based on superficial features which can be extracted very reliably, such as the presence or absence in the references of n-grams from the proposed translation. As a consequence, these measures are far from being invariant with respect to paraphrasing. In order to compensate for this problem, at least in part, most measures allow considering more than one reference translation. This has the effect of improving the correlation with human judgment, although it imposes on the evaluator the additional burden of providing multiple reference translations.

In the following we will briefly present the most widespread automatic evaluation metrics, referring to the literature for further details.

1.2.1 Levenshtein-Based Measures

A first group of measures is inherited from speech recognition and is based on computing the edit distance between the candidate translation and the reference. This distance can be computed using simple dynamic programming algorithms.

Word error rate (WER) (Nießen et al., 2000) is the sum of insertions, deletions, and substitutions normalized by the length of the reference sentence. A slight variant (WERg) normalizes this value by the length of the Levenshtein path, i.e., the sum of insertions, deletions, substitutions, and matches: this ensures that the measure is between zero (when the produced sentence is identical to the reference) and one (when the candidate must be entirely deleted, and all words in the reference must be inserted).

Position-independent word error rate (PER) (Tillmann et al., 1997b) is a variant that does not take into account the relative position of words: it simply computes the size of the intersection of the bags of words of the candidate and the reference, seen as multi-sets, and normalizes it by the size of the bag of words of the reference.

A large U.S. government project called "Global Autonomous Language Exploitation" (GALE) introduced another variant called the *translation edit rate (TER)*(Snover et al., 2006). Similarly to WER, TER counts the minimal number of insertion, deletions, and substitutions, but unlike WER it introduces a further unit-cost operation, called a "shift," which moves a whole substring from one place to another in the sentence.

In the same project a further semiautomatic *human-targeted translation edit rate (HTER)* is also used. While WER and TER only consider a pre-defined set of references, and compare candidates to them, in computing HTER a human is instructed to perform the minimal number of operations to turn the candidate translation into a grammatical and fluent sentence that conveys the same meaning as the references. Not surprisingly, Snover et al. (2006) show that HTER correlates with human judgments considerably better than TER, BLEU, and METEOR (see below), which are fully automatic.

1.2.2 N-Gram–Based Measures

A second group of measures, by far the most widespread, is based on notions derived from information retrieval, applied to the n-grams of different length that appear in the candidate translation. In particular, the basic element is the *clipped n-gram precision*, i.e., the fraction of n-grams in a set of translated sentences that can be found in the respective references.[5]

BLEU (Papineni et al., 2002) is the geometric mean of clipped n-gram precisions for different n-gram lengths (usually from one to four), multiplied by a factor (*brevity penalty*) that penalizes producing short sentences containing only highly reliable portions of the translation.

BLEU was the starting point for a measure that was used in evaluations organized by the U.S. National Institute for Standards and Technology (NIST), and is thereafter referred to as the *NIST* score (Doddington, 2002). NIST is the arithmetic mean of clipped n-gram precisions for different n-gram lengths, also multiplied by a (different) brevity penalty. Also, when computing the NIST score, n-grams are weighted according to their frequency, so that less frequent (and thus more informative) n-grams are given more weight.

1.2.3 The Importance of Recall

BLEU and NIST are forced to include a brevity penalty because they are based only on n-gram precision. N-gram recall was not introduced because it was not immediately obvious how to meaningfully define it in cases where multiple reference

5. Precision is *clipped* because counts are thresholded to the number of occurrences of n-grams in the reference, so that each n-gram occurrence in the reference can be used to "match" at most one n-gram occurrence in the proposed sentence. Note also that the precision is computed for all n-grams in a document at once, not sentence by sentence.

translations are available. A way to do so was presented in Melamed et al. (2003): the *general text matcher* (GTM) measure relies on first finding a *maximum matching* between a candidate translation and a set of references, and then computing the ratio between the size of this matching (modified to favor long matching contiguous n-grams) and the length of the translation (for precision) or the mean length of the reference (for recall). The harmonic mean of precision and recall can furthermore be taken to provide the *F-measure*, familiar in natural language processing. Two very similar measures are ROUGE-L and ROUGE-W, derived from automatic quality measures used for assessing document summaries, and extended to MT (Lin and Och, 2004a). ROUGE-S, introduced in the same paper, computes precision, recall, and F-measure based on *skip-bigram* statistics, i.e., on the number of bigrams possibly interrupted by gaps.

A further measure, which can be seen as a generalization of both BLEU and ROUGE (both -L and -S), is BLANC (Lita et al., 2005). In BLANC the score is computed as a weighted sum of all matches of all subsequences (i.e., n-grams possibly interrupted by gaps) between the candidate translation and the reference. Parameters of the scoring function can be tuned on corpora for which human judgments are available in order to improve correlation with adequacy, fluency, or any other measure that is deemed relevant.

Finally, the proposers of *METEOR* (Banerjee and Lavie, 2005) put more weight on recall than on precision in the harmonic mean, as they observed that this improved correlation with human judgment. METEOR also allows matching words which are not identical, based on stemming and possibly on additional linguistic processing.

1.2.4 Measures Using Syntax

Liu and Gildea (2005) propose a set of measures capable of taking long-distance syntactic phenomena into account. These measures require the candidates and the references to be syntactically analyzed. Inspired by BLEU and NIST, averaged precision of paths or subtrees in the syntax trees are then computed. In the same line, Giménez and Màrquez (2007b) also use linguistic processing, up to shallow semantic analysis, to extract additional statistics that are integrated in new measures.

While these measures have the drawback of requiring the availability of an accurate and robust parser for the target language, and of making the measure dependent on the selected parser, the authors show significantly improved correlation with human judgments of quality.

1.2.5 Evaluating and Combining Measures

Measures of translation quality are usually themselves evaluated according to Pearson (and less often Spearman) correlation coefficients with human judgments on some test set. Lin and Och (2004b) observe that this criterion is not stable

across data sets. They thus propose an alternative metameasure, *ORANGE*, based on the additional assumption that a good measure should tend to rank reference translations higher than machine translations. Using a machine translation system, an *n–best* list of candidate translations is generated. Each element in the list is then scored using the measure of interest against a set of m reference translations. Reference translations themselves are scored using the same measure, and a global ranking is established. From this ranking, it is possible to compute the average rank of reference translations. Averaging this average rank across all sentences in the test set provides the ORANGE score. This score is then shown to be more consistent than correlation coefficients in ranking evaluation measures on data produced by a single MT system on a given test corpus.

An interesting method to combine the complementary strengths of different measures, and at the same time evaluate evaluation measures and estimate the reliability of a test set, is QARLA (Giménez and Amigó, 2006).

1.2.6 Statistical Significance Tests

Whatever automatic measure one uses, tests of statistical significance provparametric methods are usually considered better suited to the task, especially bootstrap resampling and approximate randomization. Riezler and Maxwell (2005) provide a good discussion of these tests in the context of machine translation evaluation.

1.3 Word-Based MT

Word-based statistical MT originated with the classic work of Brown et al. (1993). Given a source sentence $\mathbf{f}$, Brown et al. seek a translation $\hat{\mathbf{e}}$ defined by the "fundamental equation of statistical MT":

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}}\, p(\mathbf{f}|\mathbf{e})\, p(\mathbf{e}). \tag{1.1}$$

Here the conditional distribution $p(\mathbf{e}|\mathbf{f})$ is decomposed into a *translation model* $p(\mathbf{f}|\mathbf{e})$ and a *language model* $p(\mathbf{e})$. By analogy with cryptography or communication theory, this is sometimes referred to as a *source-channel* (or *noisy-channel*) model, where $p(\mathbf{e})$ is a known "source" distribution, $p(\mathbf{f}|\mathbf{e})$ is a model of the process that encodes (or corrupts) it into the observed sentence $\mathbf{f}$, and the argmax is a *decoding* operation. This decomposition has the advantage of simplifying the design of $p(\mathbf{f}|\mathbf{e})$ by factoring out responsibility for ensuring that $\mathbf{e}$ is well formed into the language model $p(\mathbf{e})$ (language models will be covered in more detail in section 1.4). It also allows the language model to be trained separately on monolingual corpora, which are typically more abundant than parallel corpora.

Brown et al. elaborate a series of five generative models (numbered 1 through 5) for $p(\mathbf{f}|\mathbf{e})$ which are known collectively as the *IBM* models. Each model in the series improves on its predecessor by adding or reinterpreting parameters. The

models are trained to maximize the likelihood of a parallel corpus seen as a set of statistically independent sentence pairs,[6] with the earlier models used to provide initial parameter estimates for later models. Early stopping during expectation–maximization (EM) training is typically used to avoid overfitting.

The IBM models are defined over a hidden *alignment* variable $\mathbf{a}$ which captures word-level correspondences between $\mathbf{f}$ and $\mathbf{e}$:

$$p(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} p(\mathbf{f}, \mathbf{a}|\mathbf{e})$$

where $\mathbf{a}$ is a vector of alignment positions a_j for each word f_j in $\mathbf{f} = f_1 \dots f_J$. Each a_j takes on a value i in $[1, I]$ to indicate a connection to word e_i in $\mathbf{e} - e_1 \dots e_I$, or is 0 to indicate a null connection. Note that this scheme is asymmetrical: words in $\mathbf{f}$ may have at most a single connection, while words in $\mathbf{e}$ may have from 0 to J connections. This asymmetry greatly reduces the number of alignments which must be considered, from 2^{IJ} if arbitrary connections are allowed, to $(I+1)^J$.

1.3.1 Models 1, 2, and HMM

IBM models 1 and 2, as well as the commonly used *HMM* variant due to Vogel et al. (1996), are based on the following decomposition[7] of $p(\mathbf{f}, \mathbf{a}|\mathbf{e})$:

$$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) \approx \prod_{j=1}^{J} p(f_j|e_{a_j}) p(a_j|a_{j-1}, j, I, J).$$

These three models share the family of *lexical translation parameters* $p(f|e)$, but differ in how they parameterize alignments:

$$p(a_j|a_{j-1}, j, I, J) = \begin{cases} 1/(I+1) & \text{IBM 1} \\ p(a_j|j, I, J) & \text{IBM 2} \\ p(a_j - a_{j-1}) & \text{HMM} \end{cases},$$

i.e., in IBM 1 all connections are equally likely, in IBM 2 they depend on the absolute positions of the words being connected, and in the HMM model they depend on the displacement from the previous connection. Och and Ney (2003) discuss how to extend the HMM model to handle null connections. Maximum likelihood training using the EM algorithm is straightforward for all three models; for IBM 1 it is guaranteed to find a global maximum (Brown et al., 1993), making this model a good choice for initializing the lexical parameters. Due to the large number of lexical

6. A necessary prerequisite is identifying translated sentence pairs in parallel documents. This is nontrivial in principle, but in practice fairly simple methods based on sentence length and surface lexical cues, e.g., Simard et al. (1992), are often adequate. For more difficult corpora, a bootstrapping approach (Moore, 2002) can be used.

7. Omitting a factor for normalizing across all sentence lengths J.

parameters, it is common practice to prune out word pairs (f,e) whose probability $p(f|e)$ falls below a certain threshold after IBM1 training.

1.3.2 Models 3, 4, and 5

IBM 1/2 and HMM are based on a generative process in which each word in $\mathbf{f}$ is filled in (from left to right) by first choosing a connecting position in $\mathbf{e}$ according to the position's alignment probability, then choosing the word's identity according to its translation probability given the connected target word. In the more complex models 3, 4, and 5, the emphasis of the generative process shifts to the target sentence: for each word in $\mathbf{e}$, first the number of connected words is chosen (its *fertility*), then the identities of these words, and finally their positions in $\mathbf{f}$. These models retain word-for-word translation parameters $p(f|e)$, as well as asymmetrical alignments in which each source word may connect to at most one target word.

Model 3 incorporates a set of fertility parameters $p(\phi|\mathbf{e})$, where ϕ is the number of words connected to $\mathbf{e}$, and reinterprets model 2's alignment parameters as *distortion* parameters $p(j|i, I, J)$.

Model 4 replaces model 3's distortion parameters with ones designed to model the way the set of source words generated by a single target word tends to behave as a unit for the purpose of assigning positions. The first word in the ith unit is assigned to position j in the source sentence with probability $p(j - \overline{U_{i-1}}|e_i, f_j)$, where $\overline{U_{i-1}}$ is the average position of the words in the $(i-1)$th unit.[8] Subsequent words in the ith unit are placed with probability $p(j - U_{i,j-1}|f_j)$, where $U_{i,j-1}$ gives the position of the $(j-1)$th word in this unit.

Models 3 and 4 are *deficient* (nonnormalized) because their generative processes may assign more than one source word to the same position. Model 5 is a technical adjustment to correct for this problem.

The EM algorithm is intractable for models 3, 4, and 5 because of the expense of summing over all alignments to calculate expectations. The exact sum is therefore approximated by summing over a small set of highly probable alignments, each of whose probability can be calculated efficiently. Model 2 Viterbi alignments are used to initialize the set, which is then expanded using a greedy perturbative search involving moving or swapping individual links.

1.3.3 Search

The problem of searching for an optimal translation (the argmax operation in Eq. (1.1)) is a difficult one for the IBM models. Roughly speaking, there are two sources of complexity: finding the best bag of target words according to the many-to-one source-target mapping implicit in $p(\mathbf{f}|\mathbf{e})$; and finding the best target word

8. To combat data sparseness, Brown et al. map e_i and f_j in this expression to one of 50 equivalence classes defined over source and target vocabularies.

order according to $p(\mathbf{e})$. Knight (1999) shows that decoding is in fact NP-complete for the IBM models through separate reductions exploiting each of these sources of complexity. However, in practice, heuristic techniques work quite well. Germann et al. (2001) describe a Viterbi stack-based algorithm that operates quickly and makes relatively few search errors, at least on short sentences. As this algorithm is similar to search algorithms used with phrase-based translation models, we defer a description to section 1.5.

1.3.4 Current Status

The IBM models have been supplanted by the more recent phrase-based approach to SMT, described in section 1.5, which is conceptually simpler and produces better results. However, they retain a central role due to their ability to produce good word alignments, which are a key ingredient in training phrase-based models. Despite significant recent attention to the problem of word alignment for this and other purposes, IBM 4 alignments—typically produced using the GIZA++ toolkit (see appendix), and symmetrized using the method of Och and Ney (2000a)—remain the most often-used baseline for work in this area.

Unlike the phrase-based model and the later IBM models, models 1/2 and HMM also allow efficient computation of a smooth conditional distribution $p(\mathbf{f}|\mathbf{e})$ over bilingual sentence pairs. This makes them well suited for applications requiring analysis of existing sentence pairs, such as cross-language information retrieval.

1.4 Language Models

A language model (LM), in the basic sense of the term, is a computable probability distribution over word sequences, typically sentences, which attempts to approximate an underlying stochastic process on the basis of an observed corpus of sequences produced by that process.

Language models have many applications apart from statistical machine translation, among them: speech recognition (SR), spelling correction, handwriting recognition, optical character recognition, information retrieval. Historically, much of their development has been linked to speech recognition and often the methods developed in this context have been transposed to other areas; to a large extent this remains true today.

According to the dominant "generative" paradigm in language modeling (and to our definition above), developing a language model should actually only depend on a corpus of texts, not on the application context. The standard measure of adequacy of the language model is then its *perplexity*,[9] an information-theoretic quantity that

9. If $LL_p(T) = log_2\, p(T)$ represents the log-likelihood of the test corpus T relative to the model p, then the perplexity of p on T is defined as $2^{-LL_p(T)}$.

measures the cost of coding a test corpus with the model, and which is provably minimized when the model represents the "true" distribution of the underlying stochastic process.

Recently, some work has started to challenge the dominant paradigm, in an approach known as "discriminative" or "corrective" language modeling, where the focus is more on minimizing errors in the context of a specific application, a criterion that, due to inevitable inadequacies in the application-dependent aspects of the overall probabilistic model (such as the "acoustic model" in SR, or the "translation model" in SMT), does not coincide with perplexity minimization.

This section will mostly focus on the generative paradigm, and will give some pointers to discriminative approaches. Good general references on language models are Goodman (2001) and Rosenfeld (2000), as well as the tutorial of Charniak and Goodman (2002), which have influenced parts of this section.

1.4.1 N-Gram Models and Smoothing Techniques

Still by far the dominant technique for language modeling is the n-gram approach, where the probability of a sequence of words $w_1, w_2, \ldots, w_m$ is approximated, using the case n=3 (trigram) as an illustration, as

$$p(w_1, w_2, \ldots, w_m) \approx \prod_i p(w_i|w_{i-2}, w_{i-1}).$$

The central issue in such models is how to estimate the conditional probabilities $p(w_i|w_{i-2}, w_{i-1})$ from the corpus. The simplest way, maximum likelihood, corresponds to estimating these probabilities as a ratio of counts in the corpus (where # indicates a count of occurrences):

$$p(w_i|w_{i-2}, w_{i-1}) = \frac{\#(w_{i-2}, w_{i-1}, w_i)}{\#(w_{i-2}, w_{i-1})},$$

but this approach suffers from obvious "overfitting" defects; in particular the model assigns a zero probability to a trigram which has not been observed in the corpus. In order to address this problem, several "smoothing" techniques have been devised, which can be roughly characterized by three central representatives.

In *Jelinek-Mercer* interpolated smoothing, one writes

$$p_{JM}(w_i|w_{i-2}, w_{i-1}) = \lambda_3 \frac{\#(w_{i-2}, w_{i-1}, w_i)}{\#(w_{i-2}, w_{i-1})} + \lambda_2 \frac{\#(w_{i-1}, w_i)}{\#(w_{i-1})} + \lambda_1 \frac{\#(w_i)}{\#(\bullet)},$$

which corresponds to interpolating trigram, bigram, and unigram estimates, and where the λ weights are tuned through cross-validation on the corpus. In refined versions of this approach the weights may be tuned differently depending on different ranges of frequencies of the corresponding denominators.

In *Katz backoff* smoothing, the general idea is that when the trigram counts are low, one will *back off* to the bigram estimates:

$$p_K(w_i|w_{i-2},w_{i-1}) = \begin{cases} \frac{\#^*(w_{i-2},w_{i-1},w_i)}{\#(w_{i-2},w_{i-1})} & \text{if } \#(w_{i-2},w_{i-1},w_i) > 0 \\ \lambda\, p_K(w_i|w_{i-1}) > 0 & \text{otherwise} \end{cases},$$

where $\#^*(w_{i-2},w_{i-1},w_i)$ is a "discounted" count according to the Good-Turing formula, which has the effect of displacing some mass from observed events to unobserved events (and which is based on the insight that the number of types[10] which are observed once in the corpus is indicative of the number of unobserved types which are "just waiting there" to be observed, for which some mass should be reserved), and where λ is a normalization factor that weighs the influence of the *backoff* to bigram model.

In *Kneser-Ney backoff* smoothing, and for expository reasons considering only bigram models here, one writes

$$p_{KN}(w_i|w_{i-1}) = \begin{cases} \frac{\#(w_{i-1},w_i)-D}{\#(w_{i-1})} & \text{if } \#(w_{i-1},w_i) > 0 \\ \lambda\, \frac{\tilde{\#}(\bullet,w_i)}{\tilde{\#}(\bullet,\bullet)} & \text{otherwise} \end{cases},$$

where $D \in [0,1]$ is a fixed discounting factor, λ is a normalization factor, and where (our notation/reconstruction) $\tilde{\#}(\bullet,\bullet)$ (resp. $\tilde{\#}(\bullet,w_i)$) is the number of different bigram *types* (resp. bigram types ending in w_i) found in the corpus. Thus, a crucial difference between Kneser-Ney and other techniques is that it does not back off to a quantity that measures the relative frequency $\frac{\#(w_i)}{\#(\bullet)}$ of *occurrences* of the word w_i, which can also be written in the form $\frac{\#(\bullet,w_i)}{\#(\bullet,\bullet)} = \frac{\#(w_i)}{\#(\bullet)}$, but to a quantity $\frac{\tilde{\#}(\bullet,w_i)}{\tilde{\#}(\bullet,\bullet)}$ that measures the "context-type unspecificity" of w_i in terms of the number of different word *types* which may precede w_i. The intuition is that the less context type–specific (i.e., more context type–unspecific) w_i is, the more we would expect to recognize it in a context w_{i-1}, w_i we have never witnessed before.

It is probably fair to say that n-gram with Kneser-Ney smoothing is currently the most widely accepted language modeling technique in practice, sometimes even applied to 4-gram or 5-gram modeling when large enough corpora are available.

Caching

N-gram models are severely limited in their ability to account for nonlocal statistical dependencies. One simple and efficient technique allowing use of the nonlocal context is caching: remembering words that have been produced in the recent history, for example during a dictation session, and predicting that such words have a tendency to repeat later in the session (Kuhn and de Mori, 1990). In its simplest

10. *Types* correspond to classes of objects, as opposed to *tokens*, which correspond to occurrences of these classes. For instance, there are two tokens of the type "man" in the expression "man is a wolf to man."

form, this consists in interpolating a standard trigram model with a unigram cache model $p_{cache}(w_i|w_1,\ldots w_{i-1}) = (i-1)^{-1}(\#\text{instances of } w_i \text{ in } w_1,\ldots w_{i-1})$, but variants exist which consider instances of bigrams or trigrams in the history. One potential problem with caching is that recognition errors early in the history may have a snowball effect later on, unless the history is guaranteed to be accurate, such as in an interactive dictation environment in which the user validates the system outputs.

Class-Based Smoothing

Rather than using words as the basis for n-gram smoothing as discussed so far, another option is to first group words into classes that exhibit similar linguistic behavior, then to use these classes to model statistical dependencies. A simple example of this approach is the following:

$$p(w_i|w_{i-2},w_{i-1}) \approx p(w_i|C_i)\, p(C_i|w_{i-2},w_{i-1}),$$

where C_i is the class associated with w_i . The point of this model is that the classes have higher corpus frequencies than the individual words, and therefore conditional probabilities involving classes can be more reliably estimated on the basis of training data. There are many variants of this basic idea, along three main dimensions: (i) the classes may appear in diverse combinations on the left or right side of the conditioning sign; (ii) the association of a class to a word may be hard, with one class per word (equation shown), or soft, with several classes per word (in this case the equation shown needs to include a sum over classes); (iii) the classes may be associated with the words according to predefined categorization schemes, for instance part-of-speech tags or predefined semantic categories; the last case is especially useful for restricted target domains, for instance speech recognition for air travel reservations. At the opposite end of the spectrum, the classes may be data-driven and obtained through various clustering techniques, a criterion of a good clustering being a low perplexity of the corresponding language model.

One especially interesting application of classes is their possible use for modeling languages with a richer morphology than English, for instance by taking a class to be a lemma or a part of speech or by combining both aspects (Maltese and Mancini, 1992; El-Bèze and Derouault, 1990). Recent approaches to factored translation and language models (see sections 1.4.3 and 1.7.1) work in a similar spirit.

1.4.2 Maximum Entropy Models

Maximum entropy models (aka log-linear models), have been an important tool of statistical natural language processing (NLP) since the early 1990s, in particular in the context of statistical machine translation as we will see in the next section, but also for language modeling proper (Rosenfeld, 2000; Jelinek, 1998), where their role is more controversial.

For language modeling, these models come in two flavors. The main one, which we will call *history-based maxent models*, will be discussed first, then we will briefly discuss so-called *whole-sentence maxent models*.

History-Based Maximum Entropy Models

Generally speaking, history-based language models are models of the form

$$p(w_1, w_2, \ldots, w_m) = \prod_i p(w_i|h_i),$$

where $h_i = w_1, \ldots, w_{i-2}, w_{i-1}$ is the history, and where $p(w_i|h_i)$ is a model of the probability of the next word given its history. N-gram models take the view that $p(w_i|h_i)$ depends only on the value of the $N-1$ last words in the history, but some models attempt to extract richer information from h_i; for instance, decision trees over h_i have been used as a basis for constructing probability distributions over w_i.

A powerful approach to constructing history-based models is based on conditional maximum entropy distributions of the form

$$p(w|h) = \frac{1}{Z(h)} \exp \sum_k \lambda_k f_k(h, w),$$

where the f_ks are feature functions of the input-output pair (h, w), the λ_k are the parameters to be trained, and $Z(h)$ is a normalizing term. In some sense that can be made formally precise, such a distribution is the most "neutral" among distributions constrained to preserve the empirical expectations of the f_ks. By adding well-chosen features, one can then force the distribution to be consistent with certain empirical observations. Among features that have been proved practically useful, one finds "skipping bigrams" that model the dependency of w_i relative to w_{i-2}, skipping over w_{i-1}, and "triggers," which generalize caches and model long-range lexical influences (for instance, if *stock* appears somewhere in a document, *bond* is more likely to occur later), but in principle the addition of various other syntactic or semantic features is possible, under the usual caveat that adding too many features may lead to overfitting effects and must be controlled by feature selection procedures or some form of regularization.

History-based maximum entropy models have been reported by some to significantly decrease the perplexity of n-gram models, but other researchers are more cautious, pointing out that combinations of smoothed n-gram and cache often perform at similar levels.

Whole Sentence Maximum Entropy Models

Because the history-based approach models a sentence by predicting one word at a time, phenomena which refer to a whole sentence, such as parsability, global semantic coherence, or even sentence length are at best awkward to model in the approach. In addition, the partition function $Z(h)$, which involves a sum over all the words in the lexicon, has in principle to be computed at decoding time for each

position in the sentence, which is computationally demanding. For these reasons, the following whole-sentence maximum entropy model is sometimes considered:

$$p(s) = \frac{1}{Z}\, p_0(s)\, \exp \sum\nolimits_k \lambda_k f_k(s),$$

where s is a whole sentence, the f_ks are arbitrary features of s, $p_0(s)$ is a baseline distribution (typically corresponding to a standard n-gram model), the λ_k are parameters, and Z is a normalization constant.[11] At decoding time, Z being a constant need not be considered at all and the objective to maximize is a simple linear combination of the features.[12] On the other hand, training is computationally expensive because, *at this stage*, Z does need to be considered (it depends on the λ_ks, which vary during training), and in principle it involves an implicit sum over the space of *all* sentences s. This is infeasible, and approximations are necessary, typically in the form of MCMC (Monte Carlo Markov chain) sampling techniques.

1.4.3 Some Recent Research Trends

Syntactically Structured Language Models

There is a large and well-established body of research on statistical parsing techniques for computational linguistics. Until recently, there have been relatively few approaches to language modeling based on such techniques, in part because the focus in traditional models has been on parsing accuracy rather than on the perplexity of the associated text-generation processes (when they are well-defined), in part because most probabilistic parsing models require the availability of manually annotated treebanks, which are scarce and have limited coverage, and may not be immediately suitable to tasks such as large-scale speech recognition. Two recent language models that use statistical parsing are Chelba and Jelinek (1998) and Charniak (2001), which are both based on a form of stochastic dependency grammar, the former operating in a strict left-to-right manner and trying to predict the next word on the basis of a partial parse for its previous history, the latter assigning probabilities to the immediate descendants of a constituent conditioned on the content of its lexical head (which may be to the right of the descendant, which makes this model non–left to right). Perplexity reductions of up to 25% over a baseline trigram model have been reported, but again such reductions tend to decrease when simple improvements to the baseline are included, such as a cache mechanism.

---

11. Although introduced later in the language model literature than the previous history-based models, these nonconditional maximum entropy models are actually closer to the original formulation of the maximum entropy principle by Jaynes (1957).
12. Note, however, that decoding here means assessing a *complete* sentence s, and that these models are ill-suited for incremental evaluation of sentence *prefixes*.

Topic-Based Modeling and Related Approaches

Topic-based document modeling has been for some time now a hot topic in information retrieval, one of the best-known techniques being latent semantic analysis (LSA) or its probabilistic counterpart (PLSA). Such techniques allow words to be mapped to a real-valued vector in a low-dimensional "topic space," where Euclidian distance between vectors is an indication of the "semantic" proximity between words, as measured by their propensity to appear in lexically related documents. In Bellagarda (1997) these vectorial representations are used in conjunction with n-grams to build language models where the probability of producing a word is conditioned in part by the topical constitution of its history, as summarized by a vector that accumulates the topical contributions of each of the words in the history.

The previous approach is an instance of modeling statistical dependencies that may span over long ranges, such as a whole sentence or even a document. The neural network–based model of Bengio et al. (2003) is another approach that falls in this category. In this model, words are also represented as vectors in a low-dimensional space, and the process of generating texts is seen as one of generating sequences of such vectors. The model learns simultanously the mapping of words to vectors and the conditional probabilities of the next vector given the few previous vectors in the history. As words are "forced" into a low-dimensional vectorial representation by the learning process (in which different occurrences of a word get the same representation), words that show similar contextual behaviors tend to be mapped to vectors that are close in Euclidian space. Recently, similar techniques have been applied to language models in the context of SMT (Déchelotte et al., 2007).

Bayesian Language Modeling

Some recent approaches to document topic-modeling, such as latent Dirichlet allocation (LDA; see Blei et al., 2003) attempt to characterize the problem in strict "Bayesian" terms, that is, in terms of a hierarchical generative process where probabilistic priors are provided for the parameters. Dynamic Bayesian networks is another active area of research which also considers hierarchical time-dependent generative processes which are actually generalizations of hidden Markov models (HMM) with structured hidden layers. These methods are starting to percolate to language modeling, in models that attempt to characterize the production of word sequences through a structured generative process that incorporates a topic-modeling component (Wallach, 2006; Wang, 2005; Mochihashi and Matsumoto, 2006).

Also in the Bayesian tradition are recent attempts to provide "probabilistic-generative" explanations of the Kneser-Ney smoothing procedure in terms of the so-called Chinese restaurant process which is claimed to explain the differential treatment of type counts and occurrence counts in the procedure (Goldwater et al., 2006; Teh, 2006).

Discriminative Language Modeling

As mentioned at the beginning of this section, while *perplexity* as a measure of performance of a language model has the advantage of universality across applications, it is not always correlated with task-related measures of performance, such as the word error rate in speech recognition, or the BLEU or NIST scores in statistical machine translation. In speech recognition, for more than 15 years, this problem has been addressed, not so much in the subtask of language modeling proper, but rather in the so-called acoustic modeling subtask (recovering a word hypothesis from its acoustic realization), where acoustic models have been trained with methods such as maximum mutual information estimation (MMIE) or minimum classification error (MCE), which attempt to learn model parameters with the direct objective of minimizing recognition errors (Huang et al., 2001).

Such discriminative methods have recently gained a large following in all areas of NLP, and especially in statistical machine translation, as witnessed by several chapters in this book (chapters 7, 8, 10, 11). Concerning the use of discriminative models for language modeling proper, a representative paper is Roark et al. (2004), which applies learning based on perceptrons and conditional random fields, in a speech recognition context, to the task of tuning the parameters of a language model (weights of individual n-gram features) on the basis of a training set consisting of input-output pairs where the input is a lattice of word choices returned by a baseline speech-recognition system and the output is the correct transcription, and where the objective is to find parameters that favor the selection of the correct transcription from the choices proposed by the input lattice, as often as possible on the training set. In chapter 6, Mahé and Cancedda introduce another approach to learning a language model discriminatively in the context of machine translation, this time by using kernels rather than explicit features.

1.5 Phrase-Based MT

Phrase-based MT is currently the dominant approach in statistical MT. It incorporates five key innovations relative to the classic approach discussed in section 1.3:

- the use of log-linear models instead of a simple product of language and translation models;
- the use of multiword "phrases" instead of words as the basic unit of translation, within a simple one-to-one generative translation model;
- minimum error-rate training of log-linear models with respect to an automatic metric such as BLEU, instead of maximum likelihood training;
- a clearly defined and efficient heuristic Viterbi beam search procedure; and
- a second *rescoring* pass to select the best hypothesis from a small set of candidates identified during search.

The phrase-based approach is due to Och and Ney (2004). Our presentation in the following sections is loosely based on Koehn et al. (2003), who give a synthesis of Och's method and related approaches by other researchers.

1.5.1 Log-Linear Models

Recall that the noisy-channel approach combines contributions from a language model $p(\mathbf{e})$ and a "reversed" translation model $p(\mathbf{f}|\mathbf{e})$ by multiplying them. A slight generalization of this is to apply exponential weights in order to calibrate the contribution of each model: $p(\mathbf{e})^{\alpha_1} p(\mathbf{f}|\mathbf{e})^{\alpha_2}$. Taking logs and generalizing the language and translation models to arbitrary feature functions $h(\mathbf{f}, \mathbf{e})$ gives a log-linear analog to Eq. (1.1):

$$\begin{aligned} \hat{\mathbf{e}} &= \operatorname*{argmax}_{\mathbf{e}} \sum_i \alpha_i h_i(\mathbf{f}, \mathbf{e}) \\ &\approx \operatorname*{argmax}_{\mathbf{e},\mathbf{a}} \sum_i \alpha_i h_i(\mathbf{f}, \mathbf{a}, \mathbf{e}), \end{aligned} \tag{1.2}$$

where the standard Viterbi approximation on the second line simplifies the search problem and gives features access to the alignment $\mathbf{a}$ connecting $\mathbf{f}$ and $\mathbf{e}$. This framework is more flexible than the original noisy-channel approach because it can easily accommodate sources of information such as bilingual dictionaries which are difficult to incorporate into generative probabilistic translation models. Commonly used features are logs of forward and reversed translation model probabilities and language model probabilities, as well as a simple word count and a phrase distortion model (described in more detail below). A key assumption made by the search procedure is that features decompose linearly; that is, if $(\mathbf{f}, \mathbf{a}, \mathbf{e})$ can be split into a set of disjoint phrase triples $(\tilde{f}_k, a_k, \tilde{e}_k), k = 1 \ldots K$, then $h(\mathbf{f}, \mathbf{a}, \mathbf{e}) = \sum_{k=1}^{K} h(\tilde{f}_k, a_k, \tilde{e}_k)$. This motivates calling the framework log-linear rather than simply linear, since log probabilities have this property, but ordinary probabilities do not. It is also worth noting that $p_\alpha(\mathbf{e}|\mathbf{f}) = \exp(\sum_i \alpha_i h_i(\mathbf{f}, \mathbf{e}))/Z(\mathbf{f})$ can be interpreted as a maximum entropy model for $p(\mathbf{e}|\mathbf{f})$, where $Z(\mathbf{f})$ is a normalizing factor. This was the original formulation of the log-linear approach in Och (2002).

1.5.2 The Phrase-Based Translation Model

The key features used in Eq. (1.2) are related to the *phrase-based* model for $p(\mathbf{e}, \mathbf{a}|\mathbf{f})$.

This model is based on a simple and intuitive generative process: first, $\mathbf{f}$ is segmented into contiguous phrases (word sequences of arbitrary length), then a translation is chosen for each phrase, and finally the resulting target phrases are reordered to form $\mathbf{e}$. Unlike the IBM models, there are no parts of $\mathbf{f}$ or $\mathbf{e}$ that are not covered by a phrase, and each phrase has exactly one translation.

Segmentations are usually assumed to be uniformly distributed, but the other two parts of the generative process—translation and reordering—each give rise to log-linear features. Let $\tilde{e}_1 \ldots \tilde{e}_K$ be a segmentation of $\mathbf{e}$ into phrases, and $\tilde{f}_1 \ldots \tilde{f}_K$ be the

corresponding source phrases (i.e., the phrases in $\mathbf{f}$, in the order their translations appear in $\mathbf{e}$). Then a "reversed" translation feature can be defined by assuming that phrases are generated independently:

$$h_T(\mathbf{f}, \mathbf{a}, \mathbf{e}) = \sum_{k=1}^{K} \log p(\tilde{f}_k | \tilde{e}_k).$$

A "forward" translation feature can be defined analogously using $p(\tilde{e}_k | \tilde{f}_k)$. Koehn et al. (2003) propose a simple distortion feature for capturing reordering[13]:

$$h_D(\mathbf{f}, \mathbf{a}, \mathbf{e}) = \sum_{k=1}^{K} -|\text{begin}(\tilde{f}_k) - \text{end}(\tilde{f}_{k-1}) - 1|,$$

where $\text{begin}(\tilde{f})$ and $\text{end}(\tilde{f})$ are the initial and final word positions of $\tilde{f}$ in $\mathbf{f}$ (with $\text{end}(\tilde{f}_0) = 0$). This assigns a score of 0 to translations which preserve source phrase order, and penalizes displacements from the "expected" position of the current source phrase (immediately after the preceding phrase) by the number of words moved in either direction.

The phrase-translation distributions $p(\tilde{f}|\tilde{e})$ and $p(\tilde{e}|\tilde{f})$ are defined over a set of phrase pairs called a *phrase table*. Phrase-table induction from parallel corpora is crucial to the performance of phrase-based translation. It typically proceeds by first word-aligning the corpus, then, for each sentence pair, extracting all phrase pairs that are compatible with the given word alignment, under the criterion that a valid phrase pair must not contain links to words outside the pair. For example, in the sentence pair: *Je suis heureux / I am very happy*, with word alignment *Je/I, suis/am, heureux/very_happy*, legal phrase pairs would include *Je/I*, *Je suis/I am*, and *heureux/very happy*, but not *heureux/happy*. In general, this algorithm is fairly robust to word-alignment errors, which tend to affect recall more than precision.

The existence of a standard phrase-extraction algorithm independent of the underlying word alignment has stimulated interest in improved word-alignment techniques. As mentioned in section 1.3, the baseline approach of Och and Ney (2003) relies on IBM model (typically IBM 4) alignments carried out from each translation direction, then *symmetrizes* them into a single alignment by beginning with links in their intersection, then selectively adding links from their union, according to various heuristics. Recently proposed alternatives include combining alternate tokenizations (see chapter 5 by Elming, Habash and Crego), the use of enhanced IBM models (He, 2007), more principled symmetrization techniques (Liang et al., 2007), discriminative techniques (Blunsom and Cohn, 2006), and semisupervised techniques (Fraser and Marcu, 2006), to name only a few.

One difficulty in judging alignment quality for SMT is that the ideal metric—performance of the resulting MT system—is very expensive to compute. In a recent

13. This equation assumes that begin() and end() have access to $\mathbf{a}$, which maps the permuted source phrases $\tilde{f}_1 \ldots \tilde{f}_K$ to their positions in $\mathbf{f}$.

paper, Fraser and Marcu (2007) argue against the use of the popular *alignment error rate* metric as a stand-in, and propose an alternative which correlates better with MT performance.

Once phrase pairs have been extracted from the training corpus, it remains to estimate the phrase-translation distributions $p(\tilde{f}|\tilde{e})$ and $p(\tilde{e}|\tilde{f})$. These may be obtained directly as relative frequencies from joint counts of the number of times each phrase pair was extracted from the corpus. Compositional estimates based on lexical probabilities from the IBM models or word-alignment counts are often used in addition to relative frequencies (Koehn et al., 2003; Zens and Ney, 2004). It is interesting that the heuristic method outlined in the previous paragraphs for populating phrase tables and estimating conditional phrase probabilities seems to perform better than more principled generative algorithms (e.g., Marcu and Wong, 2002) for estimating these distributions. DeNero et al. (2006) argue that this is essentially due to the inclusive property of considering alternative segmentations simultaneously (e.g., learning both *Je/I, suis/am*, and *Je suis/I am* in the example above) rather than forcing segmentations to compete as would estimation with the EM algorithm.

1.5.3 Minimum Error-Rate Training

Given an automatic metric as discussed in section 1.2—for instance, BLEU—minimum error-rate training seeks the vector of log-linear parameters $\hat{\alpha}$ that optimize the metric on a training corpus:

$$\hat{\alpha} = \underset{\alpha}{\text{argmax}}\ \text{BLEU}(\hat{E} = \underset{E}{\text{argmax}}\ \log p_\alpha(E|F),\ R), \tag{1.3}$$

where $\log p_\alpha(E|F) = \sum_{(\mathbf{e},\mathbf{f})\in(E,F)} \log p_\alpha(\mathbf{e}|\mathbf{f})$. The inner argmax is a search with log-linear model $\log p_\alpha$, applied to a source-language corpus F to find the best translation $\hat{E}$. The outer argmax finds the α for which $\hat{E}$ maximizes BLEU with respect to a reference translation R. Och (2003) showed that this approach produces models that score better on new corpora according to the chosen metric than does a maximum likelihood criterion.

Eq. (1.3) is difficult to optimize because the inner argmax is very expensive to compute, and also because BLEU is a nondifferentiable function of α. The standard solution, proposed in Och (2003), is to approximate the inner argmax with a maximization over a small set of n-best candidate translations for F (on the order of 100 translations per source sentence). This makes it fast enough that general optimization techniques can be applied to solve the outer argmax. The success of this approximation depends on being able to identify n-best lists that are representative of the entire search space. Och does this by iterating over different values of $\hat{\alpha}$, each time using the new value of $\hat{\alpha}$ to add new candidates to the n-best lists, which are in turn used to update $\hat{\alpha}$. Bad values of $\hat{\alpha}$ will therefore add bad candidates to the lists which will allow the optimization to avoid these values in future iterations. The complete algorithm is:

1. Initialize $\hat{\alpha}$ and set the n-best set $B = \emptyset$.
2. Find $B(\hat{\alpha})$, the n-best translations for each source sentence according to $p_{\hat{\alpha}}$.
3. Set $B = B \cup B(\hat{\alpha})$. Stop if B doesn't change.
4. Set $\hat{\alpha} = \underset{\alpha}{\operatorname{argmax}}\ \mathrm{BLEU}(\hat{E} = \underset{E \in B}{\operatorname{argmax}}\, p_\alpha(E|F),\ R)$ and go to step 2.

Since the number of hypotheses produced by the decoder is finite, this algorithm is guaranteed to terminate. In practice, it converges fairly quickly, usually after ten iterations or so.

The optimization in step 4 may be solved using Powell's algorithm (Press et al., 2002). This is a general optimization algorithm that iteratively chooses lines $\alpha + \gamma\alpha'$ which must be optimized in the scalar value γ by means of a user-supplied "subroutine." Since $\log p_\alpha$ is linear in α, the score it assigns to each hypothesis in an n-best list is linear in γ. There are therefore at most $n - 1$ values of γ at which BLEU can change for a single n-best list, and at most $m(n - 1)$ values for a set of m n-best lists. By examining the intervals between these points, it is possible to efficiently obtain an exact solution to the problem of maximizing BLEU as a function of γ.

The bottleneck in Och's algorithm is the decoding operation in step 2. This makes it impractical for use on training corpora larger than about 1000 sentences, which in turn limits the number of log-linear parameters that can be reliably learned. Also, the ability of Powell's algorithm to find a good optimum appears to degrade with larger parameter sets (Och et al., 2004), so the typical number of parameters is on the order of ten. Och (2003) also proposes a smoothed version of BLEU which would allow gradient-based techniques to be used instead of Powell's algorithm, but it is not clear whether this approach would give better performance with large feature sets.

Alternatives to Och's algorithm use a different strategy for solving the central problem of costly decoding time: modify the decoder to work faster, typically by considering only monotone alignments (i.e., ones in which source and target phrases have the same order), and by using an aggressive pruning threshold. If decoding is fast enough, the outer argmax in Eq. (1.3) can be solved directly with a general optimization algorithm, e.g., downhill simplex (Zens and Ney, 2004) or simultaneous perturbation stochastic approximation (Lambert and Banchs, 2006). These approaches appear to be competitive with Och's. They have the advantage that they can optimize any parameter of the decoder, rather than just log-linear model weights, but the disadvantage of making poor estimates for features that are sensitive to monotonic decoding, for instance distortion.

Other approaches to minimum error-rate training include recent efforts to train very large sets of parameters, such as weights for Boolean phrase-pair features defined over the phrase table, on very large corpora. Liang et al. (2006) iterate the following: generate n-best lists for the corpus, controlling decoder speed by using a limited-distortion model (neighbor swaps only) and limiting to short sentences, then use the perceptron algorithm to update toward the best candidate in each n-

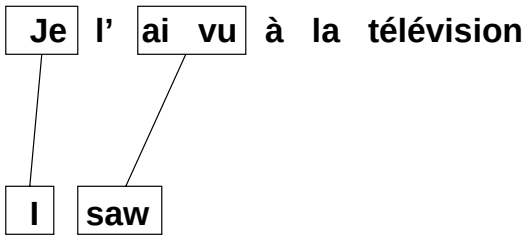

Figure 1.2 A partial hypothesis during decoding, including its alignment. This is extended by choosing a phrase that matches part of the source sentence with no alignment connection (the *uncovered* part), for instance, *à la*, and appending one of its translations from the phrase table, for instance *on*, to the target prefix, giving in this case the new prefix *I saw on*.

best list. Tillmann and Zhang (2006) iterate the following: decode and merge 1-best translations with existing n-best lists, controlling decoder speed by limiting distortion as above, then use stochastic gradient descent to minimize a "margin-inspired" distance function between n-best candidates and oracle translations generated by using the references to guide the decoder. These approaches give only fairly modest gains, possibly because of sacrifices made for decoder efficiency, and possibly because performance appears to be rather insensitive to the exact values of the phrase-translation parameters $p(\tilde{f}|\tilde{e})$.

1.5.4 Search

As we have seen, the problem of decoding Eq. (1.2) is central to minimum error-rate training, and of course in all applications of statistical MT as well. It is NP-complete for phrase-based MT, as it is for the IBM models, but somewhat simpler due to the one-to-one restriction on phrase translation. The standard Viterbi beam search algorithm (Koehn, 2004a) builds target hypotheses left to right by successively adding phrases. As each phrase is added to a hypothesis, the corresponding source phrase is recorded, so the complete phrase alignment is always known for all hypotheses, as illustrated in figure 1.2. Search terminates when the alignments for all active hypotheses are complete, i.e. when all words in the source sentence have been translated. At this point, the hypothesis that scores highest according to the model is output.

A straightforward implementation of this algorithm would create a large number of hypotheses: for each valid segmentation of the source sentence, and each bag of phrases created by choosing one translation for each source phrase in the segmentation, there would be one hypothesis for each permutation of the contents of the bag. Even when the phrase table is pruned to reduce the number of translations available for each source phrase, the number of hypotheses is still unmanageably huge for all but the shortest source sentences. Several measures are used to reduce the number of active hypotheses and the space needed to store them.

First, hypotheses are *recombined*: if any pair of hypotheses are indistinguishable by the model in the sense that extending them in the same way will lead to the same change in score, then only the higher-scoring one needs to be extended. Typically, the lower-scoring one is kept in a *lattice* (word graph) structure, for the purpose of extracting n-best lists (Ueffing et al., 2002) once search is complete. The conditions for recombination depend on the features in the model. For the standard features described above, two hypotheses must share the same last $n-1$ words (assuming an n-gram LM), they must have the same set of covered source words (though not necessarily aligned the same way), and the source phrases aligned with their last target phrase must end at the same point (for the distortion feature).

Recombination is a factoring operation that does not change the results of the search. It is typically used in conjunction with a pruning operation that *can* affect the search outcome. Pruning removes all hypotheses whose scores fall outside a given range (or *beam*) defined with respect to the current best-scoring hypothesis; or, in the case of *histogram pruning*, fall below a given rank.

Three strategies are used to make the comparison between hypotheses as fair as possible during pruning. First, the scores on which pruning is based include an estimate of the *future score*—the score of the suffix required to complete the translation—added to the current hypothesis score. If future scores are guaranteed never to underestimate the true suffix scores, then they are *admissible*, as in A* search, and no search errors will be made. This is typically too expensive in practice, however. Each feature contributes to the future score estimate, which is based on analyzing the uncovered portion of the source sentence. The highest-probability translations from the phrase table are chosen, and are assigned LM scores that assume they can be concatenated with probability 1 (i.e., the LM scores only the inside of each target phrase), and distortion scores that assume they are arranged monotonically. Phrase table and language model future scores can be precomputed for all subsequences of the source sentence prior to search, and looked up when needed.

Comparing two hypotheses that cover different numbers of source words will tend to be unfair to the hypothesis that covers the greater number, since it will have a smaller future score component, and since future scores are intentionally optimistic. To avoid this source of bias, hypotheses are partitioned into equivalence classes called *stacks*, and pruning is applied only within each stack. Stacks are usually based on the number of covered source words, but may be based on their identities as well, in order to avoid bias caused by source words or phrases that are particularly difficult to translate.

Along with recombination and pruning, a final third used to reduce the search space is a limit on the distortion cost for two source phrases that are aligned to neighboring target phrases. Any partial hypotheses that cannot be completed without exceeding this limit are removed. Interestingly, imposing such a limit, typically seven words, often improves translation performance as well as search performance.

There are a number of ways to arrange the hypothesis extension and pruning operations described in the preceding paragraphs. A typical one is to organize the search according to stacks, as summarized in the following algorithm from Koehn (2004a):

- Initialize stack 0 with an empty hypothesis.
- For each stack from 0... J-1 (where J is the number of source words):
  - For each hypothesis g in the stack:
    - For each possible extension of g, covering j source words:
      - Add the extension to stack j, checking for recombination.
      - Prune stack j.
- Output the best hypothesis from stack J.

There have been several recent improvements to the basic Viterbi search algorithm. Huang and Chiang (2007) propose *cube pruning*, which aims to reduce the number of expensive calls to the language model by generating hypotheses and performing an initial pruning step prior to applying the language model feature. Moore and Quirk (2007) use an improved distortion future score calculation and an early pruning step at the point of hypothesis extension (before LM calculation). Both techniques yield approximately an order of magnitude speedup.

1.5.5 Rescoring

The ability of the Viterbi search algorithm to generate n-best lists with minimal extra cost lends itself to a two-pass search strategy in which an initial log-linear model is used to generate an n-best list, then a second, more powerful, model is used to select new best candidates for each source sentence from this list in a *rescoring* (aka *reranking*) pass.

The advantage of this strategy is that, unlike the candidates considered during the first pass, the candidates in an n-best list can be explicitly enumerated for evaluation by the model. This means that there is virtually no restriction on the kinds of features that can be used. Examples of rescoring features that would not be practical within the first-pass log-linear model for decoding include long-range language models, "reversed" IBM 1 models for $p(f|\mathbf{e})$, and features that use IBM 1 to ensure that all words have been translated. Och et al. (2004) list many others.

To assess the scope for improvement due to rescoring, one can perform an oracle calculation in which the best candidates are chosen from the n-best list with knowledge of the reference translations.[14] This gives impressive gains, even for fairly short n-best lists containing 1000 candidates per source sentence. However, this is

14. For metrics like BLEU, which are not additive over source sentences, this can be approximated by choosing the candidate that gives the largest improvement in global score, and iterating.

somewhat misleading, because much of the gain comes from the oracle's ability to game the automatic metric by maximizing its matching criterion, and is thus not accessible to any reasonable translation model (or even any human translator). In practice, gains from rescoring are usually rather modest—often barely statistically significant; by the time n-best lists have been compiled, most of the damage has been done.

Rescoring is most often performed using log-linear models trained using one of the minimum-error techniques described in section 1.5.3. Alternatives include perceptron-based classification (learning to separate candidates at the top of the list from those at the bottom) and ordinal regression (Shen et al., 2004); and also Yamada and Muslea's ensemble training approach, presented in chapter 8.

1.5.6 Current Status

Phrase-based translation remains the dominant approach in statistical MT. However, significant gains have recently been achieved by syntactic methods (particularly on difficult language pairs such as Chinese-English; see section 1.6), by factored methods, and by system combination approaches (see section 1.7).

1.6 Syntax-Based SMT

While the first SMT models were word-based, and the mainstream models are currently phrase-based, we have witnessed recently a surge of approaches that attempt to incorporate syntactic structure, a movement that is reminiscent of the early history of rule-based systems, which started with models directly relating source strings to target strings, and gradually moved toward relating syntactic representations and even, at a later stage, logical forms and semantic representations.

The motivations for using syntax in SMT are related to consideration of fluency and adequacy of the translations produced:

- Fluency of output depends closely on the ability to handle such things as agreement, case markers, verb-controlled prepositions, order of arguments and modifiers relative to their head, and numerous other phenomena which are controlled by the syntax of the target language and can only be approximated by n-gram language models.
- Adequacy of output depends on the ability to disambiguate the input and to correctly reconstruct in the output the relative semantic roles of constituents in the input. Disambiguation is sometimes possible only on the basis of parsing the input, and reconstructing relative roles is often poorly approximated by models of reordering that penalize distortions between the source and the target word orders, as is common in phrase-based models; this problem becomes more and more severe when the source and target languages are typologically remote from each other (e.g.,

subject-verb-object languages such as English, subject-object-verb languages such as Japanese, or languages that allow relatively "free" word order such as Czech).

There are many approaches to incorporating syntax in SMT systems, of which we will describe only a few representative instances. One dimension that is useful for organizing the different approaches is the extent to which they assume some form of a priori knowledge about the syntactic structure of the source and target languages. While certain approaches require that external parsers exist for both the source and the target languages, and use parsed bilingual corpora for training their models, some approaches only require that such parsers exist for the source or for the target language,[15] while some more radical approaches do not require any externally given parser but learn aligned structured representations on the basis of an unparsed bilingual corpus. We start with these "resource-poor" approaches and move gradually toward the more "resource-intensive" ones.

1.6.1 Parser-Free Approaches

Currently probably the most representative among the parser-free approaches is Chiang (2005)'s *hierarchical phrase-based translation*. The model is in line with previous work by Wu (1997) on *inversion transduction grammars* for parsing bilingual corpora and is formally based on a generalization of these grammars, namely *synchronous context-free grammars*. Such grammars are bilateral context-free grammars that simultaneously describe constituents in a source and in a target language and have rules such as (source language is Chinese here)

$$X \rightarrow \langle X \text{ zhiyi, one of } X \rangle,$$

where the source and target expressions on the right-hand side contain terminals and "coupled" nonterminals that correspond to subconstituents which are in translation correspondence. These rules may be automatically extracted from word-aligned phrase pairs by identifying nonterminals with aligned subphrases.

One important restriction in the formalism used by Chiang is that there is only a single generic nonterminal type X, in contrast to externally motivated grammars, which would have nonterminals such as noun phrase (NP), verb phrase (VP), and so forth. Under this constraint, rules such as the above can be seen as direct generalizations of standard biphrases, where the coupled Xs correspond to sub-

15. Another aspect that distinguishes systems is whether they are tree to string, string to tree, or tree to tree, but this aspect is not as clear as the dimension involving reference to external parsers; a system that only uses an external parser for the source can still technically be tree to string or tree to tree, in the latter case through *projecting* trees from the source side of the bilingual corpus over to the target side and using the structural correspondences thus found; a similar remark is true of systems that only involve external parsers on the target side.

biphrases of these biphrases, which were themselves in translation correspondence in the training corpus and have been "anonymized" into X.

Decoding is performed by parsing the source side with the synchronous grammar and simultaneously producing a target parse. Competing derivations are scored according to a log-linear model whose weights are learned based on a minimun-error training procedure.

1.6.2 Parser on the Target

An early attempt to use syntax in SMT was presented by Yamada and Knight (2001), who considered a model for translating from Japanese to English. They use the Collins parser for English for building tree structures over the target side of the bilingual corpus and then learn a mapping from an English tree to a Japanese string through a sequence of transformations: first the nodes of the English tree are locally reordered, then some Japanese words (typically function words) are inserted in the reordered English tree, then the remaining English words are translated into Japanese words, and finally a Japanese string is produced. At training time, EM is used in order to learn the parameters of the different transformations that maximize the likelihood of the training set, and the resulting set of probabilistic transformations constitutes the "translation model" part of a noisy-channel model (hence the model is indeed eventually used for translating from Japanese to English, and not the reverse.) While the model is sometimes described as mapping Japanese strings to English trees (hence as a string-to-tree model), from the description it is clear that internally, Japanese trees are actually built; however, these Japanese trees are not obtained by reference to an independent parser of Japanese, but rather as a kind of projection of externally motivated English parses.

More recently, researchers from the same research group at the Information Sciences Institute have applied powerful formalisms, known as *tree-to-string transducers*, to relate target trees with source strings. In Marcu et al. (2006), such a model is used to translate from Chinese to English. When applied in reverse to the source string (such formalisms can be used either in a forward or reverse direction), the tree-to-string transducer behaves similarly to a context-free grammar (meaning that chart-parsing techniques can be applied to factorize the derivations above a given Chinese string) but each derivation can be seen as a recipe for gluing together English tree "stumps" and producing a complex English parse tree; thus the correspondence between derivations on the source side and trees on the target side is not as direct as in synchronous tree grammars and allows more flexible couplings. At decoding time the application of rules is under the control of a log-linear model that combines features computed on derivations, and the model weights are learnt by mininum error training. The system was claimed in 2006 to be the first

to show BLEU improvements over phrase-based models on experiments conducted over large-scale, open domain translation tasks.[16]

1.6.3 Parser on the Source

An instance of using an external parser on the source only is the work conducted at Microsoft Research by Quirk et al. (2005), who use an in-house rule-based parser, NLPWIN, that produces dependency structures for English. Given a bilingual English-French training corpus, word aligned with GIZA++,[17] the source dependency trees are projected onto the French side of the corpus and from the aligned sentence-level dependency structures obtained, a collection of aligned "treelets" is extracted. These treelets are structural analogs to the biphrases extracted in phrase-based SMT and are constrained to be connected subcomponents of the dependency structure, but not necessarily to project onto contiguous subspans of the word string. At decoding time, the source sentence is parsed, is decomposed into treelets, and a target representation is constructed by gluing together the associated target treelets, under the control of log-linear features. An important aspect of the model (permitted by the use of dependency structures) is that the target representations thus obtained are underdetermined with regard to the relative order of the dependents of a head. This order is determined by a separate model, which is independently trained; this separation of work between treelet training and order training gives flexibility to the model, as the extracted treelets themselves do not need to encapsulate word-ordering considerations.

In chapter 7, Wellington, Turian, and Melamed present another instance where an externally trained parser (Bikel's parser, trained on the Penn treebank) is used to parse the English source side of a bilingual English-French corpus and where projection techniques are used to obtain parallel trees in the target language; however the focus here is on a generic training technique for learning how to transduce a source tree into a target tree and could probably be applied as well to a situation where the target trees were obtained by an independent external parser. Training works by attempting to reconstruct the corpus target trees from the corpus source trees through a sequence of atomic decisions that incrementally build nodes of the target tree, given both the context of the source and the context of previous decisions. The training procedure interleaves feature selection actions and parameter tuning actions, using a boosted ensemble of decision trees under an l_1 regularization regime that favors sparse features.

16. This claim was based on experiments for Chinese-English in the NIST-06 campaign, and continued in NIST-08 for the same language pair. However in the case of Arabic-English, phrase-based systems still win in the later campaign.

17. Even if not mentioned explicitly, the use of GIZA++ for word-aligning a bilingual corpus is actually a shared prerequisite of most of the approaches described in this section.

1.6.4 Parsers on the Source and Target

One approach in which structural a priori knowledge of both the source and the target languages plays an important role was introduced by Cowan et al. (2006). They consider translation from German to English, and use the Dubey parser for German and a modification of the Collins parser for English in order to parse both sides of a bilingual Europarl corpus. The English parser produces structures of a specific form, aligned extended projections (AEPs), which are inspired by the formalism of lexicalized tree adjoining grammar (Frank, 2002). The focus of the paper is to learn the translation of German clauses into English clauses, as opposed to full sentences, and the AEP of an English clause can be seen as a syntactic template to which a sequence of elementary operations have been applied, such as selecting active or passive voice, instantiating the subject slot, choosing the inflection of the verb, etc. The order of such operations is linguistically motivated, for instance the inflection of the verb may depend on the subject. After the bilingual corpus has been parsed on both sides, aligned German clausal structures and English clausal AEPs are extracted, and the goal of training is to learn a sequence of decisions that will permit reconstruction of the English AEP from the German structure. The first such decision is choosing the English syntactic template, then the following decisions correspond to the elementary operations that determine the AEP. Each decision is performed on the basis of features of the German structure and of the previous decisions taken, and the training of the associated weights is done through a structured perceptron procedure. At decoding time, a beam-search procedure is applied, which attempts to find the sequence of decisions which has the largest score according to these weights. In this approach we see a clear instance where a careful linguistic design (nature and order of the elementary operations leading to an AEP) is explicitly exploited for organizing the learning procedure.

1.7 Some Other Important Directions

Statistical machine translation is a very active field of research, and the chapters in this book illustrate a range of promising directions. It would be impossible to cover all ongoing efforts: in this section we briefly touch on some that we perceive as particularly interesting.

1.7.1 Factored Models

The majority of published research on machine translation reports experiments on language pairs having English as target. Translating into other languages requires solving problems that are just negligible in English. Morphology, for instance, is very simple in English compared to most other languages, where verbs can have tens of alternative forms according to mood, tense, etc.; nouns can have different forms for nominative, accusative, dative, and so on. Dictionaries for such

languages tend to be much larger (empirical linguists speak of a lower *token/type ratio*), and reliable statistics are harder to gather. Moreover, when translating from a morphologically poor language (e.g., English) into a morphologically rich one (e.g., Russian), purely word- or phrase-based models can have a hard time, since generating the appropriate morphology might require rather sophisticated forms of analysis on the source: n-gram-based language models can only go so far.

Koehn and Hoang (2007) introduced *factored translation models*, where source words are enriched with linguistic annotation (e.g., lemmas, parts of speech, morphological tags). Separate distributions model translation from source lemmas to target lemmas and from source parts of speech and morphology to their target equivalent. A deterministic morphological generator, finally, combines target lemmas and morphological information to reconstruct target surface forms (i.e., actual words).

Factored language models, where words are represented as bundles of features and the conditioning history can be composed of heterogeneous elements (e.g., a word and a parts of speech), were introduced earlier (Bilmes and Kirchhoff (2003)). The use of *factored word-sequence kernels* in discriminatively-trained language models (chapter 6) falls in the same line of work.

1.7.2 Model Adaptation

The quality of translation and language models depends heavily on the amount of training data. Training corpora of sufficient size for a given language pair, domain, and genre might not be readily available: one is thus left with the uncomfortable choice of either training on few data points coming from the distribution of interest (*on-topic corpora*), or on many data points from a different distribution (*off-topic corpora*). Language model adaptation has been extensively investigated, especially in conjunction with speech recognition. The interest in translation model adaptation, on the other hand, is more recent.

Hildebrand et al. (2005) proposed information retrieval techniques to select from a training set sentence pairs whose source is similar to a given test set, and train only on those. Munteanu et al. (2004) went further, and proposed a classifier for identifying sentences which are a translation of one another in a *comparable corpus* (i.e., a set of similar documents in two different languages).

More recently, Foster and Kuhn (2007) introduced a method based on mixture models: the training data is divided into different components, models (both translation and language) are trained separately on each component, and are combined at translation time with weights which depend on the similarity of the source document and the training data of each component.

Similarly, Xu et al. (2007) train separate language models on different domains, and also use limited on-topic parallel data to re-estimate the weights of the features of a log-linear model. When a new document needs translation, it is first categorized into one domain and then translated using the adapted language model and feature weights.

1.7.3 System Combination

System combination techniques aim to exploit the diversity in translation outputs from different MT systems in order to improve over the best single system. A challenge in doing so is that alternate translations may have very different surface properties such as lexical choice and word order, making it difficult to blend them into a single reasonable output. Recently, Rosti et al. (2007b) proposed an effective solution that consists in choosing one system's hypothesis to establish the word order of the output. The other hypotheses are aligned to this *skeleton* using edit distance, resulting in a constrained word lattice known as a *confusion network* from which the output is derived by searching with a language model and weighted scores from the input systems. Chapter 13 by Matusov, Leusch and Neyin this book extends this approach using IBM alignments rather than edit distance for aligning hypotheses. System combination techniques have recently improved to the point where they reliably give gains over the best single system, even when the other participating systems are relatively weak.

1.7.4 Kernel Methods for Machine Translation

A rather radical departure from existing approaches to SMT is proposed by Wang et al. (2007) (see also chapter 9). Using kernels on strings it is possible to map separately sentences of the source and of the target language into distinct vector spaces (or *feature spaces*). Conceptually the translation problem can thus be decomposed into

1. mapping a source language sentence into a vector in the input feature space;
2. mapping this vector into a vector in the output feature space by means of an appropriate function;
3. mapping a vector from the output feature space into a target language sentence.

The function in the second step can be learned from a training set using an appropriate regression algorithm (such as ridge regression). In practice, the first and the second steps are conflated in that a kernel is used to implicitly map source sentences into the input feature space. The third step, the *inverse image* problem, can be very hard, depending on the kernel used on the target side.

1.8 Machine Learning for SMT

The preceding sections suggest that training a statistical machine translation system is very closely related to supervised learning. At the core of the statistical approach to MT is the attempt to map some input source language sentences $\mathbf{f}$ to some output $\mathbf{e}$. There are, however, a number of idiosyncracies which preclude the straightforward application of known machine learning techniques to SMT. In this

final section, we will relate SMT to various standard machine learning frameworks, and discuss the issue of learning with an uncertain loss, as well as the issue of dividing the MT learning problem into smaller manageable problems, as opposed to adopting an end-to-end approach.

1.8.1 Translation as a Learning Problem

In the context of translation, the output variable—a target language sentence—is formally a discrete variable. In machine learning, predicting a discrete variable usually leads to a classification framework. However, SMT clearly does not fit comfortably in this context: the output space, although discrete, has too many modalities and too much structure. The regression framework is not a much better fit: the output space is not continuous and is very unsmooth, as sentences with similar surface forms may have very different meanings (and therefore translations). In fact, MT is closer to the relatively recent framework of learning with structured output (Taskar, 2004; Tsochantaridis et al., 2005).

The work presented in chapter 9 in this book is a typical example of such an approach. Input and output data are projected into two vector spaces using the implicit mappings $\Phi(\mathbf{f})$ and $\Psi(\mathbf{e})$ provided by a kernel operating on structured data (in that case, sentences in the source and target languages). A multivariate regression model $\Psi(\mathbf{e}) \approx \mathbf{W}\Phi(\mathbf{f})$ may then be used to model the dependencies between the projected input and output, even though the original data is highly structured and does not live in vector spaces. One challenge of this approach is the *preimage problem*: given an estimate $\widehat{\Psi} = \widehat{\mathbf{W}}\Phi(\mathbf{f})$ for a new source sentence $\mathbf{f}$, which target sentence $\widehat{\mathbf{e}}$ should be chosen, such that its image through the implicit mapping, $\Psi(\widehat{\mathbf{e}})$, is "closest" to the regression estimate $\widehat{\Psi}$? This is a very difficult problem for most kernels operating on structured data, and very closely corresponds to the *decoding* step in the traditional SMT framework.

Further work will no doubt appear along those lines. In fact machine translation is a natural field of application for machine learning techniques operating on structured inputs and outputs, as large amounts of training data are available, for a variety of language pairs (e.g., Koehn, 2005; Steinberger et al., 2006). In fact, another important challenge for structured learning methods is to scale up to the corpus sizes commonly used in statistical machine translation, where millions of sentence pairs are not unusual (see, e.g., chapter 8).

It may also be interesting to draw a parallel with the ranking framework, which has been addressed in machine learning in the context of information retrieval, collaborative filtering, extractive summarization, or multiclass categorization, among others. Traditionally, machine translation has been defined as the problem of producing *one* correct translation $\mathbf{e}$ for each source sentence $\mathbf{f}$. However, an arguably equally efficient alternative would be to seek an ordered list of target hypotheses $\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \ldots$, such that correct translations are placed above incorrect ones. This may be relevant in two situations:

1. When there are many correct translations of the source sentence, all of them, not a single one, should be placed at the top of the list.

2. When the model is unable to generate any correct translation, it still makes sense to try to rank nearly correct hypotheses at the top of the list.

In the traditional approach to SMT, such as described in section 1.5, ranking is actually used in at least two ways. First, decoders based on log-linear models usually output ordered n-best lists of translation hypotheses rather than a unique, most probable translation. Second, an additional reranking step, using, for example, more and more complicated feature functions, is used to improve the n-best list by promoting "correct" hypotheses to the top of the list. In both situations, however, ranking is typically based on the output of a model trained for discrimination, not for ranking. Theoretical and empirical results in machine learning (e.g., Cortes and Mohri, 2004) suggest that models trained to minimize an error rate may not provide optimal ranking performance, especially for uneven distributions and high error rates, which is precisely the situation of most MT systems. Placing MT in the framework of a ranking problem and using techniques designed to optimize the ranking performance therefore seems like an interesting direction of investigation. This is in fact the appraoch presented by Shen et al. (2004) for the rescoring stage, and they obtain encouraging results using perceptron-based ordinal regression.

1.8.2 Learning with an Inaccurate Loss

One aspect that crucially sets machine translation apart from most other applications of machine learning is the issue of evaluation. As explained in section 1.2, even when reference translations are available, there is no exact way to calculate, or even define, the cost associated with a new translation hypothesis. This is at odds with most areas where machine learning has been applied. Indeed, most machine learning techniques, at their core, attempt to minimize some loss or risk associated with the prediction. What can be done when such a loss is not available? One typical strategy is to target a different, approximate loss, work on that instead, and hope for the best.

Standard approaches to SMT such as word-based models (section 1.3) rely on maximizing the likelihood on the training set. Within the state-of-the-art framework of phrase-based SMT (section 1.5), phrase tables and language models are typically estimated using word or phrase counts, which corresponds to maximum likelihood estimates, possibly with the addition of some smoothing or regularization. However, the likelihood is not necessarily a good indicator of translation quality. As the unattainable reference of translation evaluation is human judgment, the reliability of the various metrics described in sections 1.2 is usually assessed by their correlation

with human evaluation. The need to optimize these metrics[18] has led to *minimum error-rate training* (section 1.5), where some model parameters are trained by directly minimizing one metric. In the context of machine learning, gradient descent has been used to optimize differentiable losses. More recent work has been targeted to optimizing kernel machines on metrics such as the F-score used in information retrieval or the area under the curve (AUC) used for ranking (Callut and Dupont, 2005; Joachims, 2005). A challenging avenue for future research would be to train some of the newly proposed SMT techniques that depart from the log-linear models by directly optimizing the MT metrics, instead of relying on the standard losses such as the squared error.

An additional consideration is that automatic MT metrics focus on different aspects of the difference between the hypothesis and reference translations: n-gram precision, recall, edit distance, bag-of-word similarity, etc. Arguably, none of these is sufficient to fully account for the difference between two sentences. However, they may conceivably account for *some* of the difference. It would therefore be interesting to consider optimizing not just a single metric, but several of these metrics simultaneously. Techniques from multiobjective, or multicriteria, optimization (Steuer, 1986) may be relevant to that purpose. One of the simplest ways to do that is to combine the multiple objective functions into a single *aggregate objective function* (Giménez and Amigó, 2006). The system may then be optimized on the aggregate measure, in order to increase reliability and robustness.

Finally, the situation of MT evaluation suggests a more speculative question. Is it possible to set up a framework for learning with an imprecisely defined loss? In machine translation, we have a number of approximate losses which have measurable correlation with the "real," unknown loss. By learning on those, we surely learn something about the underlying task, provided the correlation is positive. By contrast, overtraining on the approximate metric will likely degrade the performance on the real loss. It seems to us that this is not a very commonly studied setting in machine learning. However, advances in that direction would certainly have the potential to benefit research in statistical machine translation.

1.8.3 End-to-End Learning for SMT

Current statistical translation systems involve a combination of several models (translation, language model, log-linear model, rescoring; section 1.5). The parameters associated with each of these are usually estimated more or less independently, leading to a *highly stratified* parameter estimation: the parameters of the translation model are derived from the phrase table using essentially maximum likelihood parameters; the language model parameters are obtained by smoothing the maxi-

18. Besides the quest for better translation quality, one additional motivation is that international MT evaluations are usually carried out using automatic MT evaluation metrics. Optimizing the right metric can have a direct impact on a system's ranking.

mum likelihood (or minimum perplexity) estimates; parameters of the phrase-based translation model and rescoring model are usually estimated using minimum error-rate training, etc. In addition, parts of the model, such as the distortion feature function (section 1.5.2), are parameterless, but could conceivably be made more flexible with the addition of a few parameters.

The obvious limitation of this approach is that the overall model is divided into smaller parts, each optimized locally on a loss that may be only loosely related to the overall translation goal. Instead, one would ideally like to optimize all model parameters globally, on the overall loss. Note, however, that in the context of machine translation, this means optimizing over millions of parameters of the translation and language models, in addition to the log-linear parameters. Recent advances in discriminative training of machine translation models have started addressing this issue. This is the case for two approaches described at the end of section 1.5.3. Tillmann and Zhang (2006) propose a new translation model and an associated discriminative training technique that optimizes millions of parameters using a global score (such as BLEU). Liang et al. (2006) also propose an *end-to-end* approach relying on a perceptron trained on millions of features, but which also includes translation and language model probabilities as features, thus retaining part of the stratification in the model estimation. In both cases, the models differ substantially from the current state of the art of phrase-based translation.

The issue of stratified vs. end-to-end parameter estimation therefore suggests (at least) two directions for improving translation performance. One would be to limit the stratification of current phrase-based models by estimating more parameters globally. The second is obviously to improve recent end-to-end models, which are currently competitive only with baseline versions of phrase-based models (usually a fairly standard Pharaoh system), but not with the more evolved versions used, for example, in international evaluations.

1.9 Conclusion

In this introduction, we have given an overview of current statistical machine translation techniques. We also provide pointers to the literature for readers wishing to acquire more information on specific topics. Our hope is that this chapter is self-contained and broad enough for the reader not especially familiar with SMT to now turn to and benefit from the more advanced topics addressed in the following chapters of this book.

Appendix: On-line SMT Resources

- Statistical machine translation resources (`http://www.statmt.org/`): includes links to the yearly workshop on machine translation

- Moses (`http://www.statmt.org/moses/`): SMT system implementing phrase-based translation and factored model, with beam-search decoder
- Pharaoh (`http://www.isi.edu/publications/licensed-sw/pharaoh/`): freely available decoder for phrase-based SMT
- GIZA++ (`www.fjoch.com/GIZA++.html`): toolkit implementing the IBM models
- SRILM (`http://www.speech.sri.com/projects/srilm/`): widely used SRI language modelling toolkit
- LDC (Linguistic Data Consortium, `http://www.ldc.upenn.edu/`): provider of multilingual data
- ELDA (Evaluations and Language Resources Distribution Agency, `http://www.elda.org/`): operational body of the European Language Resources Association and provider of multilingual data
- Europarl (`http://www.statmt.org/europarl/`): parallel corpus including 11 European languages
- NIST (`http://www.nist.gov/speech/tests/mt`): the annual MT evaluation carried out by NIST

I Enabling Technologies

2 Mining Patents for Parallel Corpora

Masao Utiyama
Hitoshi Isahara

Large-scale parallel corpora are indispensable language resources for machine translation. However, only a few large-scale parallel corpora are available to the public. We found that a large amount of parallel texts can be obtained by mining comparable patent corpora. This is because patents of the same subject matter are often filed in multiple countries. Such patents are called "patent families." We describe a Japanese-English patent parallel corpus created from patent families filed in Japan and the United States. The parallel corpus contains about 2 million sentence pairs that were aligned automatically. This is the largest Japanese-English parallel corpus and will be available to the public after the NTCIR-7 workshop meeting. We estimated that about 97% of the sentence pairs were correct alignments and about 90% of the alignments were adequate translations whose English sentences reflected almost perfectly the contents of the corresponding Japanese sentences.

2.1 Introduction

The rapid and steady progress in corpus-based machine translation (MT) (Nagao, 1981; Brown et al., 1993) has been supported by large parallel corpora, such as the Arabic-English and Chinese-English parallel corpora distributed by the Linguistic Data Consortium (Ma and Cieri, 2006), the Europarl corpus (Koehn, 2005) consisting of 11 European languages, and the JRC-Acquis corpus consisting of more than 20 European languages (Steinberger et al., 2006). However, large parallel corpora do not exist for many language pairs. For example, there are a few publicly available Japanese-English parallel corpora as listed in the website of the International Workshop on Spoken Language Translation (IWSLT-2007)[1] and these corpora are small compared to the above-mentioned corpora.

1. `http://iwslt07.itc.it/menu/resources.html`

Much work has been undertaken to overcome this lack of parallel corpora. For example, Resnik and Smith (2003) have proposed mining the web to collect parallel corpora for low-density language pairs. Munteanu and Marcu (2005) have extracted parallel sentences from large Chinese, Arabic, and English nonparallel newspaper corpora. Utiyama and Isahara (2003) have extracted Japanese-English parallel sentences from a noisy-parallel corpus.

In this chapter, we show that a large amount of parallel text can be obtained by mining comparable patent corpora. This is because patents of the same subject matter are often filed in multiple countries. Such patents are called *patent families*. For example, we obtained over 80,000 patent families from patents submitted to the Japan Patent Office (JPO) and the United States Patent and Trademark Office (USPTO), as described in section 2.3. From these patent families, we extracted a high-quality Japanese-English parallel corpus. This corpus and its extension will be used in the NTCIR-7 patent MT task and made available to the public after the NTCIR-7 workshop meeting, which will be held in December 2008.[2] In addition, we believe that multilingual parallel corpora for other languages could be obtained by mining patent families because patents are filed in multiple countries.

Patent translations are required in the society. For example, the JPO provides Japanese-English MT of Japanese patent applications. Consequently, it is important to collect parallel texts in the patent domain to promote corpus-based MT on that domain.

In section 2.2, we review the related work on comparable corpora. In section 2.3, we describe the resources used to develop our patent parallel corpus. In sections 2.4, 2.5, and 2.6, we describe the alignment procedure, the basic statistics of the patent parallel corpus, and the MT experiments conducted on the patent corpus.

2.2 Related Work

Comparable corpora have been important language resources for multilingual natural language processing. They have been used in mining bilingual lexicons (Fung and Yee, 1998; Rapp, 1999; Higuchi et al., 2001), parallel sentences (Zhao and Vogel, 2002; Utiyama and Isahara, 2003; Munteanu and Marcu, 2005; Fung and Cheung, 2004a,b), and parallel subsentential fragments (Munteanu and Marcu, 2006; Quirk et al., 2007).

Fung and Yee (1998) and Rapp (1999) have used newspaper corpora to extract bilingual lexicons. Higuchi et al. (2001) have used patent families filed in both Japan and the United States to extract bilingual lexicons. They used only the title and abstract fields from a number of fields (e.g., titles, abstracts, claims, and so on) in patent documents. This is because the title and abstract fields are often parallel in Japanese and English patents, even though the structures of paired patents are not

2. `http://research.nii.ac.jp/ntcir/`

always the same, e.g., the number of fields claimed in a single patent family often varies depending on the language.

Higuchi et al. (2001) have shown that the title and abstract fields in patent families are useful for mining bilingual lexicons. However, the number of sentences contained in these two fields is small compared to the overall number of sentences in the whole patents. Thus, using only these two fields does not provide enough sentences for a parallel corpus. In this chapter, we show that a large amount of parallel texts can be obtained by mining the "Detailed Description of the Preferred Embodiments" part and the "Background of the Invention" part of patent families.[3] Of the patent families examined, we found that these parts tend to be literal translations of each other, even though they usually contain noisy alignments.

Traditional sentence alignment algorithms (Gale and Church, 1993; Utsuro et al., 1994) are designed to align sentences in clean-parallel corpora and operate on the assumption that there is little noise such as reorderings, insertions, and deletions between the two renderings of a parallel document. However, this assumption does not hold for comparable or noisy-parallel corpora. In our case, for example, some information described in a Japanese patent may not be included when it is submitted to the USPTO. As a result, the patent family consisting of the original Japanese patent and the modified United States patent will contain missing text when compared.

To tackle noise in comparable corpora, Zhao and Vogel (2002) and Utiyama and Isahara (2003) first identify similar parallel texts from two corpora in different languages. They then align sentences in each text pair. Finally, they extract high-scoring sentence alignments assuming that these are cleaner than the other sentence alignments.

Zhao and Vogel (2002) and Utiyama and Isahara (2003) assume that their corpora are noisy-parallel. That is, they assume that document pairs identified by their systems are rough translations of each other. In contrast, Fung and Cheung (2004a,b) and Munteanu and Marcu (2005) do not assume document-level translations. They judge each sentence pair in isolation to decide whether those sentences are translations of each other. Consequently, they do not need document pairs being translations of each other. Munteanu and Marcu (2006) and Quirk et al. (2007) go even further. They do not assume sentence-level translations and try to extract bilingual sentential fragments (e.g., phrases) from nonparallel corpora.

In this chapter, we use Utiyama and Isahara's method (Utiyama and Isahara, 2003) to extract sentence alignments from patent families because we have found that patent families are indeed rough translations of each other.

3. We will provide additional parallel texts obtained from other fields (e.g., claims and abstracts) for the NTCIR-7 patent MT task.

2.3 Resources

Our patent parallel corpus was constructed using patent data provided for the NTCIR-6 patent retrieval task (Fujii et al., 2007). The patent data consists of

- unexamined Japanese patent applications published from 1993 to 2002, and
- USPTO patents published from 1993 to 2000.

The Japanese patent data consists of about 3.5 million documents, and the English data consists of about 1.0 million documents.

We identified 84,677 USPTO patents that originated from Japanese patents. We used the priority information described in the USPTO patents to obtain these patent pairs (families). We examined these patent families and found that the "Detailed Description of the Preferred Embodiments" part (*embodiment part* for short) and the "Background of the Invention" part (*background part* for short) of each application tend to be literal translations of each other. We thus decided to use these parts to construct our patent parallel corpus.

We used simple pattern-matching programs to extract the embodiment and background parts from the whole document pairs and obtained 77,014 embodiment part pairs and 72,589 background part pairs. We then applied the alignment procedure described in section 2.4 to these 149,603 pairs. We call these embodiment and background parts *documents*.

2.4 Alignment Procedure

2.4.1 Score of Sentence Alignment

We used Utiyama and Isahara's method (Utiyama and Isahara, 2003) to score sentence alignments. We first aligned sentences[4] in each document by using a standard dynamic programming (DP) matching method (Gale and Church, 1993; Utsuro et al., 1994). We allowed one-to-n, n-to-one ($0 \leq n \leq 5$), or two-to-two alignments when aligning the sentences. A concise description of the algorithm used is given elsewhere (Utsuro et al., 1994).[5] Here, we only discuss the similarities between Japanese and English sentences used to calculate scores of sentence alignments.

4. We split the Japanese documents into sentences by using simple heuristics and split the English documents into sentences by using a maximum entropy sentence splitter available at `http://www2.nict.go.jp/x/x161/members/mutiyama/maxent-misc.html` We manually prepared about 12,000 English patent sentences to train this sentence splitter. The precision of the splitter was over 99% for our test set.

5. The sentence alignment program we used is available at `http://www2.nict.go.jp/x/x161/members/mutiyama/software.html`

Let J_i and E_i be the word tokens of the Japanese and English sentences for the ith alignment. The similarity between J_i and E_i is[6]

$$\mathrm{SIM}(J_i, E_i) = \frac{2 \times \sum_{j \in J_i} \sum_{e \in E_i} \frac{\delta(j,e)}{\deg(j)\deg(e)}}{|J_i| + |E_i|}, \tag{2.1}$$

where j and e are word tokens and

$|J_i|$ is the number of Japanese word tokens in the ith alignment
$|E_i|$ is the number of English word tokens in the ith alignment
$\delta(j, e) = 1$ if j and e can be a translation pair, 0 otherwise
$\deg(j) = \sum_{e \in E_i} \delta(j, e)$
$\deg(e) = \sum_{j \in J_i} \delta(j, e)$

Note that $\frac{\delta(j,e)}{\deg(j)\deg(e)} = 0$ if $\delta(j, e) = \deg(j) = \deg(e) = 0$.

J_i and E_i were obtained as follows: We used ChaSen[7] to morphologically analyze the Japanese sentences and extract content words, which consisted of J_i. We used a maximum entropy tagger[8] to part-of-speech tag the English sentences and extract content words. We also used WordNet's library[9] to obtain lemmas of the words, which consisted of E_i. To calculate $\delta(j, e)$, we looked up an English-Japanese dictionary that was created by combining entries from the EDR Japanese-English bilingual dictionary, the EDR English-Japanese bilingual dictionary, the EDR Japanese-English bilingual dictionary of technical terms, and the EDR English-Japanese bilingual dictionary of technical terms.[10] The combined dictionary contained over 450,000 entries.

After obtaining the maximum similarity sentence alignments using DP matching, we calculated the similarity between a Japanese document, J, and an English document, E, (AVSIM(J, E)), as defined by Utiyama and Isahara (2003), using

$$\mathrm{AVSIM}(J, E) = \frac{\sum_{i=1}^{m} \mathrm{SIM}(J_i, E_i)}{m}, \tag{2.2}$$

where $(J_1, E_1), (J_2, E_2), \ldots (J_m, E_m)$ are the sentence alignments obtained using DP matching. A high AVSIM(J, E) value occurs when the sentence alignments in J and E take high similarity values. Thus, AVSIM(J, E) measures the similarity between J and E.

6. To penalize one-to-0 and 0-to-one alignments, we assigned $\mathrm{SIM}(J_i, E_i) = -1$ to these alignments instead of the similarity obtained by using Eq. (2.1).
7. http://chasen-legacy.sourceforge.jp/
8. http://www2.nict.go.jp/x/x161/members/mutiyama/maxent-misc.html
9. http://wordnet.princeton.edu/
10. http://www2.nict.go.jp/r/r312/EDR/

We also calculated the ratio of the number of sentences between J and E ($R(J, E)$) using

$$R(J, E) = \min(\frac{|J|}{|E|}, \frac{|E|}{|J|}), \tag{2.3}$$

where $|J|$ is the number of sentences in J, and $|E|$ is the number of sentences in E. A high $R(J, E)$-value occurs when $|J| \sim |E|$. Consequently, $R(J, E)$ can be used to measure the literalness of translation between J and E in terms of the ratio of the number of sentences.

Finally, we defined the score of alignment J_i and E_i as

$$\text{Score}(J_i, E_i) = \text{SIM}(J_i, E_i) \times \text{AVSIM}(J, E) \times R(J, E). \tag{2.4}$$

A high $\text{Score}(J_i, E_i)$ value occurs when

- sentences J_i and E_i are similar,
- documents J and E are similar,
- numbers of sentences $|J|$ and $|E|$ are similar.

$\text{Score}(J_i, E_i)$ combines both sentence and document similarities to discriminate between correct and incorrect alignments.

We use only high scoring sentence alignments to extract valid sentence alignments from noisy sentence alignments. We use Score in Eq. (2.4) as the score for a sentence alignment as described above because a variant of Score has been shown to be more appropriate than SIM for discriminating between correct and incorrect alignments (Utiyama and Isahara, 2003). When we compare the validity of two sentence alignments in the same document pair, the rank order of sentence alignments obtained by applying Score is the same as that of SIM because these alignments share common AVSIM and R. However, when we compare the validity of two sentence alignments in different document pairs, Score prefers the sentence alignment in the more similar (high AVSIM $\times$ R) document pair even if their SIM has the same value, while SIM cannot discriminate between the validity of two sentence alignments if their SIM has the same value. Therefore, Score is more appropriate than SIM when comparing sentence alignments in different document pairs because, in general, a sentence alignment in a similar document pair is more reliable than one in a dissimilar document pair.[11]

11. However, as pointed out by a reviewer, a document J which is a subset of E (or vice versa) could have a very low similarity score, $\text{AVSIM}(J, E)$. As a result, Eq. (2.4) could get a very low $\text{Score}(J_i, E_i)$, even for a pair of sentences that represent a perfect reciprocal translation. Therefore, if such cases exist in our patent corpus, many good sentence pairs may be lost. We have not yet investigated the amount of such cases in our corpus. We leave it for future work.

2.4.2 Noise Reduction in Sentence Alignments

We used the following procedure to reduce noise in the sentence alignments obtained by using the previously described aligning method on the 149,603 document pairs.

The number of sentence alignments obtained was about 7 million. From these alignments, we extracted only one-to-one sentence alignments because this type of alignment is the most important category for sentence alignment. As a result, about 4.2 million one-to-one sentence alignments were extracted. We sorted these alignments in decreasing order of scores and removed alignments whose Japanese sentences did not end with periods to reduce alignment pairs considered as noise. We also removed all but one of the identical alignments. Two individual alignments were determined to be identical if they contained the same Japanese and English sentences. Consequently, the number of alignments obtained was about 3.9 million.

We examined 20 sentence alignments ranked between 1,999,981 and 2,000,000 from the 3.9 million alignments to determine if they were accurate enough to be included in a parallel corpus. We found that 17 of the 20 alignments were almost literal translations of each other and 2 of the 20 alignments had more than 50% overlap in their contents. We also examined 20 sentence alignments ranked between 2,499,981 and 2,500,000 and found that 13 of the 20 alignments were almost literal translations of each other and 6 of the 20 alignments had more than 50% overlap. Based on these observations, we decided to extract the top 2 million one-to-one sentence alignments. Finally, we removed some sentence pairs from these top 2 million alignments that were too long (more than 100 words in either sentence) or too imbalanced (when the length of the longer sentence is more than five times the length of the shorter sentence). The number of sentence alignments thus obtained was 1,988,732. We call these 1,988,732 sentence alignments the ALL data set (*ALL* for short) in this chapter.

We also asked a translation agency to check the validity of 1000 sentence alignments randomly extracted from ALL. The agency conducted a two-step procedure to verify the data. In the first step, they marked a sentence alignment as

- A if the Japanese and English sentences matched as a whole,
- B if these sentences had more than 50% overlap in their contents,
- C otherwise,

to check if the alignment was correct. The number of alignments marked as A was 973, B was 24, and C was 3. In the second step, they marked an alignment as

- A if the English sentence reflected almost perfectly the contents of the Japanese sentence,
- B if about 80% of the contents were shared,
- C if less than 80% of the contents were shared,
- X if they could not determine the alignment as A, B, or C,

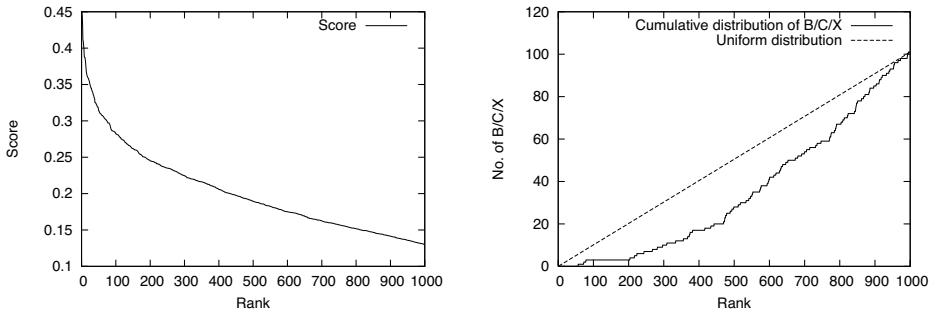

Figure 2.1 Distributions of scores and noisy alignments.

to check if the alignment was an adequate translation pair. The number of alignments marked as A was 899, B was 72, C was 26, and X was 3. Based on these evaluations, we concluded that the sentence alignments in ALL are useful for training and testing MT systems.

Next, we used these 1000 sentence alignments to investigate the relationship between the human judgments and Score given in Eq. (2.4). Figure 2.1 shows the distributions of scores and noisy alignments (marked as B, C, or X in the second step) against the ranks of sentence alignments ordered by using Score. The left figure shows that scores initially decreased rapidly for higher-ranking alignments, and then decreased gradually. The right figure shows the cumulative number of noisy alignments. The solid line indicates that noisy alignments tend to have low ranks. Note that if noisy alignments are spread uniformly among the ranks, then the cumulative number of noisy alignments follows the diagonal line. Based on the results shown in this figure, we concluded that Score ranked the sentence alignments appropriately.

2.5 Statistics of the Patent Parallel Corpus

2.5.1 Comparison of ALL and Source Data Sets

We compared the statistics of ALL with those of the source patents and sentences from which ALL was extracted to see how ALL represented the sources.

To achieve this, we used the primary international patent classification (IPC) code assigned to each USPTO patent. The IPC is a hierarchical patent classification system and consists of eight sections, ranging from A to H. We used sections G (physics), H (electricity), and B (performing operations; transporting) because these sections had larger numbers of patents than other sections. We categorized patents as O (other) if they were not included in these three sections.

As described in section 2.3, 84,677 patent pairs were extracted from the original patent data. These pairs were classified into G, H, B, or O, as listed in the Source

Table 2.1 Number of patents

IPC	ALL (%)	Source (%)
G	19,340 (37.9)	28,849 (34.1)
H	16,145 (31.6)	24,270 (28.7)
B	7,287 (14.3)	13,418 (15.8)
O	8,287 (16.2)	18,140 (21.4)
Total	51,059 (100.0)	84,677 (100.0)

Table 2.2 Number of sentence alignments

IPC	ALL (%)	Source (%)
G	946,872 (47.6)	1,813,078 (43.4)
H	624,406 (31.4)	1,269,608 (30.4)
B	204,846 (10.3)	536,007 (12.8)
O	212,608 (10.7)	559,519 (13.4)
Total	1,988,732 (100.0)	4,178,212 (100.0)

column of table 2.1. We counted the number of patents included in each section of ALL. We regarded a patent to be included in ALL when some sentence pairs in that patent were included in ALL. The number of such patents are listed in the ALL column of table 2.1. Table 2.1 shows that about 60% ($100 \times 51059/84677$) of the source patent pairs were included in ALL. It also shows that the distributions of patents with respect to the IPC code were similar between ALL and Source.

Table 2.2 lists the number of one-to-one sentence alignments in ALL and Source, where Source means the about 4.2 million one-to-one sentence alignments described in section 2.4.2. The results in this table show that about 47.6 % ($100 \times 1988732/4178212$) sentence alignments were included in ALL. The results also show that the distribution of the sentence alignments are similar between ALL and Source.

Based on these observations, we concluded that ALL represented Source well.

In the following, we use G, H, B, and O to denote the data in ALL whose IPCs were G, H, B, and O.

2.5.2 Basic Statistics

We measured the basic statistics of G, H, B, O, and ALL.

We first randomly divided patents from each of G, H, B, and O into training (TRAIN), development (DEV), development test (DEVTEST), and test (TEST) data sets. One unit of sampling was a single patent. That is, G, H, B, and O consisted of 19,340, 16,145, 7287, and 8287 patents (See table 2.1), respectively, and the patents from each group were divided into TRAIN, DEV, DEVTEST, and

Table 2.3 Number of patents

	TRAIN	DEV	DEVTEST	TEST	Total
G	17,524	630	610	576	19,340
H	14,683	487	493	482	16,145
B	6,642	201	226	218	7,287
O	7,515	262	246	264	8,287
ALL	46,364	1,580	1,575	1,540	51,059

Table 2.4 Number of sentence alignments

	TRAIN	DEV	DEVTEST	TEST	Total
G	854,136	33,133	27,505	32,098	946,872
H	566,458	20,125	19,784	18,039	624,406
B	185,778	6,239	6,865	5,964	204,846
O	193,320	6,232	6,437	6,619	212,608
ALL	1,799,692	65,729	6,0591	6,2720	1,988,732

TEST. We assigned 91% of the patents to TRAIN, and 3% of the patents to each of DEV, DEVTEST, and TEST. We merged the TRAIN, DEV, DEVTEST, and TEST of G, H, B, and O to create those of ALL. Table 2.3 lists the number of patents in these data sets and table 2.4 lists the number of sentence alignments.

2.5.3 Statistics Pertaining to MT

We measured some statistics pertaining to MT. We first measured the distribution of sentence length (in words) in ALL. The mode of the length (number of words) was 23 for the English sentences and was 27 for the Japanese sentences. (We used ChaSen to segment Japanese sentences into words.) Figure 2.2 shows the percentage of sentences for English (en) and Japanese (ja) with respect to their lengths. This figure shows that the peaks of the distributions were not sharp and that there were many long sentences in ALL. This suggests that patents contain many long sentences that are generally difficult to translate.

We then measured the vocabulary coverage. Table 2.5 lists the coverage for the types (distinct words) and tokens (running words) in each TEST section using the vocabulary in the corresponding TRAIN section for the English and Japanese data sets. These tables show that the percentages of types in TEST covered by the vocabulary in TRAIN were relatively low for both English and Japanese. However, the coverage of tokens was quite high. This suggests that patents are not so difficult to translate in terms of token coverage.

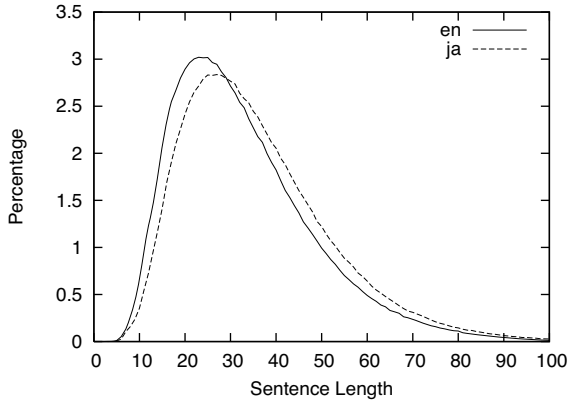

Figure 2.2 Sentence length distribution.

Table 2.5 Percentage of words in test sentences covered by training vocabulary.

(a) Coverage for English

	Type	Token
G	84.37	99.40
H	86.63	99.37
B	90.28	99.38
O	89.19	99.31
ALL	83.36	99.55

(b) Coverage for Japanese

	Type	Token
G	90.27	99.69
H	91.97	99.67
B	94.12	99.65
O	92.50	99.48
ALL	89.85	99.77

2.6 MT Experiments

2.6.1 MT System

We used the baseline system for the shared task of the 2006 NAACL/HLT workshop on statistical machine translation (Koehn and Monz, 2006) to conduct MT experiments on our patent corpus. The baseline system consisted of the Pharaoh decoder (Koehn, 2004a), SRILM (Stolcke, 2002), GIZA++ (Och and Ney, 2003), mkcls (Och, 1999), Carmel,[12] and a phrase model training code.

We followed the instructions of the shared task baseline system to train our MT systems.[13] We used the phrase model training code of the baseline system to extract phrases from TRAIN. We used the trigram language models made from TRAIN. To

12. `http://www.isi.edu/licensed-sw/carmel/`
13. The parameters for the Pharaoh decoder were "-b 0.00001 -s 100." The maximum phrase length was 7. The "grow-diag-final" method was used to extract phrases.

Table 2.6 Comparing reordering limits. Each MT system was trained on each of the G, H, B, O, and ALL TRAIN data sets, tuned for both reordering limits using each DEV data set, and applied to 1000 randomly sampled sentences extracted from each DEVTEST data set to calculate the %BLEU scores listed in these tables. The source and target languages were English-Japanese for (a) and Japanese-English for (b).

(a) English-Japanese

	no limit	limit=4
G	23.56	22.55
H	24.62	24.14
B	22.62	20.88
O	23.87	21.84
ALL	24.98	23.37

(b) Japanese-English

	no limit	limit=4
G	21.82	21.6
H	23.87	22.62
B	21.95	20.79
O	23.41	22.53
ALL	23.15	21.55

tune our MT systems, we did minimum error-rate training[14] (Och, 2003) on 1000 randomly extracted sentences from DEV using BLEU (Papineni et al., 2002) as the objective function. Our evaluation metric was %BLEU scores.[15] We tokenized and lowercased the TRAIN, DEV, DEVTEST, and TEST data sets. We conducted three MT experiments to investigate the characteristics of our patent corpus.

2.6.2 Comparing Reordering Limits

For the first experiment, we translated 1000 randomly sampled sentences in each DEVTEST data set to compare different reordering limits,[16] because Koehn et al. (2005) have reported that large reordering limits provide better performance for Japanese-English translations. We compared a reordering limit of 4 with no limitation. The results of table 2.6 show that the %BLEU scores for no limitation consistently outperformed those for limit=4. These results coincide with those of Koehn et al. (2005). Based on this experiment, we used no reordering limit in the following experiments.

2.6.3 Cross-Section MT Experiments

For the second experiment, we conducted cross-section MT experiments. The results are shown in tables 2.7 and 2.8. For example, as listed in table 2.7, when we used section G as TRAIN and used section H as TEST, we got a %BLEU score of 23.51 for English-Japanese translations, whose relative %BLEU score was 0.87 (=23.51/26.88) of the largest %BLEU score obtained when using ALL as TRAIN. In this case, we used all sentences in TRAIN of G to extract phrases and make

14. The minimum error-rate training code we used is available at `http://www2.nict.go.jp/x/x161/members/mutiyama/software.html`
15. %BLEU score is defined as BLEU $\times$ 100.
16. The parameter "-dl" for the Pharaoh decoder.

Table 2.7 %BLEU scores (relative %BLEU scores) for cross-section MT experiments (English-Japanese)

TEST: TRAIN	G	H	B	O	ALL
G	**25.89 (0.97)**	23.51 (0.87)	20.19 (0.82)	18.96 (0.76)	23.93 (0.91)
H	22.19 (0.83)	**25.81 (0.96)**	19.16 (0.78)	18.68 (0.75)	22.57 (0.86)
B	18.17 (0.68)	18.92 (0.70)	**22.54 (0.92)**	19.25 (0.77)	18.97 (0.72)
O	16.93 (0.63)	18.45 (0.69)	18.22 (0.74)	**24.15 (0.97)**	18.32 (0.70)
ALL	26.67 (1.00)	26.88 (1.00)	24.56 (1.00)	24.98 (1.00)	26.34 (1.00)

Table 2.8 %BLEU scores (relative %BLEU scores) for cross-section MT experiments (Japanese-English)

TEST: TRAIN	G	H	B	O	ALL
G	**24.06 (0.98)**	22.18 (0.90)	19.40 (0.85)	19.33 (0.80)	22.59(0.93)
H	20.91 (0.85)	**23.74 (0.97)**	18.11 (0.79)	18.60 (0.77)	21.28(0.88)
B	17.64 (0.72)	17.94 (0.73)	**21.92 (0.96)**	19.58 (0.81)	18.39(0.76)
O	17.50 (0.72)	18.43 (0.75)	18.57 (0.81)	**24.27 (1.00)**	18.67(0.77)
ALL	24.47 (1.00)	24.52 (1.00)	22.94 (1.00)	24.04 (0.99)	24.29(1.00)

a trigram language model. We used 1000 randomly sampled sentences in DEV of section G to tune our MT system. We used all sentences in TEST of section H to calculate %BLEU scores (see table 2.4 for the number of sentences in each section of TRAIN and TEST).

The results in these tables show that MT systems performed the best when the training and test sections were the same.[17]

These results suggest that patents in the same section are similar to each other, while patents in different sections are dissimilar. Consequently, we need domain adaptation when we apply our trained MT system to a section that is different from that on which it has been trained. However, as shown in the ALL rows, when we used all available training sentences, we obtained the highest %BLEU scores for all but one case. This suggests that if we have enough data to cover all sections we can achieve good performance for all sections.

Tables 2.7 and 2.8 show that both the domain and quantity of training data affect the performance of MT systems. We conducted additional experiments to see the relationship between these two factors. We trained a Japanese-English MT system

17. The results in these tables indicate that %BLEU scores for English-Japanese translations are higher than those for Japanese-English translations. This is because Japanese words are generally shorter than English words. As described in section 2.5.3, the mode of the length for English sentences was 23 and that for Japanese was 27. This suggests that it is easier to reproduce Japanese n-grams, which leads to higher %BLEU scores.

Table 2.9 %BLEU scores for the additional experiments

	B	G	H	O
Same	21.92	24.06	23.74	24.27
ALL\B	20.72	24.39	24.47	23.69
ALL	22.94	24.47	24.52	24.04

on ALL excluding B (ALL\B). We reused the feature weights of the MT system that was trained and tuned on ALL to save tuning time. We used the system to translate the sentences in TEST of sections B, G, H, and O. The %BLEU scores are shown in the ALL\B row of table 2.9. The figures listed in the row labelled Same were the %BLEU scores obtained by applying MT systems trained on each section to that section. The figures in the Same and ALL rows were taken from table 2.8.

Table 2.9 shows that the system trained on ALL outperformed the system trained on ALL\B for all sections. This suggests that it is more important to have more training data. Next, the system trained on O outperformed the systems trained on ALL and ALL\B. In this case, adding data from other domains reduced performance. The systems trained on G and H were outperformed by those trained on ALL and ALL\B. Thus, additional data helped to improve performance in these cases. Finally, the system trained on B outperformed the system trained on ALL\B despite the fact that the number of sentences in ALL\B (1,613,914) was much larger than that in B (185,778). This suggests that it is the domain that matters more than the quantity of training data.

Table 2.10 lists 15 examples of translation obtained from the Japanese-English MT system trained and tested on TRAIN and TEST of ALL. Reference translations are denoted by an R, and MT outputs are denoted by an M. The vertical bars (|) represent the phrase boundaries given by the Pharaoh decoder. These examples were sampled as follows: We first randomly sampled 1000 sentences from TEST of ALL. The correctness and adequacy of the alignment of these sentences were determined by a translation agency, as described in section 2.4.2. We then selected 899 A alignments whose English translation reflected almost perfectly the contents of the corresponding Japanese sentences. Next, we selected short sentences containing less than 21 words (including periods) because the MT outputs of long sentences are generally difficult to interpret. In the end, we had 212 translations. We sorted these 212 translations in decreasing order of *average n-gram precision*[18] and selected five sentences from the top, middle, and bottom of these sorted sentences.[19]

Table 2.10 shows that top examples (1 to 5) were very good translations. These MT translations consisted of long phrases that contributed to the fluency and

18. Average n-gram precision is defined as $\sum_{n=1}^{4} \frac{p_n}{4}$ where p_n is the modified n-gram precision as defined elsewhere (Papineni et al., 2002).
19. We skipped sentences whose MT outputs contained untranslated Japanese words when selecting these 15 sentences.

adequacy of translations. We think that the reason for these good translations is partly due to the fact that patent documents generally contain many repeated expressions. For example, example 2R is often used in patent documents. We also noticed that lcd61 in example 5M was a very specific expression and was unlikely to be repeated in different patent documents, even though it was successfully reused in our MT system to produce 5M. We found a document that contained lcd61 in TRAIN and found that it was written by the same company who wrote a patent in TEST that contained example 5R, even though these two patents were different. These examples show that even long and/or specific expressions are reused in patent documents. We think that this characteristic of patents contributed to the good translations.

The middle and bottom examples (6 to 15) were generally not good translations. These examples adequately translated individual phrases. However, they failed to adequately reorder phrases. This suggests that we need more accurate models for reordering. Thus, our patent corpus will be a good corpus for comparing various reordering models (Koehn et al., 2005; Nagata et al., 2006; Xiong et al., 2006).

2.6.4 Task-Based Evaluation of the Original Alignment Data

For the third experiment, we assessed the quality of the original alignment data in a task-based setting. In section 2.4.2, we selected the first 2 million sentence alignments based on our observations of the quality of the alignment data. In this section, we increase the size of training data to see how MT performance evolves using more data.

We used the 3.9 million one-to-one sentence alignments obtained in section 2.4.2 as our training data. From this data, we removed the alignments contained in the patents in DEV, DEVTEST, or TEST of ALL. We also removed some sentence pairs that were too long or too imbalanced, as discussed in section 2.4.2. We tokenized and lowercased this data. As a result, we obtained 3,510,846 one-to-one sentence alignments sorted by Score in Eq. (2.4).

We conducted controlled Japanese-English and English-Japanese MT experiments using these 3.5 million sentence alignments. We used the common (a) word alignment data, (b) language models, (c) feature weights, and (d) test data. We changed the size of word alignment data when we built phrase tables in the following experiments.

The common settings were obtained as follows. First, (a) we made word alignment data from all sentence alignments using GIZA++. We randomly divided all the sentence alignment data into two halves, applied GIZA++ separately to each half, and combined them to obtain all word alignment data. (b) We made Japanese and English trigram language models from the first 3.5 million sentence alignments. (c) We reused the feature weights of the English-Japanese and Japanese-English MT systems that were trained and tuned on ALL as described in section 2.6.3. (d) We randomly sampled 2000 sentences from TEST of ALL as the test data.

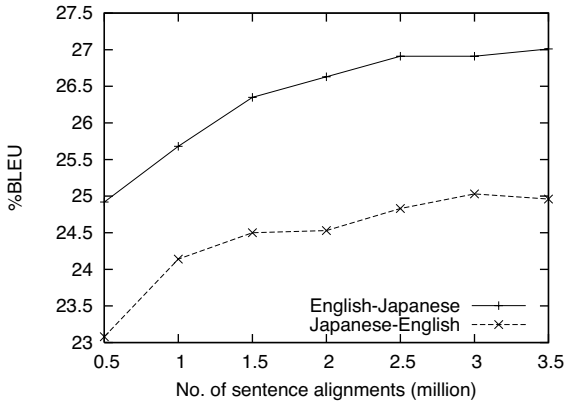

Figure 2.3 Relationship between the %BLEU scores and the number of sentence alignments (in millions).

Finally, we used the first 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, and 3.5 million sentence alignments to make phrase tables. The %BLEU scores obtained with these phrase tables in the common settings are shown in figure 2.3.

Figure 2.3 shows that the %BLEU scores for the English-Japanese MT experiments reached a plateau around 2.5 million sentence alignments. The %BLEU scores for the Japanese-English experiments increased up to 3.0 million sentence alignments and then dropped when 3.5 million alignments were used as the training data.

These observations indicate that, up to certain points, the increase in the size of the training data offsets the decrease in alignment quality. However, the performance of the MT systems reached a plateau or even decreased after those points due to noise in the alignment data. Therefore, based on the results from these experiments and the results shown in figure 2.1, we conclude that Utiyama and Isahara's method effectively sorted the sentence alignments in decreasing order of their quality.

2.7 Conclusion

Large-scale parallel corpora are indispensable language resources for MT. However, there are only a few publicly available large-scale parallel corpora.

We have developed a Japanese-English patent parallel corpus created from Japanese and U.S. patent data provided for the NTCIR-6 patent retrieval task. We used Utiyama and Isahara's method and extracted about 2 million clean sentence alignments. This is the largest Japanese-English parallel corpus to date. Its size is comparable to other large-scale parallel corpora. This corpus and its extension will

be used in the NTCIR-7 patent MT task and made available to the public after the 7th NTCIR-7 workshop meeting.

We hope that the patent corpus described in this chapter will promote MT research in general and the Japanese-English patent MT research in particular.

Table 2.10 Examples of reference (R) and machine (M) translations

Top	
1R	the printer 200 will now be described .
1M	next , the printer | 200 will now be described . |
2R	preferred embodiments of the present invention will be described hereinbelow with reference to the accompanying drawings .
2M	hereinafter , | preferred embodiments of the | present invention | will be described with reference | to the accompanying drawings | . |
3R	more specifically , variable tr (k) is defined by the following equation .
3M	namely , the variable | tr (| k) | is defined | by the following equation | . |
4R	wd signal is further applied to a command decoder 24 and a data comparator 23 .
4M	further , | signal | wd | is | also applied to a | command decoder 24 | and a data comparator | 23 | . |
5R	at this time , the selected page is displayed on the lcd 61 .
5M	at this time , the | selected | page is | displayed on | the | lcd61 | . |
Middle	
6R	further , reference numbers 201-219 indicate newly-added circuit elements .
6M	further , | reference numerals | 201 to 219 | is newly | added to | the circuit | elements . |
7R	for this purpose , a magnetic head 3 for recording is provided near the disk 1 .
7M	therefore , the | recording | magnetic head | 3 is | provided adjacent to the | disk 1 | . |
8R	accordingly , the energy exerting an influence on the occupant can be reduced .
8M	as a result , the | occupant | on | energy | can be reduced . |
9R	note that nothing is connected to the 1-bit output terminals q0 , q1 of the up-counter 131 .
9M	the | output terminals q0 | , q1 | , the | number | of bits of the | up counter | 131 | is also not connected . |
10R	application program 20 is executed under support of operating system 18 .
10M	an operating system | 20 | of the | support | 18 | under the | application program | is executed . |
Bottom	
11R	numeral 14 denotes a suction surface non-separation streamline , which improves the p-q characteristic and reduces noise .
11M	the | back pressure | , and | no | peeling | surface | 14 | , and | noise | is reduced . | improving characteristics of the | p or q | represents a | stream line |
12R	the use of a robot for deburring work is a known prior art .
12M	deburring | operation | using the | robot | is conventionally | known | technique . |
13R	rdp indicates an address to which a cpu accesses presently .
13M	the cpu | rdp | is currently | being accessed | address | is shown . |
14R	the same is true with regard to the b signal independently of the r signal .
14M	this is | regardless of signals | r | and b signals | similarly . |
15R	the structure of the airbag device 1 will be explained hereinafter .
15M	the air bag apparatus | 1 | are as follows . |

3 Automatic Construction of Multilingual Name Dictionaries

Bruno Pouliquen
Ralf Steinberger

Machine translation and other natural language processing systems often experience performance loss if they have to process texts with unknown words, such as proper names. Proper name dictionaries are rare and can never be complete because new names are being made up all the time. A solution to overcome this performance loss could be to recognize and mark a named entity in text before translating it and to carry over the named entity untranslated. This would also help avoid the accidental translation of a name such as Bill Black, e.g., into French as "facture noire". An even better translation would be achieved if the target language spelling of the name would be used, and this seems even crucial when translating from languages with a different script, such as Chinese, Arabic, or Cyrillic. We will show that multilingual name dictionaries are furthermore helpful for a number of other text analysis applications, including information retrieval, topic detection and tracking, relation and event extraction, and more. We then present a method and a system to recognize named entities of the types "person" and – to some extent – "organisation" in multilingual text collections and to automatically identify which of the newly identified names are variants of a known name. By doing this for currently 19 languages and in the course of years, a multilingual name dictionary has been built up that contains to date over 630,000 names plus over 135,000 known variants, with up to 170 multilingual variants for a single name. The automatically generated name dictionary is used daily, for various purposes, in the publicly accessible multilingual news aggregation and analysis system NewsExplorer.

3.1 Introduction and Motivation

There is an – often not explicitly mentioned – assumption that names do not need translating. To some extent, this is true, at least for person names in related lan-

guages such as those spoken in western European countries. The usefulness of name translation is much more obvious for languages using different writing systems, such as the Chinese, Arabic, and Cyrillic scripts, because it is not immediately obvious to most readers that *Џорџ В. Буш* , *Τζωρτζ Μπους*, and *Джордж Уокер Буш* all refer to the same person, President *George W. Bush*. It is less widely known that even among languages using the Roman script, names are sometimes transliterated. The Latvian equivalence of the same name, for example, is *Džordžs V. Bušs*. We argue that the usefulness of name dictionaries is not restricted to cases of cross-script or intrascript transliteration, but that many more name variants occur within closely related languages and even within the same language. While it may be obvious to human readers that *George W. Busch* (found in German text), *George Walter Bushi* (Estonian), and *Georges Bush* (French) refer to the same person,[1] many automatic text analysis applications require an exact match of name variants, meaning that they will benefit from a reference list of possible name variants.

3.1.1 Contents

The following subsections will serve as an introduction to the presented work on automatically producing multilingual name variant dictionaries. They describe the effect proper names have on machine translation (MT; section 3.1.2) and mention some other text analysis applications that require name dictionaries (section 3.1.3). Section 3.1.4 summarizes the various reasons why name variants exist. Current efforts to produce name dictionaries, as well as some background information on named entity recognition (NER) and name variant matching are summarized in section 3.2 on related work.

The sections thereafter describe lightweight methods to recognize named entities in many different languages (section 3.3) and to deal with highly inflected languages such of those of the Balto-Slavonic language group (section 3.4), followed by a short summary of named entity recognition evaluation results for the presented system (section 3.5). Lightweight procedures are essential when aiming at gathering name variants from a large variety of languages. We will then describe a method to identify and merge name variants that does not rely on language pair–specific resources or methods (section 3.6). The method, consisting of the steps transliteration, name normalization, and approximate matching, can be applied to name variants of the same language, of different languages using the same writing system, or to languages using different scripts. The special feature of this approach is that – unlike state-of-the-art approaches – it does not require any language pair–specific resources, which

1. See `http://press.jrc.it/NewsExplorer/entities/en/1.html` for a list of name variants for the president of the United States, as found in real-life news collections in the course of several years of news analysis. The live entity page `http://langtech.jrc.it/entities/` listing the names identified only since midnight (Central European Time) shows that the usage of name variants is the rule rather than an exception.

makes it easy to add new languages. Section 3.7 concludes and points to future work.

The methods presented here are being applied daily to tens of thousands of news texts in many different languages. The results are publicly accessible in the multilingual news aggregation and analysis system *NewsExplorer*, available at `http://press.jrc.it/NewsExplorer` NewsExplorer clusters related news items written in the same language on a daily basis and links each news cluster with related news clusters from the previous days (topic detection and tracking) and across languages (cross-lingual topic tracking). Finally, NewsExplorer cumulatively collects information on name mentions, including name variants, co-occurrence statistics with other names, name attributes mentioned across the many languages, etc.

3.1.2 Proper Names and Machine Translation

Starting from the observation that "name translation has proven to be a challenge for machine translation providers," Hirschman et al. (2000) identified the following three types of problems related to proper names:

1. Translation of proper names as if they were normal meaningful words (e.g., the name of the former German Chancellor *Helmut Kohl* translated as *Helmut Cabbage*).

2. Idiomatic rather than literal translation of names; this mostly concerns organization names (e.g., *Escuela de Derecho de Harvard* should not be backtransliterated as *Harvard School of the Right*, but the original *Harvard Law School* should be used).

3. Rendering of names in a format that is unusable by target language processing. This is mainly an issue of transliteration, as foreign characters (such as those of the Cyrillic alphabet) cannot be displayed or read in other languages.

We evaluated various Spanish-to-English MT systems with respect to proper names and stated error rates of between 46% and 49% at token level and 31% to 35% at entity level. Based on these results, we recommend that named entities should be identified in the source language before the MT process is started in order to avoid translation errors, by simply copying over the names into the target language.

Vilar et al. (2006) compared name translation errors to other types of MT errors, based on their Chinese to English MT system trained on 200 million words. They found that the relative frequency of errors related to proper names is 8.9% (5.4% for person names). The biggest sources of errors in their study were incorrect words (wrong lexical choice or incorrect morphological form), followed by missing words (27.5%). It goes without saying that the relative impact of names on overall MT performance evaluation depends heavily on the text type and thus the number of (especially unknown) names in the text.

3.1.3 Relevance of Multilingual Name Dictionaries to Other Text Analysis Applications

Multilingual name dictionaries linking name variants to the same entity are useful for a range of further applications. As part of the *Topic Detection and Tracking* (TDT) competitions,[2] Larkey et al. (2004) worked on cross-lingual topic tracking, and specifically on identifying which Mandarin Chinese and Arabic language news texts are related to English news texts about a chosen event. For that TDT exercise, participants used either MT or bilingual dictionaries to map Chinese and Arabic texts to the corresponding English texts. Larkey et al. observed that the results for cross-lingual topic tracking suffer from poorly translated proper names or from translated names being spelled differently than in the English news. They took this observation to support their so-called *native language hypothesis*, which says that it is better to process the documents monolingually (NER, topic tracking) and to then only map the result with the English results, rather than translating document by document and operating on the machine-translated texts. Hakkani-Tür et al. (2007) came to the same conclusion regarding information retrieval. This insight is not surprising as different news stories can be distinguished quite accurately based on the persons, organizations, and locations involved. Steinberger et al. (2004) therefore propose to use normalized named entities plus other features as anchors to link related news across languages.

Friburger and Maurel (2002) showed that the identification and usage of proper names, and especially of geographical references, significantly improves document similarity calculation and clustering. Hyland et al. (1999) clustered news and detected topics exploiting the unique combinations of various named entities to link related documents. However, according to Friburger and Maurel (2002), the usage of named entities alone is not sufficient.

Ignat et al. (2005) present two applications that rely on named entities to provide cross-lingual information access. The first one is a *cross-lingual glossing* application that identifies named entities in foreign language text, highlights them, and displays the named entity in the language of the user. The idea is to help users decide on the potential relevance of a foreign language text so that they can decide whether or not to get the text translated. The second application gives an overview of large multilingual thematically related news collections by using an interactive geographical map showing which locations are being referred to how often. In addition to the geographical map, the application also displays frequency lists of named entities and of specialist terms. Each document collection can thus be visualized in an intuitive manner and users can explore it via each of the three entity types.

2. See `http://www.nist.gov/speech/tests/tdt/index.htm` for an introduction to the TDT exercises.

Pouliquen et al. (2007a) produce multilingual social networks based on the co-occurrence of the same names in news texts written in 19 different languages. In Pouliquen et al. (2007b), a multilingual quotation network is produced based on who mentions whom in direct speech quotations (reported speech) in 11 different languages. Tanev (2007) describes work that aims at learning to extract English language patterns that describe specific relationships that hold between persons, such as family, support, criticism relationship, and more.

Identifying name mentions and their variants in documents and storing the extracted metainformation in a knowledge base has the benefit of providing users with a powerful metasearch capacity. In *NewsExplorer* (Steinberger et al. (2005)), for instance, users can search for news and historically collected name attribute information. Due to the name dictionaries used, customers querying for a name will find news articles independently of the spelling of the name.

Generally speaking, any work that aims at extracting information about entities, their relations, the events in which they are involved, etc., will benefit from multilingual name variant dictionaries because they will allow identification of a coreference between entity mentions even if the names are spelled differently. Such dictionaries will be particularly useful when the applications are multilingual or cross-lingual. However, as the next subsection shows, even monolingual applications will benefit from name dictionaries because the same entities are often referred to with different surface strings.

3.1.4 Why do Name Variants Exist?

The same person can be referred to by different name variants. The main reasons for these variations are listed below. Not all of the variant types should be included in a name dictionary:

1. The reuse of name parts to avoid repetition (e.g., *Condoleezza Rice*, *Ms. Rice*, and *Secretary of State Rice*).

2. Morphological variants such as added suffixes (e.g., in Polish, *Tonym Blairem, Toniego Blaira, Tony'ego Blaira* may be found for the name *Tony Blair*).

3. Spelling mistakes (e.g., *Condolleezzaa Rice*, *Condolizza Rice*, *Condoleezza Rica*, etc., all repeatedly found in real news texts).

4. Adaptation of names to local spelling rules (e.g., the Polish name *Lech Wałęsa* is almost always found as *Lech Walesa* in other languages because the letters "ł" and "ę" are not part of their alphabet. Similarly, *Émile Lahoud* is often written *Emile Lahoud*).

5. Transliteration differences due to different transliteration rules or different target languages (e.g., the same Russian name *Михаил Фрадков* has been found to be transliterated as *Mikhail Fradkov* in English, *Michail Fradkow* in German, *Mikhaïl Fradkov* in French, and *Mijaíl Fradkov* in Spanish).

Table 3.1 Most frequent name variants for the President of Sudan, Omar al-Bashir, automatically extracted from news articles. The language of the text where the variant was found is indicated in brackets. Names in languages other than the 19 active NewsExplorer languages were automatically retrieved from Wikipedia. (See `http://press.jrc.it/NewsExplorer/entities/en/934.html`)

Omar al-Bashir (Eu,sv)	Omar Hassan Ahmed al-Bashir (Eu,en)	Omar Hassan el Beschir (de)
Omar Hassan al-Bashir (Eu,sv)	Omar al-Bachir (es,pt)	Omar Hassan Ahmed el Baschir (de)
Omar el-Béchir (fr,pt)	Омар Ел-Башир (bg)	Umar Hassan Ahmad al-Bashir (en)
Omar el-Bashir (Eu,sl)	Omar Hassan al-Baschir (de)	Omar al Béchir (fr)
Omar al-Beshir (Eu,pt)	Omar Al- Bashir (en,pt)	عمر حسن أحمد البشير (ar)
Omar Hassan (de,sv)	Umar Hasan al-Bashir (en)	Omar al-Hassan (en)
Omar al-Béchir (fr,pt)	Omar Hassan Al Bashir (en,pt)	Omar Ahmad al-Bashir (en)
Omar Hassan Ahmad al-Bashir (Eu,pt)	Omar El-Béchir (fr,pt)	Omar -Hassan al-Beshir (en)
عمر البشير (ar,fa)	Omar Al Beshir (en)	Omar el- Béchir (fr)
Omar al Bashir (de,pt)	Omar el Beshir (de,it)	Omar Beschir (de)
Omar Bashir (en,pt)	Omar Al Bechir (fr,pt)	Omar Hassan Ahmed Al-Bashir (en)
Umar al-Bashir (en)	Omar Hassan Al-Baschir (de)	Omar al-Béchir (fr)
Omar Al-Bachir (fr)	Omar Hassan el-Béchir (fr)	Omar Hassan El Bechir (fr)
عمر حسن البشير (ar,fa)	Umar el-Bashir (en)	オマル・アル＝バシール (ja)
Omar el Baschir (Eu,de)	Omar Hassan A. Al-Bashir (en)	Ομάρ Μπασίρ (el)
Омар Башир (bg,ru)	Umar Hassan al-Bashir (en)	Омар ал Башир (ru)
Omar el-Beshir (en,pt)	Omar Hasan al-Bachir (es)	Омар ал-Башир (ru)
Omar al Baschir (de)	Omar al Beschir (de)	Omar el-Bachir (ft)
Omar el Bashir (de,nl)	Omar Hassan al Bachir (en,fr)	Omar Beshir (sv)
Omar el-Bechir (es,pt)	Omer Al-Bashir (en)	Omar Hassan al-Beshir (en)
Omar Hassan al Bashir (de,it)	Omar al Bachir (es,fr)	Omer al Bašir (sl)
Omer al-Bashir (en)	Omar Al- Baschir (de)	Omar El Beshir (en)
Omar Hasan al Bachir (es)	Omar Hassan Ahmed el-Bashir (en)	Omar El-Beshir (en)
Omar Hassan Al-Bashir (de,nl)	Omar Hassan el-Bashir (en,es)	Omar Hassen al Bachir (fr)
Omar al-Baschir (de)	Omar Bechir (es,pt)	Omar El-Bechir (fr)
Omar Al Bashir (en,pt)	Omar Hassan al-Béchir (fr,nl)	Omar el-Báchir (pt)
Omar al-Bechir (fr,pt)	Omar Al-Béchir (fr)	Omar Hasan el-Bechir (es)
Omar Hassan Bashir (en,sv)	Omer Hassan al-Bashir (en)	Omar Hasan Ahmad al-Bashir (Eu,nl)
Omar El Bashir (en,nl)	Omar el-Béshir (pt)	Omar al-Baszir (pl)
Omar el Bechir (it,pt)	Omer al- Bashir (en)	Omar El-Béshir (pt)
Omar el Beschir (de)	Omar al Beshir (de,it)	Amer Bashir (it)
Omar El-Bashir (en)	Umar Hasan Ahmad al-Bashir (en,it)	Omar al-Behsir (en)
Omar El Béchir (en,pt)	Omar Bachir (fr,pt)	Omar Al Béchir (fr)

6. In the specific case of Arabic, where short vowels are usually not written, vowels need to be inserted during transliteration, which can cause large numbers of variants (e.g., the Arabic name *Mohammed* consists of only the consonants *Mhmd*, which explains the different Romanized variants *Mohammed*, *Muhameed*, *Muhamad*, etc.).

It is our experience that name variants are not only found in the news in different languages but even within the English language news we find many variants. Table 3.1 shows that many variants for the same name can be found within the same language (33 variants in English alone). Our aim is to identify all variants as belonging to the same person and name identifier, so that users can search and find all documents mentioning a person, independently of the spelling.

3.2 Related Work

In this section, we will mention existing efforts to build multilingual name dictionaries (section 3.2.1) and will briefly report on related work concerning NER (section 3.2.2) and name variant matching (section 3.2.3).

3.2.1 Existing Name Dictionaries or Efforts to Build Such Lists

Bouchou et al. (2005) are working on compiling a multilingual name dictionary, ProLex, through a manual effort, including also historical name variants, geolocations, and more.

Hassan et al. (2007) exploit both parallel and comparable corpora to compile Arabic-English name dictionaries. They produce comparable corpora by translating an existing Arabic text into English, and by searching for real-life English documents automatically by querying search engines with terms extracted from the English MT result. As they will find various name candidates for each Arabic name, they use the phonetic string-matching algorithm *Editex* (Zobel and Dart, 1996) and a filter based on the length of the names to identify the best English language name candidate. Editex is similar to the more widely known *edit distance*. However, Editex considers the phonetic similarity of letters or letter groups, stating for instance that the distance between the letter "t" and its voiced counterpart "d" is only half that of unrelated letters (e.g., "t" and "m"). For an overview of other possible string distance metrics, see Cohen et al. (2003).

Klementiev and Roth (2006b) also exploit comparable corpora to compile a list of Russian-English name pairs. They propose an iterative algorithm based on the observation that named entities have a similar time distribution in news articles (synchronicity of names).

In section 3.2.3, we will present some more work on compiling bilingual name dictionaries by using transliteration rules, either machine-learned from bilingual name example lists or based on phonetic observations.

3.2.2 Named Entity Recognition

NER is a known research area (e.g., MUC-6; Nadeau and Sekine, 2007). Work on *multilingual* NER started much more recently (e.g., MLNER; Poibeau, 2003; Tjong Kim Sang and De Meulder, 2003).

People's names can be recognized in text through various methods:

1. through a lookup procedure if a list of known names exists;
2. by analyzing the local lexical context (e.g., *"President" Name Surname*);
3. because part of a sequence of candidate words is a known name component (e.g., *"John" Surname*), or

4. because the sequence of surrounding parts-of-speech (or other) patterns indicates to a tagger that a certain word group is likely to be a name.

It is rather common to use machine learning approaches to learn context patterns that recognize names by looking at words or syntactic structures surrounding known names. For the European languages, it is sufficient to consider only uppercase words as name candidates, although common lowercase name parts need to be allowed (e.g., Dutch *van der*, Arabic *al* and *bin*, French *de la,* etc.). Other languages, such as Arabic, do not distinguish case. In the work presented here, we currently use methods 1 to 3 above, but we do not use part-of-speech taggers, because we do not have access to such software for all languages of interest. Until now, the focus has been on people's names, but we also recognize some organization names, using mainly method 3 (i.e., because the candidate contains organization-specific name parts such as *International* or *Bank*).

The Joint Research Centre (JRC)'s NER tools differ in some features from other similar tools. These differences are due to the specific needs of the NewsExplorer environment: First, we aim at covering many languages rather than at optimizing the tools for a specific language, because NewsExplorer already now covers 19 languages and more languages are to come. Second, we aim at identifying each name at least once per text or even per cluster of articles and we have no specific interest in identifying every single occurrence of the name in a text. This is due to the fact that users need to see the names mentioned in each *cluster* of related news and that they should be able to find all clusters in which a certain name was mentioned. The aim is thus to optimize the recall of the name recognition system *per cluster or per document*, instead of per name instance. Third, we aim at recognizing names with at least two name parts (typically first name and last name, e.g., *Angela Merkel*) and ignore references to names consisting of only the last name (e.g., *Chancellor Merkel*). The decision to ignore one-word name parts is due to the facts that single words are more likely to be ambiguous and that it is safer to use the combination of first and last names to match name variants (see section 3.6.3). The name recognition recall is nevertheless very satisfying because news articles usually mention the full name of a person at least once per article, together with the title or function of the person. However, when applying the NER tools to other text types, we had to relax and extend the patterns in order to allow the recognition of last names only (e.g., *Chancellor Merkel*) and combinations of first name initials and last names (e.g., *A. Merkel*).

3.2.3 Name Variant Matching

Most past work on detecting name variants in text focuses on coreference resolution for partially overlapping strings (*Condoleezza Rice* vs. *Ms. Rice*, etc.) and on categorizing the named entities found (e.g., identifying whether *Victoria* is a person or a city). Morphological variants of names are typically dealt with by using lemmatization or stemming software. For this purpose, morphological analyzers are used

to identify variants and to return the normal form (usually the nominative). This presumably is the best-performing approach (although names are often unknown words so that morphological analysis errors are likely), but morphological tools are not easily available for many languages, or they are costly. As an alternative, Klementiev and Roth (2006b) propose to simply map name variants by truncating all letters after the first five letters and to map on the basis of the remaining "stem."

Freeman et al. (2006) combine the Editex distance (Zobel and Dart, 1996) with a large number of manually written language-specific normalization rules (both for English and for Arabic) to create a name mapping system for Arabic and English.

To our knowledge, there are no NER systems that automatically match name variants due to wrong spelling or varying transliteration, although approximate matching tools exist for name variant matching inside databases. A number of different techniques have been developed for this purpose, including edit distance and letter n-gram overlap (Cohen et al., 2003).

For names written in different writing systems (e.g., Cyrillic or Arabic alphabets; Chinese ideographs), transliteration rules are frequently used. These can be handwritten (e.g., Freeman et al., 2006) or learned from large lists of bilingual name equivalences (e.g., Knight and Graehl, 1997; Sherif and Kondrak, 2007). The American *automatic content extraction* programme ACE-07 (2007) organized its 2007 benchmark test with the title *entity translation*. The aim of the competition was the automatic extraction of named entities from Arabic and Chinese texts and their representations in English. In such a bilingual context, pronunciation rules can be used to infer possible target language spellings. Within the news domain and the multilingual environment of the NewsExplorer, however, we cannot take a certain pronunciation for granted. The French name *Chirac* found in Italian news, for instance, must not be pronounced according to Italian phonetic rules as it would otherwise end up as /*kirak*/. Conversely, the Italian name *Vannino Chiti* should – even in French news – be pronounced according to Italian rules /*Vanino Kiti*/.

The fact that the national origin of names in news texts is not known and that pronunciation rules cannot be taken for granted is often ignored and cannot be stressed enough. Identifying the origin of a name and guessing its pronunciation is often difficult. The name *Arthur Japin*, found in a Dutch newspaper, could be pronounced according to the Flemish-Dutch rules(/artür yapin/), according to the French rules (/artür japiñ/), in the English way (/arthur djapin/), or others (the Dutch novelist actually uses the French pronunciation and should be transliterated in Russian as Артур Жапа and not Артур Джапин or Артур Япин). The origin of a name cannot be accurately determined from the string alone. Konstantopoulos (2007) reports an accuracy of 60% when automatically guessing the origin of names relying only on the surface string. This is low, but the task clearly is difficult: the comparative result for human guessing was even lower, namely 43%.

The approach described in the following sections differs from the state of the art in that we use the same empirically determined spelling normalization rules for all languages to produce a simplified, or "normalized" form that can be used to

compare all variants found. For morphological variants, we use the language-specific suffix stripping or suffix replacement rules described in section 3.4.1.

3.3 Multilingual Recognition of New Names

The following subsections will describe the lightweight process to identify new named entities daily in multilingual news texts (section 3.3.2) and to mine Wikipedia for further name variants in additional languages (section 3.3.3). The algorithm described in sections 3.3.2 and 3.4 are applied daily on the multilingual NewsExplorer news data, described in section 3.3.1.

3.3.1 Background: Multilingual News Data

NewsExplorer (Steinberger et al., 2005) is part of the *Europe Media Monitor* family of applications (EMM) (Best et al., 2005). NewsExplorer receives from EMM an average of 40,000 news articles in 35 languages, scraped from about 1400 news portals in Europe and around the world. EMM converts all articles into a standard UTF-8-encoded RSS format and classifies them into a given set of several hundred categories. The articles received within the last few minutes and hours are displayed on the live NewsBrief website (`http://press.jrc.it/`), which is updated every ten minutes. For 19 of the 35 languages, NewsExplorer clusters related articles once per day, separately for each language, in order to group news about the same event or subject. From each of these clusters, person, organization and location names are extracted. The information on the entities found in each cluster, combined with the database information on names and name variants and further information, is then used to link equivalent news clusters across the whole range of languages and for all language pair combinations. A database keeps the information where and when each name was found, which other names were found in the same cluster, and which name attributes were found next to the name. This process is described in Steinberger and Pouliquen (2007).

3.3.2 A Lightweight Recognition Process Allowing High Multilinguality

The rules to identify new names consist of lexical patterns which have mostly been collected in a bootstrapping procedure: We first collected open source lists of titles (*Dr., Mr., President*, etc.) and wrote simple local patterns in PERL to recognize names in a collection of three months of English, French, and German news. We then looked at the most frequent left- and right-hand side contexts of the resulting list of known names and hand-selected the good candidates. For English alone, we have collected about 3300 local patterns, consisting of titles (*Dr.*, *Mr*, etc.), country adjectives (such as *Estonian*), professions (*actor*, *tennis player*, etc.), and other specific patterns, e.g., expressing age (e.g., *[0-9]+ year-old)*, functions (*ambassador to [A-Z][a-z]+*), or actions (e.g., in French *a affirmé*; in Portuguese *explicou*; in

Spanish *aprovechó*). We refer to these local patterns as *trigger words*. Combinations of the different types of trigger words are also recognized (e.g., *Polish ambassador to Iraq*). Additionally, we check all groups of uppercase words found in text against long lists of known first names in the 19 NewsExplorer languages. If one part of the name candidate is a known first name, the group will also be marked as a named entity.

To reduce the number of wrongly identified names in sequences of uppercase words (e.g., in English language news heads, where content words are often spelled in uppercase), we also make use of large numbers of *name stop words*. These name stop words are lists of common words which, during the early development phases, were frequently wrongly identified as name parts. When found in a name candidate string, they are used as separators that will not be allowed as part of the name. In English, for instance, the days of the week are on the name stop word list, because they are spelled in uppercase and frequently lead to wrong names. For instance, in phrases such as *During the speech of Chancellor Angela Merkel Friday . . .*, the name stop word avoids identifying the whole string *Angela Merkel Friday* as a name.

The patterns additionally allow some common (uppercase or lowercase) name parts such as *Mc, de, bin, al-, van der*, *O', M'* in order to recognize names like *Paul McCartney, Cécile de France, Osama bin Laden, Rafik al-Hariri*, *Rafael van der Vaart, Martin O'Neill*, and *Mohamed M'Barek*.

The mentioned patterns allow the program to recognize new names (e.g., in *the American doctor John Smith*), but a stored list of such patterns is additionally useful to give users supplementary information about persons. In the previous example, for instance, the user will see that *John Smith* probably is an American doctor. When a name is often used with the same trigger words, this information can be used to qualify names automatically. In that way, *George W. Bush* will be recognized as being the *American president*, *Rafik Hariri* as being a *former Lebanese prime minister*, etc.

For each added language, native speakers translate the existing pattern lists and use the same bootstrapping procedure to complete the patterns. As of September 2007, NewsExplorer performs named entity recognition in 19 languages. The stored associated titles for 15 of the 19 languages are updated daily. This NER approach has been designed with the requirement in mind that the development of resources for new languages should be quick and easy. Steinberger et al. (Forthcoming) describe how this can be achieved by using mostly language-independent rules with language-specific resource files, and how the latter can be produced with bootstrapping methods requiring minimal effort.

3.3.3 Enriching the Name Database Using Wikipedia

For known names, the system periodically checks whether a corresponding entry exists in the Wikipedia online encyclopedia.[3] If it exists, the URL of the Wikipedia page will be added to the NewsExplorer page to provide readers with more information on the person, if required. Additionally, the system downloads the picture for the person or organization and checks whether Wikipedia lists name variants not yet known to the NewsExplorer database. Additional variants found are then added to the knowledge base. This is particularly useful for name variants in foreign scripts (Arabic, Russian, Chinese, etc.).

3.4 Lookup of Known Names and Their Morphological Variants

This section describes the method used to recognize *known* names (names already in the database), their inflections, and common spelling variants in text.

3.4.1 Dealing with Inflection

Both the context rules to recognize new names (described in section 3.3) and the lookup of known names (see section 3.4.2) rely on exact matches between the word forms found in the text and those found in the name lists and in the rules. Inflected names are thus a major problem for NER and must be dealt with one way or another. For instance, in the Croatian text snippet *predsjednika Stjepana Mesića*, the Croatian president's name *Stjepan Mesić* would not be found even though his name is in the NewsExplorer name database because the name lookup procedure would fail. For more highly inflected languages such as those of the Balto-Slavonic or Finno-Ugric families, a minimum of morphological treatment is thus required. For an overview of morphological and other tricky phenomena for Balto-Slavonic languages, see Przepiórkowski (2007). As it is both costly and difficult to get hold of morphological tools for the range of NewsExplorer languages, we adopted a simplistic, but relatively reliable solution that consists of pregenerating a large number of possible inflected morphological variants for the frequent known names, and for first names. If any of these pregenerated variants is then found in text, they will be recognized and can be linked to the base form. For details, see Pouliquen et al. (2005).

The name inflections can be generated by collecting the most common name suffixes and suffix replacement rules and by then applying them to all known names of the database. While coming up with accurate rules to produce the full morphological paradigm for each name is a linguistically highly challenging task, producing lists of the most common suffixes and suffix replacement rules is much

3. `http://www.wikipedia.org/`, last visited 3/28/2007.

less difficult. The reasons are twofold: first, overgenerating inflection forms is not a problem because the overgenerated nonexisting word forms would simply not be found in text; second, the suffixes and rules can be collected empirically and semiautomatically by searching text collections of the new language for (uppercase) strings whose beginning is the same as the names in the name database.

The name inflection rules are encoded as a set of regular expressions generating different endings. As it is not always possible to determine the gender of a person by looking at the name (e.g., *Andrea* is a male first name in Italian but a female first name in German), the rules always generate both masculine and feminine name inflections. In Slovene, for example, we allow the following suffixes for each name: *-e, -a, -o, -u, -om, -em, -m, -ju, -jem, -ja*. If the name already ends with "-a", "-e" or "-o", the final vowel is optional (allowing the form *Pierrom Gemayelom* for *Pierre Gemayel*). Even without knowing Slovene, it is possible to produce lists of possible suffixes on the basis of these forms, but native speaker competence will, of course, improve the results a lot. It goes without saying that there will always be cases that cannot be dealt with by using this method. Cases of infix change or vowel harmonization, for instance, will remain unsolved. Obviously, there is always a small risk that an overgenerated form matches a real word of that language, but we are not aware of any such cases having occurred.

While the empirical and admittedly crude method described here will not be sufficient for many linguistic purposes, the performance is perfectly acceptable for NewsExplorer's lookup application, due to the inherent redundancy in the large data volume of the news analysis system.

3.4.2 Lookup Procedure

The NewsExplorer database currently contains about 630,000 names plus about 135,000 name variants (status September 2007); however, less than a quarter of these are found repeatedly. We feed a FLEX finite state automaton (Paxson, 1995) with the 50,000 most frequent known names (found in at least five different news clusters) and their additional 80,000 variants. Our lookup process allows the usage of character-level regular expressions and it allows finding proper names in languages that do not use spaces between words, like Chinese. Additionally, we generate regular expressions so that frequent spelling variants will be recognized, as well.

These include:

1. Hyphen/space alternations: For hyphenated names such as *Marie-Odile* or *Ahmad al-Fahd al-Sabah*, we generate alternation patterns (`Marie[\ \-]Odile`).

2. Diacritic alternations: Words with diacritics are frequently also found without their diacritics, e.g., *Émile Lahoud* may be found as *Emile Lahoud*; we thus generate the pattern `(É|E)mile[\ ]Lahoud`.

3. Declensions of names: We pregenerate morphological variants for all known names in the languages that decline person names, as described in section 3.4.1. In Slovene, for example, we can find the following declensions of the name *Javier*

Solana: *Javierja Solane, Javiera Solane, Javierom Solano, Javierjem Solano, Javierja Solano.*

In the Slovene automaton, the regular expression generated for the name *José Van-Dúnem* is then

```
Jos(é|e)?(e|a|o|u|om|em|m|ju|jem|ja)?[\]
Van[\ \-]D(ú|u)nem(e|a|o|u|om|em|m|ju|jem|ja)?
```

The live webpage `http://langtech.jrc.it/entities/` shows the most frequently found names plus all their name variants successfully looked up since last midnight CET. Figure 3.1 shows how large a variety of names and their declensions can be found within a few hours.

3.5 Evaluation of Person Name Recognition

Most existing systems aim at optimizing NER for a given language, to recognize every mention of the name, and to categorize the name into the given classes person, organization, location, etc. The focus of our own work, however, is rather different, as mentioned in section 3.2.2. In early 2005, we nevertheless carried out a standard evaluation of the performance of our person name recognition tool for various languages, which yielded the *Precision*, *Recall*, and *F-measure* results shown in table 3.2. Each test set consisted of a random selection of about 100 newspaper articles, for which experts listed all person names that were present in the text. For each article, we then compared if the automatically recognized person names were also selected by the expert (to get *Precision*), and if all the manually extracted names were also automatically found (to get *Recall*). These two values were combined using the F1-measure.

The results are clearly less good than for the best monolingual NER systems that use part-of-speech taggers. The *Precision* is nevertheless reasonably high. In the NewsExplorer setting, where names need to be detected in redundant *clusters* of news instead of in individual articles, the lower *Recall* is not a big issue because names are usually found in at least one of the articles so that the person information for the cluster is mostly complete.

The low *Recall* score could be due to the nature of our heterogeneous test set: The set not only includes articles from many different domains (politics, sports results, discussions of television programs, etc.) but also from international newspapers from all over the world (this is particularly true for the English-language texts). The system has to analyze articles containing text such as: *Phe Naimahawan, of Chiang Mai's Mae Ai district, has been selected (...) to represent Thailand in a swimming event (...). Phe is being helped by Wanthanee Rungruangspakul, a law lecturer.* Without part-of-speech tagging, it is difficult to guess that *Phe Naimahawan* is a person name. However, in the same text, we were able to guess the name *Wanthanee Rungruangspakul* thanks to the trigger word *law lecturer*.

Figure 3.1 Screenshot from the publicly accessible live site `http://langtech.jrc.it/entities/` showing the person names and their variants plus the text snippets in which the name was found since midnight CET. The example of *Vladimir Putin* shows that – even in such a short time period – a large number of spelling variants and morphological variants can be found. The screenshot shows text snippets with different orthographies in French, Greek, Bulgarian, Polish, and Arabic.

Table 3.2 Evaluation of person name recognition in various languages. The number of rules (i.e., trigger words) gives an idea of the expected coverage for this language. The third and fourth columns show the size of the test set (number of texts, number of manually identified person names).

Language	*#rules*	*#texts*	*#names*	*Precision*	*Recall*	*F-measure*
English	1100	100	405	92	84	88
French	1050	103	329	96	95	95
German	2400	100	327	90	96	93
Spanish	580	94	274	85	84	84
Italian	440	100	298	92	90	91
Russian	447	61	157	81	69	74

The lower *Precision* for German was predictable as in German every noun is uppercased, which sometimes results in the system recognizing common nouns as proper names. In the example: *Die österreichische Eishockey Nationalmannschaft bekommt während der Heim-WM noch Verstärkung*, the word combination *Eishockey Nationalmannschaft (ice hockey national team)* was wrongly identified as being a person name due to the trigger word *österreichische* (Austrian).

Errors in Spanish were due to various facts. One of them was that we did not have any Basque first names in our name lists and that many Basque names were found in the test set (e.g., *Gorka Aztiria Echeverría*). Another reason was that our system frequently only recognized part of the typically Spanish compound names (e.g., *Elías Antonio Saca*, where the process recognized only *Antonio Saca*).

Finally, several organization names were wrongly classified by the algorithm as person names.

The explanation for the Russian results mainly is that, at the time, our name database contained only a dozen Russian names so that the system had to guess most names, which is harder than looking up known names. Since the evaluation was carried out, our NER system has been improved, but no formal evaluation has been carried out since these changes have been made.

3.6 Identification and Merging of Name Variants

The name variants displayed on the person pages in NewsExplorer are not generated, but they were all collected from real live news texts. Once a day, NewsExplorer compares each new name found to all the other names in the database. If the new name is found to be a variant of an existing name, it is automatically merged with the existing name. Otherwise, it is added to the database with a new numerical identifier. This section explains how variants belonging to the same name are identified. More specifically, section 3.6.1 explains the transliteration step for names in languages not written with the Roman script. Section 3.6.2 describes the language-independent name normalization rules, and section 3.6.3 finally describes how we identify which name variants should be merged to the same name.

3.6.1 Transliteration of Names Not Written with the Roman Alphabet

The common approach to identifying name equivalents across writing systems is to transliterate letters or groups of letters from one writing system (e.g., Cyrillic) into the other (e.g., Roman script), using either handwritten or automatically learned equivalence rules (see also section 3.2.3) and to then compare the results. As the phonetic systems of different languages differ and not every foreign language sound can be represented accurately in the other language, transliteration and backtransliteration frequently result in different spellings. For instance, the French name *Valery Giscard d'Estaing* is transliterated into Russian as *Валери Жискар д'Эстен* and the result of backtransliteration is *Valeri Jiskar d'Esten*. Klementiev and Roth (2006b) report that Russian to English transliteration was successful in only about 38% of cases, even though they had avoided the additional problem of name inflections by stemming the names.

In our own approach, we combined standard, handwritten transliteration rules with the subsequent normalization step described in section 3.6.2. In the case of the former French president, this results in *valeri giskar desten*, which is rather different from its original spelling (*Valery Giscard d'Estaing*), but similar enough to the normalized form of the original name *valeri giskard destaing* for the fuzzy matching techniques (see section 3.6.3) to be successful.

In the case of Arabic, an additional step needs to be carried out because in Arabic short vowels are usually not written. For this reason, we also delete the vowels from

Table 3.3 A selection of name normalization rules with examples

accented character ⇒ nonaccented equivalent	Éamon Ó Cuív ⇒ Eamon O Cuiv
y ⇒ i	Serguey Kislyak ⇒ Serguei Kisliak
double consonant ⇒ single consonant	Sybille Bammer ⇒ Sibile Bamer
ou ⇒ u	Malik Saidoulaiev ⇒ Malik Saidulaiev
wl (beginning of name) ⇒ vl	Wladimir Putin ⇒ Vladimir Putin
ow (end of name) ⇒ ov	Michail Fradkow ⇒ Michail Fradkov
ck ⇒ k	Franck Lampard ⇒ Frank Lampard
x ⇒ ks	Alexei Kudrin ⇒ Aleksei Kudrin
al- ⇒ {empty}	Rafik al-Hariri ⇒ Rafik Hariri

the name written in Roman script before applying the fuzzy matching techniques. We refer to this normalized and vowelless name representation as the *consonant signature*. The name *Condoleezza Rice*, normalized to *kondoleza rice*, will thus have the consonant signature (*kndlz rc*), which is more similar to the Arabic consonant signature (after transliteration and the removal of the remaining long vowels) *kndlz rs*: the edit similarity between these two variants is 0.875 and thus rather high. For an evaluation on small Arabic, Farsi, and Russian test sets, see Pouliquen et al. (2005).

3.6.2 "Normalization" of Name Variants

Comparing each of the approximately 450 new names found every day with all existing 630,000 names in the database (plus their 135,000 variants) would be computationally rather heavy, independently of the approximate matching algorithm used. When using edit distance, the overall process would have a complexity of $O(N \cdot M \cdot n \cdot m \cdot a)$, with N being the number of new names, M the number of existing names in the database, n the average length of new names, m the average length of existing names, and a the average number of aliases per name. Additionally, certain regularities for at least some of the name variants can be observed. For instance: the Russian name suffix "ov" ("ов") is frequently transliterated either as "ov" (e.g. in English or French) or as "ow" (in German); diacritics on names are frequently omitted in English and other languages (*François* and *Wałęsa* are often found as *Francois* and *Walesa*); the English letter combination "sh" and the German "sch" are usually transliterated into Slovene as "š," etc.

We exploit these empirically observed regularities by writing normalization rules that merge the observed variants to the same *normalized form* (see table 3.3 for rule examples). This normalized form does not claim to represent any linguistic reality, but is purely motivated by pragmatic needs. We observed that most of the variations concern vowels (especially when the name was transliterated from Arabic). Therefore we also remove vowels from the normalized form in order to

Table 3.4 Examples of normalized forms and consonant signatures

Name	*Normalised form*	*Consonant signature*
Mohammed Siad Barre, Mohamed Siad Barré, Мохаммед Сиад Барре, محمد سياد بري	mohamed siad bare	**mhmd sd br**
Mahmoud Ahmadinejad, Mahmūd Ahmadīnežād	mahmud ahmadinejad	**mhmd hmdnjd**
Сергей Куприянов, Sergei Kupriyanov, Sergei Kuprianow, Sergueï Kouprianov	sergei kuprianov	**srg kprnv**
Ban Ki-moon, Ban Ki Moon, Пан Ги Мун	ban ki mun	**bn k mn**

compute the consonant signature. For new names written with the Roman alphabet, all names are first normalized, then reduced to consonants and then compared to the prestored consonant signatures of the names in the database. Table 3.4 shows examples for normalized forms and consonant signatures. This comparison is, of course, rather fast and efficient.

3.6.3 Approximate Matching of (Normalized) Name Variants

We automatically compare each of the approximately 450 newly identified names per day with all other known names from the database. Due to the large volume of names, we apply fuzzy matching rules only to an automatic preselection of new names. This preselection is done by comparing the consonant signature of the new names with the pre-stored consonant signatures of all known names and their variants. Only if there is an exact match, are the following fuzzy matching rules applied. Otherwise, the new names are entered with a new identifier into the database.

The similarity calculation between the preselected new name and the similar known name(s) is based on edit distance (Zobel and Dart, 1995). This measure is applied twice, each time to a different representation of the same name pair: First it is applied to the normalized form (*with* vowels) and then to the lowercased nonnormalized name, as it was found in the text (after transliteration, if applicable). The first similarity has a relative weight of 0.8, the second of 0.2. If the combined similarity value is above the threshold of 0.94, the candidates are automatically merged. Otherwise the new name is entered into the name database with a new identifier. Note that the newly identified name is merged with existing names if it is similar enough to *any* of the stored name variants. The threshold of 0.94 was

Table 3.5 Pairs of candidates for the automatic merging, and their combined fuzzy matching similarity values. When the value is 0.94 or higher, the two variants are automatically merged. The last column shows the human judgment on the merging decision.

Name 1	Name 2	Similarity	Merged?	Same person?
Alexander Lukashenko	Aleksander Lukaşenko	0.99	Yes	Yes
Yuri Boyko	Юрий Бойко	0.98	Yes	Yes
Abdullah bin Abdul Aziz	Abdullah bin Abdel Aziz	0.96	Yes	Yes
Barzan al-Tikriti	Barazan al-Takriti	0.94	Yes	Yes
Ammar al-Houri	عمار حوري	0.94	Yes	Yes
Michael Sprung	Michael Spreng	0.93	No	No
Gerhard Schröder	Gerhard Schruder	0.93	No	Yes (typo)
Jorge Costa	Jorge Acosta	0.92	No	No
Paris Hilton	Парис Хилтън	0.92	No	Yes
Georges Sarre	Georges Serre	0.91	No	No
Raffaele Bucca	Rafael Beca	0.82	No	No
Ursula Lehr	Ursula Lher	0.81	No	Yes (typo)

determined empirically by looking at large numbers of merger candidates. In this test set, all name variant candidates with a similarity above 0.94 were indeed good candidates (i.e., the precision was 100%). By lowering the threshold, some more good candidates could have been found, but we would merge some name variants that do not belong to the same person. The focus of the merging is thus on high precision rather than on good recall. Table 3.5 shows a few name merger candidates and their combined similarity values. The shown example set includes candidates above the threshold (automatically merged) and below (kept separate).

In previous work (Pouliquen et al., 2005), we used the cosine vector space similarity measure (representing proper names as vectors of bigrams and trigrams of characters) but observed that edit distance is more suitable for proper name comparison. Vector space-based similarity measures have the advantage that they allow for an inversion of the sequence of characters, but have the disadvantage that they give more vague results when names contain common substrings but at different places. In our case, it is quite unlikely that two names have a completely different order of characters. In newspaper articles, full names are almost always written with the first name followed by the family name. Hungarian is a notable exception, as it usually mentions the family name first for local names (Kálmán, 1978), while using the inverse order for foreign names. Specific rules will have to be written to tackle this problem.

3.7 Conclusion and Future Work

In the past, dictionaries were developed for general or subject-specific vocabularies, but not for proper names. However, name dictionaries, which may include cross-lingual, cross-script, and also monolingual name variants, are a precious resource that can help improve the output of many text analysis applications. These include machine translation, information retrieval, topic tracking, relation and event extraction, the automatic generation of social networks based on information found in free text, and more. While work on automatically extracting information to feed name dictionaries is still rather scarce, many scientists now work on automatically learning transliteration rules and name equivalences from bilingual name lists, especially for Arabic and Russian.

We have presented work on recognizing new names in multilingual news collections in 19 languages and on an automatic procedure to determine whether any new name is likely to be a variant of a known name or whether it is a name in its own right. For that purpose, each name is normalized – using language pair-independent rules – and then compared to each of the known names in the database using a combination of two similarity measures. The language-independence of the rules is of particular importance because names found in news texts can come from any country and could be pronounced according to the pronunciation rules of any language on the globe.

The presented method is applied daily in the multilingual news aggregation and analysis application *NewsExplorer*. It identifies an average of 400 new names and of 50 new name variants per day.

Work we would like to carry out in the future is to exploit the multilingual name database, which currently contains about 630,000 names plus an additional 135,000 name variants. Similarly to the work Mulloni (2007) carried out on predicting cognate variants across languages, we would like to learn typical name transliteration and equivalence rules. For instance, the database knowledge could be used to learn Latvian transliteration rules (*Džordžs V. Bušs* is the Latvian equivalent to *George W. Bush*) or typical patterns for the transliteration of a Russian name like *Устинов* to English *Ustinov*, French *Oustinov*, or German *Ustinow*.

Acknowledgments

We are grateful to our colleagues from the *Web Mining and Intelligence* team for providing us with the precious multilingual news data. We are particularly grateful to the team leaders Clive Best and Erik van der Goot, as well as to Flavio Fuart, who produced the attractive and robust web interfaces that show our results.

4 Named Entity Transliteration and Discovery in Multilingual Corpora

Alexandre Klementiev
Dan Roth

Named entity recognition (NER) is an important part of many natural language processing tasks. Current approaches often employ machine learning techniques and require supervised data. However, many languages lack such resources. This chapter[1] presents an (almost) unsupervised learning algorithm for automatic discovery of named entities (NEs) in a resource-free language, given bilingual corpora in which it is weakly temporally aligned with a resource-rich language. NEs have similar time distributions across such corpora, and often some of the tokens in a multiword NE are transliterated. We develop an algorithm that exploits both observations iteratively. The algorithm makes use of a new, frequency-based, metric for time distributions and a resource-free discriminative approach to transliteration. Seeded with a small number of transliteration pairs, our algorithm discovers multiword NEs, and takes advantage of a dictionary (if one exists) to account for translated or partially translated NEs. We evaluate the algorithm on an English-Russian corpus, and show a high level of NEs discovery in Russian.

4.1 Introduction

Named entity recognition has received significant attention in natural language processing (NLP) research in recent years since it is regarded as a significant component of higher-level NLP tasks such as information distillation and question answering. Most modern approaches to NER employ machine learning techniques, which require supervised training data. However, for many languages, these resources do not exist. Moreover, it is often difficult to find linguistic expertise either for annotating data or specifying domain knowledge. On the other hand, comparable multilingual

1. This chapter unifies and extends work from Klementiev and Roth (2006a,b).

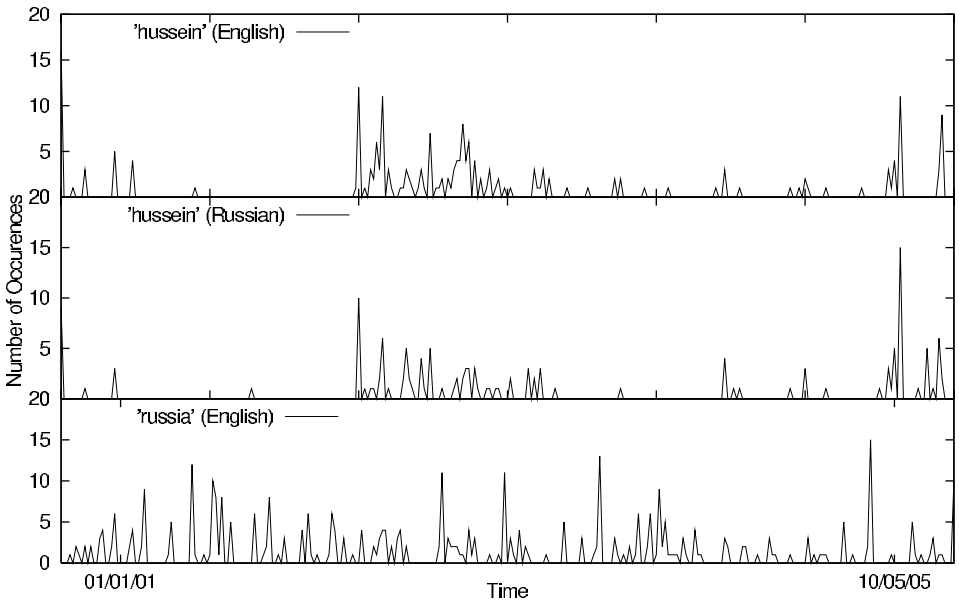

Figure 4.1 Temporal histograms for *Hussein* (*top*), its Russian transliteration (*middle*), and of the word *Russia* (*bottom*).

data (such as multilingual news streams) is becoming increasingly available (section 4.4). Unique properties of such corpora may allow us to transfer annotation across to resource-poor domains, relieving the supervision bottleneck.

In this chapter, we make two independent observations about named entities encountered in such corpora, and use them to develop an algorithm that extracts pairs of NEs across languages. Specifically, given a bilingual corpus that is weakly temporally aligned, and a capability to annotate the text in one of the languages with NEs, our algorithm identifies the corresponding NEs in the second language text, and annotates them with the appropriate type, as in the source text.

The first observation is that NEs in one language in such corpora tend to co-occur with their counterparts in the other. E.g., figure 4.1 shows a histogram of the number of occurrences of the word *Hussein* and its Russian transliteration in our bilingual news corpus spanning the years 2001 through late 2005. One can see several common peaks in the two histograms, the largest one being around the time of the beginning of the war in Iraq. The word *Russia*, on the other hand, has a distinctly different temporal signature. We can exploit such weak synchronicity of NEs across languages to associate them. In order to score a pair of entities across languages, we compute the similarity of their time distributions.

The second observation is that NEs often contain or are entirely made up of words that are phonetically transliterated or have a common etymological origin across languages (e.g., *parliament* in English and *парламент*, its Russian translation),

Table 4.1 Examples of English NEs and their transliterated Russian counterparts.

English NE	Russian NE
lilic	лилич
fletcher	флетчер
bradford	брэдфорд
isabel	изабель
hoffmann	гофман
kathmandu	катманду

and thus are phonetically similar. Table 4.1 shows an example list of NEs and their possible Russian transliterations.

Approaches that attempt to use these two characteristics separately to identify NEs across languages would have significant shortcomings. Transliteration-based approaches require a good model, typically handcrafted or trained on a clean set of transliteration pairs. On the other hand, time sequence similarity–based approaches would incorrectly match words which happen to have similar time signatures (e.g., *Taliban* and *Afghanistan* in recent news).

We introduce an algorithm called *co-ranking*, which exploits these observations simultaneously to match NEs on one side of the bilingual corpus to their counterparts on the other.

We first train a transliteration model on single-word NEs. During training, for a given NE in one language, the current model chooses a list of top-ranked transliteration candidates in another language. A metric based on the discrete Fourier transform (Arfken, 1985) is then used (section 4.3.1) to rerank the list and choose the candidate best temporally aligned with the given NE. Finally, pairs of source language NEs and the top candidates from the reranked candidate lists are used for the next iteration of the transliteration model training.

Once the model is trained, NE discovery proceeds as follows. For a given NE, the transliteration model selects a candidate list for each constituent word. If a dictionary is available, each such candidate list is augmented with translations (if they exist). Translations will be the correct choice for some NE words (e.g., for *queen* in *Queen Victoria*), and transliterations for others (e.g., *Bush* in *Steven Bush*). We expect temporal sequence alignment to resolve many of such ambiguities. Temporal alignment score is used to rerank translation/transliteration candidate lists for each constituent word. The top candidates from each re-ranked list are then merged into a possible target language NE. Finally, we verify that the candidate NE actually occurs in the target corpus.

A major challenge inherent in discovering transliterated NEs is the fact that a single entity may be represented by multiple transliteration strings. One reason is language morphology. For example, in Russian, depending on the case being used, the same noun may appear with various endings. Another reason is the lack of transliteration standards. Again, in Russian, several possible transliterations of an

English entity may be acceptable, as long as they are phonetically similar to the source.

Thus, in order to rely on the time sequences we obtain, we need to be able to group variants of the same NE into an equivalence class, and collect their aggregate mention counts. We would then score time sequences of these equivalence classes. For instance, we would like to count the aggregate number of occurrences of {*Herzegovina, Hercegovina*} on the English side in order to map it accurately to the equivalence class of that NE's variants we may see on the Russian side of our corpus (e.g., {*Герцеговина, Герцеговину, Герцеговины, Герцеговиной*}). In the rest of the chapter, whenever we refer to a named entity or an NE constituent word, we imply its equivalence class.

One of the objectives of this work was to use as little of the knowledge of both languages as possible. In order to effectively rely on the quality of time sequence scoring, we used a simple, knowledge-poor approach to group NE variants for the languages of our corpus (section 4.3.1). Although we expect that better use of language-specific knowledge would improve the results, it would defeat one of the goals of this work.

A demo of this work, as well as the software and the data used in the experiments is available at `http://L2R.cs.uiuc.edu/~cogcomp/`

4.2 Previous Work

There has been other work on discovering NEs automatically with minimal supervision. Both Cucerzan and Yarowsky (1999), and Collins and Singer (1999) present algorithms to obtain NEs from untagged corpora. However, they focus on the *classification* stage of already segmented entities, and make use of contextual and morphological clues that require knowledge of the language beyond the level we want to assume with respect to the target language.

The use of similarity of time distributions for information extraction, in general, and NE extraction, in particular, is not new. Hetland (2004) surveys recent methods for scoring time sequences for similarity. Shinyama and Sekine (2004) used the idea to discover NEs, but in a single language, English, across two news sources. Moreover, we use a different temporal distribution similarity function and show it to be better in section 4.4.3.

A large amount of previous work exists on transliteration models. Most are *generative* and consider the task of *producing* an appropriate transliteration for a given word, and thus require considerable knowledge of the languages. For example, AbdulJaleel and Larkey (2003) and Jung et al. (2000) train English-Arabic and English-Korean generative transliteration models, respectively. Knight and Graehl (1997) build a generative model for backward transliteration from Japanese to English. Sproat et al. (2006) produce transliterations by combining the scores of temporal and phonetic transliteration models, whereas we also propose a method to train a transliteration model with little supervision.

While generative models are often robust, they tend to make independence assumptions that do not hold in data. The discriminative learning framework advocated by Roth (1998, 1999) as an alternative to generative models is now used widely in NLP, even in the context of word alignment (Taskar et al., 2006; Moore, 2005). We make use of it here too, to learn a discriminative transliteration model that requires little knowledge of the target language.

Bilingual lexicon extraction from non parallel corpora (e.g., Rapp, 1995; Koehn and Knight, 2002; Déjean et al., 2002) is the line of research most related to our work. Focusing on named entities, however, allows us to exploit properties specific to them (transliteration and temporal alignment). Furthermore, NEs hold particular significance for NLP tasks such as information extraction.

4.3 *Co-Ranking*: An Algorithm for NE Discovery

In essence, the algorithm we present (figure 4.2) uses temporal alignment as a supervision signal to iteratively train a transliteration model $\mathcal{M}$. On each iteration, for each NE in the source language corpus $\mathcal{S}$ it selects a list of top-ranked transliteration candidates from the target language corpus $\mathcal{T}$ according to the current model (line 6). It then uses temporal alignment (with thresholding) to rerank the list and select the best transliteration candidate for the next round of training (lines 8 and 10).

Similarly, in testing or discovery (figure 4.3), candidate lists are collected (line 6) for each constituent word of each source NE using the trained model $\mathcal{M}$. Optionally, the lists $\mathcal{NE}^i_{\mathcal{T}}$ are augmented with the dictionary translations of the respective source word (line 7). The lists are then reranked without thresholding (line 9), and collected into a multiword target NE candidate $\mathcal{E}_{\mathcal{T}}$. Finally, we discard $\mathcal{E}_{\mathcal{T}}$ which do not actually occur (in any order of the constituent words) in target corpus $\mathcal{T}$.

4.3.1 Time Sequence Generation and Matching

In order to generate a time sequence for a given word, we sort the set of (time-stamped) documents of our corpus into a sequence of equally sized temporal bins. We then count the number of occurrences of the word in each bin, and normalize the sequence.

We use a method called the F-index (Hetland, 2004) to implement the *score* similarity function (figure 4.2, line 8, and figure 4.3, line 9). We first run a discrete Fourier transform (DFT) on a time sequence to extract its Fourier expansion coefficients. The score of a pair of time sequences is then computed as a Euclidean distance between their expansion coefficient vectors. We compare this approach to two commonly used alternative metrics in section 4.4.3.

As we mentioned in the introduction, an NE may map to more than one transliteration in another language. Identification of the entity's equivalence class of transliterations is important for accurately obtaining its time sequence.

Algorithm *Co-ranking [training]*
Input: Bilingual corpus $(\mathcal{S}, \mathcal{T})$, set of named entities $\mathcal{NE_S}$ from $\mathcal{S}$, threshold θ
Output: Transliteration model $\mathcal{M}$

Initialize $\mathcal{M}$.
$\forall \mathcal{E} \in \mathcal{NE_S}$, collect time distribution $\mathcal{Q_{ES}}$.
repeat
    $\mathcal{D} \leftarrow \emptyset$.
    for each $\mathcal{E_S} \in \mathcal{NE_S}$
        Use $\mathcal{M}$ to collect candidates $\mathcal{NE_T} \in \mathcal{T}$ with high translit. scores.
        Collect time distribution $\mathcal{Q_{ET}}$ for each candidate in $\mathcal{NE_T}$.
        Select candidate $\mathcal{E_T} \in \mathcal{NE_T}$ with the best $\omega = score(\mathcal{Q_{ES}}, \mathcal{Q_{ET}})$.
        if $\omega > \theta$
            $\mathcal{D} \leftarrow \mathcal{D} \bigcup \{(\mathcal{E_S}, \mathcal{E_T})\}$.
    Use $\mathcal{D}$ to train $\mathcal{M}$
until Discovered training set $\mathcal{D}$ no longer changes between iterations.
return $\mathcal{M}$.

Figure 4.2 Iterative transliteration model training with single-word NEs.

Algorithm *Co-ranking [testing]*
Input: Bilingual corpus $(\mathcal{S}, \mathcal{T})$, set of named entities $\mathcal{NE_S}$ from $\mathcal{S}$, transliteration model $\mathcal{M}$, dictionary *dict* (otional)
Output: Set of NE pairs $\mathcal{D}$ from $\mathcal{S}$ and $\mathcal{T}$

$\mathcal{D} \leftarrow \emptyset$.
for each $\mathcal{E_S} \in \mathcal{NE_S}$
    $\mathcal{E_T} \leftarrow ()$.
    for each constituent word $\mathcal{E}^i_\mathcal{S}$ in $\mathcal{E_S}$
        Collect time distribution $\mathcal{Q}^i_\mathcal{ES}$ for $\mathcal{E}^i_\mathcal{S}$.
        Use $\mathcal{M}$ to collect candidates $\mathcal{NE}^i_\mathcal{T} \in \mathcal{T}$ with high translit. scores.
        (optional) $\mathcal{NE}^i_\mathcal{T} \leftarrow \mathcal{NE}^i_\mathcal{T} \bigcup dict(\mathcal{E}^i_\mathcal{S})$.
        Collect time distribution $\mathcal{Q}^i_\mathcal{ET}$ for each candidate in $\mathcal{NE}^i_\mathcal{T}$.
        Select candidate $\mathcal{E}^i_\mathcal{T} \in \mathcal{NE}^i_\mathcal{T}$ with the best $\omega = score(\mathcal{Q}^i_\mathcal{ES}, \mathcal{Q}^i_\mathcal{ET})$.
        $\mathcal{E_T} \leftarrow \mathcal{E_T} + \mathcal{E}^i_\mathcal{T}$.
    if $Occurs(\mathcal{E_T})$
        $\mathcal{D} \leftarrow \mathcal{D} \bigcup \{(\mathcal{E_S}, \mathcal{E_T})\}$.
return $\mathcal{D}$.

Figure 4.3 Testing phase.

In order to keep to our objective of requiring as little language knowledge as possible, we took a rather simplistic approach for both languages of our corpus. For Russian, two words were considered variants of the same NE if they share a prefix of size five or longer. Each unique word had its own equivalence class for the English side of the corpus, although, in principle, more sophisticated approaches to group entity mentions (such as in Li et al., 2004) could be incorporated. A cumulative distribution was then collected for such equivalence classes.

4.3.2 Transliteration Model

Unlike most of the previous work considering *generative* transliteration models, we take the *discriminative* approach. Indeed, we do not need to generate transliterations for unseen named entities. Instead, we aim to match NEs in the source language to their counterparts present in the target language side of our corpus in order to transfer annotation.

We train a linear model to decide whether a word $\mathcal{E}_\mathcal{T} \in \mathcal{T}$ is a transliteration of an NE $\mathcal{E}_\mathcal{S} \in \mathcal{S}$. The words in the pair are partitioned into a set of substrings $s_\mathcal{S}$ and $s_\mathcal{T}$ up to a particular length (including the empty string _). Couplings of the substrings $(s_\mathcal{S}, s_\mathcal{T})$ from both sets produce features we use for training. Note that couplings with the empty string represent insertions/omissions.

Consider the following example: $(\mathcal{E}_\mathcal{S}, \mathcal{E}_\mathcal{T})$ = (powell, пауэлл). We build a feature vector from this example in the following manner:

1. We split both words into all possible substrings of up to size two:
   - $\mathcal{E}_\mathcal{S} \rightarrow \{\_, p, o, w, e, l, l, po, ow, we, el, ll\}$
   - $\mathcal{E}_\mathcal{T} \rightarrow \{\_, n,\ a,\ y,\ э,\ л,\ л,\ na,\ ay,\ yэ,\ эл,\ лл\}$
2. We then build a feature vector by coupling substrings from the two sets:
   - $((p, \_), (p, a), ...(w, yэ), ...(el, эл), ...(ll, лл))$

We use the observation that transliteration tends to preserve phonetic sequence to limit the number of couplings. For example, we can disallow the coupling of substrings whose starting positions are too far apart: thus, we might not consider a pairing $(po, yэ)$ in the above example. In our experiments, we paired substrings if their positions in their respective words differed by -1, 0, or 1.

We use the perceptron (Rosenblatt, 1958) algorithm to train the model. The model activation provides the score we use to select best transliterations on line 6. Our version of perceptron takes a variable number of features in its examples; each example is a subset of all features seen so far that are active in the input. As the iterative algorithm observes more data, it discovers and makes use of more features. This model is called the *infinite attribute model* (Blum, 1992) and it follows the perceptron version of SNoW (Carlson et al., 1999).

Positive examples used for iterative training are pairs of NEs and their best temporally aligned transliteration candidates. Alignment score thresholding is used to implement the tradeoff between the quality and the number of the positive examples selected for the next round. Negative examples are English non-NEs paired with random Russian words.

4.4 Experimental Study

We ran experiments using a bilingual comparable English-Russian news corpus[2] we built by crawling a Russian news website (`www.lenta.ru`). The site provides loose translations of (and pointers to) the original English texts. We collected pairs of articles spanning from 1/1/2001 through 10/05/2005. Each side of the corpus consists of 2327 documents, with zero to eight documents per day; the total sizes of the English and Russian sides are roughly 940K and 380K tokens respectively. The English side was tagged with a publicly available NER system based on the SNoW learning architecture (Carlson et al., 1999), that is available on the same site. This set of English NEs was pruned by hand to remove incorrectly classified words to obtain 978 single-word NEs.

Temporal distributions were collected with bin size of one day, as described in section 4.3.1. In order to reduce running time, some limited preprocessing was done on the Russian side. In particular, all equivalence classes, whose temporal distributions were close to uniform (i.e., words with a similar likelihood of occurrence throughout the corpus) were deemed common and not considered as NE candidates. Unique words were thus grouped into 14,781 equivalence classes.

Unless mentioned otherwise, the transliteration model was initialized with a set of 20 pairs of English NEs and their Russian transliterations. Negative examples here and during the rest of the training were pairs of non-NE English and Russian words selected uniformly randomly from the respective corpora.

As the transliteration model improves throughout training, new examples and thus new features are discovered. All but the top 3000 features from positive and 3000 from negative examples were pruned based on the number of their occurrences so far. Features remaining at the end of training were used for NE discovery. Insertions/omissions features (section 4.3.2) were not used in the experiments as they provided no tangible benefit for the languages of our corpus.

In each iteration, we used the current transliteration model to find a list of the 30 best transliteration equivalence classes for each NE. We then computed the time sequence similarity score between an NE and each class from its list to find the one with the best matching time sequence. If its similarity score surpassed a set threshold, it was added to the list of positive examples for the next round of training. Positive examples were constructed by pairing an NE with the common stem of its transliteration equivalence class. We used the same number of positive and negative examples.

We used the Mueller English-Russian dictionary to obtain translations in our multiword NE experiments. Lists of transliteration candidates were augmented with up to ten dictionary translations.

For evaluation, a random subset of 727 out of the total of 978 NEs were matched to correct transliterations by a language expert (partly due to the fact that some of

2. The corpus, code, and demo are available at `http://L2R.cs.uiuc.edu/~cogcomp/`

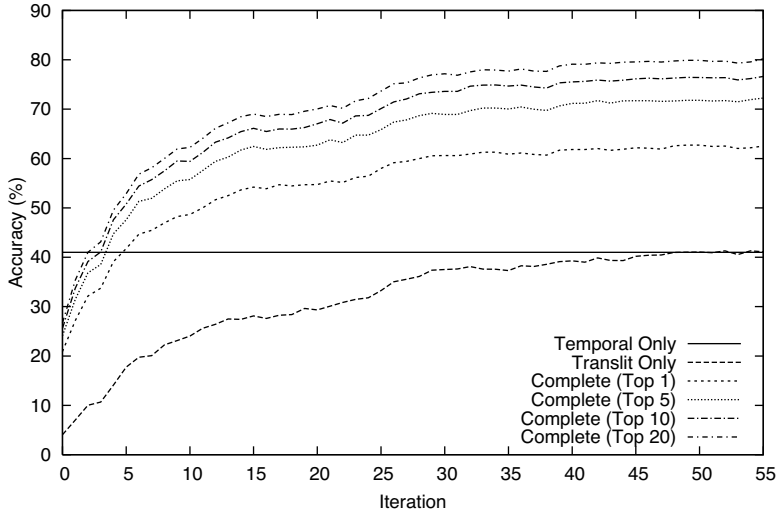

Figure 4.4 Proportion of correct NEs in the top-N discovered candidates vs. training iteration (averaged over five runs initialized with different random sets of 20 examples). The complete algorithm outperforms both the transliteration model and temporal sequence matching when used on their own.

the English NEs were not mentioned in the Russian side of the corpus). Accuracy was computed as the percentage of NEs correctly identified by the algorithm. Note that although multiple correct Russian transliterations are possible for a given English NE, the evaluation set included only a single one (due to the prohibitive amount of labor required of the language expert otherwise). Thus, evaluation results tend to be conservative.

In the multiword NE experiment, 177 random multiword (two or more words) NEs and their transliterations/translations discovered by the algorithm were verified by a language expert. Again, phrases which were incorrectly tagged as NEs by the source language NE tagger were discarded.

4.4.1 NE Discovery

Single-Word NEs

Figure 4.4 shows the proportion of correctly discovered NE transliteration equivalence classes throughout the training stage. The figure also shows the accuracy if transliterations are selected according to the current transliteration model (top-scoring candidate) and temporal sequence matching alone.

The complete algorithm experiments included counting if the correct transliteration appeared as the top-scoring candidate (*Top 1*), was present in the top 5 (*Top 5*), top 10 (*Top 10*), or top 20 (*Top 20*) candidates chosen by the algorithm.

Table 4.2 A sample of the features discovered by the algorithm during training

Feature	Num. in neg.	Num. in pos.	Percent in pos.
(a, а)	1432	6820	82.65
(n, н)	1182	5517	82.36
(r, р)	1011	5278	83.92
(gh, г)	5	137	96.48
(hm, м)	0	100	100
(tz, ц)	0	78	100
(j, х)	3	71	95.95
(j, дж)	0	198	100
(ic, ич)	11	403	97.34
(an, ан)	22	1365	98.41

Both the transliteration model and the temporal alignment alone achieve the accuracy of about 41%. The combined algorithm achieves about 63%, showing a significant improvement over either of the two methods alone. Moreover, the correct NEs appear among the top 5 candidates 72% of the time; among the top 10, 77%; and among the top 20, 80%.

Discovered Features

Table 4.2 lists a few interesting features discovered by the algorithm during training. As expected, single-letter pairs that have similar pronunciation in both languages are highly indicative of a transliteration. English two-letter sequences *gh* and *hm* correspond to a single-letter sequences in Russian, since *h* is often silent. The letter *j* is pronounced differently in names of Hispanic origin and is thus mapped to two distinct letter sequences in Russian. Some features are particularly useful for the specific training corpus. For example, the news corpus often refers to Serbian surnames ending in *ic*.

Intuition

In order to understand what happens to the transliteration model as the training proceeds, let us consider the following example. Table 4.3 shows parts of candidate transliteration lists[3] for NE *forsyth* for two iterations of the algorithm. The weak transliteration model selects the correct transliteration (italicized) as the 24th best transliteration in the first iteration. Time sequence scoring function chooses it to be one of the training examples for the next round of training of the model. By the eighth iteration, the model has improved to select it as the best transliteration.

3. Each candidate is represented by an equivalence class: a common prefix and a set of endings found in text.

Table 4.3 Lists of Russian transliteration candidates for *forsyth* for two iterations of the algorithm. As the transliteration model improves, the correct transliteration moves up the list.

	Iteration 0		**Iteration 7**
1	скоре {-е, -й, -йшего, -йший}	1	*форсайт* {*-а, -, -у*}
2	оформ {-лено, -ил, . . . }	2	оформ {-лено, -ил, -ить, . . . }
3	кокрэйн {-а, -}	3	проры {-вом, -ва, -ли, . . . }
4	флоре {-нс, -нц, -, -нции}	4	фросс
	•	5	фоссет {-т, -та, -ту, -а, -у}
	•		•
24	*форсайт* {*-а, -, -у*}		•
	•		•

Not all correct transliterations make it to the top of the candidates list (the transliteration model by itself is never as accurate as the complete algorithm in figure 4.4). That is not required, however, as the model only needs to be good enough to place the correct transliteration anywhere in the candidate list.

Not surprisingly, some of the top transliteration candidates start sounding like the NE itself, as training progresses. In table 4.3, candidates for *forsyth* on Iteration 7 include *fross* and *fossett*.

Multiword NEs

Once the transliteration model was trained, we ran the algorithm to discover multiword NEs, augmenting candidate sets of dictionary words with their translations as described in section 4.3. Of all multiword named entity pairs discovered by the algorithm, about 68% were matched correctly. The discovered Russian NEs included entirely transliterated, partially translated, and entirely translated NEs. Some of them are shown in table 4.4.

Table 4.4 Example of correct transliterations discovered by the algorithm.

English NE	**Russian NE equivalence class**
carla del ponte	карла{-, -йл} дель понте
marc dutroux	марк дютру
pangbourne	пангбурн
supreme council	верхо{-вный, ...} совет{...}
congolese	конго{-, -лезской}
north carolina	север{...} карол{-ина, ...}
junichiro koizumi	дзюнитиро коидзуми
rehman	реман{-, -а}

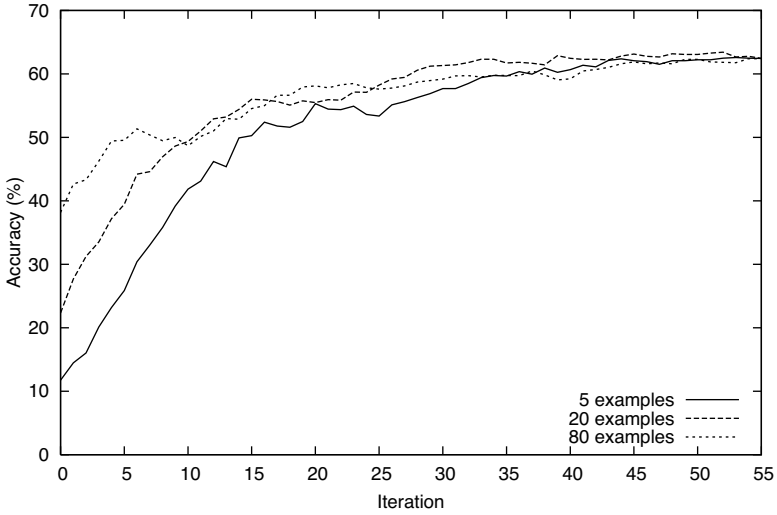

Figure 4.5 Proportion of correctly discovered NE pairs vs. the initial example set size (averaged over three runs each). Decreasing the number of examples does not have an impact on the performance of the later iterations.

4.4.2 Initial Example Set Size

We ran a series of experiments to see how the size of the initial training set affects the accuracy of the model as training progresses (figure 4.5). Although the performance of the early iterations is significantly affected by the size of the initial training example set, the algorithm quickly improves its performance. As we decrease the size from 80 to 20 and then to 5, the accuracy of the first iteration drops by over 15% and 10% respectively. However, in about 50 iterations all three perform similarly.

The few examples in the initial training set produce features corresponding to substring pairs characteristic of English-Russian transliterations. A model trained on these (few) examples chooses other transliterations containing the same substring pairs. In turn, the chosen positive examples contain other characteristic substring pairs, which will be used by the model (via the *infinite attribute domain*; Blum, 1992) to select more positive examples on the next round, and so on. The smaller the initial training set, the longer it takes to discover the characteristic features, and the longer it takes for the algorithm to converge.

One would also expect the size of the training set necessary for the algorithm to improve to depend on the level of temporal alignment of the two sides of the corpus. Indeed, the weaker the temporal supervision, the more we need to endow the model so that it can select cleaner candidates in the early iterations.

Table 4.5 Proportion of correctly discovered NEs vs. corpus misalignment (k, *left*) and vs. sliding window size (w, *right*) for each of the three measures. The DFT-based measure provides significant advantages over commonly used metrics for weakly aligned corpora.

	k=1	k=3	k=5
Cosine	41.3	5.8	1.7
Pearson	41.1	5.8	1.7
DFT	41.0	12.4	4.8

	w=1	w=2	w=3
Cosine	5.8	13.5	18.4
Pearson	5.8	13.5	18.2
DFT	12.4	20.6	27.9

4.4.3 Comparison of Time Sequence Scoring Functions

We compared the DFT-based time sequence similarity scoring function we use in this chapter to the commonly used *cosine* (Salton and McGill, 1986) and *Pearson*'s correlation measures in order to assess its performance and robustness to misalignment between two sides of the corpus.

We perturbed the Russian side of the corpus in the following way. Articles from each day were randomly moved (with uniform probability) within a k-day window. We ran single-word NE temporal sequence matching alone on the perturbed corpora using each of the three measures (table 4.5, left).

Some accuracy drop due to misalignment could be accommodated for by using a larger temporal bin for collecting occurrence counts. We tried various (sliding) window sizes w for a perturbed corpus with $k = 3$ (table 4.5, right).

The DFT metric outperforms the other measures significantly in most cases. NEs tend to have distributions with few pronounced peaks. If two such distributions are not well aligned, we expect both Pearson and cosine measures to produce low scores, whereas the DFT metric should catch their similarities in the frequency domain.

4.5 Conclusions

We have proposed a novel algorithm for cross-lingual multiword NE discovery in a bilingual weakly temporally aligned corpus. We have demonstrated that using two independent sources of information (transliteration and temporal similarity) together to guide NE extraction gives better performance than using either of them alone (figure 4.4).

The algorithm requires almost no supervision or linguistic knowledge. Indeed, we used a very small bootstrapping training set and made a simple assumption in order to group morphological variants of the same word into equivalence classes in Russian.

We also developed a linear discriminative transliteration model, and presented a method to automatically generate features. For time sequence matching, we used a scoring metric novel in this domain and provided experimental evidence that it outperforms two other metrics traditionally used.

4.6 Future Work

The algorithm can be naturally extended to comparable corpora of more than two languages. Pairwise time sequence scoring and transliteration models should give better confidence in NE matches.

The ultimate goal of this work is to automatically tag NEs so that they can be used for training of an NER system for a new language. To this end, we would like to compare the performance of an NER system trained on a corpus tagged using this approach to one trained on a handtagged corpus.

Acknowledgments

We thank Richard Sproat, ChengXiang Zhai, and Kevin Small for their useful feedback during this work, and the anonymous referees for their helpful comments. This research is supported by the NSF grant ITR IIS-0428472, the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program, and a DOI grant under the Reflex program.

5 Combination of Statistical Word Alignments Based on Multiple Preprocessing Schemes

Jakob Elming
Nizar Habash
Josep M. Crego

Word alignments over parallel corpora have become an essential supporting technology to a variety of natural language processing applications. We present an approach to using multiple preprocessing (tokenization) schemes to improve statistical word alignments. In this approach, the text to align is tokenized before statistical alignment and is then remapped to its original form afterward. Multiple tokenizations yield multiple *remappings* (remapped alignments), which are then combined using supervised machine learning. Our results show that the remapping strategy improves alignment correctness by itself. We also show that the combination of multiple remappings improves measurably over a commonly used state-of-the-art baseline. We obtain a relative reduction of alignment error rate of about 38% on a blind test set.

5.1 Introduction

Word alignments over parallel corpora have become an essential supporting technology to a variety of natural language processing (NLP) applications, most prominent among which is statistical machine translation (SMT). Although phrase-based approaches to SMT tend to be robust to word-alignment errors (Lopez and Resnik, 2006), improving word alignment is still meaningful for other NLP research that is more sensitive to alignment quality, e.g., projection of information across parallel corpora (Yarowsky et al., 2001).

In this chapter, we present a novel approach to using and combining multiple preprocessing (tokenization) schemes to improve word alignment. In our approach, the text to align is tokenized before statistical alignment and is then remapped

to its original form afterward. Multiple tokenizations yield multiple *remappings* (remapped alignments), which are then combined using supervised machine learning. The intuition here is similar to the combination of different preprocessing schemes for a morphologically rich language as part of SMT (Sadat and Habash, 2006) except that the focus is on improving the alignment quality. The language pair we work with is Arabic-English.

In the following two sections, we present related work and Arabic preprocessing schemes. Sections 5.4 and 5.5 present our approach to alignment preprocessing and combination, respectively. Alignment results are presented in section 5.6. We also present some results and analysis on the contribution of improved alignments to SMT in section 5.7.

5.2 Related Work

Recently, several successful attempts have been made at using supervised machine learning for word alignment (Liu et al., 2005; Taskar et al., 2005; Moore, 2005; Ittycheriah and Roukos, 2005; Fraser and Marcu, 2006; Cherry and Lin, 2006). This approach often makes for faster alignment and is easier to add new features to, compared to generative models. With the exception of Moore (2005) and Fraser and Marcu (2006), the above-mentioned publications do not entirely discard the generative models. Instead, they integrate IBM model predictions as features. We extend this approach by including IBM alignment information based on multiple preprocessing schemes in the alignment process.

In other related work, Tillmann et al. (1997a) use several preprocessing strategies on both source and target language to make them more alike with regard to sentence length and word order. Lee (2004) only changes the word segmentation of the morphologically complex language (Arabic) to induce morphological and syntactic symmetry between the parallel sentences.

We differ from previous work by including alignment information based on multiple preprocessing schemes in the alignment process. We do not decide on a certain scheme to make source and target sentences more symmetrical with regard to the number of tokens and their content. Instead, it is left to the alignment algorithm to decide under which circumstances to prefer alignment information based on one preprocessing scheme over information based on another scheme.

The intuition behind using different preprocessing schemes for word alignment is a simple extension of the same intuition for preprocessing parallel data for SMT: namely, that reduction of word sparsity often improves translation quality (and in our case alignment quality). This reduction can be achieved by increasing training data or via morphologically driven preprocessing (Goldwater and McClosky, 2005). Recent publications on the effect of morphology on SMT quality focused on morphologically rich languages such as German (Nießen and Ney, 2004); Spanish, Catalan, and Serbian (Popović and Ney, 2004); Czech (Goldwater and McClosky, 2005); and Arabic (Lee, 2004; Habash and Sadat, 2006). They all studied the effects

Table 5.1 Arabic preprocessing scheme variants for وسيكتبها! "and he will write it!"

Preprocessing scheme		Example	
Name	**Definition**	**Arabic script**	**Transliteration**
NONE	natural text	وسيكتبها!	*wsyktbhA!*
AR	simple tokenization	وسيكتبها !	*wsyktbhA !*
*D*1	decliticize CONJ	و+ سيكتبها !	*w+ syktbhA !*
*D*2	decliticize CONJ, PART	و+ س+ يكتبها !	*w+ s+ yktbhA !*
TB	Arabic treebank tokenization	و+ سيكتب +ها !	*w+ syktb +hA !*
*D*3	decliticize all clitics	و+ س+ يكتب +ها !	*w+ s+ yktb +hA !*

of various kinds of tokenization, lemmatization, and part-of-speech (POS) tagging and show a positive effect on SMT quality. However, to our knowledge, no study tried to tease out the effect of tokenization on word alignment. Sadat and Habash (2006) investigated the effect of combining multiple preprocessing schemes on MT quality in a phrase-based SMT system. In this chapter, we focus on alignment improvement independent of SMT.

5.3 Arabic Preprocessing Schemes

Arabic is a morphologically complex language with a large set of morphological features. As such, the set of possible preprocessing schemes is rather large (Habash and Sadat, 2006). We follow the use of the terms *preprocessing scheme* and *preprocessing technique* as used by Habash and Sadat (2006). We focus here on a subset of schemes pertaining to Arabic attachable clitics. Arabic has a set of attachable clitics to be distinguished from inflectional features such as gender, number, person, and voice. These clitics are written attached to the word and thus increase its ambiguity. We can classify three degrees of cliticization that are applicable in a strict order to a word base:

```
[CONJ+
    [PART+
        [Al+ BASE +PRON]]]
```

At the deepest level, the BASE can have a definite article +ال (*Al+*[1] "the") or a member of the class of pronominal clitics, +PRON, (e.g., +ها *+hA* "her/it/its"). Pronominal enclitics can attach to nouns (as possessives) or verbs and prepositions (as objects). Next comes the class of particles (PART+), (e.g., +ل *l+* "to/for" or

1. All Arabic transliterations are provided in the Habash-Soudi-Buckwalter transliteration scheme (Habash et al., 2007).

+س *s+* "will/future"). Most shallow is the class of conjunctions (CONJ+), (+و *w+* "and" and +ف *f+* "then").

We use the following five schemes: *AR*, *D*1, *D*2, *D*3, and *TB*. Definitions and contrastive examples of these schemes are presented in table 5.1. To create these schemes, we use MADA (the morphological analysis and disambiguation for Arabic), an off-the-shelf resource for Arabic morphological disambiguation (Habash and Rambow, 2005), and TOKAN, a general Arabic tokenizer (Habash, 2007).

5.4 Preprocessing Schemes for Alignment

5.4.1 Giza++ Alignments

The basic alignments used as baselines and by the combiner in this work are created with the Giza++ statistical aligner (Och and Ney, 2003). Giza++ is an implementation of the IBM models (Brown et al., 1993) with some extensions. The IBM models 1 to 5 use increasingly sophisticated modeling to achieve better alignments based on nonlinguistic, statistical information about word occurrences in language-parallel sentences.

A limitation of the IBM models is that they create asymmetrical alignments, i.e., they only allow one-to-many linking from source to target. In order to make the alignments symmetrical, heuristics that combine two alignments trained in opposite directions are often applied. By combining the one-to-many and many-to-one alignments, it is possible to obtain a symmetrical many-to-many alignment. For our baseline alignment, we chose the symmetrization heuristic "grow-diag-final" (*gdf*) commonly used in phrase-based SMT (Koehn et al., 2003). This heuristic adds links to the intersection of two asymmetrical statistical alignments in an attempt to assign every word a link.

5.4.2 Alignment Remapping

Using a preprocessing scheme for word alignment breaks the process of applying Giza++ on some parallel text into three steps: preprocessing, word alignment, and remapping. In preprocessing, the words to align are tokenized into smaller units. Then they are passed along to Giza++ for alignment (default settings). Finally, the Giza++ alignments are mapped back (remapped) to the original word form before preprocessing. In this work, *all* words are *AR* tokens because our hand-aligned training and test data are in this scheme (section 5.6.1). However, the alignment is done using different schemes. For instance, take the first word in table 5.1, *wsyktbhA*; if the *D*3 preprocessing scheme is applied to it before alignment, it is turned into four tokens (*w+ s+ yktb +hA*). Giza++ will link these tokens to different words on the English side (e.g., "and he will write it"). In the remapping step, the union of

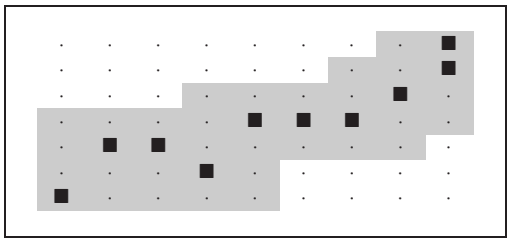

Figure 5.1 Word alignment illustrating the pruned search space for the combiner.

these links is assigned to the original word *wsyktbhA*. We refer to such alignments as remappings.

5.5 Alignment Combination

After creating the multiple remappings, we pass them as features into an alignment combiner. The combiner is also given a variety of additional features, which we discuss later in this section. The combiner is a binary classifier that determines for each source-target word pair whether they are linked or not. Given the large size of the data used, we use a simplifying heuristic that allows us to minimize the number of source-target pairs used in training. Only links evidenced by at least one of the initial alignments and their immediate neighbors are included. This is exemplified by the matrix in figure 5.1 which illustrates an alignment (the black squares) and the search space (the gray area) attained by expanding to all immediate surrounding neighbors of all links, i.e., all points bordering on the side or corner of a link. This provides the bottom left link with only three neighbors, while a centered link has eight neighbors. All other links (the white area) are considered nonexistent. This choice removes a large portion of the search space, but at the expense of raising the lower performance boundary for the system. On the development data set, 78.6% of the search space is removed at the expense of removing 2.2% of the correct links.

The combiner we use here is implemented using a rule-based classifier, Ripper (Cohen, 1996). The reasons we use Ripper as opposed to other machine learning approaches are (a) Ripper produces human readable rules that allow better understanding of the kind of decisions being made; and (b) Ripper is relatively fast compared to other machine learning approaches we examined given the very large nature of the training data we use.[2] The combiner is trained using supervised data (human annotated alignments), which we discuss in section 5.6.1.

2. In a small pilot experiment, Ripper used four hours of training time, and TinySVM used four days (`http://chasen.org/~taku/software/TinySVM/`).

In the rest of this section we describe the different machine learning features given to the combiner. We break the combination features into two types: word/sentence features and remapping features.

Word/Sentence Features

- *Word form (WW)*: The source and target word forms.
- *POS (WP)*: The source and target part-of-speech tags. For Arabic, we use the Bies POS tagset (Maamouri et al., 2004) as output by MADA. For English, we use MXPOST (Ratnaparkhi, 1996) trained on the PennTreeBank (Marcus et al., 1993).
- *Location (WL)*: The source and target *relative* sentence position (the ratio of absolute position to sentence length). We also include the difference between the source and the target relative position.
- *Frequency (WF)*: The source and target word frequency computed as the number of occurrences of the word form in training data. We also use the ratio of source to target frequency.
- *Similarity (WS)*: This feature is motivated by the fact that proper nouns in different languages often resemble each other, e.g., صدام حسين, *SdAm Hsyn*, and "saddam hussein". We use the equivalence classes proposed by Freeman et al. (2006) to normalize Arabic and English word forms (e.g., the former example becomes "sdam hsyn" and "sadam husyn"). Then we employ the longest common subsequence as a similarity measure. This produces the longest (not necessarily contiguous) sequence that the two compared sequences have in common. The similarity score is calculated as the intersection (i.e., the number of characters in the longest common subsequence) over the union (i.e., intersection + nonmatching characters) (the former example gets a similarity score of $8/(8+2) = 0.8$).

Remapping Features

- *Link (RL)*: For each possible source-target link, we include (a) a binary value indicating whether the link exists according to each remapping; (b) a cumulative sum of the remappings supporting this link; and (c) co-occurrence information for this link. This last value is calculated for each source-target word pair as a weighted average of the product of the relative frequency of co-occurrence in both directions for each remapping. The weight assigned to each remapping is computed empirically.[3] Only the binary link information provides a different value for each remapping. The other two give one combined value based on all included remappings.

3. We use the alignment error rate (AER) on the development data normalized so all weights sum to one. See section 5.6.3.

- *Neighbor (RN)*: The same information as Link, but for each of the (three to eight) immediate neighbors of the current possible link individually. These features inform the current possible link about whether its surrounding points are likely to be links. This is motivated by the fact that alignments tend toward making up a diagonal line of adjacent points in the alignment matrix.
- *Cross (RC)*: These include (a) the number of source words linked to the current target word; (b) the number of target words linked to the current source word; (c) the sum of all links to either the current source word or the current target word; (d) the ratio of the co-occurrence mass between the current target word and the current source word to the total mass between all target words and the current source word; (e) the same ratio as in (d) but in the other direction; and (f) the ratio of the total co-occurrence mass assigned to either the current source word or the current target word to the co-occurrence mass between the current target word and the current source word. With these features, we obtain a relation to the rest of the sentence. This provides information on whether there are better ways of linking the current source and target word respectively.

5.6 Evaluation

A basic assumption for our investigation is that statistical word alignments based on different preprocessing schemes will lead to different systematically detectable advantages. A machine learning algorithm should as a consequence profit from the information made available by doing word alignment based on several different preprocessing schemes as opposed to a single scheme. In order to test this hypothesis, we conduct the following four experiments with the goal of assessing

1. the contribution of alignment remapping (section 5.6.2);
2. the contribution of combination features for a single alignment, i.e., independent of the combination task (section 5.6.3);
3. the contribution of the individual features (section 5.6.4); and
4. the best-performing combination of alignment remappings (section 5.6.5).

All of these experiments are done using a development set. We then pick our best-performing system and use it on a blind test set in section 5.6.6. We also present an analysis of the rules we learn in section 5.6.7 and an error analysis of our best development system in section 5.6.8. Next, we discuss the experimental setup and metrics used in all of these experiments.

5.6.1 Experimental Data and Metrics

Data Sets

The gold standard alignments we use here are part of the IBM Arabic-English aligned corpus (IBMAC) (Ittycheriah and Roukos, 2005). Of its total 13.9K sentence pairs, we only use 8.8K sentences because the rest of the corpus uses different normalizations for numerals that make the two sets incompatible. We break this data into 6.6K sentences for training and 2.2K sentences for development by letting every fourth line go to the development set. As for test data, we use the IBMAC's test set (NIST MTEval 2003 — 663 sentences with four references, all Arabic-English–aligned). Experiments in sections 5.6.3, 5.6.4, and 5.6.5 used only 2.2K of the gold alignment training data (not the same as the development set) to minimize computation time. As for our test experiment (section 5.6.6), we use our best system with all of the available data (8.8K).

To get initial Giza++ alignments, we use an Arabic-English parallel corpus of about 5 million words of newswire (LDC-NEWS) for training data together with the annotated set. The parallel text includes Arabic News (LDC2004T17), eTIRR (LDC2004E72), English translation of Arabic treebank (LDC2005E46), and Ummah (LDC2004T18).[4]

Since the IBMAC and LDC-NEWS have much overlap, we take care to remove duplications in LDC-NEWS to avoid biasing our experiments. The additional data (LDC-NEWS minus IBMAC) was prepared to match the preprocessing scheme used in IBMAC (*AR* with some additional character normalizations). We match the preprocessing and normalizations on our additional data to that of IBMAC's Arabic and English preprocessing (Ittycheriah and Roukos, 2005).

Metrics

The standard evaluation metric within word alignment is the alignment error rate (Och and Ney, 2000b), which requires gold alignments that are marked as "sure" or "probable." Since the IBMAC gold alignments we use are not marked as such, AER reduces to 1 - F-score (Ittycheriah and Roukos, 2005):

$$Pr = \frac{|A \cap S|}{|A|} \quad Rc = \frac{|A \cap S|}{|S|} \quad AER = 1 - \frac{2PrRc}{Pr+Rc} \ ,$$

where A links are proposed and S links are gold. Following common practice, NULL links are not included in the evaluation (Ayan, 2005; Ittycheriah and Roukos, 2005). In addition to AER, we also use precision (Pr) and recall (Rc) in some cases to better compare different systems.

4. All of the training data we use is available from the Linguistic Data Consortium (LDC): `http://www.ldc.upenn.edu/`

Table 5.2 AER and word count for each alignment remapping in both directions and combined using the GDF heuristic

Remapping	Word count	DIR	INV	GDF
AR	47721	24.67	31.68	24.77
$D1$	50584	23.07	28.16	22.90
$D2$	52900	22.17	25.29	21.63
TB	54507	21.50	23.93	21.04
$D3$	65787	20.76	22.35	20.45

The baseline we measure against in all of the experiments in this section is the symmetrization algorithm "grow-diag-final" (*gdf*) discussed earlier in section 5.4. The AER of this baseline is 24.77 for the development set and 22.99 for the test set.

5.6.2 The Contribution of Alignment Remapping

We experimented with five alignment remappings in two directions: *dir* (Ar-En) and *inv* (En-Ar). Table 5.2 shows the AER associated with each of the ten alignment remappings and the remapping of their corresponding *gdf* alignment. Table 5.2 also contains information on the word count of the schemes, which corresponds to an English text with 58,201 word tokens. The more segmented a preprocessing scheme (i.e., the greater the word count), the lower the AER for either direction and for *gdf* of the corresponding remapping. The order of the schemes from worst to best is AR, $D1$, $D2$, TB, and $D3$. INV alignments are always worse than DIR alignments. This indicates the difference between Arabic and English morphology. The more you split up the Arabic words, the easier it becomes to match them to their English correspondences. Even when the Arabic word count exceeds the English with more than 7500 tokens, we still get an improvement. The results reveal that the remapping strategy in itself is an interesting approach to alignment. When interested in word-aligned text in a specific preprocessing scheme, it might be worth doing word alignment in a different scheme followed by a remapping step. The best result we obtained through remapping is that of $D3_{gdf}$ which had a 20.45% AER (17.4% relative decrease from the baseline).

5.6.3 The Contribution of Combination Features

This experiment is conducted to specify the order for combining the alignment remappings when finding the overall best system (see section 5.6.5). For each of the basic ten (non-*gdf*) alignment remappings, we trained a version of the combiner that uses all the relevant features but has access to *only* one alignment at a time.

The results of evaluating on the development data are shown in table 5.3. We see a substantial improvement resulting from applying the alignment combination as a

Table 5.3 AER for the combination system when alignment remappings are varied

Alignment remapping	AER
AR_{inv}	20.79
$D1_{inv}$	19.30
$D2_{inv}$	17.77
AR_{dir}	17.26
TB_{inv}	16.77
$D1_{dir}$	16.35
TB_{dir}	16.14
$D3_{inv}$	15.83
$D2_{dir}$	15.56
$D3_{dir}$	14.50

Table 5.4 The effect of varying feature clusters in the combination system

Feature cluster	Remove	Add cumulative
AC: Alignment cross link	16.32	19.97
AN: Alignment neighbor link	16.14	17.29
AL: Alignment basic link	16.02	17.07
WF: Word frequency	15.28	15.49
WP: Word position	15.01	14.82
WW: Word form	14.97	14.75
WL: Word location	14.78	14.77
WS: Word similarity	14.77	14.50

supervised alignment correction system. For the ten alignment remappings the AER ranges from 14.5 to 20.79, giving an average relative improvement of 29.9% (down from 20.76 to 31.68 in columns two and three in table 5.2). The relative order of all alignments remains the same with this improvement except for TB_{dir} which moves from #2 to #4. In addition to determining the order of combination, the scores in table 5.3 are also used to weigh the co-occurrence information supplied by each alignment remapping as described in footnote 3 in section 5.5.

5.6.4 The Contribution of Individual Features

In order to validate the importance of each feature cluster to the alignment algorithm, a two-step experiment is conducted. First, each feature cluster is removed individually from the best-performing system from the previous experiment (AER($D3_{dir}$) = 14.50). The increase in AER indicates the importance of this fea-

Table 5.5 Determining the best combination of alignment remappings

Alignment remapping combination	AER
$D3_{dir}$	**14.50**
$D3_{dir}D2_{dir}$	**14.12**
$D3_{dir}D2_{dir}D3_{inv}$	**12.81**
$D3_{dir}D2_{dir}D3_{inv}TB_{dir}$	13.05
$D3_{dir}D2_{dir}D3_{inv}D1_{dir}$	**12.75**
$D3_{dir}D2_{dir}D3_{inv}D1_{dir}TB_{inv}$	12.78
$D3_{dir}D2_{dir}D3_{inv}D1_{dir}AR_{dir}$	12.84
$D3_{dir}D2_{dir}D3_{inv}D1_{dir}D2_{inv}$	12.76
$D3_{dir}D2_{dir}D3_{inv}D1_{dir}D1_{inv}$	12.93
$D3_{dir}D2_{dir}D3_{inv}D1_{dir}AR_{inv}$	**12.69**

ture cluster. Second, the features are added cumulatively in the order of importance to determine the best combination of features.

The results listed in table 5.4 show that all of the features help the alignment algorithm, and the best combination of features includes all of them. Not surprisingly, the alignment features are more important than the word features.

5.6.5 Alignment Combination Experiments

To determine the best subset of alignment remappings to combine, we ordered the remappings given their AER performance when used individually in the combination system (section 5.6.3). This was done by forward selection. Starting with the best performer ($D3_{dir}$), we continue adding alignments in the order of their performance so long the combination's AER score is decreased. Our combination results are listed in table 5.5. The best alignment combination used alignments from four different schemes which confirms our intuition that such combination is useful.

We further trained our best combination on all of the training data (6.6K sentences) as opposed to only 2.2K training sentences (see section 5.6.1). The best combination performance improves slightly to 12.24 from 12.69.

5.6.6 Test Set Evaluation

We ran our best system trained on all of the IBMAC data (training and development), on all of the unseen IBMAC test set. The results are shown in table 5.6 comparing training on all seen data (training and development) to just using the training data. The development set shows a relative improvement of 50.6% (24.77 to 12.24). On the test data, we also achieve a substantial relative improvement of 38.3% when using all training data (22.99 to 14.19).

Table 5.6 Development vs. test results: AER (precision / recall)

Data	Development	Test
Baseline	24.77 (76.45 / 74.04)	22.99 (72.39 / 82.25)
TRAIN (6.6K)	**12.24** (88.43 / 87.11)	14.31 (80.17 / 92.02)
ALL (8.8K)	—	**14.19** (80.46 / 91.93)

On the test data, the initial search space reduction heuristic behaves much as on the development and training data. The search space is reduced by around 80% and in the processes only 1.4% of the correct links are removed. In other words, the lower boundary for the system is an AER of 1.4.

The test baseline is lower than the development baseline, yet the best AER on test is higher than development. The precision and recall measures give additional insights into this issue. The test baseline is much higher in terms of its recall compared to the development baseline; however its precision is lacking. This tradeoff pattern is preserved in our best systems. This large difference between precision and recall also corresponds to a disproportionate number of links in the test baseline compared to the test reference: test alignment links are 14% more than test reference, compared to development alignment links, which are 3% *less* than their reference. One possible explanation of this difference between development and test is that the test data in fact contains four replicas in the Arabic with different English translations (see section 5.6.1). Since all of the initial alignments were done jointly, the performance on this subset may be biased, especially in terms of recall. Nonetheless, our approach improves both precision and recall for both development and test. The last experiment, using all of the data for training, gives a small boost to the test AER score, but the improvement seems to be specifically in terms of an increase in precision matched with a tiny decrease in recall.

Ittycheriah and Roukos (2005) used only the top 50 sentences in IBMAC test data. Our best AER result on their test set is 14.02 (baseline is 22.48) which is higher than their reported result (12.2 with 20.5 baseline (Arabic-to-English Giza++)). The two results are not strictly comparable because (a) Ittycheriah and Roukos (2005) used additional gold-aligned data that was not released and (b) they used an additional 500K sentences from the LDC UN corpus for Giza training that was created by adapting to the source side of the test set – the details of such adaptation were not provided and thus it was not clear how to replicate to compare fairly. Clearly this additional data is helpful since even their baseline is higher than ours.[5]

5. Abraham Ittycheriah, personal communication, 2006.

5.6.7 Alignment Rule Analysis

The rules provided by Ripper have the advantage of giving us an insight into the choices made by the classifier. In this section, we look closely at three rules selected from our best-performing system.

First, the number 1 rule learned by Ripper is also the simplest and most commonly applied. It has a precision of 97.0% and a recall of 67.3%. The rule simply states that a link should be assigned if both $D3_{dir}$ and $D3_{inv}$ contain this link. In other words, the backbone of the combination alignment is the intersection of both directions of the $D3$ remappings.

Second, the number 2 rule learned by Ripper is more complex and thus less general. It has a precision of 99.0% and a recall of 2.4% (of what is left after rule number 1 applies). The rule contains the following conditions:

1. RC(a) $D2_{dir} = 1$
2. RL(a) $AR_{inv} = 1$
3. RC(f) >= 0.15
4. WS >= 0.44

The first condition states that according to the $D2_{dir}$ remapping, the Arabic word should only link to this English word. In fact, due to the unidirectionality of Giza++ alignments, this means that the two words should only align to each other (in the $D2_{dir}$ remapping). The second condition says that the AR_{inv} remapping should contain the link. The third condition requires that this link carry at least 15% of the combined lexical probability of the source and target words being linked to any word. Finally, the fourth condition states that the two word forms should, at least to a certain degree, be similar. The majority of cases handled by this rule are multiword expressions (in Arabic, English, or both) where the words being linked by the rule are similar to some degree but the links were missed by $D3_{dir}$ or $D3_{inv}$ (thus, rule number 1 did not apply).

The last rule we examine here applies as the 23rd rule of the set. It has a precision of 89.8% and a recall of 0.9% (of remaining links). The rule contains the following conditions:

1. WP(Arabic) = NN
2. WP(English) = DT
3. RN(right) $D3_{dir} = 1$
4. WL(difference) <= 0.05
5. RC(c) <= 9

The first two conditions state that the Arabic word is a noun, and the English is a determiner. The third states that the right neighbor should be linked according to the $D3_{dir}$ remapping. In other words, this reveals that the Arabic word should be linked to the following English word as well, according to $D3_{dir}$. The difference

Table 5.7 A categorization of alignment errors found in error analysis of baseline and best-performing system.

Error type	Baseline frequency	Best system frequency	Error reduction	Best system gold errors
Closed	236 (51%)	117 (54%)	50%	18 (15%)
Open	117 (25%)	33 (15%)	72%	0 (0%)
Comp.	89 (19%)	50 (23%)	44%	19 (38%)
Num.	13 (3%)	9 (4%)	31%	0 (0%)
Punc.	10 (2%)	9 (4%)	10%	2 (22%)
Total	465	218	53%	39 (18%)

in relative sentence position should be small, i.e., the words should appear in about the same place in the sentence. And finally, the two words should not have a lot of other linking options in the available remappings. In other words, an Arabic noun should link to an English determiner if the Arabic noun is also linked to the following English word (quite possibly a noun). This rule handles the fact that the determiner is often a part of the Arabic word, which is not the case in English. Only the *D3* tokenization scheme separates the *Al+* determiner in Arabic.

5.6.8 Error Analysis

We conducted a detailed error analysis on 50 sentences from our development set's baseline and best system. The sample included 1011 Arabic words and 1293 English words. We found 465 erroneous alignments (including null alignments) in the baseline and 218 errors in our best system. We classified errors as follows. *Closed-class*[6] errors involve the misalignment of a closed-class word in Arabic or English. *Open-class* errors involve open-class words such as nouns and verbs. *Numeral* and *punctuation* errors involve numbers and punctuation misalignments, respectively. Finally, *compositional* errors are complex errors involving noncompositional expressions (as in the idiom *half-brother* mapping to the Arabic اخ غير شقيق *Ax gyr šqyq*, lit. *brother, not full brother*) or compositional translation divergences (as in *aggravate* mapping to the Arabic زاد تفاقما *zAd tfAqmA*, lit. *increase aggravation*) (Dorr et al., 2002). Open-class and closed-class errors are strictly defined here to not involve compositional errors. We also computed *gold* errors in our best system; these are cases inconsistent with the alignment guidelines as explained in Ittycheriah and Roukos (2005).

6. As opposed to an open-class, a closed class is a relatively small group of words that is usually not extended by new words. Determiners, prepositions, and pronouns are examples of closed-class words.

Table 5.7 presents the results of the error analysis. The first column lists the different error classes. The second and third columns list the frequency of errors in the baseline and best system, respectively. The percentages in parentheses indicate the ratio of the error type in that column. The fourth column specifies the error reduction in our best system from the baseline. Overall, we reduce the errors by over 50%. The largest reduction is in the open-class errors followed by closed-class errors. The relative distribution of errors is similar in baseline and best system except that open-class and compositional errors exchange ranks: open-class errors are second in the baseline but third in our best system. The last column lists the frequency of gold errors. The percentages in parentheses are ratios against the corresponding best-system frequencies. The gold errors comprised 18% of all of the errors in our best system. They are generally split between closed-class and compositional errors.

The errors in our best system are consistent with previous studies where a majority of errors are associated with closed-class words (especially determiners such as *the*). Closed-class errors can be attributed to their high frequency as opposed to the open-class errors which are more a result of their low frequency or out-of-vocabulary status. Compositionality errors are complex and seem to result from there being no clear definition on what is compositional on one hand and from a lack of a multiword alignment model in our system. The need for a way to enforce a well-formed multiword alignment (or phrasal constructs in general) is also responsible for many of the closed-class errors. Such a multiword alignment model would be an interesting extension of this research since it explores the meaning of an *alignment token* at different levels above and below the *word*. Perhaps, one could use a phrase chunker or even a parser to add constraints on either alignment or combination steps (Cherry and Lin, 2006).

Punctuation errors are perhaps a result of lower use of punctuation in Arabic as opposed to English, thus, there is a lot of sparsity in the training data. Number errors are a result of neutralizing all numerals in our system, i.e., merging them all to a single number category. If the word form of the numbers had not been a neutralized expression, the word similarity measure would probably have made the system capable of handling numbers better.

5.7 Postface: Machine Translation and Alignment Improvements

In addition to evaluating the combined alignments against the gold standard alignment, we evaluated their effect on machine translation (MT) using two different statistical MT approaches: standard phrase-based MT (PHMT) implemented as Pharaoh (Koehn, 2004a) and n-gram-based MT (NGMT) (Mariño et al., 2006).

Standard PHMT uses bilingual phrase tables as its translation model. In contrast, the translation model employed in NGMT is expressed using bilingual units, termed *tuples*, and is estimated as an n-gram language model (Mariño et al., 2006). Phrases are extracted as multiple segmentations of each sentence pair, allowing for multiple overlapping spans of translations. In contrast, tuples are extracted from a single

segmentation of each sentence pair. The resulting sequences of tuples are used to estimate a bilingual n-gram language model. Additionally, tuples are extracted with *reordered* source words using the *unfold* method described in (Crego et al., 2005). Unfold uses word alignments to rewrite the source sentence following the target sentence word order. Our intuition for comparing these two approaches is that the single segmentation employed to extract tuples and the reordering component of the *unfold* extraction method make them more sensitive to alignment quality than basic phrase extraction in PHMT.

5.7.1 Experimental Setup

For training, we used the same data described in section 5.6.1. All evaluated systems use the same surface trigram language model, trained on approximately 340 million words of English newswire text from the English Gigaword corpus.[7] English language model (LM) preprocessing simply included downcasing, separating punctuation from words, and splitting off "'s." The trigram language models were implemented using the SRILM toolkit (Stolcke, 2002).

For testing, we used the standard NIST MTEval data sets for the years 2004 and 2005 (MT04 and MT05, respectively). Parameter tuning was done on the NIST MTEval 2002 set.[8] Both BLEU (Papineni et al., 2002) and NIST (Doddington, 2002) metric scores are reported in addition to multireference word error rate (mWER). All scores are computed against four references with n-grams of maximum length 4. For PHMT, tuning was done with minimum error-rate training (MERT) (Och, 2003); however, tuning was done using the Simplex algorithm (Nelder and Mead, 1965) for NGMT. Tuning is the only difference between the two implementations we compare that is not a real difference in approach. The different choices were determined by the available implementations.

5.7.2 Results

We compare the performance of the baseline alignment (AR_{gdf}) against our best combined alignments system (COMBAL) in PHMT and NGMT. The results are shown in the first four data rows in table 5.8. These results are consistent with previously published work showing that alignment improvements (as measured by AER) do not always map into MT improvements (as measured by BLEU, NIST, or mWER). The PHMT and NGMT results are not much different either. None of the differences are statistically significant. BLEU 95% confidence intervals are ± 0.0232 and ± 0.0133 respectively for MT04 and MT05.

In order to better understand the interaction between MT and alignments, we did an NGMT-specific combination of the baseline and COMBAL alignments. In

7. Distributed by the Linguistic Data Consortium: http://www.ldc.upenn.edu
8. http://www.nist.gov/speech/tests/mt/

Table 5.8 Evaluating the effect of improved alignments on MT.

MT04					
MT approach	**Alignment**	**AER**	**BLEU**	**NIST**	**mWER**
PHMT	AR_{gdf}	22.99	39.18	9.74	53.63
PHMT	COMBAL	14.19	38.48	9.62	53.51
NGMT	AR_{gdf}	22.99	38.12	9.62	55.34
NGMT	COMBAL	14.19	37.82	9.67	54.53
NGMT	AR_{gdf}+COMBAL	17.47	38.34	9.73	54.70
NGMT	COMBAL+AR_{gdf}	16.98	37.76	9.69	54.28
MT05					
MT approach	**Alignment**	**AER**	**BLEU**	**NIST**	**mWER**
PHMT	AR_{gdf}	22.99	41.97	9.96	51.38
PHMT	COMBAL	14.19	40.93	9.85	51.99
NGMT	AR_{gdf}	22.99	40.23	9.60	54.24
NGMT	COMBAL	14.19	39.28	9.54	54.08
NGMT	AR_{gdf}+COMBAL	17.47	40.85	9.74	52.67
NGMT	COMBAL+AR_{gdf}	16.98	41.05	9.81	52.23

this combination, we used the number of tuples produced by the unfold method on every sentence in the training data to select the alignment from either baseline or COMBAL. The alignment that produced the largest number of tuples was preferred, as it contained tuples of a smaller size. Large tuples have important sparseness problems, and are less reusable in test (Crego et al., 2005).

The baseline was preferred 35% of the time, the COMBAL 23% of the time, and the rest was a tie. We produced two combined alignments based on the bias of the tie (baseline or COMBAL). The results are shown in the last two data rows in table 5.8. The order of the alignment name shows the tie-breaking bias (to first-named alignment). The associated AER is computed over the same test set mentioned earlier in section 5.6.6. A small but consistent improvement in MT BLEU and NIST scores was achieved using the combination (with baseline bias) over either alignment. The AER associated with the combination is in between the two AER values of baseline and COMBAL. This result suggests that alignment improvement for MT should perhaps be optimized toward specific MT approaches/implementations rather than generic gold alignments.

Although the results from MT performance are generally weak, we believe this is a problem of either the gold standard used or the alignment error metric, or both. We think our approach to alignment combination and using different tokenization is generic enough that it can be used with other metrics and gold standards with little modification.

5.8 Conclusion

We have presented an approach for using and combining multiple alignments created using different preprocessing schemes. Our results show that the remapping strategy improves alignment correctness by itself. We also show that the combination of multiple remappings improves word alignment measurably over a commonly used state-of-the-art baseline. We obtain a relative reduction of alignment error rate of about 38% on a blind test set.

We also confirmed previous findings about the robustness of SMT to word alignment. The gain from improving word-alignment quality does not transfer to translation quality. In this case, an improvement actually seems to hurt performance, perhaps because the approach diverges from a purely statistical approach. The results indicate that AER is the wrong metric to optimize toward, when the purpose of the word alignment is as an information source for machine translation.

Acknowledgments

N.H. was supported by the Defense Advanced Research Projects Agency (DARPA) under contract no. HR0011-06-C-0023. Any opinions, findings, and conclusions or recommendations expressed in this chapter are those of the authors and do not necessarily reflect the views of DARPA. We thank Mona Diab, Owen Rambow, Martin Jansche, Bonnie Dorr, Fazil Ayan, and Abe Ittycheriah for helpful discussions. We thank IBM for making their hand-aligned data available to the research community.

6 Linguistically Enriched Word-Sequence Kernels for Discriminative Language Modeling

Pierre Mahé
Nicola Cancedda

This chapter introduces a method for taking advantage of background linguistic resources in statistical machine translation. Morphological, syntactic, and possibly semantic properties of words are combined by means of an enriched word-sequence kernel. In contrast to alternative formulations, linguistic resources are integrated in such a way as to generate rich composite features defined across the various word representations. Word-sequence kernels find natural applications in the context of discriminative language modeling, where they can help correct specific problems of the translation process. As a first step in this direction, experiments on an artificial problem consisting in the detection of word misordering demonstrate the interest of the proposed kernel construction.

6.1 Motivations

Language modeling consists in estimating a probability distribution over the sentences (actually, sequences of words) of a language. This process is central to statistical machine translation (SMT), initially formulated following the noisy-channel model (Brown et al., 1993), in which the probability $p(t|s)$ of observing a sentence t in the target language conditionally on a sentence s in the source language is expressed as

$$p(t|s) \propto p(s|t)p(t). \tag{6.1}$$

This decouples the modeling problem into

1. estimating a *translation model* $p(s|t)$ to quantify how well t conveys the information contained in the source sentence s,

2. estimating a (target) *language model* $p(t)$ to assess the likelihood of t as a sentence in the target language.

Modern SMT systems are based on a broader class of log-linear models (Och and Ney, 2002) that considerably generalize the noisy-channel model[1] by introducing arbitrary real-valued feature functions $h_i(s,t)$:

$$p(t|s) \propto \exp\Big(\sum_i \alpha_i h_i(s,t)\Big). \tag{6.2}$$

Nevertheless, language models remain central to such systems, where they invariably appear as a feature function $h_{\mathrm{lm}}(s,t) = \log p(t)$.

In practice, the most widely used family of language models is the class of *n-gram* models defined as

$$p(t) = \prod_{i=1}^{|t|} p(t_i|t_{i-1},\ldots,t_{i-N+1}), \tag{6.3}$$

where t is a sequence of $|t|$ words in the target language, assumed to be "padded" with $N-1$ empty symbols so that this equation is well defined. These models basically make a Markov independence assumption between the words of a sentence, and capture local regularities of the language. They are usually trained by maximizing the likelihood of a large set of sentences in the target language, specific smoothing techniques being applied to cope with the high dimensionality of the parameter space (Chen and Goodman, 1996).

Generative language models (LMs) such as these are fully justified in the noisy-channel framework. Notice, however, that LMs are usually the primary instrument for promoting translation fluency, based on the tacit assumption that likelihood and fluency correlate well. In the context of SMT, an alternative —or at least complementary— approach is to view language modeling as a discrimination problem: a classifier can be trained to distinguish between fluent and disfluent sentences, and the underlying scoring function can subsequently be used as a feature of the log-linear model (Eq. (6.2)). As a first step in this direction, we present in this chapter a simple way to incorporate background linguistic resources in discriminative language models. More precisely, we define linguistically enriched kernel functions (Shawe-Taylor and Cristianini, 2004) where words are not only represented by their surface forms but according to a set of linguistic features that, following the terminology adopted in Bilmes and Kirchhoff (2003) and Koehn and Hoang (2007), we call *factors*. Our main contribution is an extension of word-sequence kernels (Cancedda et al., 2003) that makes effective use of the different factors through the introduction of composite linguistic features.

The rest of the chapter is structured as follows. After a brief introduction to word-sequence kernels, the next section describes the notion of factored representation

1. Note indeed that $p(t|s) \propto p(s|t)p(t) = \exp\big(\log p(s|t) + \log p(t)\big)$.

and details our kernel formulation. Section 6.3 validates the kernel construction on an artificial discrimination task reproducing some of the conditions encountered in translation, and section 6.4 discusses related and future work.

6.2 Linguistically Enriched Word-Sequence Kernels

In this section, we give a brief and intuitive introduction to word-sequence kernels, define the notion of factored representation derived from multiple linguistic representations, and propose a novel way to integrate such factors into a global word-sequence kernel.

6.2.1 Word-Sequence Kernels

Sequence kernels derive a measure of similarity between sequences by means of their common subsequences. In this chapter, we focus on the *gap-weighted subsequence* formulation of sequence kernels that was first introduced to compare sequences of characters (Lodhi et al., 2002), and later extended to compare sequences of words (Cancedda et al., 2003).

Letting $|x|$ be the length of a sequence x, and $x[\mathbf{i}]$ be its restriction to the symbols indexed by the vector $\mathbf{i}$, the set of subsequences of size n associated with x can be written formally as

$$\mathcal{S}_n(x) = \Big\{ x[\mathbf{i}] \,;\; \mathbf{i} = [i_1, \ldots, i_n] \in \{1, \ldots, |x|\}^n,\; i_1 < i_2 < \cdots < i_n \Big\}.$$

The indices $i_1, \ldots, i_n$ that define a subsequence are solely constrained to be (strictly) increasing. As a result, a subsequence can contain *gaps*: some positions can be skipped while extracting the symbols of the subsequence. For example, in the sequence $x =$ "this" "is" "a" "sequence," the vector of indices $\mathbf{i} = [1, 4]$ defines the subsequence $x[\mathbf{i}] =$ "this" "sequence" that contains two gaps, corresponding to the words "is" and "a." We will denote by $g(x[\mathbf{i}])$ this total number of gaps in the following,[2] which can be seen to be given by $i_n - i_1 + 1 - n$.

Letting $\mathbf{1}(.)$ be a generic indicator function being 1 iff its argument is true, we can write the gap-weighted subsequence kernel as a standard inner product, defined for the pair of sequences (x, y) drawn from an alphabet Σ as

$$k_n(x, y) = \langle \phi(x), \phi(y) \rangle,$$

where $\phi(x) = \big(\phi_u(x)\big)_{u \in \Sigma^n}$ maps the sequence x into a feature space indexed by Σ^n, the set of sequences of length n that can be drawn from the alphabet Σ; and $\phi_u(x) = \sum_{s_x \in \mathcal{S}_n(x)} \mathbf{1}(s_x = u) \lambda^{g(s_x)}$ counts the number of times a particular

2. The function g is introduced for convenience in the notation. Strictly speaking, $x[\mathbf{i}]$ does not contain enough internal structure to compute $g(x[\mathbf{i}])$.

sequence u is found within $\mathcal{S}_n(x)$, each occurrence being downweighted by the number of gaps it contains, according to a parameter $\lambda \in [0, 1]$.

The parameter λ is called the gap penalization factor. When it tends to zero, any subsequence containing gaps gets so downweighted that the kernel boils down to counting the number of contiguous subsequences. When λ increases, the penalization of gaps decreases, and for $\lambda = 1$, every common subsequence contributes equally to the kernel value, independently of its number of gaps. We refer the interested reader to Shawe-Taylor and Cristianini (2004) for a thorough introduction to sequence kernels.

The size of the feature space associated with this kernel is therefore $|\Sigma|^n$, which can be very large in practice, especially for word-sequence kernels where $|\Sigma|$ typically exceed $10,000$. Nevertheless, this kernel can be computed efficiently exploiting properties common to all *convolution* kernels (Haussler, 1999). Indeed, it is easy to see that the kernel can equivalently be written as

$$k_n(x, y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \mathbf{1}(s_x = s_y) \lambda^{g(s_x)} \lambda^{g(s_y)}, \tag{6.4}$$

meaning that it can be computed directly by comparing all pairs of subsequences that can be extracted from x and y. For that purpose, Lodhi et al. (2002) introduce a dynamic programming algorithm having a complexity of $O(n|x||y|)$. Interestingly, when computing the order-n kernel k_n, this algorithm computes the kernels k_i for $i < n$ as intermediaries, which offers the possibility to compute a "blended" or "up to n" kernel $k = \sum_{i=1}^{n} \mu_i k_i$ at almost no extra cost, the parameters $\mu_i \geq 0$ controlling the relative contribution of the kernel computed at different orders.

We note finally that because the number of subsequences associated with a sequence increases with its length, the value of the kernel is highly dependent on the size of the sequences to be compared. In order to compensate for this size effect, it is common to consider a normalized version of the kernel, defined as $\tilde{k}_n(x, y) = k_n(x, y)/\sqrt{k_n(x, x) k_n(y, y)}$.

6.2.2 Factored Representation and Kernel Combination

We now consider the case where words are not only characterized by their surface forms but along several linguistic dimensions. This work focuses more precisely on linguistic features associating a single and discrete tag to each word, which, following the terminology of Bilmes and Kirchhoff (2003) and Koehn and Hoang (2007), we refer to as *factors* in what follows. This situation occurs naturally for common linguistic features such as word lemmas and parts of speech, upon which we base our experimental study. The formalism we adopt is nevertheless very general, and section 6.4 discusses the introduction of higher-level features.

Under this representation, a word u is formally defined as a set of p factors $\{u^{(d)}\}_{d=1:p}$ drawn from different alphabets Σ_d. A simple way to define a kernel on

such multidimensional sequences is to resort to a linear combination of individual kernels:

$$k_n^{\text{lin}}(x, y) = \sum_{d=1}^{p} w_d k_n(x^{(d)}, y^{(d)}), \tag{6.5}$$

where $x^{(d)}$ denotes the dth sequence of word factors associated with the sequence x, k_n is the kernel[3] of Eq. (6.4), and the weights $w_d \geq 0$ control the relative contributions of the different factors in the global kernel. Despite its simplicity, this integration scheme is known to be effective in practice (Lanckriet et al., 2004), and was recently pursued for word-sense disambiguation under the denomination of *syntagmatic kernels* (Giuliano et al., 2006).

Because summing a set of inner products is equivalent to (i) concatenating the corresponding feature vectors and (ii) computing a single inner product, this linear combination has the effect of concatenating the feature spaces of the individual kernels. In our case, the individual feature spaces are indexed by $(\Sigma_d)^n$, and it follows easily that the kernel in Eq. (6.5) can be written as a standard inner product:

$$k_n^{\text{lin}}(x, y) = \langle \phi(x), \phi(y) \rangle,$$

where $\phi(x) = \big(\phi_u(x)\big)_{u \in \Sigma}$ is indexed by $\Sigma = \bigcup_{d=1}^{p} (\Sigma_d)^n$, the set of sequences of length n that can be drawn separately from the different alphabets Σ_d; and for a sequence $u \in (\Sigma_d)^n$, $\phi_u(x) = \sqrt{w_d} \sum_{s_x \in \mathcal{S}_n(x^{(d)})} \mathbf{1}(s_x = u) \lambda^{g(s_x)}$ counts, in the appropriate dimension $x^{(d)}$, the (gap-weighted) number of occurrences of u, multiplied by the square root of the weight w_d associated with the dth factor.

6.2.3 Factored Kernel

The above kernel definition has the limitation of leading to a disconnected integration of the linguistic resources, in the sense that each sequence indexing its feature space involves symbols drawn from a single alphabet. It is often the case that linguistic phenomena can be expressed at an abstract grammatical or morphological level, and yet are *lexicalized*, that is, they depend on specific words (or lemmas). For instance, we might want to be able to express the fact that the pattern "`phone PRON up`" is fluent. This would generalize better than "`phone her up`," "`phone him up`," "`phone you up`," etc. separately, and would be more accurate than "`VB PRON PREP`," which would also promote "`phone him out`" as equally fluent. With this in mind we now introduce an alternative definition of kernels for factored representations, which is able to combine the different alphabets into composite subsequences.

3. Note that for simplicity, we introduce a single kernel order n to compare the different sequences, but one could use factor-specific orders $n(d)$.

Because a pair of sequences is identical if it is made of pairwise identical symbols, the sequence kernel of Eq. (6.4) can equivalently be written as

$$\begin{aligned} k_n(x,y) &= \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \mathbf{1}(s_x = s_y) \lambda^{g(s_x)} \lambda^{g(s_y)} \\ &= \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \lambda^{g(s_x)+g(s_y)} \prod_{i=1}^{n} \mathbf{1}(s_x[i] = s_y[i]). \end{aligned}$$

This simple fact allows us to considerably enrich the kernel definition by turning the binary function checking whether a pair of symbols is identical or not into a flexible kernel function meant to quantify their similarity. This is know as the *soft-matching* extension of sequence kernels (Shawe-Taylor and Cristianini, 2004), and can formally be written as

$$k_n^{\text{soft}}(x,y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \lambda^{g(s_x)+g(s_y)} \prod_{i=1}^{n} k_\Sigma(s_x[i], s_y[i]),$$

where $k_\Sigma : \Sigma \times \Sigma \to \mathbb{R}$ is a kernel between symbols. Soft-matching kernels do not have an explicit interpretation as inner products in general, but provided the elementary kernel k_Σ is a proper kernel function, this construction is known to be valid by the closure properties of the class of kernel functions (Haussler, 1999; Shawe-Taylor and Cristianini, 2004).

To accommodate factored representations, we propose to resort to this soft-matching formulation. Different schemes can be considered to define a kernel k_Σ comparing vectors of symbols. In this work, we focus on the following definition:

$$k_\Sigma(u,v) = \sum_{d=1}^{p} w_d k^{(d)}(u^{(d)}, v^{(d)}),$$

which provides a direct way to integrate multiple factors guaranteeing symmetry and positive semidefiniteness. In this definition, the kernel $k^{(d)}$ compares the dth word factors, and the weights $w_d \geq 0$ control the relative contribution of the different word factors in the word similarity measure. When word factors consist of exactly one tag, as it is assumed in this work, the kernels $k^{(d)}$ can naturally be defined as binary functions checking whether the factors are identical or not. Altogether, this leads to the following kernel definition:

$$k_n^{\text{fact}}(x,y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \lambda^{g(s_x)+g(s_y)} \prod_{i=1}^{n} \sum_{d=1}^{p} w_d \mathbf{1}(s_x^{(d)}[i] = s_y^{(d)}[i]), \quad (6.6)$$

that we call the *factored kernel*.

As mentioned above, soft-matching sequence kernels do not have an explicit interpretation as inner products in general. With this restricted definition, however, we can provide such an interpretation, which allows relating the kernel construction to other integration schemes. For that purpose we introduce a notion of *composite*

sequences: sequences drawn from multiple alphabets. For example, a sequence $u \in \Sigma_1 \times \Sigma_2 \times \Sigma_1$ is a composite sequence of size 3: it corresponds to the concatenation of three symbols, the first and third ones being drawn from a first alphabet Σ_1 and the second one being drawn from a second alphabet Σ_2. The set of *composite subsequences* of size n associated with a factored sequence x can be defined as

$$\mathcal{S}_n^*(x) = \big\{x^{(\mathbf{d})}[\mathbf{i}] = x^{(d_1)}[i_i] \dots x^{(d_n)}[i_n], \\ \mathbf{i} \in \{1, \dots, |x|\}^n, \mathbf{d} \in \{1, \dots, p\}^n, i_1 < i_2 < \dots < i_n\big\}.$$

A composite subsequence $x^{(\mathbf{d})}[\mathbf{i}] \in \mathcal{S}_n^*(x)$ is defined by two n-tuples of indices:

- a tuple of word indices $\mathbf{i}$, as in the standard (unidimensional) case,
- a tuple of factor indices $\mathbf{d}$, defining the alphabets involved in the composition.

Based on these definitions, we can state the following proposition, whose proof is postponed to the appendix.

Proposition 6.1 *The factored kernel of Eq. (6.6) can be written as a standard inner product:*

$$k_n^{fact}(x, y) = \langle \phi(x), \phi(y) \rangle,$$

where $\phi(x) = \big(\phi_u(x)\big)_{u \in \Sigma}$ *is indexed by* $\Sigma = \big(\bigcup_{d=1}^{p} \Sigma_d\big)^n$*, the set of composite sequences of size* n *that can be drawn from the union of the different alphabets* Σ_d*; and for the composite sequence* $u \in \Sigma_{f_1(u)} \times \dots \times \Sigma_{f_n(u)}$*, where* $f_i(u) \in \{1, \dots, p\}$ *indexes the alphabet from which the* i*th symbol of the sequence is drawn, we have* $\phi_u(x) = w(u) \sum_{s_x \in \mathcal{S}_n^*(x)} \mathbf{1}(s_x = u) \lambda^{g(s_x)}$*, with* $w(u) = \prod_{i=1}^{n} \sqrt{w_{f_i(u)}}$.

In comparison with the linear combination of kernels, the central point of this factored integration is therefore the ability to detect composite subsequences in which symbols are drawn from $\bigcup_{d=1}^{p} \Sigma_d$, the union of the individual alphabets. Intuitively, the factored kernel can be seen as (i) merging the individual alphabets into a global alphabet, and (ii) counting the resulting composite subsequences in the factored representation. On the other hand, linearly combining the kernels corresponds to (i) counting subsequences in the different linguistic representations, and (ii) merging the resulting feature vectors. The next section illustrates the benefit of being able to consider composite subsequences. Moreover, the factored integration can be seen to be slightly more efficient from the computational point of view. Indeed, while the linear combination of kernel basically multiplies the initial complexity by a factor p, leading to an overall complexity of $O(np|x|y|)$, the soft-matching extension has the same $O(n|x|y|)$ complexity as the hard-matching one, provided the word similarity measure is computed in advance. The complete k_Σ

Gram matrix would in our case be of size $(|\Sigma_1||\Sigma_2|\dots|\Sigma_p|)^2$, and it would thus be highly impractical to precompute it and store it in memory. In general, computing *on the fly* only the entries of this matrix needed to compare x and y requires $|x|\times|y|$ evaluations of the kernel k_Σ. In our case, k_Σ has a complexity of p, which leads to an overall complexity of $O\big((n+p)|x||y|\big)$ to compare x and y.

6.2.4 Illustration

We now give an illustration of the different approaches introduced above on a simple example: comparing the sentences `He comes from London` and `She came to Paris` based on a factored representation defined from the surface forms of the words, their lemmas, and their parts of speech (POS):

surface	He	comes	from	London	She	came	to	Paris
lemma	*he*	*come*	*from*	*London*	*she*	*come*	*to*	*Paris*
POS	PRON	VERB	PREP	PROPER	PRON	VERB	PREP	PROPER

These sentences clearly have a related meaning and a similar structure. An occurrence of one as a positive example in the training set should therefore provide evidence of the fluency of the other. However, since they don't have a single word in common, the original word-sequence kernel, which solely considers their surface forms, is not able to detect this similarity at all. Integrating the different linguistic dimensions proves to be useful in this respect. Restricting ourselves to the detection of common subsequences of length 4, we first note a match along the POS dimension through the sequence `PRON-VERB-PREP-PROPER`. The linearly combined kernel (Eq. (6.5)) and the factored kernel (Eq. (6.6)) are both able to take into account this match into the global similarity measure. The first important difference between the two formulations is related to their parametrization. Indeed, they respectively give a weight of w_{pos} and w_{pos}^4 to this match in the global kernel. More important, however, is the ability of the factored kernel to detect in addition the composite match `PRON-`*come*`-PREP-PROPER` involving both word lemmas and POS. On this particular example, this composite sequence bears a richer information with respect to the semantic content shared by the sentences. Experiments in section 6.3 reveal that such composite subsequences are worth including in the kernel. As an ending remark, we note that this composite subsequence is given a weight equal to $w_{\text{pos}}^3 \times w_{\text{lemma}}$, inbetween the weights of w_{pos}^4 and w_{lemma}^4 given to sequences of length 4 consisting exclusively of POS tags and word lemmas respectively.

6.2.5 Rational Kernel Interpretation

We conclude this section by an interpretation of the above definitions in the context of rational kernels (Cortes et al., 2004), meant to provide an intuitive illustration of the different kernel constructions. Loosely speaking, the framework of rational

kernels offers the possibility to define kernels between finite-state automata by means of a generic operation of transducers composition. Two key elements enter this process: the pair of automata A and B to be compared, and a transducer T in charge of their comparison. In some sense, the transducer T is designed to extract patterns from automata through an operation of transducer composition, denoted as $A \circ T$ when it is composed with the automaton A. At a further level of composition, the operation $(A \circ T) \circ (B \circ T)^{-1} = A \circ (T \circ T^{-1}) \circ B$ makes it possible to extract simultaneously patterns from A and B, through the composed transducer $T \circ T^{-1}$. Under general conditions on the transducer T and the input automata A and B, positive definite kernels can be obtained from this construction.

As illustrated in figure 6.1 a sequence $s = (s_1, \ldots, s_n)$ corresponds to a trivial automaton defined by $n+1$ states $(e_0, \ldots, e_n)$ and n transitions between successive states (e_i, e_{i+1}), corresponding to the emission of the n symbols of the sequence. Under this representation, Cortes et al. (2004) show that the gap-weighted subsequence kernel of Eq. (6.4) can be computed by means of a simple transducer T. Without going into details, composing the automaton A encoding a sequence and this particular transducer T leads to a transducer $A \circ T$ such that there is a bijection between its set of successful paths and the set of (noncontiguous) subsequences of the sequence encoded by A. More details about this construction can be found in Cortes et al. (2004). From the analysis of the previous sections, we note that both the linear combination of individual kernels (Eq. (6.5)) and the factored kernel (Eq. (6.6)) can be seen as rational kernels involving the same transducer T, but modified input automata in which

- in the linear combination of kernels, the internal states and the sequence and transitions are duplicated in order to encode the separate emission of symbols drawn from the different alphabets;
- in the factored kernel, the transitions from the states i to the states $i + 1$ in the automata are replaced by a set of d transitions encoding the emission of the different word factors.

These representations are illustrated in figure 6.1.

6.3 Experimental Validation

In order to validate the proposed kernel construction, we consider an artificial problem consisting of discriminating naturally occurring sentences from disrupted sentences. For that purpose, a training and a test set of respectively 5000 and 2000 English sentences are extracted from the Europarl corpus.[4] Within each set, half of the sentences are kept as such, and we disrupt the sentences of the other half by randomly permuting pairs of consecutive words with probability 0.16. Preliminary

4. `http://www.statmt.org/europarl/`

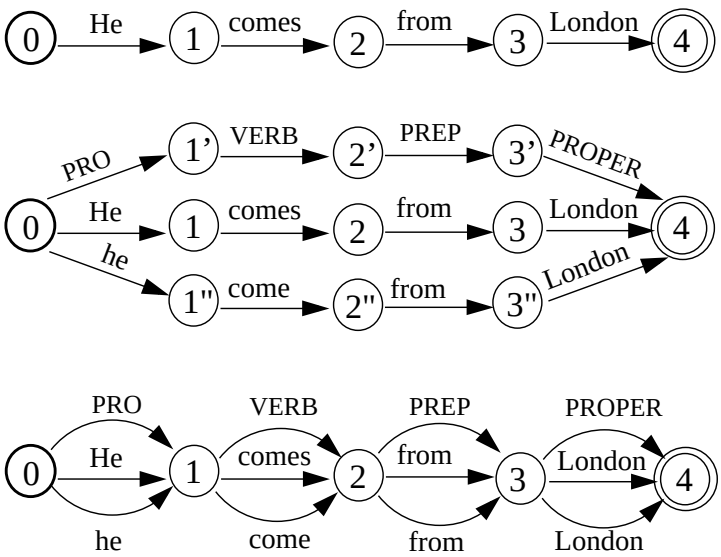

Figure 6.1 Finite-state automaton interpretation of sentences in rational kernels. *Top*: single-factor word-sequence kernel; *middle*: linear combination of kernels; *bottom*: factored word-sequence kernel.

experiments[5] revealed consistent results for values smaller and greater than 0.16, so we decided to focus on this intermediate level of disruption. Note that the perturbations are done iteratively, and as a result, a given word can be carried far away from its original position by a series of swaps. Albeit artificial, this problem is related to the translation task where such misorderings can seriously degrade the fluency of the produced sentences. As illustrated in section 6.2.4, we consider a tridimensional factored representation in which, besides its surface form, a word is represented by its lemma and part of speech (POS). In these experiments, tagging is done by the Xerox Incremental Parser (XIP) developed at Xerox Research Center Europe (Aït-Mokhtar et al., 2002), and is performed after perturbing the sentences,

The different kernels were implemented in C, following the dynamic programming (DP) approach described in Lodhi et al. (2002). The only modification consists in adding an extra step for computing the required k_Σ values before running the DP recursion, as mentioned at the end of section 6.2.3, and detailed in Shawe-Taylor and Cristianini (2004). We used the python PyML package[6] to perform SVM classification. The "C" soft margin parameter of the SVM is systematically optimized by cross-validation on the training set and we report classification accuracy on the test set.

5. detailed in `http://www.smart-project.eu/D31.pdf`
6. `http://pyml.sourceforge.net/`

6.3.1 Kernels on Individual Factors

We start by studying the behavior of the standard word-sequence kernel (6.4) when it is based on a single factor, according to (i) the factor considered; (ii) the length n of the subsequences, taken in $\{2, 3, 4\}$; and (iii) the gap penalization factor λ, taken in $\{0.001, 0.1, 0.5, 1.0\}$. Results are presented in table 6.1 for the "only-n" case, where the kernel is based exclusively on subsequences of size n, and table 6.2 for the "blended" or "up to n" case, where the kernel is defined as the sum of the (normalized) kernels computed from subsequences of size 1 to n.

Several conclusions can be drawn from table 6.1. First, we note that, in general, better results are obtained for small λ, or in other words, when we favor contiguous subsequences. This observation makes sense considering the problem at hand. Indeed, since negative examples are obtained by swapping pairs of consecutive words, it is not surprising that gaps do not prove useful in this context. Second, we obtain comparable results with surface forms and lemmas, and systematically better results with POS. The first point can be explained by the fact that English is a language of limited inflection, and as a result, the surface form of the words and their lemmas often coincide. The fact that parts of speech give better results is not so surprising either with respect to the problem considered: sequences of POS tags intuitively constitute powerful features to detect word swapping. Finally, we note that for each factor the classification performance decreases when n increases, especially for surface forms and lemmas. This behavior is most likely due to the fact that the feature space associated with the sequence kernel has a dimensionality of $|\Sigma|^n$, which increases exponentially with n. As a result, feature vectors get sparser when n increases, which tends to render them orthogonal. This effect is more severe for surface forms and lemmas than it is for parts of speech because their alphabets, and as a result, their associated feature spaces, are significantly bigger (tens of thousands vs. tens of symbols). This issue is well known when dealing with sequence kernels, and although postprocessing the kernel matrix in order to reduce its diagonal dominance can help compensate for this (Weston et al., 2003), we did not further investigate this point.

Concerning table 6.2, we note that the behavior of the kernel with respect to the considered factor and the value of λ is globally similar: better results are in general obtained with small values of λ and POS outperform surface forms and lemmas, which give comparable results. Concerning the integration of subsequences of different lengths, we first note that integrating the unigram ($n = 1$) and the bigram ($n = 2$) kernels seems to slightly improve the results obtained with bigrams only for surface forms and lemmas. This result is quite surprising since unigrams should not bring any information with respect to the problem considered. We don't have a precise explanation for that, and conjecture that there is some bias in the distribution of words within the two classes of sentences. This seems to be confirmed by the fact that unigram kernels computed from surface forms and lemmas give results around 52%, where one would expect 50% of correct classification if words were identically distributed within each class of sentences. Although POS results can

Table 6.1 Test set accuracy obtained with the word-sequence kernel (Eq. (6.4)) computed on isolated factors – "only-n" version

	$n=2$				$n=3$				$n=4$			
λ	0.001	0.1	0.5	1.0	0.001	0.1	0.5	1.0	0.001	0.1	0.5	1.0
Surface	**85.1**	84.3	80.1	56.1	73.0	73.1	72.0	58.1	65.1	64.5	63.7	57.0
Lemma	**85.6**	85.0	79.3	56.3	73.9	73.6	73.0	58.5	65.1	64.7	64.7	57.6
POS	**89.7**	89.6	85.1	63.9	88.1	88.3	87.8	69.3	85.5	86.7	87.5	74.5

Table 6.2 Test set accuracy obtained with the word-sequence kernel (Eq. (6.4)) computed on isolated factors – "up to n" version: $k = \sum_{i=1}^{n} k_i$

	$n=2$				$n=3$				$n=4$			
λ	0.001	0.1	0.5	1.0	0.001	0.1	0.5	1.0	0.001	0.1	0.5	1.0
Surface	**85.8**	85.3	79.3	54.9	85.6	85.0	80.2	55.9	84.6	84.8	79.3	55.8
Lemma	**86.4**	85.6	78.8	55.2	86.3	85.8	79.8	55.9	85.1	85.4	79.1	56.2
POS	89.5	89.5	85.3	66.2	89.7	**90.0**	87.8	70.9	89.2	89.3	86.9	74.4

be seen to slightly improve, further integrating trigrams ($n = 3$) globally leaves the results unchanged. At order 4 however, results start decreasing, especially for surface forms and lemmas, which might be related to the issue of diagonal dominance of the kernel matrix at high orders. Finally, we note that while we decided to simply consider the sum of the kernels computed at different orders in our experiments, better results might be obtained using a general linear combination $k = \sum_i \mu_i k_i$ with parameters μ_i tuned from the training set.

6.3.2 Integration of Factors

We now turn to the central part of the study: the integration of the different factors. According to the above analysis, we systematically set the value of the gap penalization factor λ to 0.001. To evaluate the influence of the individual factors, we consider different vectors of weights, summarized in the following table for kernels of order n:

	combined			factored		
	surface	lemma	POS	surface	lemma	POS
w_1	1	1	1	1	1	1
w_2	1	1	2	1	1	$\sqrt[n]{2}$
w_3	1	1	5	1	1	$\sqrt[n]{5}$
w_4	1	1	10	1	1	$\sqrt[n]{10}$
w_5	1	2	5	1	$\sqrt[n]{2}$	$\sqrt[n]{5}$

In the first four weight vectors, the contribution of the POS factor, which gives individually the best results, is gradually increased. The last weight vector further increases lemmas over surface forms. The weights used in the factored kernel of order n are set to the nth root of the weights used in the linear combination of kernels in order to make a direct comparison between the two approaches. Indeed, recall from section 6.2 that a subsequence drawn from the dth factor is given a weight of $\sqrt{w_d}$ in the linearly combined kernel (see section 6.2.2) and a weight of $\prod_{i=1}^{n} \sqrt{w_d}$ in the factored formulation (see section 6.2.3). Setting the weight used in the factored kernel to the nth root of the corresponding weight involved in the linear combination of kernels gives the same weight to the set of subsequences that appear in both formulations. The composite features that are additionally included in the factored kernel are given weights inbetween the weights given to sequences involving a single factor, as illustrated in section 6.2.4.[7]

Figure 6.2 shows the results obtained with the linear combination of individual kernels and the factored kernel, for n taken in $\{2, 3\}$, in the "only n" case (left) and the "up to n" case (right). The main conclusion that can be drawn from this figure is that factored kernels (dark bars) systematically outperform linear combinations of kernels (light bars), which indicates that introducing composite subsequences in the model is indeed beneficial. We also note, in comparison with tables 6.1 and 6.2, that integrating the different factors is always beneficial: both the linearly combined and factored kernels systematically outperform identically parameterized kernels computed on individual factors. Concerning the contribution of the individual factors, we note quite surprisingly that the influence of the weight vector is marginal, although increasing the influence of POS over other factors seems to help a little (especially in the "only n" case). Finally, we note a drop in performance for the "only n" case when the order of the kernel is increased from 2 to 3, while results stay globally similar in the "up to n" case, which is consistent with observations made from individual factors (tables 6.1 and 6.2). Altogether, optimal results are obtained by the factored kernel taken "up to $n = 3$," showing a 2% absolute improvement over the POS baseline, and 1% over the linear combination of kernels.

6.3.3 Comparison to N-Gram Models

In order to provide baseline results, we turn standard n-gram language models (6.3) into binary classification algorithms according to the following procedure. First, we train an n-gram language model on the regular (that is, the nondisrupted) version of the training set used to learn the SVM models in the previous experiments.

7. The equality between weights associated with the features common to both formulations only holds in the "only-n" case. Indeed, in the "up to 2" case, for instance, unigrams and bigrams drawn from the dth factor are given the same weight (equal to $\sqrt{w_d}$) in the linearly combined kernel, which is not the case in the factored kernel (bigrams have a weight of $\sqrt{w_d}$ and unigrams a weight of $(\sqrt{w_d})^{1/2}$).

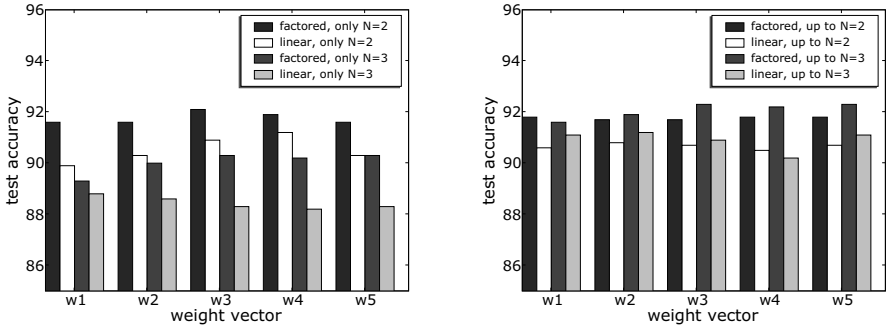

Figure 6.2 Test set classification accuracy obtained by the linear combination of kernels and the factored kernel, for different vectors of weights (defined in the text) and $\lambda = 0.001$. *Left*: "only n" version; *right*: "up to n" version.

Training was done using the SRI language modeling toolkit[8] with Kneser-Ney as smoothing method. This language model is then used to compute the (per word) perplexity of the sentences of the (disrupted) test set. Finally, we make predictions by thresholding these perplexity values: sentences having a perplexity lower than the threshold are classified as positive, and sentences with greater perplexity are classified as negative. While this threshold could be automatically selected, using cross-validation techniques for instance, in these experiments we simply pick the threshold maximizing the accuracy on the test set. The same procedure was applied to learn a POS language model, where words are represented by their parts of speech instead of their surface forms. Results are presented in table 6.3 for language models based on n-grams of order 2 to 6. Note that since the decision threshold is optimized on the test set, these results constitute an upper bound of the classification accuracy one could obtain with such a procedure.

We can note from this table that optimal results are obtained for $n = 2$ when the model is based on surface forms, and for $n = 4$ when it is based on POS. This is most likely due to the fact that POS-based n-gram models have many fewer parameters than standard language models, and can consequently be estimated more accurately from a limited amount of training data. As opposed to the SVM models involved in previous experiments, n-gram models are trained generically, that is, not specifically for this particular discrimination task. Nevertheless, this approach leads to comparable (for POS), and even better results (for surface forms) when words are represented according to a single factor. The results are still outperformed by kernels on factored representations, as well as linear combinations of individual kernels.

Finally, table 6.4 shows the results obtained when the (surface form) language model is learned from the whole Europarl corpus with the exception of the test set.

8. `http://www.speech.sri.com/projects/srilm/`

Table 6.3 N-gram baseline – Test set classification accuracy using n-gram models trained on the nondisrupted training set of 5000 sentences

n	2	3	4	5	6
Surface forms	**87.5**	87.0	87.1	87.0	87.0
Parts of speech	87.6	89.2	**90.1**	89.8	89.2

Table 6.4 N-gram baseline – Test set classification accuracy using n-gram models trained on the whole Europarl corpus but the test set

n	2	3	4	5	6
Surface forms	94.7	95.2	95.5	95.4	**95.6**

This raises the size of the training set of the language model from 5000 to around 1.4 million sentences. In comparison with table 6.3, we note that considerably better results can be obtained by simply increasing the size of the training set. Moreover, we note that optimal results are now obtained with n-grams of order 3 and greater, which reflects the fact that larger data sets allow the estimation of models of a higher complexity. With an accuracy of 95.6% this approach greatly outperforms the results we obtained with kernel-based approaches, which suggests that important gains could be obtained by applying the kernel-based approach to larger data sets.

6.4 Conclusion and Future Work

In this chapter we have considered different approaches to define kernels between linguistically enriched representation of sentences. In particular, we have introduced a global way to integrate multiple linguistic resources. On an artificial problem meant to reproduce a typical issue of SMT, namely word misordering, the proposed construction was shown to outperform a standard alternative formulation.

Our future work will be mainly dedicated to the actual integration of these techniques in SMT systems. A rapid way to assess their impact, which does not require applying complex modifications to the decoder, is to adopt a reranking approach. Reranking casts translation into a two-step process. To translate a given sentence, the first step is to produce an "n-best list" of candidate translations by the decoder: these are the n top-ranked translations according to the log-linear model (Eq. (6.2)). In a second step, this list of candidates is reranked to find a better candidate than the one returned by default by the decoder (that is, the first one in the n-best list). When informative features not directly accessible by the decoder are used for reranking, this approach can improve the fluency of the produced sentences, and is now a standard component of SMT systems (see, for instance, Och et al. (2004) and Shen et al. (2004)). The mainstream approach to

learn reranking models is to use perceptron algorithms trained from a development set, kept aside from training, according to automatic criteria such as the BLEU or the NIST scores. There are at least two simple ways to integrate kernels in such models:

- Perceptrons being straightforward to "kernelize," a first approach would be to integrate directly the kernels in their scoring functions. This would be very similar to the approach presented in Roark et al. (2004) based on n-gram features, which proved effective in the context of speech recognition.
- An alternative approach, in the direct continuity of this work, would be to first train an SVM model to distinguish between fluent and disfluent sentences, and to use the resulting scoring function as a single additional feature to learn the reranking model.

On the practical side, the large-scale experiments carried out with the whole Europarl corpus and standard n-gram models revealed the necessity to be able to consider large data sets. In this respect the quadratic complexity of the gap-weighted subsequence kernel constitutes a major obstacle. However, we note that this quadratic complexity is due to the initial requirement that the kernel handles gaps, which did not prove to be useful in our experiments. Focusing on contiguous subsequences paves the way to computationally cheaper algorithms based, for instance, on trie-trees or suffix trees, that reduce the complexity from quadratic to linear (Shawe-Taylor and Cristianini, 2004; Vishwanathan and Smola, 2004). These algorithms can be further extended to accommodate for a restricted number of gaps and mismatches within the subsequences (Leslie and Kuang, 2004), but to the best of our knowledge, they are not compatible with the soft-matching extension of sequence kernels in the general case. For the restricted formulation considered in this chapter, however, generalizations of these algorithms may be designed because of the possibility to explicit the feature space of the kernel.

Another direction that we want to explore is the introduction of additional linguistic features related to the morphology and the semantic content of the words. Coming back to the illustration of section 6.2.4, semantic features could help detect that `Paris` and `London` both refer to cities, and morphological features could indicate that the two pronouns `he` and `she` are singular, and third person. However, the introduction of such features raises new questions because they typically do not consist of a single and discrete tag, which was the basic assumption of this work. Indeed, the morphological description of a word usually comes as a set of tags whose number and type can vary according to its part of speech. For instance, nouns and adjectives are typically characterized by `CASE`, `NUMBER`, and `GENDER` tags, verbs usually have additional `TENSE` and `MOOD` tags, and adverbs have no morphological variations.[9] Similarly, a semantic representation of words can, for instance, be based

9. Note that this type of morphological information is highly language-dependent.

on WordNet[10] synsets, on automatically derived word classes (Brown et al., 1992), or on their projection onto a latent semantic space (Landauer et al., 1998; Bengio et al., 2003). It should be clear that although this chapter focused on linguistic factors assigning a single and discrete tag to words, any source of information can be considered provided an appropriate kernel k_Σ is available. Different alternatives are considered in Costa et al. (2006) and Giuliano et al. (2006) to define kernels between sets of WordNet synsets for instance. Although this is most likely incompatible with the use of the above-mentioned linear complexity algorithms, factored kernels offer a natural and sound formalism to integrate such higher-level word representations.

Acknowledgments

This work was supported by the IST Programme of the European Community under the SMART project, IST-2005-033917, and under the PASCAL Network of Excellence, IST-2002-506778. We thank Matti Vuorinen and Juho Rousu for preparing the data set used in this paper, and the anonymous reviewers for their useful comments.

Appendix: Proof of Proposition 6.1

We detail the feature space interpretation of the factored kernel (Eq. (6.6)):

$$k_n^{\text{fact}}(x,y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \lambda^{g(s_x)+g(s_y)} \prod_{i=1}^{n} \sum_{d=1}^{p} w_d \mathbf{1}(s_x^{(d)}[i] = s_y^{(d)}[i]).$$

First, by a simple distributivity argument, we remove the summation over word factors (indexed by d) from the product over subsequences indices (indexed by i) and replace it by an exhaustive summation according to a vector of factor indices $\mathbf{d} = [d_1, \ldots, d_n] \in \{1, p\}^n$:

$$k_n^{\text{fact}}(x,y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \lambda^{g(s_x)+g(s_y)} \sum_{\mathbf{d}} \prod_{i=1}^{n} w_{d_i} \mathbf{1}(s_x^{(d_i)}[i] = s_y^{(d_i)}[i]).$$

Then, using the converse operation to that invoked at the very beginning of section 6.2.3 in the derivation of the soft-matching kernel, we replace the product of binary functions checking whether the n pairs of word factors $(s_x^{(d_i)}[i], s_y^{(d_i)}[i])$ are identical

10. http://wordnet.princeton.edu/

by a binary function checking whether the two sequences of word factors are globally identical:

$$k_n^{\text{fact}}(x,y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \lambda^{g(s_x)+g(s_y)} \sum_{\mathbf{d}} w(\mathbf{d}) \mathbf{1}(s_x^{(\mathbf{d})} = s_y^{(\mathbf{d})}),$$

with $w(\mathbf{d}) = \prod_{d=1}^{n} w_{d_i}$.

Using the definition of *composite subsequences* of length n associated with the sequence x introduced in section 6.2.3:

$$\mathcal{S}_n^*(x) = \big\{ x^{(\mathbf{d})}[\mathbf{i}] = x^{(d_1)}[i_i] \dots x^{(d_n)}[i_n], \\ \mathbf{i} \in \{1,\dots,|x|\}^n, \mathbf{d} \in \{1,\dots,p\}^n, i_1 < i_2 < \dots < i_n \big\},$$

we can include the summation over the vector $\mathbf{d}$ of word factors within the summation over subsequences themselves:

$$k_n^{\text{fact}}(x,y) = \sum_{s_x \in \mathcal{S}_n^*(x)} \sum_{s_y \in \mathcal{S}_n^*(y)} \lambda^{g(s_x)+g(s_y)} w(s_x) \mathbf{1}(s_x = s_y),$$

where for $s_x = x^{(\mathbf{d})}[\mathbf{i}] \in \mathcal{S}_n^*(x)$ and $s_y = y^{(\mathbf{d}')}[\mathbf{i}'] \in \mathcal{S}_n^*(y)$,

$$\mathbf{1}(s_x = s_y) = \begin{cases} 1 \text{ if } \mathbf{d} = \mathbf{d}' \text{ and } x^{(d_k)}[i_k] = y^{(d_k)}[i'_k] \text{ for k=1,...,n} \\ 0 \text{ otherwise,} \end{cases}$$

and for $s_x = x^{(\mathbf{d})}[\mathbf{i}]$, $w(s_x) = \prod_{i=1}^{n} w_{d_i}$.

Finally, because $\mathbf{1}(s_x = s_y) = 1$ implies $w(s_x) = w(s_y)$ for the pair of composite subsequences (s_x, s_y), we can write the above equation as

$$k_n^{\text{fact}}(x,y) = \sum_{s_x \in \mathcal{S}_n^*(x)} \sum_{s_y \in \mathcal{S}_n^*(y)} \lambda^{g(s_x)+g(s_y)} \sqrt{w(s_x)} \sqrt{w(s_y)} \mathbf{1}(s_x = s_y),$$

which leads to an expression similar to that of Eq. (6.4). The kernel can consequently be written as a standard inner product $k_n^{\text{fact}}(x,y) = \langle \phi(x), \phi(y) \rangle$, where

- $\phi(x) = (\phi_u(x))_{u \in \Sigma}$ with $\Sigma = \left(\bigcup_{d=1}^{p} \Sigma_d \right)^n$ is the set of all possible composite sequences of length n,
- $\phi_u(x) = \sum_{s_x \in \mathcal{S}_n^*(x)} \mathbf{1}(s_x = u) \lambda^{g(s_x)} \sqrt{w(s_x)}$.

The proof is concluded by noting that for $u \in \prod_{i=1}^{n} \Sigma_{f_i(u)}$ and $s_x = x^{(\mathbf{d})}[\mathbf{i}]$, the equality $\mathbf{1}(s_x = u) = 1$ implies $d_i = f_i(u)$ for $i = 1,\dots,n$, and therefore $w(s_x) = \prod_{i=1}^{n} w_{d_i} = \prod_{i=1}^{n} w_{f_i(u)}$.

II Machine Translation

7 Toward Purely Discriminative Training for Tree-Structured Translation Models

Benjamin Wellington
Joseph Turian
I. Dan Melamed

7.1 Introduction

Discriminative training methods have recently led to significant advances in the state of the art of machine translation (MT). Another promising trend is the incorporation of syntactic information into MT systems. Combining these trends is difficult for reasons of system complexity and computational complexity. This study makes progress toward a syntax-aware MT system whose every component is trained discriminatively. Our main innovation is an approach to discriminative learning that is computationally efficient enough for large statistical MT systems, yet whose accuracy on translation subtasks is near the state of the art.

Our approach to predicting a translation string is to predict its parse tree, and then read the string off the tree. Predicting a target tree given a source tree is equivalent to predicting a synchronous tree (*bitree*) that is consistent with the source tree. Our method for training tree transducers was to train an inference engine to predict bitrees. The *inference engine* employs the traditional AI technique of predicting a structure by searching over possible sequences of inferences, where each inference predicts a part of the eventual structure. Thus, to train a model for predicting bitrees, it is sufficient to train it to predict correct inferences. However, unlike most approaches employed in natural language processing (NLP), the proposed method makes no independence assumptions: the function that evaluates each inference can use arbitrary information not only from the input but also from all previous inferences.

Let us define some terms to help explain how our algorithm predicts a tree. An *item* is a node in the tree. Every *state* in the search space consists of a set of items, representing nodes that have been inferred since the algorithm started. States whose items form a complete tree are final states. An *inference* is a (state, item) pair, i.e., a state and an item to be added to it. Each inference represents a transition from one state to another. A state is *correct* if it is possible to infer zero or more items

to obtain the final state that corresponds to the training data tree. Similarly, an inference is correct if it leads to a correct state.

Given input s, the inference engine searches the possible complete trees $T(s)$ for the tree $\hat{t} \in T(s)$ that has minimum *cost* $C_\Theta(t)$ under model Θ:

$$\hat{t} = \arg\min_{t \in T(s)} C_\Theta(t) = \arg\min_{t \in T(s)} \left(\sum_{j=1}^{|t|} c_\Theta(i_j) \right). \tag{7.1}$$

The i_j are the inferences involved in constructing tree t. $c_\Theta(i)$ is the cost of an individual inference i. The number of states in the search space is typically exponential in the size of the input. The freedom to compute $c_\Theta(i)$ using arbitrary nonlocal information from anywhere in inference i's state precludes exact solutions by ordinary dynamic programming. We know of two effective ways to approach such large search problems. The first is to restrict the order in which items can be inferred, for example, bottom-up. The second is to make the simplifying assumption that the cost of adding a given item to a state is the same for all states. Under this assumption, the fraction of any state's cost due to a particular item can be computed just once per item, instead of once per state. However, in contrast to traditional context-free parsing algorithms, that computation can involve context-sensitive features.

7.2 Related Work

A probabilistic treatment of Eq. (7.1) uses $C_\Theta(t) = \Pr(t|s)$, the probability that a text s in a source language will translate into a text t in the target language. Brown et al. (1993) used Bayes's rule to decompose this posterior into a language model $\Pr(t)$ and a translation model $\Pr(s|t)$:

$$\Pr(t|s) = \frac{\Pr(t)\Pr(s|t)}{\sum_t \Pr(t)\Pr(s|t)} \propto \Pr(t)\Pr(s|t). \tag{7.2}$$

The problem with this model is that finding the maximum probability tree is difficult. Foster (2000) instead directly modeled the posterior $\Pr(t|s)$ using a chain-rule expansion:

$$\Pr(t|s) = \prod_{i=1}^{|t|} \Pr(t_i|t_{i-1}, \ldots, t_1, s), \tag{7.3}$$

which is a special case of Eq. (7.1). In Foster (2000), each inference t_i adds the ith token in t. Hence, the translation text was predicted one word at a time, where each prior word in the translation text was known at the point of guessing the next. Foster was the first to build a large-scale log-linear translation model. More specifically, Foster modeled the probability of individual decisions $\Pr(t_i|t_{i-1}, \ldots, t_1, s)$ using an

unregularized locally normalized log-linear model (Berger et al., 1996). There are several differences between our approach and Foster's:

- His learning method was unregularized. Unregularized models tend to overfit, especially in the presence of useful features that always occur with a certain target output. Our learning approach uses ℓ_1-regularization.
- To reduce overfitting, Foster performed automatic feature selection as a separate step before training. This approach can inadvertently remove important features. In contrast, our learning method does automatic feature selection during training.
- Foster's model is induced over just the *atomic* features he defined. Our approach achieves more powerful modeling by combining *atomic* features in useful ways.
- Foster's model was locally normalized, i.e., there was conservation of mass among the candidate inferences at each state. Local normalization can cause label bias (Lafferty et al., 2001). Our models are unnormalized, which avoids label bias (Turian, 2007, sections 4.1 and 7.1).
- The inference process of Foster is to infer tokens in the target string, one by one. Our inference process is to build a target-side syntax tree bottom-up, one node at a time, which can be viewed as translation by parsing (Melamed, 2004).

Och and Ney (2002) described a generalization of the approach of Brown et al. (1993), which allowed different information sources in the form of features. Och and Ney's approach was to perform minimum error-rate training (MERT), using the loss function of one's choice. One such approach was that of Koehn et al. (2003), who built a phrase-based statistical MT system that worked by calculating the probability that one phrase would translate to another and compiling those probabilities into a phrase table. In their model, discriminative methods were used to tune a handful of metaparameters, namely the parameters of (1) the log probability of the output sentence t under a language model, (2) the score of translating a source phrase into a target phrase based on the phrase table just described, (3) a distortion score, and (4) a length penalty. Several authors have even applied this method to syntax-aware systems (Chiang, 2005; Quirk et al., 2005). However, while MERT was a significant step forward in applying discriminative training to statistical MT, it can be performed reliably on only a small number of parameters. For this reason, it cannot be used with approaches like that of Foster (2000), in which there are tens of thousands of features, let alone approaches like ours involving millions of features. Another limitation of MERT is that the submodels may or may not be optimized for the same objective as the metaparameters. On a task like machine translation between common languages, where training data is abundant, a more elegant and more accurate system might be obtained by training all parts of the system with a single regularized objective.

More recent attempts to take full advantage of discriminative methods are the perceptron-based approaches of Tillmann and Zhang (2005) and Cowan et al. (2006), and the perceptron-like approach of Liang et al. (2006). All of these took advantage of discriminative methods to learn parameters for models with millions

of features. All of them were limited by the instability of the perceptron algorithm. Tillmann and Zhang (2006) used a more principled approach, involving stochastic gradient descent over a regularized loss function in a reranking framework. From a machine-learning point of view, this latter work was perhaps the most advanced (Bottou and Bousquet, 2008). However, all of these recent efforts were applied only to finite-state translation models.

Our work is the first, to our knowledge, to apply principled discriminative learning directly to the millions of parameters of a tree-structured translation model.

7.3 Learning Method

The goal of learning is to induce c_Θ, which is the cost function for individual inferences. c_Θ is used in Eq. (7.1) to find the minimum cost output structure. An important design decision in learning the inference cost function c_Θ is the choice of feature set. Given the typically large number of possible features, the learning method must satisfy two criteria. First, it must be able to learn effectively even if the number of irrelevant features is exponential in the number of examples. It is too time-consuming to manually figure out the right feature set for such problems. Second, the learned function must be sparse. Otherwise, it would be too large for the memory of an ordinary computer, and therefore impractical. In this section, we describe how c_Θ is induced.

7.3.1 Problem Representation

The training data used for translation initially comes in the form of bitrees. These gold-standard trees are used to generate training examples, each of which is a candidate inference: starting at the initial state, we randomly choose a sequence of correct inferences that lead to the (gold-standard) final state.[1] All the candidate inferences that can possibly follow each state in this sequence become part of the training set. The vast majority of these inferences will lead to incorrect states, which makes them negative examples. More specifically, the training set I consists of candidate inferences. Each inference i in the training set I can be expanded to a tuple $\langle X(i), y(i)\rangle$. $X(i)$ is a feature vector describing i, with each element in $\{0, 1\}$. We will use $X_f(i)$ to refer to the element of $X(i)$ that pertains to feature f. $y(i) = +1$ if i is correct, and is $y(i) = -1$ if not.

An advantage of this method of generating training examples is that it does not require a working inference engine and can be run prior to any training. A

1. The order of inferences is nondeterministic, so there may be many paths to the gold-standard final state. Although the monolingual experiments of Turian (2007, section 6.4) indicate that having a deterministic inference logic is preferable, for the task of tree transduction it is not clear if there is a sensible canonical ordering.

disadvantage of this approach is that it does not teach the model to recover from mistakes. This is because example generation does not use the model in any way. Our training set is identical to the inferences that would be scored during search using an oracle cost function. The oracle cost function $\hat{c}(i)$ is ∞ if $y(i) = -1$ and 0 if $y(i) = +1$. If we wanted to improve accuracy by teaching the inference engine to recover from its mistakes, we could use inference during example generation and iterate training. With a deterministic logic, using the oracle cost function to generate training examples is similar to the first iteration of SEARN (Daumé et al., 2005, 2006).

7.3.2 Objective Function

The training method induces a real-valued inference evaluation function $h_\Theta(i)$. We will describe how to transform $h_\Theta(i)$ to $c_\Theta(i)$ in Eq. (7.5). In this chapter, h_Θ is a linear model parameterized by a real vector Θ, which has one entry for each feature f: $h_\Theta(i) = \Theta \cdot X(i) = \sum_f \Theta_f \cdot X_f(i)$. The sign of $h_\Theta(i)$ predicts the y-value of i and the magnitude gives the confidence in this prediction. The training procedure adjusts Θ to minimize the expected risk R_Θ over training set I. R_Θ is the *objective* or *risk* function, which is the sum of *loss* function L_Θ and *regularization* term Ω_Θ. The loss measures the extent to which the model underfits the training data. The regularization term is a measure of model complexity, and is used to avoid overfitting the training data. We use the log-loss and ℓ_1-regularization, so we have

$$\begin{aligned} R_\Theta(I) &= L_\Theta(I) + \Omega_\Theta = \left[\sum_{i \in I} l_\Theta(i)\right] + \Omega_\Theta \\ &= \left[\sum_{i \in I} [\ln(1 + \exp(-\mu_\Theta(i)))]\right] + \left[\lambda \cdot \sum_f |\Theta_f|\right]. \end{aligned} \tag{7.4}$$

λ is a parameter that controls the strength of the regularizer and $\mu_\Theta(i) = y(i) \cdot h_\Theta(i)$ is the *margin* of example i. The tree cost C_Θ (Eq. (7.1)) is obtained by computing the objective function with $y(i) = +1$ for every inference in the tree, and treating the penalty term Ω_Θ as constant:

$$c_\Theta(i) = l_\Theta(\langle X(i), y = +1 \rangle) = \ln(1 + \exp(-h_\Theta(i))). \tag{7.5}$$

The experiments in section 7.4.2 motivate the use of ℓ_1 regularization instead of ℓ_2 regularization.

7.3.3 Minimizing the Risk

We minimize the risk R_Θ using a form of forward stagewise additive modeling, a procedure Guyon and Elisseeff (2003) refer to as *embedded feature selection*. At

each iteration in training, we pick one or more parameters of the model to adjust (*feature selection*), then adjust these parameters (*model update*).

Feature Selection

We want to choose parameters that allow us to decrease the risk quickly. We define the *gain* of a feature f as

$$\begin{aligned} G_\Theta(I; f) = \max(&\text{decrease in risk as we increase } f\text{'s parameter value,} \\ &\text{decrease in risk as we decrease } f\text{'s parameter value,} \\ &\text{decrease in risk if we leave } f\text{'s parameter value unchanged).} \end{aligned} \tag{7.6}$$

More specifically, we pick the features with the steepest gradients (Mason et al., 1999, 2000; Perkins et al., 2003):

$$G_\Theta(I; f) = \max\left(\lim_{\epsilon \to 0^+} \frac{\partial\, -R_\Theta}{\partial \Theta_f}(\Theta_f + \epsilon),\ \lim_{\epsilon \to 0^-} \frac{\partial\, -R_\Theta}{\partial\, -\Theta_f}(\Theta_f + \epsilon),\ 0\right). \tag{7.7}$$

The limits are taken from above and below, respectively. This analysis technique allows us to determine the gain for any continuous function, regardless of whether it is continuously differentiable or not. To determine the gain function in Eq. (7.7) for a particular risk, we consider two cases:

- If we are using the ℓ_1 penalty ($\Omega_\Theta = \sum_{f \in F} |\Theta_f|$) and $\Theta_f = 0$, then we are at the gradient discontinuity and we have

$$G_\Theta(I; f) = \max\left(\left|\frac{\partial L_\Theta}{\partial \Theta_f}\right| - \lambda,\ 0\right). \tag{7.8}$$

Observe that unless the magnitude of the gradient of the empirical loss $|\partial L_\Theta(I)/\partial \Theta_f|$ exceeds the penalty term λ, the gain is zero and the risk increases as we adjust parameter Θ_f away from zero in either direction. In other words, the parameter value is trapped in a "corner" of the risk. In this manner the polyhedral structure of the ℓ_1 norm tends to keep the model sparse (Riezler and Vasserman, 2004). Dudík et al. (2007) offer another perspective, pointing out that ℓ_1 regularization is "truly sparse": if some feature's parameter value is zero when the risk is minimized, then the optimal parameter value will remain at zero even under slight perturbations of the feature's expected value and of the regularization penalty. However, if the gain is nonzero, $G_\Theta(I; f)$ is the magnitude of the gradient of the risk as we adjust Θ_f in the direction that reduces R_Θ.

- If we are using the ℓ_2 penalty ($\Omega_\Theta = \sum_{f \in F} {\Theta_f}^2$), then we have

$$G_\Theta(I; f) = \left|\frac{\partial L_\Theta}{\partial \Theta_f} + \lambda \cdot 2 \cdot \Theta_f\right|. \tag{7.9}$$

The ℓ_2-regularization term disappears when $\Theta_f = 0$, and so—for unused features—ℓ_2-regularized feature selection is indistinguishable from unregularized feature selection. The only difference is that the ℓ_2-penalty term reduces the magnitude of the optimal parameter setting for each feature. This is why ℓ_2-regularization typically leads to models where many features are active (nonzero).

Let us define the *weight* of an example i under the current model as the rate at which loss decreases as the margin of i increases:

$$w_\Theta(i) = -\frac{\partial l_\Theta(i)}{\partial \mu_\Theta(i)} = \frac{1}{1 + \exp(\mu_\Theta(i))}. \tag{7.10}$$

To determine $\frac{\partial L_\Theta}{\partial \Theta_f}$, the gradient of the unpenalized loss L_Θ with respect to the parameter Θ_f of feature f, which is used in Eq. (7.8) and (7.9), we have

$$\begin{aligned} \frac{\partial L_\Theta(I)}{\partial \Theta_f} &= \sum_{i \in I} \frac{\partial l_\Theta(i)}{\partial \Theta_f} = \sum_{i \in I} \frac{\partial l_\Theta(i)}{\partial \mu_\Theta(i)} \cdot \frac{\partial \mu_\Theta(i)}{\partial \Theta_f} \\ &= -\sum_{i \in I} w_\Theta(i) \cdot [y(i) \cdot X_f(i)] = -\sum_{\substack{i \in I: \\ X_f(i)=1}} w_\Theta(i) \cdot y(i). \end{aligned} \tag{7.11}$$

Turian (2007, section 4.2) contains detailed derivations of the gain functions in this section, as well as more discussion and comparison to related learning approaches.

Compound Feature Selection

Although h_Θ is just a linear discriminant, it can nonetheless learn effectively if feature space F is high-dimensional. Features encode information about the inference in question. A priori, we define only a set A of simple atomic features (sometimes also called *attributes* or *primitive* features).

Feature *construction* or *induction* methods are learning methods in which we induce a more powerful machine than a linear discriminant over just the attributes, and the power of the machine can be data-dependent.

Our learner induces *compound* features, each of which is a conjunction of possibly negated atomic features. Each atomic feature can have one of three values (yes/no/don't care), so the compound feature space F has size $3^{|A|}$, exponential in the number of atomic features. Each feature selection iteration selects a set of *domain-partitioning* features. Domain-partitioning features $\tilde{F}$ cover the domain and are nonoverlapping for every possible inference i in the space of inferences:

cover : $\exists f \in \tilde{F}$ s.t. $X_f(i) = 1$.

nonoverlapping : $\forall f, f' \in \tilde{F}$, if $X_f(i) = 1$ and $X_{f'}(i) = 1$ then $f = f'$.

In other words, $\tilde{F}$ partitions the inference space if for any i, there is a unique feature f in the partition $\tilde{F}$ such that $X_f(i) = 1$, and $X_{f'}(i) = 0$ for all other features.

One way to choose a set of domain-partitioning compound features is through greedy splitting of the inference space. This is the approach taken by decision trees,

for some particular splitting criterion. Since we wish to pick splits that allow us to reduce risk, our splitting criterion uses the gain function. In particular, assume that we have two different two-feature partitions of some subspace of the inference space. The features f_1 and f_2 are one partition, and $f_1{}'$ and $f_2{}'$ are the other. We treat the gain of each partition as $G_\Theta(f_1) + G_\Theta(f_2)$ and $G_\Theta(f_1{}') + G_\Theta(f_2{}')$, respectively, and prefer the partition with higher gain. The work of Mason et al. (1999, 2000), who studied a related problem, motivate this choice theoretically using a first-order Taylor expansion.

Model Update

After we have selected one or more high-gain features, we update the model. *Parallel* update methods can adjust the parameter values of overlapping features to minimize the risk. *Stagewise* or *sequential* update methods adjust the parameter values of nonoverlapping features to minimize the risk, each of which can be optimized independently, e.g., using line search. Since domain-partitioning features are nonoverlapping, we use sequential updates to choose parameter values.

Boosting Regularized Decision Trees

To summarize, our approach to minimizing the risk is divided into feature selection and model update. Feature selection is performed using gradient descent in the compound feature space through a greedy splitting procedure to partition the inference space. Model update is performed using a sequential update method.

Our specific implementation of this risk minimization approach is to boost an ensemble of confidence-rated decision trees. This boosted ensemble of confidence-rated decision trees represents Θ. We write $\Delta\Theta(t)$ to represent the parameter values chosen by tree t, and for ensemble T we have $\Theta = \sum_{t \in T} \Delta\Theta(t)$. Each internal node is split on an atomic feature. The path from the root to each node n in a decision tree corresponds to a *compound* feature f, in which case we write $\varphi(n) = f$. An example i percolates down to node n iff $X_{\varphi(n)} = 1$. Each leaf node n keeps track of delta-parameter value $\Delta\Theta_{\varphi(n)}(t)$. To score an example i using a decision tree, we percolate the example down to a leaf n and return confidence $\Delta\Theta_{\varphi(n)}(t)$. The score $h_\Theta(i)$ given to an example i by the whole ensemble is the sum of the confidences returned by all trees in the ensemble.

Figure 7.1 presents our training algorithm. At the beginning of training, the ensemble is empty, $\Theta = \mathbf{0}$, and λ is set to ∞. We grow the ensemble until the risk cannot be further reduced for the current choice of λ. So for some choice of penalty factor λ', our model is the ensemble up until when λ was decayed below λ'. We then relax the regularizer by decreasing λ and continue training. We use the decay factor $\eta = 0.9$ as our *learning rate*. In this way, instead of choosing the best λ heuristically, we can optimize it during a single training run. This approach of progressively relaxing the regularizer during a single training run finds the entire

Algorithm 7.1 Outline of the training algorithm.

procedure TRAIN(I)
  ensemble $\leftarrow \emptyset$
  regularization parameter $\lambda \leftarrow \infty$
  while not converged **do**
    $g \leftarrow \max_{a \in A}(|\partial L_\Theta / \partial \Theta_\emptyset|, |\partial L_\Theta / \partial \Theta_a|, |\partial L_\Theta / \partial \Theta_{\neg a}|)$ ▷ Find the best unpenalized gain of any root split
    $\lambda \leftarrow \min(\lambda, \eta \cdot g)$ ▷ Maybe decay the penalty parameter so that training can progress
    $\Delta\Theta(t) \leftarrow \text{BUILDTREE}_\Theta(\lambda, I)$
    if $R_{\Theta + \Delta\Theta(t)} < R_\Theta + \epsilon$ **then** ▷ If we have reduced loss by some threshold
      $\Theta \leftarrow \Theta + \Delta\Theta(t)$ ▷ Then keep the tree and update the model
    else ▷ Otherwise, we have converged for this choice of λ
      $\lambda \leftarrow \eta \cdot \lambda$ ▷ Decay the penalty parameter

procedure BUILDTREE$_\Theta$(λ, I)
  $t \leftarrow$ root node only
  while some leaf in t can be split **do** ▷ See Eq. (7.13)
    split the leaf to maximize gain ▷ See Eq. (7.12)
  percolate every $i \in I$ to a leaf node
  for each leaf n in t **do**
    $\Delta\Theta_{\varphi(n)}(t) \leftarrow \arg\min_{\Delta\Theta_{\varphi(n)}(t)}(R_{\Theta + \Delta\Theta_{\varphi(n)}(t)}(I_{\varphi(n)}))$ ▷ Choose $\Delta\Theta_{\varphi(n)}(t)$ to minimize R using a line search
  return $\Delta\Theta(t)$

regularization path (e.g., Hastie et al., 2004) and is a form of continuation method for global optimization (e.g., Chapelle et al., 2006).

Each invocation of BUILDTREE has several steps. First, we choose some compound features that will allow us to decrease the risk function. We do this by building a decision tree whose leaf node paths represent the chosen compound features. Second, we confidence-rate each leaf to minimize the risk over the examples that percolate down to that leaf. Finally, we return the decision tree. TRAIN appends it to the ensemble and update parameter vector Θ accordingly. In this manner, compound feature selection is performed incrementally during training, as opposed to a priori.

The construction of each decision tree begins with a sole leaf node, the root node, which corresponds to a dummy "always true" feature $\emptyset$. By "always true", we mean that $X_\emptyset(i) = 1$ for any example i. We recursively split leaf nodes by choosing the best atomic splitting feature that will allow us to increase the gain. Specifically, we consider splitting each leaf node n using atomic feature $\hat{a}$, where $f = \varphi(n)$ and

$$\hat{a} = \arg\max_{a \in A} \left[G_\Theta(I; f \wedge a) + G_\Theta(I; f \wedge \neg a) \right]. \tag{7.12}$$

(We use f to mean an actual feature, not a feature index.) Splitting using $\hat{a}$ would create children nodes n_1 and n_2, with $\varphi(n_1) = f \wedge \hat{a}$ and $\varphi(n_2) = f \wedge \neg\hat{a}$. We split

Table 7.1 Data sizes in 000's

	sent.	English words		French words	
	pairs	types	tokens	types	tokens
training1	10	11	210	14	232
training2	100	29	2100	38	2300
tuning	1	3.5	21	4.2	23
devel	1	3.5	21	4.1	23
test	1	3.5	21	4.1	23

node n using $\hat{a}$ only if the total gain of these two children exceeds the gain of the unsplit node, i.e., if

$$G_\Theta(I; f \wedge \hat{a}) + G_\Theta(I; f \wedge \neg\hat{a}) > G_\Theta(I; f). \tag{7.13}$$

Otherwise, n remains a leaf node of the decision tree, and $\Theta_{\varphi(n)}$ becomes one of the values to be optimized during the parameter update step.

Parameter update is done sequentially on only the most recently added compound features, which correspond to the leaves of the new decision tree. After the entire tree is built, we percolate each example down to its appropriate leaf node. As indicated earlier, a convenient property of decision trees is that the leaves' compound features are mutually exclusive, so their parameters can be directly optimized independently of each other. We use a line search to choose for each leaf node n the parameter $\Theta_{\varphi(n)}$ that minimizes the risk over the examples in n.

7.4 Experiments

7.4.1 Data

The data for our experiments came from the English and French components of the EuroParl corpus (Koehn, 2005). From this corpus, we extracted sentence pairs where both sentences had between 5 and 40 words, and where the ratio of their lengths was no more than 2:1. We then extracted disjoint training, tuning, development, and test sets. The tuning, development, and test sets were 1000 sentence pairs each. For some experiments we used 10,000 sentence pairs of training data; for others we used 100,000. Descriptive statistics for these corpora are in table 7.1.

We parsed the English half of the training, tuning, development, and test bitexts using Dan Bikel's parser (Bikel, 2004), which was trained on the Penn treebank (Marcus et al., 1993). On each of our two training sets, we induced word alignments using the default configuration of GIZA++ (Och and Ney, 2003). The training set word alignments and English parse trees were fed into the default French-English

hierarchical alignment algorithm distributed with the GenPar system (Burbank et al., 2005), to produce binarized tree alignments.

7.4.2 Word Transduction

Our first set of experiments evaluated our approach on the task of translating individual words from English to French. The input was a single English word, which we'll call the "focus" word, along with a vector of features (described below). The output was a single French word, possibly NULL. The proposed translation was compared to a gold-standard translation.

The gold-standard word pairs that we used for this task were extracted from the tree alignments described above. Thus, the gold standard was a set of GIZA++ Viterbi word alignments filtered by a tree cohesion constraint. Regardless of whether they are created manually or automatically, word alignments are known to be highly unreliable. This property of the data imposed a very low artificial ceiling on all of our results, but it did not significantly interfere with our goal of controlled experiments to compare learning methods. To keep our measurements consistent across different training data sizes, the word alignments used for testing were the ones induced by GIZA++ when trained on the larger training set. The number of trials was equal to the number of source words for which GIZA++ predicts an alignment.

In contrast to Vickrey et al. (2005), we did not allow multiword "phrases" as possible translations. Our hypothesis, based on some evidence in section 7.4.3 and in Quirk and Menezes (2006), is that the best MT systems of the future will not need to deal in such objects. In our study, phrases might have raised our absolute scores, but they would have confounded our understanding of the results. Our experiment design also differs from Vickrey et al. (2005) in that we trained classifiers for *all* words in the training data.[2] There were 161K word predictions in the smaller (10,000 sentence pairs) training set, 1866K in the larger training set, 17.8K predictions in the tuning set, 14.2K predictions in the development set, and 17.5K predictions in the test set.

Using the smaller training set and guessing the most frequent translation of each source word achieves a baseline accuracy of 47.54% on the development set. With this baseline, we compared three methods for training word transducers on the word alignments described above. The first was the method described in section 7.3 for inducing ℓ_1-regularized log-linear models over the compound feature space. The second method was similar to Vickrey et al. (2005): induce ℓ_2-regularized log-linear models over the *atomic* feature space.[3] The third method was LaSVM (Bordes et al., 2005), an online SVM algorithm designed for large data sets.

---

2. David Vickrey (personal communication) informed us that Vickrey et al. (2005) omitted punctuation and function words, which are the most difficult in this task.

3. We do not induce ℓ_2-regularized log-linear models over the *compound* feature space because Turian (2007, section 6.4) found that these models were too large even for monolingual parsing experiments. To induce ℓ_2-regularized log-linear models over the

Table 7.2 Percent accuracy on the development set and sizes of word-to-word classifiers trained on 10K or 100K sentence pairs. The feature sets used were (W)indow, (D)ependency, and (C)o-occurrence. ℓ_1 size is the number of compound feature types.

	W	W+D	W+C	W+D+C
10,000 training sentences — baseline = 47.54				
ℓ_2	54.09	54.33	52.36	52.88
ℓ_1	53.96	54.13	53.29	53.75
LaSVM	53.38	51.93	49.13	50.71
pruned ℓ_2	47.37	46.01	46.68	45.01
ℓ_1 size	54.1K	41.7K	37.8K	38.7K
ℓ_2 size	1.67M	2.51M	5.63M	6.47M
pruned ℓ_2 size	54.1K	41.7K	37.8K	38.7K
100,000 training sentences — baseline = 51.94				
ℓ_1	62.00	62.42	61.98	62.40
ℓ_1 size	736K	703K	316K	322K

For each training method, we experimented with several kinds of features, which we call "window," "co-occurrence," and "dependency." Window features included source words and part-of-speech (POS) tags within a two-word window around the focus word, along with their relative positions (from −2 to +2). Co-occurrence features included all words and POS tags from the whole source sentence, without position information. Dependency features were compiled from the automatically generated English parse trees. The dependency features of each focus word were

- the label of its maximal projection (i.e., the highest node that has the focus word as its lexical head, which might be a leaf, in which case that label is a POS tag);
- the label and lexical head of the parent of the maximal projection;
- the label and lexical head of all dependents of the maximal projection;
- all the labels of all head-children (recursively) of the maximal projection.

The window features were present in all experimental conditions. The presence/absence of co-occurrence and dependency features yielded four "configurations."

Using each of these configurations, each training method produced a confidence-rating binary classifier for each translation of each English word seen in the training data. In all cases, the test procedure was to choose the French word predicted with the highest confidence. All methods, including the baseline, predicted NULL for source words that were not seen in training data.

Table 7.2 shows the size and accuracy of all three methods on the development set, after training on 10,000 sentence pairs, for each of the four configurations. The best configurations of the two logistic regression methods far exceed the baseline,

atomic feature space, we used MegaM by Hal Daumé, available at `http://www.cs.utah.edu/~hal/megam/`

but otherwise they were statistically indistinguishable. The accuracy of LaSVM was similar to the regression methods when using only the window features, but it was significantly worse with the larger feature sets.

More interesting were the differences in model sizes. The ℓ_2-regularized models were bigger than the ℓ_1-regularized models by two orders of magnitude. The ℓ_2-regularized models grew in size to accommodate each new feature type. In contrast, the ℓ_1-regularized models *decreased* in size when given more useful features, without significantly losing accuracy. This trend was even stronger on the larger training set, where more of the features were more reliable.

The size of models produced by LaSVM grew linearly with the number of examples, because for source words like "the," about 90% of the examples became support vectors. This behavior makes it infeasible to scale up LaSVM to significantly larger data sets, because it would need to compare each new example to all support vectors, resulting in near-quadratic runtime complexity.

To scale up to 100,000 sentence pairs of training data with just the window features, the ℓ_2 classifiers would need about 25 billion parameters, which could not fit in the memory of our computers. To make them fit, we could set all but the largest feature parameters to zero. We tried this on 10,000 training sentence pairs. The number of features allowed to remain active in each ℓ_2 classifier was the number of active features in the ℓ_1 classifier. Table 7.2 shows the accuracy of these "pruned" ℓ_2-regularized classifiers on the development set, when trained on 10,000 sentence pairs. With the playing field leveled, the ℓ_1 classifiers were far more effective.

In preliminary experiments, we also tried perceptron-style updates, as suggested by Tillmann and Zhang (2005). However, for reasons given by Tewari and Bartlett (2005), the high-entropy decisions involved in our structured prediction setting often prevented convergence to useful classifiers. Likewise, Christoph Tillmann informed us (personal communication) that, to ensure convergence, he had to choose features very carefully, even for his finite-state MT system.

Regularization schemes that don't produce sparse representations seem unsuitable for problems on the scale of machine translation. For this reason, we used only ℓ_1-regularized log-loss for the rest of our experiments. Table 7.2 shows the accuracy and model size of the ℓ_1-regularized classifier on the development set, when trained on 100,000 sentence pairs, using each of the four configurations. Our classifier far exceeded the baseline. The test set results for the best models (window + dependency features) were quite close to those on the development set: 54.64% with the smaller training set, and 62.88% with the larger.

7.4.3 Bag Transduction

The word-to-word translation task is a good starting point, but any conclusions that we might draw from it are inherently biased by the algorithm used to map source words to target words in the test data. Our next set of experiments was on a task with more external validity – predict a translation for *each* source word in the test data, regardless of whether GIZA++ predicted an alignment for it. The

difficulty with this task, of course, is that we have no deterministic word alignment to use as a gold standard. Our solution was to pool the word translations in each source sentence and compare them to the bag of words in the target sentence. We still predicted exactly one translation per source word, and that translation could be NULL. Thus, the number of target words predicted for each source sentence was less than or equal to the number of words in that source sentence. The evaluation measures for this experiment were precision, recall, and F-measure, with respect to the bag of words in the test target sentence.

We compared the four configurations of our ℓ_1-regularized classifiers on this task to the most-frequent-translation baseline. We also evaluated a mixture model, where a classifier for each source word was chosen from the best one of the four configurations, based on that configuration's accuracy on that source word in the tuning data. As an additional gauge of external validity, we performed the same task using the best publicly available machine translation system (Koehn et al., 2003). This comparison was enlightening but necessarily unfair. As mentioned above, our long-term goal is to build a system whose every component is discriminatively trained to optimize the objective function. We did not want to confound our study with techniques such as "phrase" induction, word-class induction, nondiscriminatively trained target language models, etc. On the other hand, modern MT systems are designed for use with such information sources, and cannot be fairly evaluated without them. So, we ran Pharaoh in two configurations. The first used the default system configuration, with a target language model trained on the target half of the training data. The second allowed Pharaoh to use its phrase tables but without a target language model. This second configuration allowed us to compare the accuracy of our classifiers to Pharaoh specifically on the subtask of MT for which they were designed.

The results are shown in table 7.3. The table shows that our method far exceeds the baseline. Since we predict only one French target word per English source word, the recall of our bag transducer was severely handicapped by the tendency of French sentences to be longer than their English equivalents. This handicap is reflected in the one-to-one upper bound shown in the table. With a language model, Pharaoh's recall exceeded that of our best model by slightly less than this 13.7% handicap. However, we were surprised to discover that the bag transducer's precision was much higher than Pharaoh's when they compete on a more level playing field (without a language model), and higher even when Pharaoh was given a language model. Table 7.4 shows the accuracy of the best models on the test set. These results closely follow those on the development set.

This result suggests that it might not be necessary to induce "phrases" on the source side. Quirk and Menezes (2006) offer additional evidence for this hypothesis. After all, the main benefits of phrases on the source side are in capturing lexical context and local word reordering patterns. Our bag transducers capture lexical context in their feature vectors. Word order is irrelevant for bag transduction.

The main advantage of phrase-based models on this task is in proposing more words on the target side, which eliminates the one-to-one upper bound on recall.

Table 7.3 (P)recision, (R)ecall, and (F)-measure for bag transduction of the development set. The discriminative transducers were trained with ℓ_1 regularization.

	P	R	F
training on 10,000 sentence pairs			
baseline	48.09	41.26	44.42
window only	53.93	42.81	47.73
dependency	53.97	42.72	47.69
co-occurrence	53.31	42.45	47.26
co-oc. + dep.	53.58	42.67	47.50
mixture model	54.05	42.95	47.87
training on 100,000 sentence pairs			
baseline	51.91	40.79	45.68
window only	58.79	44.26	50.50
dependency	59.12	44.57	50.82
co-occurrence	59.06	44.36	50.67
co-oc. + dep.	59.19	44.60	50.87
mixture model	59.03	44.55	50.78
Pharaoh w/o LM	32.20	54.62	40.51
Pharaoh with LM	56.20	57.49	56.84
1-to-1 upper bound	100.00	86.31	92.65

Table 7.4 (P)recision, (R)ecall, and (F)-measure of bag transducers on the test set

	P	R	F
training on 10,000 sentence pairs			
dependency	54.36	42.75	47.86
mixture model	54.27	42.81	47.86
training on 100,000 sentence pairs			
co-oc. + dep.	59.49	44.19	50.71
mixture model	59.62	44.38	50.88
Pharaoh w/o LM	32.55	54.62	40.80
Pharaoh with LM	57.01	57.84	57.45

Another advantage is that phrase-based models require no parsing, and thus require no hand-annotated treebank to be available for training a parser for the source language. Note, however, that when training on 100,000 sentences, our best model not relying on parse tree information (the co-oc. model) was only a tiny bit less accurate than the best model overall (the co-oc. + dep. model).

7.4.4 Tree Transduction

We experimented with a simplistic tree transducer, which involves only two types of inference. The first type transduces leaves; the second type transduces internal

nodes. The transduction of leaves is exactly the word-to-word translation task described in section 7.4.2. Leaves that are transduced to NULL are deterministically erased. Internal nodes are transduced merely by permuting the order of their children, where one of the possible permutations it to retain the original order. This transducer is grossly inadequate for modeling real bitext (Galley et al., 2004): it cannot account for many kinds of noise and for many real translingual phenomena, such as head-switching and discontinuous constituents, which are important for accurate MT. It cannot even capture common phrasal translations such as *there is / il y a*. However, it is sufficient for controlled comparison of learning methods. The learning method will be the same when we use more sophisticated tree transducers. Another advantage of this experimental design is that it uses minimal linguistic cleverness and is likely to apply to many language pairs, in contrast to other studies of constituent/dependent reordering that are more language-specific (Xia and McCord, 2004; Collins et al., 2005).

To reduce data sparseness, each internal node with more than two children was binarized, so that the multiclass permutation classification for the original node was reduced to a sequence of binary classifications. This reduction is different from the usual multiclass reduction to binary: in addition to making the classifier binary instead of multiclass, the reduction decomposes the label so that some parts of it can be predicted before others. For example, without this reduction, a node with children $\langle A, B, C\rangle$ can be transduced to any of six possible permutations, requiring a six-class classifier. After binarization, the same six possible permutations can be obtained by first permuting $\langle A, B\rangle$, and then permuting the result with C, or by first permuting $\langle B, C\rangle$ and then permuting the result with A. This reduction eliminates some of the possible permutations for nodes with four or more children (Wu, 1997).

Our monolingual parser indicated which node is the head-child of each internal node. Some additional permutations were filtered out using this information: two sibling nodes that were *not* the head-children of their parent were not allowed to participate in a permutation until one of them was permuted with the head-child sibling. Thus, if C was the head-child in the previous example, then $\langle A, B\rangle$ could not be permuted first; $\langle B, C\rangle$ had to be permuted first, before permuting with A.

We search for the tree with minimum total cost, as specified in Eq. (7.1). We compared two models of inference cost—one generative and one discriminative.

The generative model was based on a top-down tree transducer (Comon et al., 2002) that stochastically generates the target tree given the source tree. The generative process starts by generating the target root given the source root. It then proceeds top-down, generating every target node conditioned on its parent and on the corresponding node in the source tree. Let π be the function that maps every node to its parent, and let η be the function that maps every target node to

Table 7.5 (P)recision, (R)ecall, and (F)-measure of transducers using 100,000 sentence pairs of training data

Source parser	Transduction model	Exponent 1			Exponent 2		
		P	R	F	P	R	F
generative	generative	51.29	38.30	43.85	22.62	16.90	19.35
generative	discriminative	59.89	39.53	47.62	26.94	17.78	21.42
discriminative	generative	50.51	37.76	43.21	22.04	16.47	18.85
discriminative	discriminative	**62.36**	39.06	**48.04**	**28.02**	17.55	**21.59**
	Pharaoh (w/o LM)	32.19	**54.62**	40.51	12.37	**20.99**	15.57

its corresponding source. If we view the target tree as consisting of nodes n with n_0 being the root node, then the probability of the target tree t is

$$\Pr(t) = \Pr(n_0|\eta(n_0)) \cdot \prod_{n \neq n_0 \in t} \Pr(n|\pi(n), \eta(n)). \tag{7.14}$$

For the generative model, the cost of an inference i is the negative logarithm of the probability of the node $n(i)$ that it infers: $l(i) = -\log \Pr[n(i)|\pi(n(i)), \eta(n(i))]$. We estimated the parameters of this transducer using the Viterbi approximation of the inside-outside algorithm described by Graehl and Knight (2004). Following Zhang and Gildea (2005), we lexicalized the nodes so that their probabilities capture bilexical dependencies.

The discriminative model was trained using the method in section 7.3, with the inference cost computed as described in Eq. (7.5). A separate classifier was induced for each possible translation of each source word seen in training data, to evaluate candidate transductions of leaf nodes. Additional classifiers were induced to confidence-rate candidate permutations of sibling nodes. Recall that each permutation involved a head-child node and one of its siblings. Since our input trees were lexicalized, it was easy to determine the lexical head of both the head-child and the other node participating in each permutation. Features were then compiled separately for each of these words according to the "window" and "dependency" feature types described in section 7.4.2. Since the tree was transduced bottom-up, the word-to-word translation of the lexical head of any node was already known by the time it participated in a permutation. So, in addition to dependents on the source side, there were also features to encode their translations. The final kind of feature used to predict permutations was whole synchronous context-free production rules, in bilexical, monolexical, and unlexicalized forms. We cannot imagine a generative process that could explain the data using this combination of features. Our hypothesis was that the discriminative approach would be more accurate, because its evaluation of each inference could take into account a great variety of information in the tree, including its entire yield (string), not just the information in nearby nodes.

For both models, the search for the optimal tree was organized by an agenda, as is typically done for tree inference algorithms. For efficiency, we used a chart, and

pruned items whose score was less than 10^{-3} times the score of the best item in the same chart cell. We also pruned items from cells whenever the number of items in the same cell exceeded 40. Our entire tree transduction algorithm can be viewed as translation by parsing (Melamed, 2004) where the source side of the output bitree was constrained by the input (source) tree.

We compared the generative and discriminative models by reading out the string encoded in their predicted trees, and comparing that string to the target sentence in the test corpus. In pilot experiments we used the BLEU measure commonly used for such comparisons (Papineni et al., 2002). To our surprise, BLEU reported unbelievably high accuracy for our discriminative transducer, exceeding the accuracy of Pharaoh even with a language model. Subsequently, we discovered that BLEU was incorrectly inflating our scores by internally retokenizing our French output. This behavior, together with the growing evidence against using BLEU for syntax-aware MT (Callison-Burch et al., 2006), convinced us to use the more transparent precision, recall, and F-measure, as computed by GTM (Turian et al., 2003). With the exponent set to 1.0, the F-measure is essentially the unigram overlap ratio, except it avoids double-counting. With a higher exponent, the F-measure accounts for overlap of all n-grams (i.e., for all values of n), again without double-counting.

During testing, we compared two kinds of input parse trees for each kind of tree transducer. The first kind was generated by the parser of Bikel (2004). The second kind was generated by the parser of Turian and Melamed (2006), which was trained in a purely discriminative manner. Table 7.5 shows the results. The discriminatively trained transducer far outperformed the generatively trained transducer on all measures, at a statistical significance level of 0.001 using the Wilcoxon signed ranks test. In addition, the discriminatively trained transducer performed better when it started with parse trees from a purely discriminative parser.

7.5 Conclusion

We have presented a method for training all the parameters of a syntax-aware statistical machine translation system in a discriminative manner. The system outperforms a generative syntax-aware baseline. We have not yet added all the standard information sources that are necessary for a state-of-the-art MT system, but the scalability of our system suggests that we have overcome the main obstacle to doing so.

Our next step will be to generalize the tree transducer into a bitree transducer, so that it can modify the target side of the bitree after it is inferred from the source side. In particular, we shall add a new type of inference to predict additional words on the target side, starting from words that are already there, and from all the information available on the source side (cf. Toutanova and Suzuki, 2007). This kind of inference will enable the transducer to translate strings with fewer words into strings with more words. Item costs assigned by a target language model will

likely be an important source of information for this kind of inference, just as it is in most other kinds of MT systems.

Acknowledgments

We would like to thank Yoshua Bengio, Léon Bottou, Patrick Haffner, Fernando Pereira, Christopher Pike, Cynthia Rudin, the anonymous reviewers, and the editors for their helpful comments and constructive criticism. This research was sponsored by NSF grants #0238406 and #0415933. The work described in this chapter was carried out while all of the authors were at New York University.

8 Reranking for Large-Scale Statistical Machine Translation

Kenji Yamada
Ion Muslea

Statistical machine translation systems conduct a nonexhaustive search of the (extremely large) space of all possible translations by keeping a list of the current n-best candidates. In practice, it was observed that the ranking of the candidates within the n-best list can be fairly poor, which means that the system is unable to return the best of the available N translations. In this chapter we propose a novel algorithm for reranking these n-best candidates. Our approach was successfully applied to large-scale, state-of-the-art commercial systems that are trained on up to three orders of magnitude more data than previously reported in reranking studies. In order to reach this goal, we create an ensemble of rerankers that are trained in parallel, each of them using just a fraction of the available data. Our empirical evaluation on two mature language pairs, Chinese-English and French-English, shows improvements of around 0.5 and 0.2 BLEU on corpora of 80 million and 1.1 billion words, respectively.

8.1 Introduction

Statistical machine translation (SMT) systems, which are trained on parallel corpora of bilingual text (e.g., French and English), typically work as follows: for each sentence to be translated, they generate a plethora of possible translations, from which they keep a smaller n-best list of the most likely translations. Even though the typical n-best list contains mostly high-quality candidates, the actual ranking is far from accurate: as shown in Och et al. (2004), the n-best list usually contains many translations that are of higher quality than the top-ranked candidate.

In order to deal with this problem, researchers have proposed a variety of reranking approaches. Previous work (Liang et al., 2006; Shen and Joshi, 2005; Roark et al., 2004) shows that machine learning can be successfully used for this

task. However, as the existing experiments were conducted on small corpora (the rerankers were trained on less than 1 million words), it is unclear how the results scale to real-world applications, in which one has to deal with several orders of magnitude more data.

We present here an efficient approach for building reranking systems for commercial, large-scale SMT systems. In this chapter, we make two main contributions. First, we introduce a novel reranking algorithm that can be trained efficiently on more than a billion words. In order to speed up its training process, we learn in parallel an ensemble of rerankers, each of which is using only a fraction of the training data. Second, we show that such rerankers can improve the performance of state-of-the-art SMT systems; more precisely, we used our novel approach for two mature language pairs, Chinese-English and French-English, which were trained on 80 million and 1.1 billion words, respectively. Our empirical results show that reranking has improved the quality of the translation for both systems by approximately 0.5 and 0.2 BLEU points (Papineni et al., 2002), respectively.

8.2 Background

Our reranking algorithm is designed to improve the performance of a baseline SMT system similar to the one described in Och et al. (2004). In keeping with the typical SMT methodology, such systems are trained on a large training set, tuned on a small development set, and then evaluated on a blind test set. In the remainder of this section we explain how the various data sets are used to build the system.

In order to build an SMT system that translates from the source to the target language (say, French into English), one starts with a large training corpus (i.e., parallel text) that is used to learn the system's translation and language models. The former consists of phrase pairs that are likely translations of each other, while the latter consists of the probabilities of the various phrases in the target language. For example, the translation model may learn that, according to the (imperfect) training data, it is highly likely that *J'ai faim* should be translated into *I'm hungry*, even though it may be sometimes be translated—less accurately—into *I'm so hungry!* In contrast, the language model contains the probabilities of encountering various English phrases in a large English corpus; e.g., it may be significantly more likely to encounter the phrase *I'm hungry* than *I'm so hungry!*

In practice, a typical training set is a large corpus (e.g., billions of words) that is obtained by concatenating documents from a plethora of genres that were translated with various levels of accuracy. Consequently, in order to optimize the system's performance, researchers perform an additional tuning step in which they use a *development set*; this development set is fairly small in size (typically around 1000 sentences), but contains high-quality translations of genres similar to the ones in the blind test set.

During this tuning phase, for each source sentence in the development set, the system adjusts the weights of a log-linear model to favor the most accurate

among the n-best translations (i.e., the one that is most similar to the reference translation). This log-linear model combines a variety of features such as the probabilities obtained from the language and translation models, the length (in words) of the translated sentence, etc. The ranking within each n-best list output by the SMT system is based on the so-called *decoder cost* (Och et al., 2004), which consists of the weighted combination of all the features according to the log-linear model.

In this chapter, we introduce a reranking algorithm that takes as input the ranked n-best lists for all sentences in both the training and development sets. Besides the decoder cost described above, we also use as features the phrases from the translation model that were used to generate the translations in the n-best list. We will show that our reranker improves the performance of the baseline system, and that it can be efficiently trained on n-best lists from tens of millions of sentences to find the most salient among the millions of available features.

8.3 Related Work

In recent years, ranking has been an active area of research in a variety of communities, from machine learning (Crammer and Singer, 2002; Bordes et al., 2007; Busse et al., 2007; Cao et al., 2007; Xu and Fern, 2007) to information retrieval (Brin and Page, 1998; Joachims et al., 2007) to natural language processing (Ratnaparkhi, 1997; Charniak, 2000; Ji et al., 2006). Intuitively, ranking can be defined as ordering a finite number of items according to a criterion such as their relevance to a given query or the preferences of a particular consumer. In contrast, reranking, which is used in natural language parsing (Collins and Koo, 2005; Collins and Duffy, 2002b; Shen and Joshi, 2003) and statistical machine translation (Shen and Joshi, 2005; Liang et al., 2006; Hasan et al., 2006), typically refers to the task of improving an existing ranking that was created based solely on local features available to an underlying generative model.

There are several reasons why general-purpose ranking algorithms are not appropriate for our task of reranking the n-best translations produced by an SMT system:

- Ranking algorithms are traditionally applied to collections of items that are directly comparable to each other (e.g., *I prefer this book to that book*). In contrast, when performing reranking for SMT, the only meaningful comparisons are among the various translations of the *same sentence* (i.e., translations coming from the same n-best list). From a reranking perspective, it does not make sense to compare - say - the translation $T91$ of sentence $S2$ with the translation $T13$ of sentence $S12$.
- In contrast to traditional ranking, reranking does not attempt to rank *all* n-best candidates, but rather to find *the best* of them (or, at the very least, one that is superior to the top-ranked one in the original ranking).

- Most of the existing ranking approaches were applied to a relatively small number of possible ranks; for example, the algorithms proposed in Herbrich et al. (2000), Crammer and Singer (2002), and Harrington (2003) were used in domains that had fewer than ten possible ranks.
- Last but not least, when dealing with large corpora, there are serious scalability issues. For example, our French-English training corpus consists of 91 million parallel sentences, each of which has a 200-best list of possible translations. Even after exploiting the locality of the reranking, one cannot hope to efficiently train a system on all $O(200 \times 199 \times 91,000,000)$ possible candidate pairs that exist in our corpus (i.e., the 200×199 possible pairs for each of the 91 million training sentences).

Our approach is most similar in spirit to the fast perceptron training algorithm proposed in Shen and Joshi (2005). In their experiments, they trained using only the n-best lists for the 1000 sentences in the development set; furthermore, they used as features only the 20 best individual features from Och et al. (2004). Even though this approach showed promising results, it has the drawback of ignoring the data from the significantly larger training set and many potentially informative features.

In contrast to Shen and Joshi (2005), we train the reranker on the entire corpus and investigate the use of millions of lexical features from the translation model. Given the size of our data set, training a single perceptron on the entire corpus would be extremely inefficient. Instead, we create an ensemble of perceptrons that is trained in parallel on a fraction of the data. This approach is similar to bagging (Breiman, 1996), except that we use arbitrary splits of 5000 sentences (given that each split consist of 0.1% of the corpus, it does not make sense to use sampling with replacement). After the perceptrons are trained, the ensemble selects the best of the N candidates by averaging the predictions of all perceptrons.

Our approach also has similarities to the one presented in Liang et al. (2006): both algorithms are based on perceptron-style training and use a large number of features. Liang et al. (2006) have also tried to use the translation model features discussed here, supplemented by additional language model and part-of-speech features. However, their approach focuses on improving the tuning algorithm, rather than reranking. Furthermore, our reranker is capable of exploiting three orders of magnitude more training data, and we manage to improve a far more performant baseline system (BLEU 57.33 vs. 29.8).

8.4 Our Approach

In this section we describe the main idea behind our approach to reranking. The next two sections discuss in detail the way in which we applied this method to two different commercial SMT systems.

Algorithm 8.1 Perceptron training algorithm.

Initialize w_0
For each epoch t
  For each sentence
    For each nonoracle hypothesis x_i in the n-best
      if $w_t \cdot x_i < w_t \cdot x_{\text{oracle}}$ then
        // misclassification
        $w_{t+1} = w_t + (x_i - x_{\text{oracle}}) \times \alpha$
        where $\alpha = \text{BP1}(x_{\text{oracle}}) - \text{BP1}(x_i)$
      else $w_{t+1} = w_t$
      $t = t + 1$
Output: either w_t or $\sum_i w_i/t$

As we already mentioned, the goal of reranking is to find, for each of the n-best lists, the most accurate translation according to the reference translation. We refer to these translations as oracles, and we use them to learn a reranker that discriminates between oracles and nonoracles. In order to identify the oracles in the n-best lists used for training, we use a floored version of the BLEU metric called BLEU+1 (BP1).[1]

Intuitively, our reranking algorithm can be seen as a perceptron that repeatedly visits all the n-best lists provided for training. For each n-best list, it compares the oracle against each nonoracle translation. If the reranker makes a mistake on a particular comparison (i.e., it predicts that the nonoracle is a better translation than the oracle), the perceptron's weights are updated proportionally to the difference between the oracle and the nonoracle translation.

Figure 8.1 provides a formal description of our perceptron training algorithm. The weights w_0 are initialized so that they reproduce the original ranking produced by the baseline system (i.e., they are sorted based on the decoder cost); more precisely, the weight of the decoder cost is set to one, while all other weights are set to zero. As mentioned above, the weights are updated only when a nonoracle translation outperforms the corresponding oracle. After the predefined number of epochs, the final reranker can either use the last set of weights or can opt for the weights $\sum_i w_i/t$, which correspond to the averaged perceptron algorithm (Freund and Shapire, 1999).

We experiment here with two variations of the traditional perceptron algorithm: the *averaged perceptron* (Freund and Shapire, 1999) and the *dev-interleaved* perceptron. The former is an extremely simple and efficient approximation of the traditional maximal-margin classifiers. The latter is introduced in this chapter and is

1. As described in Papineni et al. (2001), the BLEU metric is a geometric average of the n-gram precision in a translation, and it is meant to measure the document-level translation quality, not at the sentence level. At the sentence level, especially for short sentences, BLEU can be extremely inaccurate and is commonly replaced by BP1 (Och et al., 2004).

inspired by the common practice within the SMT community of repeatedly replicating the development corpus until the amount of the development data is comparable to the one of the training corpus (Yamada et al., 2007). This is motivated by two considerations: (1) the amount of training data is orders of magnitude larger than the amount of development data, and (2) the statistical distribution within the training corpus is typically different from that of the test set. In order to maintain a balance between the weights' updates caused by the training and development sentences, we have decided to interleave two corpora (e.g., the first sentence comes from the training corpus, the second one from the development one, etc.), thus the name of *dev-interleaved* for the new algorithm. Interleaving, rather than concatenating, is necessary because the perceptron algorithm can update the weight vector by each training sentence.

Last but not least, in order to deal with the large amount of training data, we do not train a single perceptron, but rather we create an ensemble of up to 1000 perceptrons. Each individual perceptron is trained based on the algorithm in figure 8.1, and it uses only a small fraction of the training data; as the perceptrons can be trained independently of each other, this represents a highly parallelizable process. After training all individual perceptrons, the final weights are obtained by averaging the weights from each individual perceptron. It is possible to apply a sophisticated averaging scheme such as weighted average, but we only do a simple averaging with no normalization.

8.5 Experiment 1: Reranking for the Chinese-to-English System

In order to build the baseline Chinese-to-English system, we used a setup similar to the one described in Och et al. (2004). The training corpus contains 4.76 million sentences (approximately 80 million words on the English side), while the development and the blind test set consist of 993 and 919 sentences, respectively. Each Chinese sentence from the training set has just a single translation (of unknown quality), while the ones from the development and test set have four high-quality human translations each. Based on the training corpus, the baseline system has learned a translation model that consists of approximately 19 million phrase pairs and a trigram language model for the 80 million English words. An additional trigram language model was learned from a 200 million word monolingual English corpus. The baseline system used a tuning algorithm (Och, 2003) with the development corpus.

8.5.1 Training the Reranker

In order to create the training set for our reranker, we first used the baseline system to generate the 200 best translations for each Chinese sentence in the training and development corpus; then we used the BLEU+1 metric to find the oracle within each 200-best list (i.e., the most accurate of the 200 translations).

As discussed earlier, the features used to describe each of the n-best translations are the decoder score, together with the most informative of the 19 million phrase pairs that were used to generate that particular translation. More precisely, we pruned out all the phrase pairs that occurred extremely frequently or infrequently in the training corpus (i.e., more than 100,000 times or only once, respectively). After this pruning step, we were left with about 4 million phrase pairs to be used as features (the value of such a feature indicates how many times the phrase pair was used to generate that particular translation).

In order to ensure that each perceptron in the ensemble was learned as quickly as possible, we decided to train each individual perceptron on 5000 sentences, thus creating an ensemble of 953 perceptrons. For each of the 5000 sentences we had a 200-best list, and the perceptron compared each of the 5000 oracles against the corresponding other 199 translations; consequently, during an epoch of perceptron training, the training algorithm performed almost 1 million comparisons (199×5000).

After the last training epoch, we created our reranker by averaging the weights learned for the 953 individual perceptrons (i.e., the reranker was a perceptron that used the same features as the ensemble). We used this newly constructed perceptron to rerank the 200 best translations for each sentence in the test set. The performance of our algorithm was evaluated by computing the BLEU score of the one-best translation according to the reranker. (We also measured the performance with other metrics: see appendix.)

8.5.2 The Results

Figure 8.1 shows our main results for the Chinese-to-English experiment. The x-axis shows the number of training epochs, while the y-axis shows the BLEU score of the translations selected by the reranker. We use the normal perceptron as baseline, and we compare its performance against that of three other types of perceptrons: *averaged*, *dev-interleaved*, and *averaged, dev-interleaved*.

The baseline system (at epoch 0) yields a BLEU score of 31.19, while the best BLEU score reached by any of the four systems is 31.77 (the averaged, dev-interleaved perceptron after epoch 4). This improvement of 0.58 BLEU is statistically significant at $p < .001$ (Riezler and Maxwell, 2005).

Figures 8.2, 8.3, and 8.4 show the results obtained when varying the size of the training data used for each individual perceptron in the ensemble: 50,000, 500,000, and 4.76 million sentences, respectively (which lead to ensembles of sizes 96, 10, and 1). The first two scenarios lead to results similar to those obtained by the 953-perceptron ensemble, while the last one shows a less stable behavior.

Finally, figures 8.5 and 8.6 show the performance in terms of the number of classification errors. Similar to the previous figures, the x-axis shows the number of epochs, but the y-axis shows the number of misclassification errors that occur during the ensemble training. The misclassification error rate was measured against the training data and the test data, respectively. For the training corpus, the

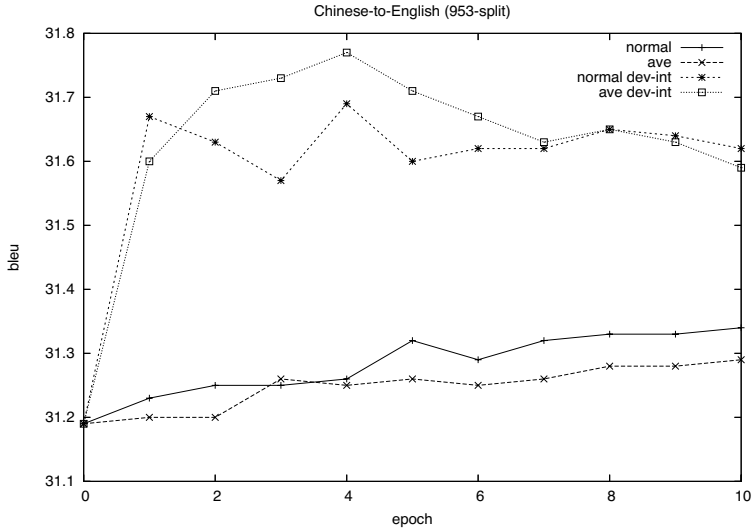

Figure 8.1 Chinese-to-English experiment: 953-split.

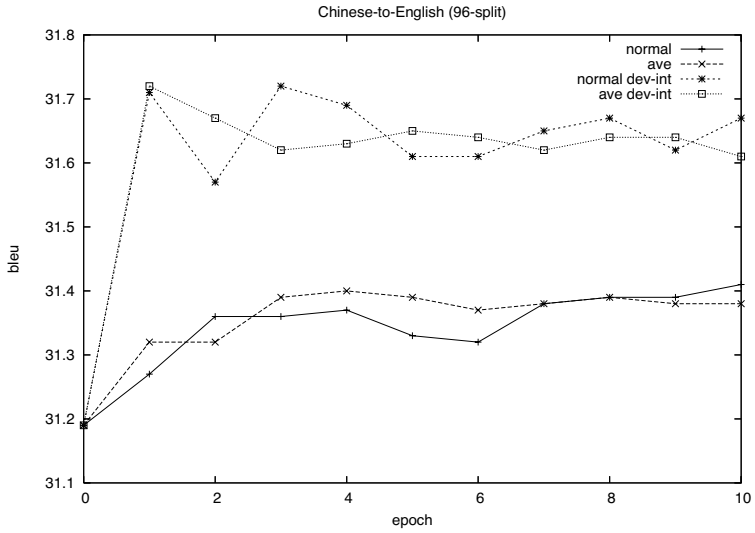

Figure 8.2 Chinese-to-English experiment: 96-split.

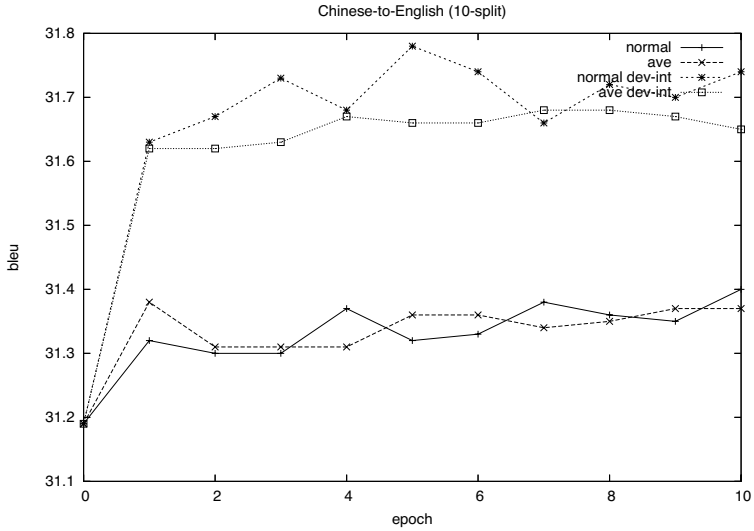

Figure 8.3 Chinese-to-English experiment: 10-split.

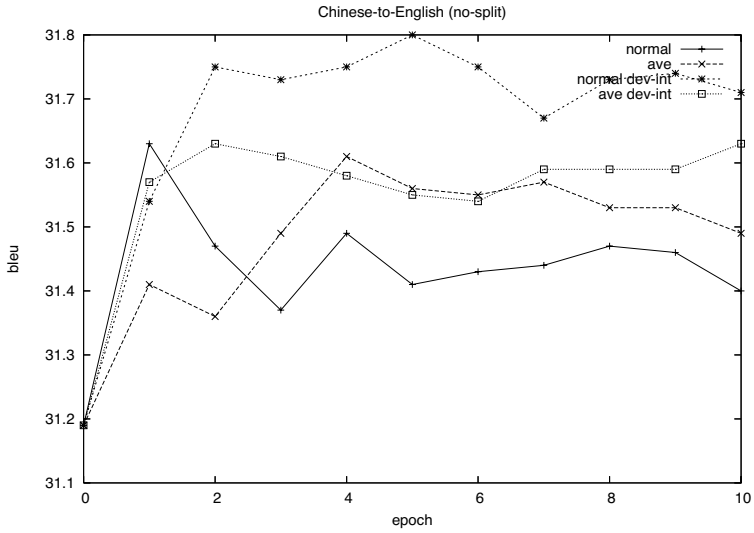

Figure 8.4 Chinese-to-English experiment: no split.

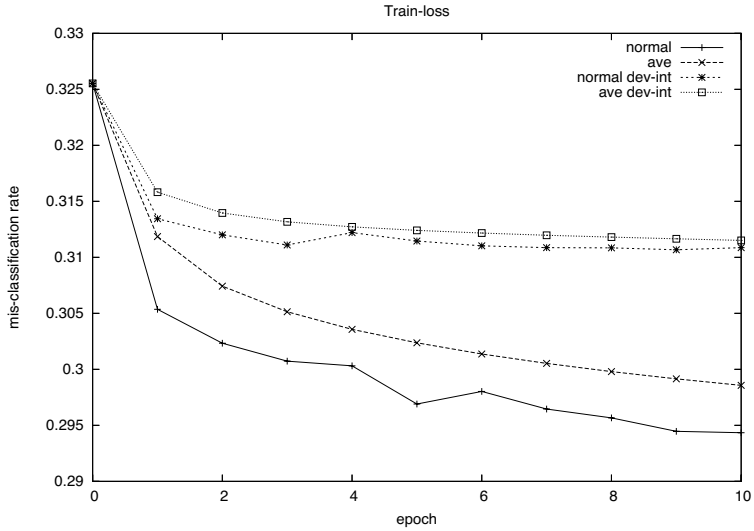

Figure 8.5 Chinese-to-English experiment: train-loss.

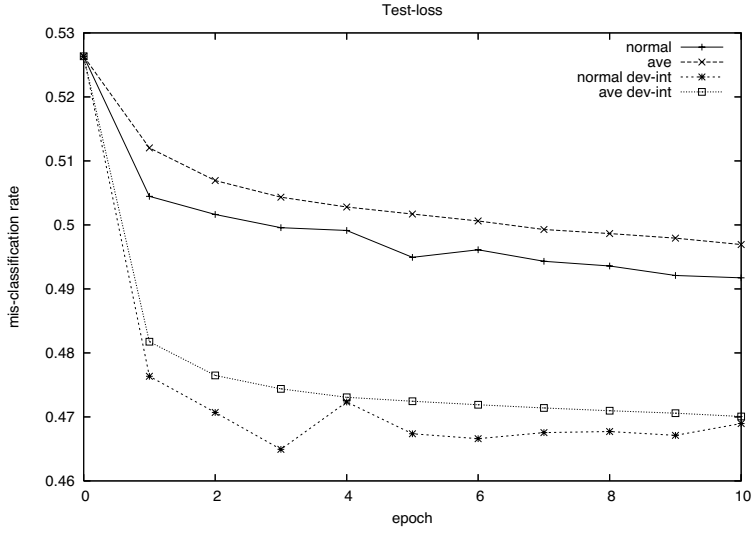

Figure 8.6 Chinese-to-English experiment: test-loss.

misclassification rate in the baseline system is 0.326 (epoch 0 in figure 8.5), which means that 32.6% of the nonoracle translations were wrongly classified as better translations than the oracle one. In other words, the oracle translation was ranked, on average, around 32.6% from the top in the n-best output.

8.6 Experiment 2: Reranking for the French-to-English System

The approach used to apply reranking to our French-to-English system is extremely similar to the one presented in the previous section. The few differences come from the fact that the baseline French-to-English system was trained from more than an order of magnitude more data. More precisely, the training corpus consists of 91 million sentences (1.1 billion words on the English side), from which 79 million phrase pairs were extracted. The development and the test corpus consist of 1013 and 1020 sentences, respectively. Even without using an additional language model, the baseline system yields a BLEU score of 57.33.

The feature selection was similar to that of Chinese-to-English, except we changed the values of the pruning thresholds due to the larger amount of data (i.e., 1 million for the frequent and 5 million for the infrequent phrase pairs). After pruning, we were left with about 23 million phrase pairs.

In order to avoid creating an extremely large ensemble of perceptrons that would take too long to train, we used only the 200-best lists from every tenth sentence in the corpus, for a total of more than 9.1 million sentences. Each individual perceptron was trained on 5000 sentences, which led to an ensemble of 1828 perceptrons. In keeping with the setup from the Chinese-to-English experiments, we also used again the normal, the averaged, the dev-interleaved, and the averaged, dev-interleaved perceptrons.

8.6.1 The Results

As seen in figure 8.7, the best BLEU score (57.61) was obtained after epoch 2 of the averaged, dev-interleaved perceptron. The improvement from the baseline is 0.28, which is statistically significant at $p < .005$.

Similarly to the Chinese-to-English experiments, we also tried splitting into different numbers of training sentences for each perceptron; besides the default value of 5000 sentences, we also tried 50,000, 500,000, and 9.1 million sentences, leading to ensembles of 186, 19, and a single perceptron (see figures 8.8, 8.9, and 8.10, respectively). Finally, figures 8.11 and 8.12 show the misclassification rate of the 1828-perceptron ensemble for the training data and the test data, respectively.

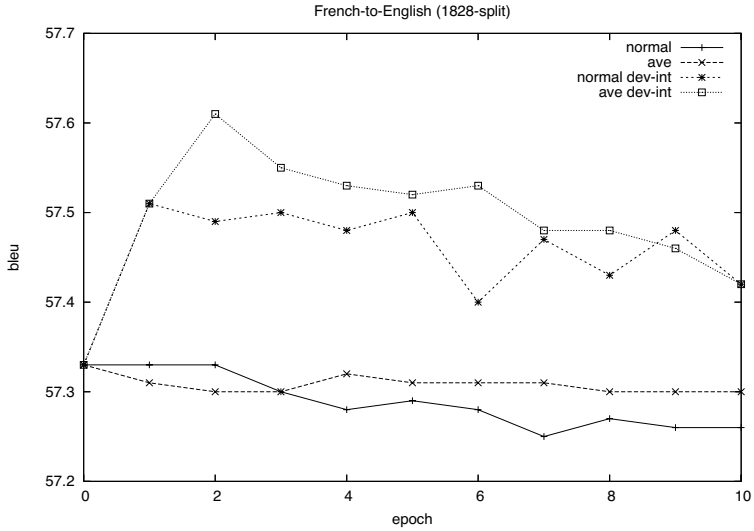

Figure 8.7 French-to-English experiment: 1828-split.

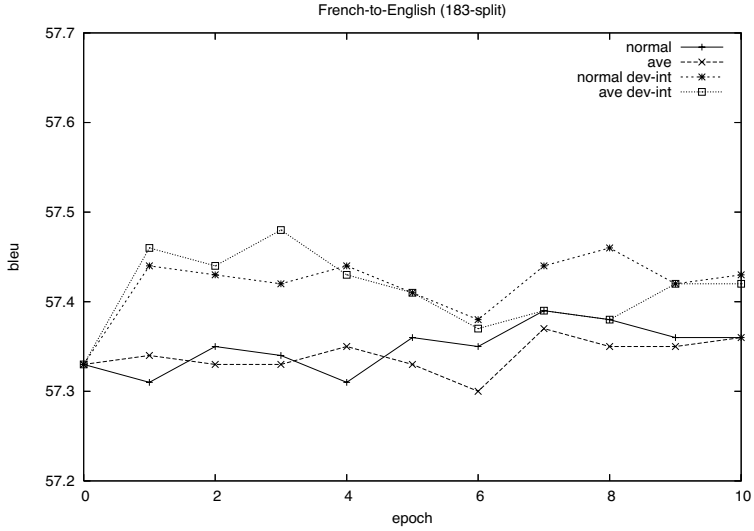

Figure 8.8 French-to-English experiment: 183-split.

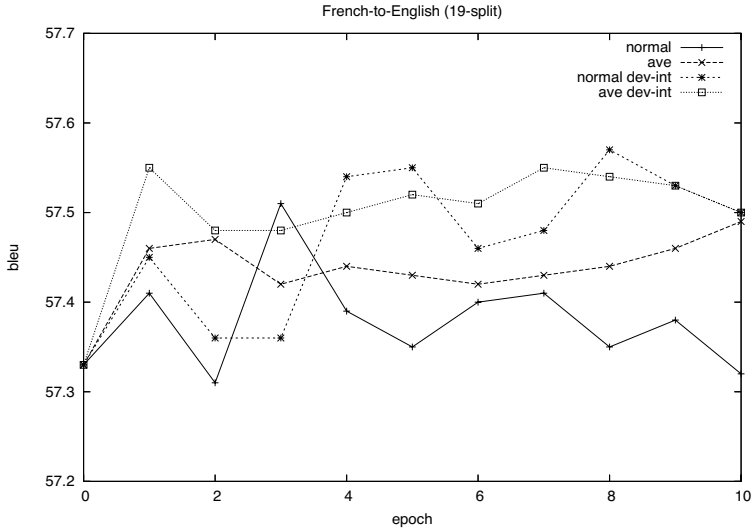

Figure 8.9 French-to-English experiment: 19-split.

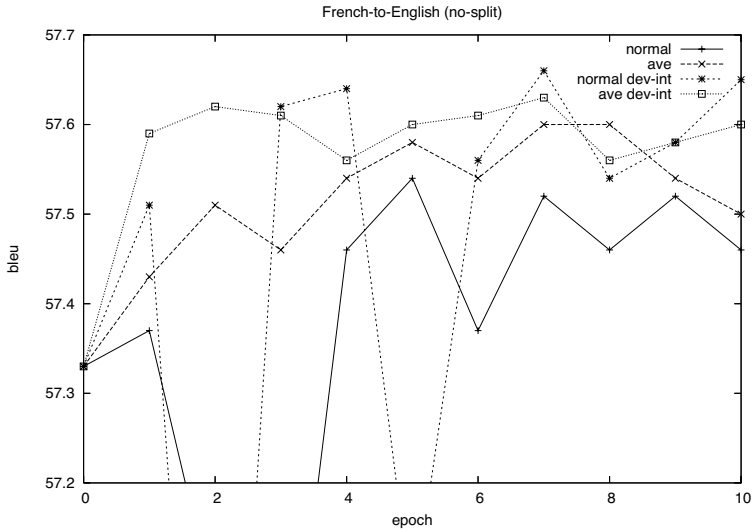

Figure 8.10 French-to-English experiment: no-split.

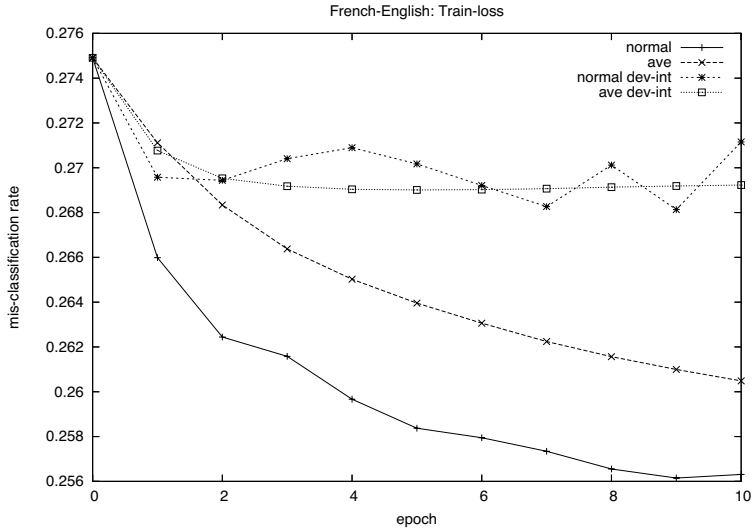

Figure 8.11 French-to-English experiment: train-loss.

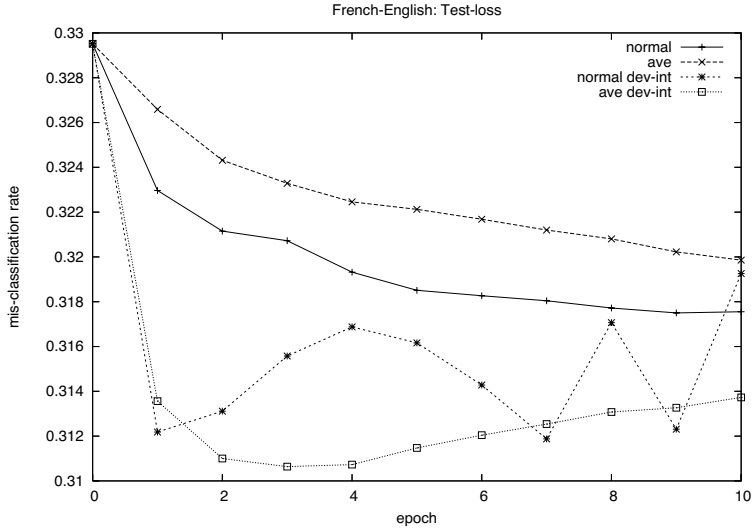

Figure 8.12 French-to-English experiment: test-loss.

8.7 Discussion

Our experimental results show that the reranking algorithm improves the BLEU score by around 0.5 and 0.2 for the Chinese-to-English and French-to-English systems, respectively. We think that the smaller improvement for the second system is due to the fact that it is significantly more difficult to improve the BLEU score of an excellent system (i.e., BLEU of 55) than one of lesser quality (BLEU of just 31).

The explanation above is also supported by the following observation: for both the training and the test corpora, the initial training error is higher for the Chinese-to-English system (0.326 and 0.526 in figures 8.5 and 8.6) than the French-to-English system (0.275 and 0.329 in figures 8.11 and 8.12).

This large gap between the training and the test error rates of both systems ($0.526 - 0.326 = 0.2$ and $0.329 - 0.275 = 0.054$) also reflects the discrepancy between the distributions of training and test data. Our dev-interleaved perceptron represents an effective solution to this problem, especially for the Chinese-to-English system. In contrast, the averaged perceptron leads to smaller improvements, but behaves in a more stable manner. By combining the two methods within the averaged, dev-interleaved perceptron, we obtain the best of both worlds.

The idea of training an ensemble of perceptrons was also extremely successful. For both the Chinese and French systems, the larger the size of the ensemble, the more stable the behavior in terms of the BLEU score. This approach also has the advantage of being highly parallelizable: by training a perceptron per CPU, we managed to train the reranker in several hours rather than a few weeks.

In most experiments, the best BLEU score is obtained after two to four training epochs: more epochs further reduce the classification error, but the BLEU improvement is extremely small. Ideally, an additional development corpus should be used to determine the exact number of training epochs.

In the French-to-English experiment, the nonaveraged, dev-interleaved perceptron exhibits relatively large oscillations of the BLEU from epoch to epoch (see figures 8.7, 8.11, and 8.12). We think that this is due to the fact that the phrase pair features do not take into account the larger context (i.e., the relative position of the phrase in the translation, together with the phrases preceding or following it); consequently, the same phrase pair may be extremely useful for some sentences, but inappropriate in other contexts.

8.8 Conclusion

In this chapter we have introduced a novel approach to reranking the n-best list produced by a statistical machine translation system. We have used an ensemble of perceptrons that are trained in parallel, each of them on just a fraction of the available data. We performed experiments on two large-scale commercial systems:

a Chinese-to-English system trained on 80 million words and a French-to-English system trained on 1.1 billion words. Our reranker obtained statistically significant improvements of about 0.5 and 0.2 BLEU points on the Chinese-to-English and the French-to-English system, respectively. In future work, we are primarily interested in exploiting additional features that could further improve the reranking (e.g., language model (Roark et al., 2004) or syntactic features (Hasan et al., 2006)).

Appendix

We also measured the performance of our reranker with metrics other than BLEU: the NIST score (Doddington, 2002), the word error rate (WER), and the position-independent error rate (PER) (Tillmann et al., 1997b). The NIST score is similar to BLEU, but it is weighted by the n-gram frequencies. WER is the edit distance between the SMT output and the reference translations (i.e, the number of words that must be inserted, deleted, or substituted), divided by the length of the reference sentence. Finally, PER is similar to WER, but it ignores the word order.

We took the experimental results from figures 8.1 and 8.7 and measured the improvements with NIST, WER, and PER (i.e., the selected translations by the reranker are the same). They are shown in figures 8.13 and 8.14. Each figure shows one experiment with the four metrics. The upper-left panel in each figure shows the results with the BLEU metric, and it is identical to the corresponding figure in the previous section.

Note that the training behavior of the reranker looks similar, independently of the metrics used: the best results are seen in training epochs 2 to 4, and the averaged, dev-interleaved perceptron outperforms the other three approaches. The only peculiar result is the one using PER for the French-to-English system; we do not currently have an explanation for this behavior.

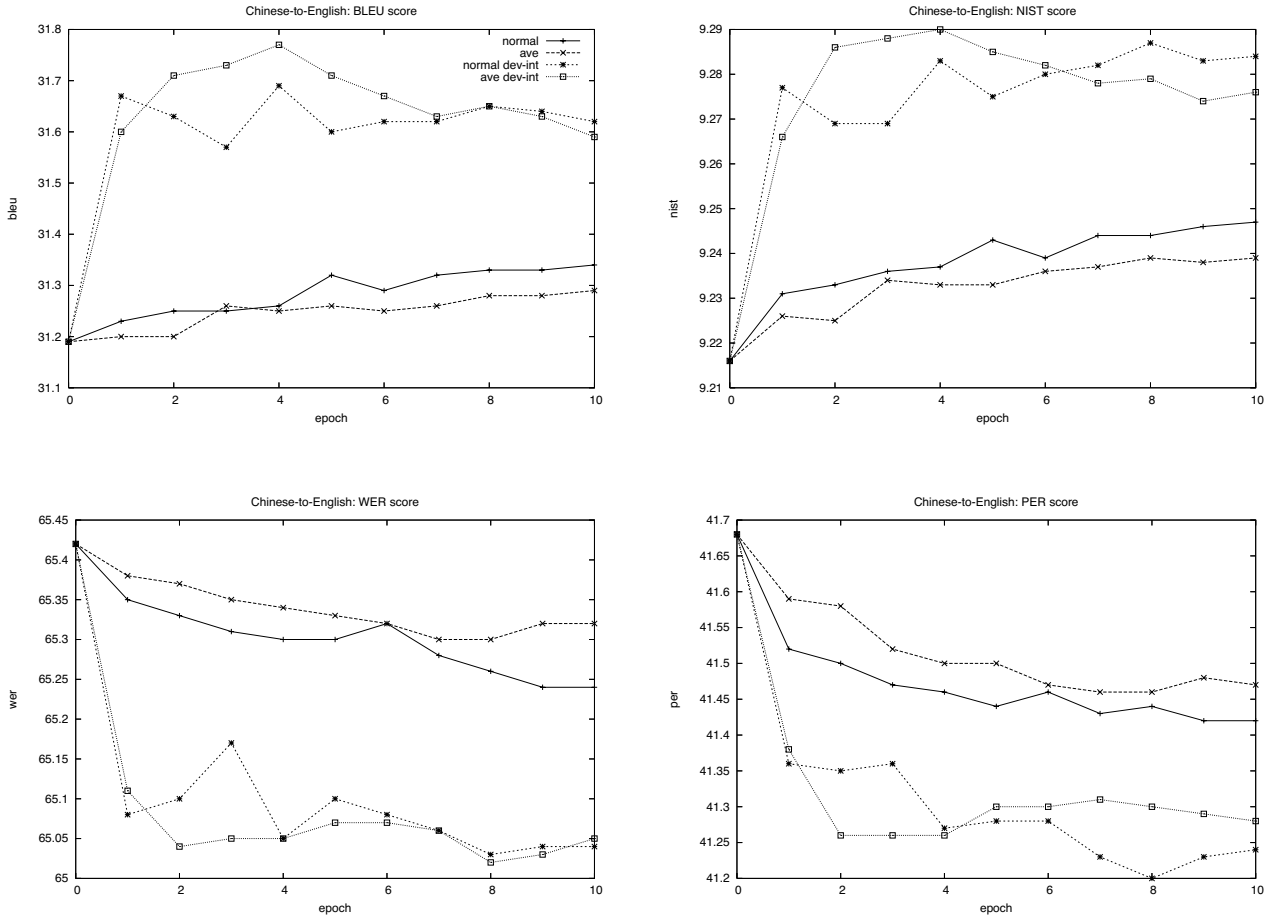

Figure 8.13 Chinese-to-English results with different evaluation metrics.

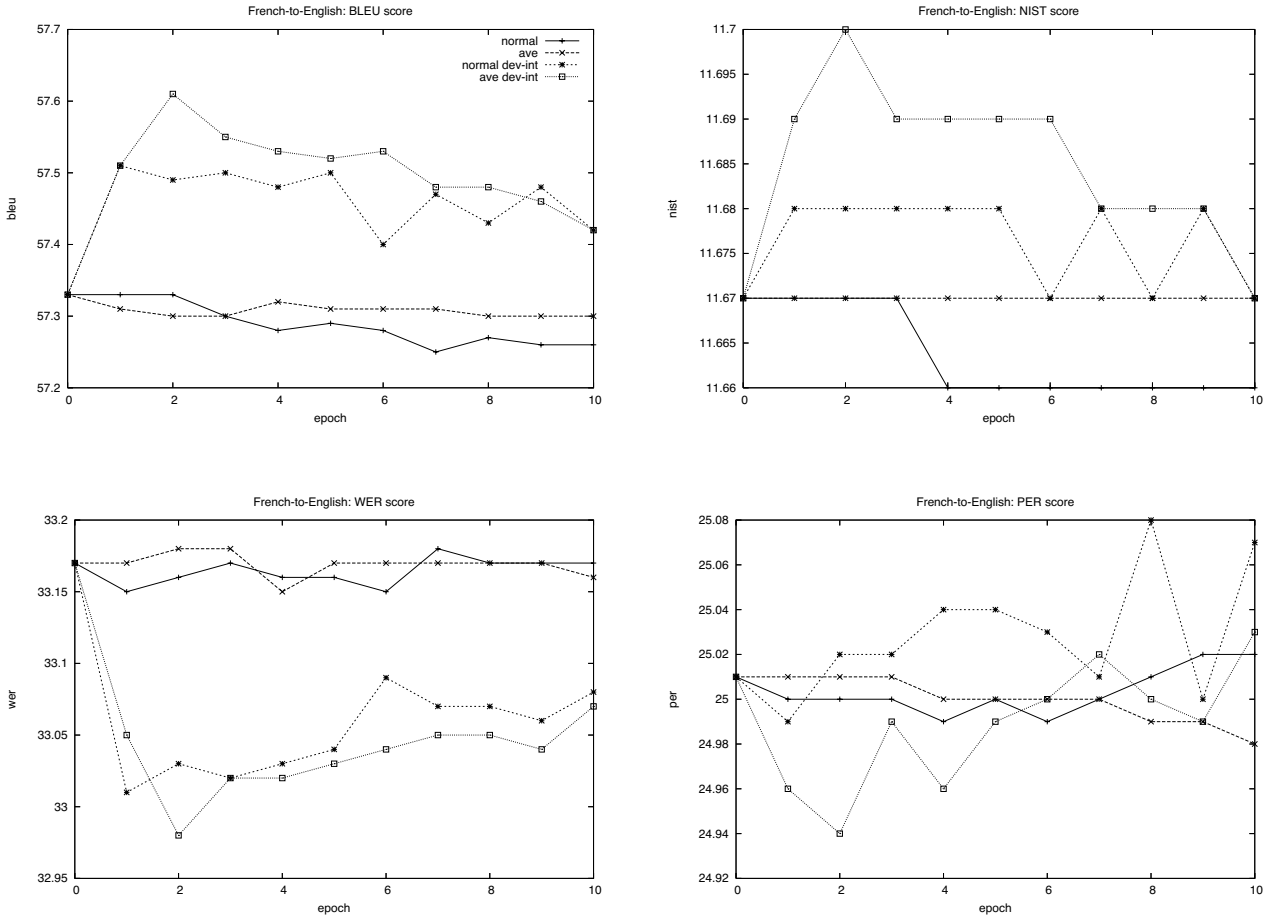

Figure 8.14 French-to-English results with different evaluation metrics.

9 Kernel-Based Machine Translation

Zhuoran Wang
John Shawe-Taylor

In this chapter, we introduce a novel machine translation framework based on kernel regression techniques. In our model, the translation task is viewed as a string-to-string mapping, for which ridge regression is employed with both source and target sentences embedded into their respective kernel-induced feature spaces. Not only does it suggest a more straightforward and flexible way to model the translational equivalence problem, compared to previous probabilistic models that usually require strong assumptions of conditional independences, this method can also be expected to capture much higher-dimensional correspondences between inputs and outputs. We propose scalable training for it based on the blockwise matrix inversion formula, as well as sparse approximations via retrieval-based subset selection techniques. However, because of the complexities of kernel methods, the contribution of this work is still mainly conceptual. We report experimental results on a small-scale reduced-domain corpus, to demonstrate the potential advantages of our method when compared with an existing phrase-based log-linear model.[1]

9.1 Introduction

Many problems in text processing and related fields, e.g., part-of-speech tagging and optical character recognition, can be regarded as string-to-string mappings. Cortes et al. (2005) presented a novel regression framework for the learning of this kind of transduction problem, in which both the input and the output strings were embedded into their respective reproducing kernel Hilbert spaces (RKHS), usually known as feature spaces, and then ridge regression was applied to learn the mapping from the input feature space to the output one. In addition, Cortes et al. (2005) also proposed a solution to seek a preimage, i.e., predicted output string, from an

1. Parts of this work have appeared in Wang et al. (2007).

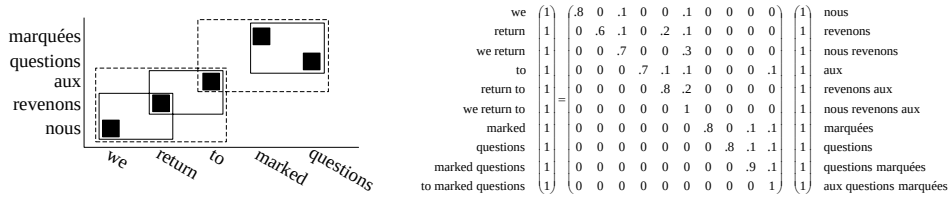

Figure 9.1 From phrase alignment to linear mapping: a simple demonstration.

n-gram string kernel (Cortes et al., 2004) induced output feature space, which can be summarized as finding a Eulerian circuit of a De Bruijn graph associated with the predicted n-gram counts.

In this chapter, we follow the work of Cortes et al. (2005), and apply the kernel regression technique to the particular task of statistical machine translation (SMT), which is viewed as a mapping from word strings (sentences) in the source language to word strings in the target language. Concretely, if we define the feature space $\mathcal{H}_x$ of our source language $\mathcal{X}$ as all its possible informative word chunks (word substrings), and define the mapping $\Phi : \mathcal{X} \to \mathcal{H}_x$, then a sentence $\mathbf{x} \in \mathcal{X}$ can be expressed by its feature vector $\Phi(\mathbf{x}) \in \mathcal{H}_x$. Each component of $\Phi(\mathbf{x})$ is indexed by a word chunk with the value being the frequency of it in $\mathbf{x}$. The definition of the feature space $\mathcal{H}_y$ of our target language $\mathcal{Y}$ can be made in a similar way, with corresponding mapping $\Psi : \mathcal{Y} \to \mathcal{H}_y$. Now in the machine translation task, we are trying to seek $\mathbf{W}$, a matrix-represented linear operator, such that

$$\Psi(\mathbf{y}) = \mathbf{W}\Phi(\mathbf{x}) \tag{9.1}$$

to predict the translation $\mathbf{y}$ for an arbitrary source sentence $\mathbf{x}$. We argue that the linear regression view of SMT is natural, as from phrase alignment diagrams widely used in the training of existing phrase-based SMT models we are able to explicitly write out many possible projection matrices $\mathbf{W}$, for which a simple example is demonstrated in figure 9.1.

Thus, the motivation of this work is to investigate an algorithm to learn the $\mathbf{W}$ that captures potentially very high-dimensional correspondences among the source and target features on a given training set, and generalizes well for unseen examples. First, the ridge regression formulation of the learning problem, the kernel function selected to induce the implicit feature spaces, and related scalable training and sparse approximation issues are introduced in detail in section 9.2. Since in the SMT case the vocabulary is so large that the word chunk counts in our prediction will be too noisy to construct a good enough De Bruijn graph to yield a meaningful output sentence, the preimage solution will be a further challenge. In section 9.3 we explore the decoding algorithm for this particular task, which follows the beam search procedure, parses the word lattice generated based on a lexicon, and seeks the candidate translation whose feature vector is closest to our prediction measured by Euclidean distance. Then, in section 9.4, experimental results are reported, where our method is compared to a typical existing phrase-based log-linear model,

and achieves higher performance. We give further discussions on the integration of additional language models and the utilization of richer linguistic knowledge in our framework in section 9.5, and conclude in section 9.6.

9.2 Regression Modeling for SMT

9.2.1 Kernel Ridge Regression

Given a set of training samples, i.e., bilingual sentence pairs $S = \{(\mathbf{x}_i, \mathbf{y}_i) : \mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y}, i = 1, \ldots, m.\}$, least squares regression learns the linear operator $\mathbf{W}$ in Eq. (9.1) by seeking the one minimizing the squared loss in $\mathcal{H}_y$ on S, as

$$\min \quad \|\mathbf{W}\mathbf{M}_\Phi - \mathbf{M}_\Psi\|_F^2 \ , \tag{9.2}$$

where $\mathbf{M}_\Phi = [\Phi(\mathbf{x}_1), ..., \Phi(\mathbf{x}_m)]$, $\mathbf{M}_\Psi = [\Psi(\mathbf{y}_1), ..., \Psi(\mathbf{y}_m)]$, and $\|\cdot\|_F$ denotes the Frobenius norm, which is a matrix norm defined as the square root of the sum of the absolute squares of the elements in that matrix. However, if there are some duplicated or very similar samples in the training set, $\mathbf{M}_\Phi$ will be ill-conditioned or singular, yielding a large number of solutions. A well-known improvement to this problem is ridge regression (Hoerl and Kennard, 1970), which gives preference to the solution with a smaller norm by adding a regularization term to the minimization as

$$\min \quad \|\mathbf{W}\mathbf{M}_\Phi - \mathbf{M}_\Psi\|_F^2 + \nu\|\mathbf{W}\|_F^2 \ , \tag{9.3}$$

where ν is a coefficient adjusting the effect of the regularization. This regularization improves the conditioning of the problem, thus enabling a numerical solution.

The explicit solution of the ridge regression problem can be computed as follows. Differentiating the expression and setting it to zero gives

$$\mathbf{W} = \mathbf{A}\mathbf{M}_\Phi^\top,$$

where

$$\mathbf{A} = -\frac{1}{\nu}(\mathbf{W}\mathbf{M}_\Phi - \mathbf{M}_\Psi).$$

Substituting the expression for $\mathbf{W}$ in the equation for $\mathbf{A}$ gives

$$\mathbf{A}(\mathbf{K}_\Phi + \nu\mathbf{I}) = \mathbf{M}_\Psi,$$

implying

$$\mathbf{W} = \mathbf{M}_\Psi(\mathbf{K}_\Phi + \nu\mathbf{I})^{-1}\mathbf{M}_\Phi^\top, \tag{9.4}$$

where $\mathbf{I}$ is the identity matrix, and $\mathbf{K}_\Phi = \mathbf{M}_\Phi^\top \mathbf{M}_\Phi = (\kappa_\Phi(\mathbf{x}_i, \mathbf{x}_j)_{1 \le i,j \le m})$. Note here, we use the kernel function

$$\kappa_\Phi(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle = \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j) \tag{9.5}$$

to denote the inner product between two feature vectors. If the feature spaces are properly defined, the "kernel trick" will allow us to avoid dealing with the very high-dimensional feature vectors explicitly (Shawe-Taylor and Cristianini, 2004). A detailed explanation of these issues in our case will be left to section 9.2.2. In addition, we will denote by $\kappa_\Psi(\cdot, \cdot)$ the kernel with respect to $\Psi(\cdot)$ in future discussions.

Inserting Eq. (9.4) into Eq. (9.1), we obtain our prediction as

$$\Psi(\mathbf{y}) = \mathbf{M}_\Psi (\mathbf{K}_\Phi + \nu \mathbf{I})^{-1} \mathbf{k}_\Phi(\mathbf{x}), \tag{9.6}$$

where $\mathbf{k}_\Phi(\mathbf{x}) = (\kappa_\Phi(\mathbf{x}, \mathbf{x}_i)_{1 \le i \le m})$ is an $m \times 1$ column matrix.

9.2.2 N-Gram String Kernel

In the practical learning (the matrix inversion in Eq. (9.4)) and prediction (see section 9.3.1, Eq. (9.14)) processes, only the inner products of feature vectors are required, which can be computed with the kernel function implicitly without evaluating the explicit coordinates of points in the feature spaces. In the SMT case, the n-gram string kernel (Cortes et al., 2004) that compares two strings by counting how many contiguous substrings of length n they have in common is a good choice to implicitly induce our $\mathcal{H}_x$ and $\mathcal{H}_y$.

We denote by $\mathbf{x}^{i:j}$ a substring of sentence $\mathbf{x}$ starting with the ith word and ending with the jth. Then for two sentences $\mathbf{x}$ and $\mathbf{z}$, the n-gram string kernel is computed as

$$\kappa_n(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{|\mathbf{x}|-n+1} \sum_{j=1}^{|\mathbf{z}|-n+1} [\![\mathbf{x}^{i:i+n-1} = \mathbf{z}^{j:j+n-1}]\!]. \tag{9.7}$$

Here, $|\cdot|$ denotes the length of the sentence, and $[\![\cdot]\!]$ is the indicator function for the predicate.

As the length of informative word chunks varies, we use a blended n-gram string kernel to count the substrings of length from 1 to n simultaneously. That is,

$$\kappa(\mathbf{x}, \mathbf{z}) = \sum_{p=1}^{n} \kappa_p(\mathbf{x}, \mathbf{z}). \tag{9.8}$$

Note here, instead of explicitly defining the informative work chunks, with this kernel actually all the potential word n-grams are used as our features.

9.2.3 Large-Scale Training

As a kernel method, this model suffers from the major drawback of high computational complexities. The matrix inversion operation in Eq. (9.4) requires $O(m^3)$ time and $O(m^2)$ space. On the other hand, for SMT usually the training set is so huge that loading the $m \times m$ kernel matrix into memory at one time is infeasible. Moreover, our preliminary experiments show that the translation quality is very sensitive to the accuracy of the inverse, and therefore, sparse greedy matrix approximation approaches (Smola and Schölkopf, 2000; Bach and Jordan, 2002; Vincent and Bengio, 2002) will not help in this case. Thus, to make the application of this model practical, we need some tricks in the implementation.

Fortunately, we can split the matrix $(\mathbf{K}_\Phi + \nu\mathbf{I})$ into blocks, as

$$(\mathbf{K}_\Phi + \nu\mathbf{I}) = \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix}.$$

The blockwise matrix inversion formula (Golub and Van Loan, 1996) gives

$$\begin{bmatrix} A & B \\ B^\top & C \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}BS_A^{-1}B^\top A^{-1} & -A^{-1}BS_A^{-1} \\ -(A^{-1}BS_A^{-1})^\top & S_A^{-1} \end{bmatrix}, \tag{9.9}$$

where $S_A = C - B^\top A^{-1}B$ is called the Schur complement of A. This blockwise calculation provides a memory-efficient solution for large-scale training problems. The inverse matrix now can be computed in smaller blocks, with only one or two blocks loaded into memory at a time. In addition, the blocks can be processed in parallel, which speeds up the computing time to some extent.

9.2.4 Retrieval-Based Sparse Approximation

In the RKHS defined above, we will be able to obtain a good translation as long as we find a hyperplane passing through, or close enough to, the test data point. So we do not necessarily need all the training samples, but only those covering the features of the test point well, in other words, relevant to the test point. Since to translate a sentence the source of it is given in advance, we can retrieve a set of training sentence pairs whose source is close to it, and train a regression model only on this small relevant set to predict the translation. Similar strategies have been used in the earlier research in domain adaptation for SMT systems, which are usually found to help significantly. Representative works include, for example, Eck et al. (2004) and Zhao et al. (2004) where online language models have been trained for their SMT systems on the retrieved documents or sentences, and Hildebrand et al. (2005) who proposed the method to train translation models only on the sentence pairs retrieved from the training set, whose source is similar to a given test set. A recent work (Foster and Kuhn, 2007) has further extended these techniques in that it splits the training set into different components (domains) and weights the

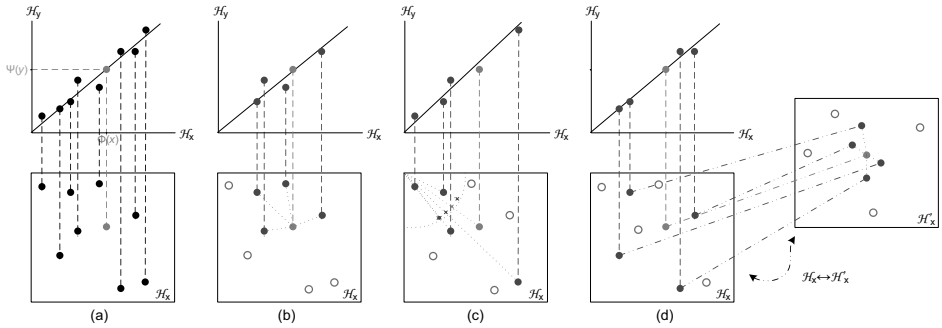

Figure 9.2 Retrieval-based sparse approximation.

models trained separately on those component corpora according to their degrees of relevance to the test domain. While in our case not only does this method adapt the translation model to the test domain, it also achieves a sparse approximation to the original regression hyperplane, which makes this expensive kernel method possible to be applied to large-scale data sets.

This sparse approximation process is demonstrated in figure 9.2 with different metrics to measure how "close" two sentences are. If $(\Phi(\mathbf{x}), \Psi(\mathbf{y}))$ is a test point in the joint feature spaces $\mathcal{H}_x \times \mathcal{H}_y$ and figure 9.2(a) represents the original regression hyperplane learned on the whole training set, the particular methods we use to retrieve the relevant set are described as follows:

- *Euclidean distance*: As shown in figure 9.2(b), the training samples are subselected by picking out those whose source feature vectors have smallest Euclidean distances to $\Phi(\mathbf{x})$ in $\mathcal{H}_x$, where the Euclidean distance between two vectors $\Phi(\mathbf{x})$ and $\Phi(\mathbf{z})$ is computed as

$$D(\mathbf{x}, \mathbf{z}) = \|\Phi(\mathbf{x}) - \Phi(\mathbf{z})\| = \sqrt{\kappa_\Phi(\mathbf{x}, \mathbf{x}) - 2\kappa_\Phi(\mathbf{x}, \mathbf{z}) + \kappa_\Phi(\mathbf{z}, \mathbf{z})}. \tag{9.10}$$

- *Cosine angle distance*: Figure 9.2(c) shows the metric to select the training samples by cosine angle distance in $\mathcal{H}_x$, where the smaller the angle between two vectors is, the closer they are regarded. The angle between $\Phi(\mathbf{x})$ and $\Phi(\mathbf{z})$ is calculated as

$$D(\mathbf{x}, \mathbf{z}) = \arccos \frac{\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle}{\|\Phi(\mathbf{x})\| \|\Phi(\mathbf{z})\|} = \arccos \frac{\kappa_\Phi(\mathbf{x}, \mathbf{z})}{\sqrt{\kappa_\Phi(\mathbf{x}, \mathbf{x}) \kappa_\Phi(\mathbf{z}, \mathbf{z})}}. \tag{9.11}$$

- *tf-idf metric*: Of course it is not necessary to retrieve the data based on the measurements only in $\mathcal{H}_x$. We can map the source part of the training and test data into some other space, and measure the relevance there, as illustrated in figure 9.2(d). One of the methods is the tf-idf (term frequency – inverse document frequency) metric from information retrieval (IR). In the tf-idf metric, a document (sentence in our case) is viewed as a bag of words. Each word is weighted by the

product of its tf-score and idf-score. The tf-score of a word w_i in a sentence $\mathbf{x}$ is its frequency in that sentence normalized by the sentence length, as

$$\mathrm{tf}(w_i, \mathbf{x}) = \frac{C(w_i, \mathbf{x})}{|\mathbf{x}|}, \tag{9.12}$$

where $C(\cdot, \cdot)$ represents the counts, and its idf-score is computed as the logarithm of the quotient of the total number of sentences in the training set and the number of the sentences containing it:

$$\mathrm{idf}(w_i) = \log \frac{|S_{\mathcal{X}}|}{|\{\mathbf{z}|\mathbf{z} \in S_{\mathcal{X}}, w_i \in \mathbf{z}\}|}, \tag{9.13}$$

where we use $S_{\mathcal{X}}$ to denote the source portion of the training set. Finally, the cosine angle distance is used to measure the similarity of two sentences in this weighted bag-of-words feature space.

- *Edit distance*: Besides the tf-idf metric that captures the key words in the sentences, we try the edit distance measure as well, with the expectation to extract the training samples which have similar sentence patterns to the test sentence. The edit distance here refers to the Levenshtein distance, which is given by the minimum number of operations needed to transform one sentence into the other, where an operation is an insertion, deletion, or substitution of a single word.

9.3 Decoding

9.3.1 Preimage Problem

The preimage problem is to find the target sentence $\mathbf{y}$ from the feature vector $\Psi(\mathbf{y})$ predicted in Eq. (9.1). However, in this very high-dimensional case, the predicted $\Psi(\mathbf{y})$ will be very noisy. If we simply round its components to obtain integer counts as in Cortes et al. (2005), some n-grams may be missing, which fails to construct a connected De Bruijn graph, while we will also get many single isolated noisy nodes. Thus, the preimage solution here is achieved by seeking the $\hat{\mathbf{y}}$ that has the minimum loss between its feature vector $\Psi(\hat{\mathbf{y}})$ and our prediction $\Psi(\mathbf{y})$, as

$$\begin{aligned} \hat{\mathbf{y}} &= \arg\min_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \|\mathbf{W}\Phi(\mathbf{x}) - \Psi(\mathbf{y})\|^2 \\ &= \arg\min_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \kappa_\Psi(\mathbf{y}, \mathbf{y}) - 2\mathbf{k}_\Psi(\mathbf{y})(\mathbf{K}_\Phi + \nu\mathbf{I})^{-1}\mathbf{k}_\Phi(\mathbf{x}), \end{aligned} \tag{9.14}$$

where $\mathcal{Y}(\mathbf{x}) \subset \mathcal{Y}$ denotes a finite set covering all potential translations for the given source sentence $\mathbf{x}$, and $\mathbf{k}_\Psi(\mathbf{y}) = (\kappa_\Psi(\mathbf{y}, \mathbf{y}_i)_{1 \le i \le m})$. A proper $\mathcal{Y}(\mathbf{x})$ can be generated according to a lexicon that contains possible translations for every component (word or phrase) in $\mathbf{x}$. But the size of it will grow exponentially with the length of $\mathbf{x}$, which poses an implementation problem for a search algorithm.

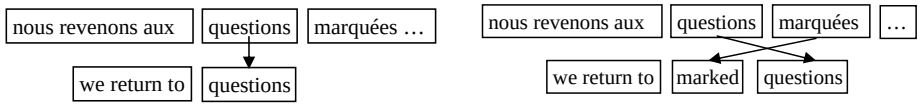

Figure 9.3 Search states with restricted distortion: (a) appending a translated phrase directly; (b) two adjacent phrases exchanging their positions.

9.3.2 Beam Search

In earlier systems, several heuristic search methods have been developed, of which a typical one is introduced in Koehn (2004a) for phrase-based models. Koehn's (2004a) beam search decoder generates the target sentence from left to right in the form of hypotheses (search states). The search process starts from an initial (empty) hypothesis. At each time, it expands a hypothesis by extending the output with a translation of a phrase not yet translated in the source sentence to create a new hypothesis. It estimates a current score and a future score for the new hypothesis, inserts it into a stack indexed by the number of the source words translated, and reranks and prunes the hypotheses in the stack at the same time according to their scores. When all the words in the source sentence are covered, the search reaches final states, among which the highest-scored one is selected as the best translation.

In previous models, the scores for a hypothesis are usually computed incrementally by the sum of a serial of log-probabilities, each of which depends on the features in a local area only, e.g., a phrase pair for a translation model, a word n-gram for a language model, etc. This is very convenient as the scores can be accumulated phrase by phrase during the expansion of hypotheses. However, in our case, because of the $\kappa_\Psi(\mathbf{y}, \mathbf{y})$ item in Eq. (9.14) the expression cannot be decomposed into a sum of subfunctions each involving feature components in a local area only. It means we will not be able to estimate exactly how well a part of the source sentence is translated until we obtain a translation for the entire sentence, which prevents us doing a straightforward beam search similar to Koehn (2004a). Note here, at the same time $\kappa_\Psi(\mathbf{y}, \mathbf{y})$ has the effect of penalizing the bias of the second item in Eq. (9.14) on longer sentences, and therefore it controls the length of the prediction.

To simplify the situation, we restrict the reordering (distortion) of phrases that yield the output sentences by only allowing adjacent phrases to exchange their positions. Now, if we go back to the implementation of a beam search, the current distortion restriction guarantees that in each expansion of the search states (hypotheses) we have $\mathbf{x}^{1:l_x}$ translated to a $\mathbf{y}^{1:l_y}$, either like state (a) or like state (b) in figure 9.3, where l_x is the number of words translated in the source sentence, and l_y is the number of words obtained in the translation.

We assume that if $\mathbf{y}$ is a good translation of $\mathbf{x}$, then $\mathbf{y}^{1:l_y}$ is a good translation of $\mathbf{x}^{1:l_x}$ as well. So we can expect that the squared loss $\|\mathbf{W}\Phi(\mathbf{x}^{1:l_x}) - \Psi(\mathbf{y}^{1:l_y})\|^2$ is

small for the hypothesis yielding a good translation. Accordingly, the hypotheses in the search stacks can thus be reranked with the following score function:

$$\text{Score}(\mathbf{x}^{1:l_x}, \mathbf{y}^{1:l_y}) = \kappa_\Psi(\mathbf{y}^{1:l_y}, \mathbf{y}^{1:l_y}) - 2\mathbf{k}_\Psi(\mathbf{y}^{1:l_y})(\mathbf{K}_\Phi + \nu\mathbf{I})^{-1}\mathbf{k}_\Phi(\mathbf{x}^{1:l_x}). \quad (9.15)$$

Therefore, to solve the preimage problem, we just employ the same beam search algorithm as Koehn (2004a), except we limit the derivation of new hypotheses with the distortion restriction mentioned above.

9.3.3 Complexity Analysis

Inevitably, this score function brings more runtime complexities when compared with traditional probabilistic methods. The time complexity of a naive implementation of the blended n-gram string kernel between two sentences $\mathbf{x}$ and $\mathbf{z}$ is $O(n|\mathbf{x}||\mathbf{z}|)$. So if based on the training set S defined at the beginning of section 9.2.1, the score function in Eq. (9.15) results in a runtime complexity of $O\left(nl_y \sum_{i=1}^{m} |\mathbf{y}_i|\right)$. Note here that $(\mathbf{K}_\Phi + \nu\mathbf{I})^{-1}\mathbf{k}_\Phi(\mathbf{x}^{1:l_x})$ can be precomputed for l_x from 1 to $|\mathbf{x}|$ before the beam search, which calls for $O(m|\mathbf{x}|)$ space and $O\left(n|\mathbf{x}|^2 \sum_{i=1}^{m} |\mathbf{x}_i| + m^2\right)$ time. A trie-based computation of the string kernel (Shawe-Taylor and Cristianini, 2004, chapter 11), can improve the time complexity to $O(n(|\mathbf{x}| + |\mathbf{z}|))$. In addition, further efficiency can be achieved by this approach to compute the whole column $(\kappa(\mathbf{x}, \mathbf{x}_i)_{0 \le i \le m})$ instead of evaluating each entry independently, which is $O\left(n\left(|\mathbf{x}| + \sum_{i=1}^{m} |\mathbf{x}_i|\right)\right)$. Finally, we can improve the complexities of our decoder to $O\left(n\left(l_y + \sum_{i=1}^{m} |\mathbf{y}_i|\right)\right)$ for scoring and $O\left(n\left(|\mathbf{x}|^2 + \sum_{i=1}^{m} |\mathbf{x}_i|\right) + m^2\right)$ for precomputing.

9.4 Experiments

9.4.1 Corpora

Considering the complexities of our model, we choose a small-scale reduced-domain corpus to conceptually demonstrate its effectiveness. The corpus used in the following experiments is the technical manuals of Xerox Corporation in three language pairs, including French-English, German-English, and Spanish-English. In each of them, there are 25 documents which are mostly user guides, operator's guides, or system administration guides, provided with various types of hardware (copiers, printers, multifunction devices, etc.). For each language pair, three documents are taken as our test sets (Test), another two are reserved as a developing set (Dev) for the tuning of the parameters of our systems, and all the remaining 20 documents are used as the training set (Train). Some characteristics of the corpora are summarized in table 9.1.

Table 9.1 Statistics of corpora

		French	English	German	English	Spanish	English
Train	Sentences	56,999		50,122		58,982	
	Tokens	678,832	618,409	533,174	583,473	721,440	639,817
	Vocabulary	13,813	11,545	23,757	11,237	15,104	11,727
Dev	Sentences	1758		1721		1838	
	Tokens	22,682	21,005	17,474	20,407	24,449	21,661
	OOV words	194	170	467	168	142	168
	LM perplexity	–	46.72	–	47.39	–	44.99
Test	Sentences	2198		1788		2285	
	Tokens	24,426	22,589	19,333	21,863	26,326	23,779
	OOV words	156	113	536	104	187	103
	LM perplexity	–	33.39	–	34.91	–	33.36

OOV: out-of-vocabulary; LM: language model.

9.4.2 System Settings

To compare with previous work, we take Pharaoh (Koehn, 2004a), a phrase-based log-linear model, as our baseline system. We set Pharaoh's beam size to 100 and the maximum number of translations per source phrase to 20. The distortion limit of Pharaoh is set to 7, which equals the maximum phrase length. Note here, in this preliminary work this setting reduces Pharaoh's advantages on word reordering, so makes it commensurable to the restricted distortion (allowing swapping for adjacent phrases only) in our regression model.

In addition, there are two parameters to be set for our kernel ridge regression model: the maximum n-gram length n used in the string kernel, and the regularization coefficient ν. As in this case it will be too expensive to do cross-validation we test different values and observe the performance. Preliminary experiments show that translation qualities are insensitive to ν and 0.01 would be a proper value for it. Then, models with n increasing from 2 to 7 are trained on a smaller portion of the training set in each language pair, and evaluated on the respective developing set. As a result, we find that the blended trigram kernel always gives the best performance. This may be due to the distributions of longer word n-grams being too sparse to learn reliable correspondences from resulting in overfitting. Therefore we fix this setting ($n = 3$ and $\nu = 0.01$) in the following experiments.

9.4.3 Ridge Regression Experiments

We train Pharaoh and our blended trigram kernel regression model on the three training sets respectively, and compare their performance on the test sets. In addition, Kneser-Ney smoothed trigram language models are trained for Pharaoh with the SRILM toolkit (Stolcke, 2002) on the English portions of the training sets. Its log-linear parameters are tuned on the development sets based on the minimum-

Table 9.2 Pharaoh vs. kernel ridge regression (KRR). All values are in percentage, except NIST scores.

		BLEU	NIST	METEOR	TER	WER	PER
French–English	Pharaoh	48.38	7.98	**66.77**	38.26	31.71	23.95
	KRR	**49.01**	**8.49**	66.43	**34.78**	**29.12**	**21.50**
German–English	Pharaoh	**37.95**	**7.04**	**58.80**	47.53	41.86	29.80
	KRR	37.64	7.03	58.58	**43.98**	**38.37**	**28.34**
Spanish–English	Pharaoh	55.17	9.08	73.87	28.88	25.70	16.63
	KRR	**56.40**	**9.45**	**73.89**	**26.27**	**23.15**	**15.14**

error-rate training (MERT) method (Och, 2003). To facilitate comparison, we use the same phrase translation table as Pharaoh's to decode our regression model, where the beam size is initially set to 100. (Further experiments on decoding search error rates can be found in section 9.4.5.)

The translation qualities are evaluated by different metrics, including BLEU (Papineni et al., 2002), NIST (Doddington, 2002), METEOR (Banerjee and Lavie, 2005), translation edit rate (TER) (Snover et al., 2006), word error rate (WER), and position-independent word error rate (PER), as shown in table 9.2. In the evaluations only one single reference translation is provided for each sentence. The results are statistically significant with $p < .001$ based on the bootstrap test (Koehn, 2004b).

It can be found that the kernel ridge regression model outperforms Pharaoh in most of the tasks, especially with significant advantages in TER, WER, and PER. But there is a uniform slight decline of our method on BLEU, NIST, and METEOR in the German-English task. A main reason for this could be that German is a morphologically rich language. Accordingly, the evidence for the high-dimensional correspondences among the n-gram features would be too sparse. More detailed analysis can be gained by looking into some sample translation results in table 9.3. If sufficient translation examples can be found in the training set, the kernel ridge regression model will generate very well-structured outputs, such as the first French-English sample and the second Spanish-English sample. But its drawback is that as the n-gram features connecting the phrases are actually functioning similar to a language model, if some of them are weighted too low in the prediction (because of the lack of support of training examples), the output sentence may become broken, e.g., as in the second French-English sample and the two German-English samples. Additional language models will undoubtedly be helpful, of which further discussions are left to section 9.5.

9.4.4 Sparse Approximation Experiments

We test the sparse approximation methods proposed in section 9.2.4 on the French-English portion of the corpus. For each test sentence, a set of relevant training samples are extracted according to the measure of cosine angle distance, Euclidean

Table 9.3 Example translation results

French-English	
Src:	créez un compte utilisateur sur le serveur de numérisation à utiliser comme compte utilisateur du serveur de distribution .
Ref:	create a user account on the scan server , to be used as the distribution server user account .
KRR:	create an user account on the scan server , to be used as the distribution server account .
Pha:	create a user account on the scan server to use as a user account on the distribution server .
Src:	reportez-vous maintenant à la carte d' installation et de configuration correspondant à votre système d' exploitation .
Ref:	refer now to the installation and setup reference card for your operating system .
KRR:	refer to now card installation and configuration for your operating system .
Pha:	now refer to the card installation and configuration for your operating system .
German-English	
Src:	als scan-server kann eines der folgenden systeme benutzt werden :
Ref:	any of the following to function as the scan server :
KRR:	the scan server be of the following used systems :
Pha:	scan server one of the following systems can be used :
Src:	das gerät wird neu gestartet und gibt einen konfigurationsbericht aus . dieser vorgang dauert etwa drei minuten .
Ref:	the device will reboot and print a configuration sheet in approximately 3 minutes .
KRR:	the device will reboot and this a configuration report off the process takes approximately three minutes .
Pha:	the device will reboot and provides a configuration report off . this process takes approximately three minutes .
Spanish-English	
Src:	haga clic en sí en el cuadro de diálogo modificar la carpeta inicio si desea añadir un icono de exploración a la barra de tareas de windows .
Ref:	click yes on the modify startup folder dialog if you wish to add a desktop scanning icon to the windows taskbar .
KRR:	click yes dialog the modify startup folder if to add a scanning icon in the windows taskbar .
Pha:	click yes in the dialog the modify startup folder if you want to add a scanning icon to the windows taskbar .
Src:	asegúrese de que la conexión eléctrica de su máuina satisfaga estos requisitos .
Ref:	ensure that the power connection for your machine satisfies these requirements .
KRR:	ensure that the power connection for your machine meets these requirements .
Pha:	ensure that the attachment electrical machine meets your requirements .

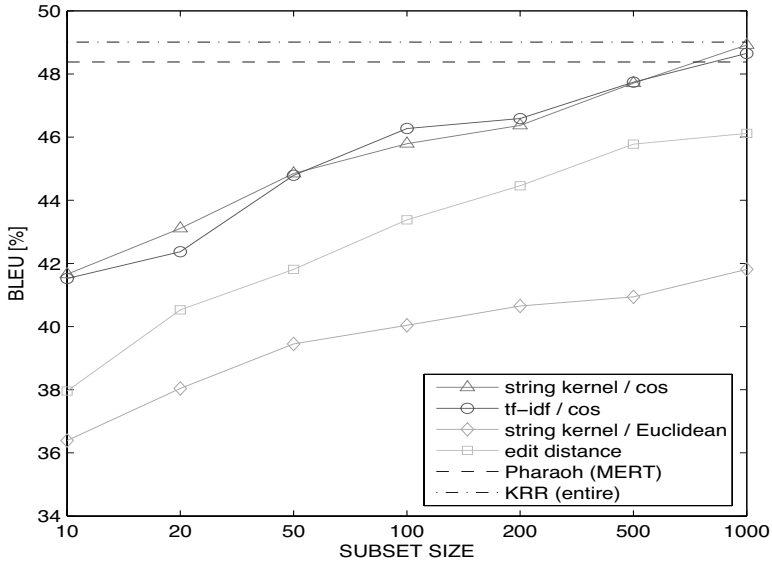

Figure 9.4 Retrieval-based sparse approximation: BLEU score vs. relevant set size.

distance, tf-idf metric, and edit distance, respectively. The BLEU score performance curves of different metrics over relevant set size are illustrated in figure 9.4, with the comparison to Pharaoh and kernel ridge regression trained on the entire training set.

It is shown that the cosine angle distance and the tf-idf metric work almost equally well in this task, and their BLEU score performance is roughly linear to the extracted subset set size in log-domain. When the relevant set size reaches 1000, they outperform Pharaoh by around 0.5% BLEU score, which approaches the result of the regression model trained on the entire training set. The trend suggests that we might be able to gain higher BLEU scores if more relevant training samples are extracted for each test sentence. However, preliminary experiments show that on this data set a relevant set larger than 1000 does not help. This could be because in this closed set, after the retrieved subset reaches a certain size, the rear sentences will be of little relevance to the test point and therefore would correspond to adding noise. Nevertheless, it can be expected that this trend would hold if we were able to retrieve adequate relevant training sentence pairs from an open domain.

9.4.5 Search Errors

Finally, we consider the search errors of our beam search decoding algorithm. We pick out 1000 sentences of length between 5 and 20 words from each test set. All the candidate translations of each test sentence are generated based on the restricted word distortion, which are then reranked by the kernel ridge regression model. Comparing the results to the outputs of our beam search decoder with different beam size, we give a view of the search error rates in table 9.4. An error occurs if

Table 9.4 Search error rate vs. beam size

Beam Size	10	20	50	100	200	500	1000
French–English	4.0%	1.9%	1.3%	1.1%	0.9%	0.8%	0.6%
German–English	6.8%	5.0%	4.1%	2.1%	2.1%	2.1%	2.1%
Spanish–English	3.3%	1.7%	1.1%	1.0%	1.0%	1.0%	0.9%

the decoder failed to find the best ranked sentence. Although we have no effective way to evaluate the decoding quality for long sentences (more than 20 words), the overall performance shows that this decoding strategy is acceptable.

9.5 Further Discussions

At this stage, the superiority of our regression model is limited, but there is ample room for improvement. In this section, we will focus the discussion on the following two aspects: language modeling and linguistic knowledge.

9.5.1 Language Modeling

Language models have proved to be very helpful for SMT, but they are not used in our current system, which potentially is a major drawback. The most straightforward solution will be to add a weight to adjust the strength of the regression-based translation scores and the language model score during the decoding procedure. It can be further extended to using the regression-based translation model just as one of the components in the log-linear framework. Moveover, our regression framework also has the capability to embed a language model into the regression itself. As language models are typically n-gram-based, they match the definition of our feature space. For instance, if we take a trigram model as an example:

$$P(\mathbf{y}) = \prod_{i=1}^{n} P(y_i|y_{i-2}y_{i-1}),$$

where $\mathbf{y} = y_1 y_2 \dots y_n$ is a sentence of length n, and y_i denotes the ith word in it, we can express the logarithm of it as

$$\log P(\mathbf{y}) = \mathbf{V}^\top \Psi(\mathbf{y}),$$

where $\mathbf{V}$ is a vector with components $\mathbf{V}_{y''y'y} = \log P(y|y''y')$, and y, y' and y'' are arbitrary words. Note here, in order to match our blended trigram indexed feature vector $\Psi(\mathbf{y})$, we can make $\mathbf{V}$ of the same dimension as it, with the components corresponding to unigrams and bigrams set to 0 or possibly encoding lower-order language models. Furthermore, in the general case $\mathbf{V}$ is not necessarily based on log-probability, but can be any form of weight vector scoring a sentence $\mathbf{y}$ based on its n-gram features. Accordingly, given source sentence $\mathbf{x}$, our prediction can be scored

directly by the language model $\mathbf{V}$ as $\mathbf{V}^\top \mathbf{W}\Phi(\mathbf{x})$. We add this language model score as an additional loss to the objective function of the kernel ridge regression, as

$$\min \quad \|\mathbf{W}\mathbf{M}_\Phi - \mathbf{M}_\Psi\|_F^2 + \nu_1 \|\mathbf{W}\|_F^2 - \nu_2 \mathbf{V}^\top \mathbf{W}\mathbf{M}_\Phi \mathbf{1} \ , \tag{9.16}$$

where ν_2 is a coefficient to adjust the effect of the language model loss, and $\mathbf{1}$ denotes a vector with components 1. The point is that now we are seeking the matrix $\mathbf{W}$ which balances between the prediction being close to the target feature vector (measured by Euclidean distance) and being a fluent target sentence. By differentiating the expression with respect to $\mathbf{W}$ and setting the result to zero, we can obtain the explicit solution as

$$\mathbf{W} = (\mathbf{M}_\Psi + \nu_2 \mathbf{V}\mathbf{1}^\top)(\mathbf{K}_\Phi + \nu_1 \mathbf{I})^{-1} \mathbf{M}_\Phi^\top. \tag{9.17}$$

9.5.2 Linguistic Knowledge

Besides plain text, our kernel regression method offers a flexible framework for the incorporation of various types of features. Higher-level linguistic knowledge can be utilized easily by selecting corresponding kernels, e.g., string kernels for part-of-speech tag sequences or convolution kernels for parse trees (Collins and Duffy, 2002a), though new decoding problems might arise in some cases. In addition, soft-matching string kernels (Shawe-Taylor and Cristianini, 2004, chapter 11) can be applied to morphologically rich languages to relieve the data sparseness problem. A recent state-of-the-art approach to SMT, the factored translation model introduced by Koehn and Hoang (2007), has also enabled the integration of linguistic annotations into phrase-based translation models. Compared to their method, where the factors can only be self-influential, our model allows interactions among different linguistic features, which could be a potential advantage.

9.6 Conclusion

In this chapter, we present a novel framework for machine translation based on kernel ridge regression, by which comparable performance to previous work has been achieved. Although at this stage the main contribution is still conceptual, its feasibility in an application to the SMT field is demonstrated. As a kernel method, this framework has the advantage of capturing the correspondences among the features of inputs and outputs in a very high-dimensional space. But the drawback is that its computational complexities are much higher than probabilistic models. A solution is sparse approximation as proposed above, which poses the problem of extracting a sufficient amount of relevant bilingual training samples for a given input. Other essential improvements to this model could be the integration of additional language models and the utilization of linguistic knowledge. These issues are left open here, but will be explored in our future work.

Acknowledgments

This work is supported by the European Commission under the IST Project SMART (FP6-033917). We thank Roland Kuhn, George Foster, Cyril Goutte, and Pierre Isabelle for their suggestions and useful discussions on the sparse approximation methods in section 9.2.4. We also thank Xerox Research Centre Europe for providing the corpus used in above experiments.

10 Statistical Machine Translation through Global Lexical Selection and Sentence Reconstruction

Srinivas Bangalore
Stephan Kanthak
Patrick Haffner

Machine translation of a source language sentence involves selecting appropriate target language words and ordering the selected words to produce a well-formed target language sentence. Most of the previous work on statistical machine translation relies on (*local*) associations of target words/phrases with source words/phrases for lexical selection. In contrast, in this chapter, we present a novel approach to lexical selection where the target words are associated with the entire source sentence (*global*) without the need to compute local associations. Further, we present a technique for reconstructing the target language sentence from the selected words. We compare the results of this approach against those obtained from a finite-state based statistical machine translation system which relies on local lexical associations.

10.1 Introduction

The problem of machine translation can be viewed as consisting of two subproblems: (a) lexical selection, where appropriate target language lexical items are chosen for each source language lexical item and (b) lexical reordering, where the chosen target language lexical items are rearranged to produce a meaningful target language string. Most of the previous work on statistical machine translation, as exemplified in Brown et al. (1993), employs a word-alignment algorithm (such as GIZA++ (Och and Ney, 2003)) that provides local associations between source words and target words. The source-to-target word alignments are sometimes augmented with target-to-source word alignments in order to improve the precision of these local associations. Further, the word-level alignments are extended to phrase-level alignments in order to increase the extent of local associations. The phrasal

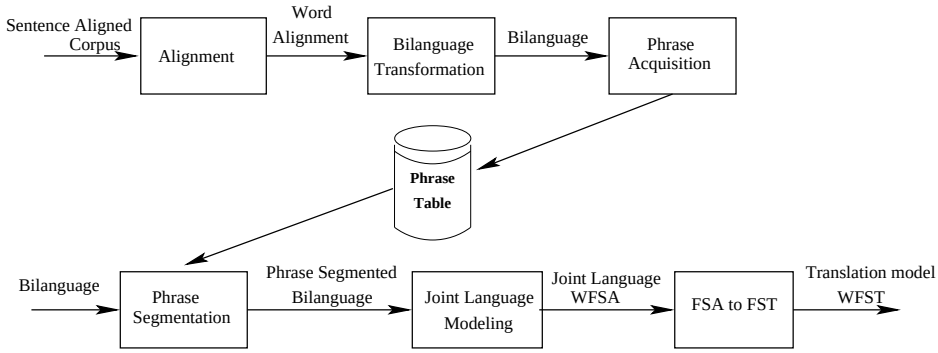

Figure 10.1 Training phases for our system.

associations compile some amount of (*local*) lexical reordering of the target words – those permitted by the size of the phrase. Most of the state-of-the-art machine translation systems use these phrase-level associations in conjunction with a target language model to produce the target sentence. Some of these systems also include a model for global reordering that takes the form of a *distortion* model based on string transductions (Kanthak et al., 2005; Tillmann and Zhang, 2006; Nagata et al., 2006; Xiong et al., 2006) or use hierarchical transduction models motivated by syntax (Wu, 1997; Alshawi et al., 1998; Yamada and Knight, 2001; Chiang, 2005).

In this chapter, we present an alternate approach to lexical selection and lexical reordering. For lexical selection, in contrast to the local approaches of associating target words to source words, we associate the target words to the entire source sentence. The intuition is that there may be lexicosyntactic features of the source sentence (not necessarily a single source word) that might trigger the presence of a target word in the target sentence. Furthermore, it might be difficult to exactly associate a target word to a source word in many situations: (a) when the translations are not exact but paraphrases, and (b) when the target language does not have one lexical item to express the same concept that is expressed by a source word. The extensions of word alignments to phrasal alignments attempt to address some of these situations in addition to alleviating the noise in word-level alignments.

As a consequence of the global lexical selection approach, we no longer have a tight association between source language words/phrases and target language words/phrases. The result of lexical selection is simply a bag of words(phrases) in the target language and the target sentence has to be reconstructed using this bag of words. The target words in the bag, however, might be enhanced with rich syntactic information that could aid in the reconstruction of the target sentence. This approach to lexical selection and sentence reconstruction has the potential to circumvent the limitations of methods based on word alignment for translation between languages with significantly different word order (English-Japanese, for example).

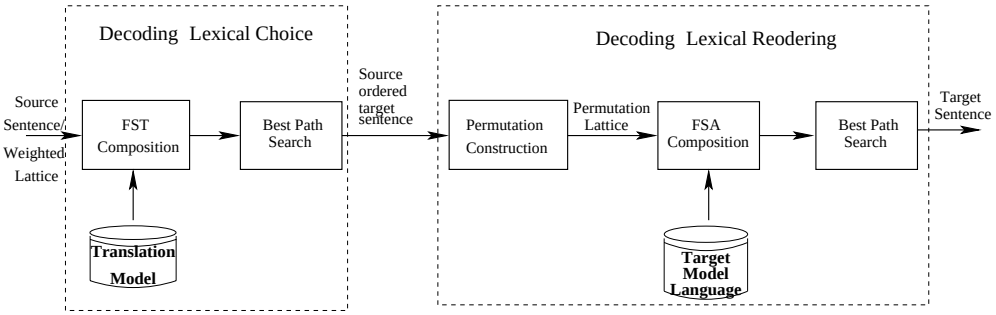

Figure 10.2 Decoding phases for our system.

In this chapter, we present the details of training a global lexical selection model using classification techniques and sentence reconstruction models using permutation automata. We also present a stochastic finite-state transducer (SFST) as an example of an approach that relies on local associations and use it to compare and contrast our approach.

The outline of the chapter is as follows. In section 10.2, we describe in detail the different stages used to train an SFST translation model and discuss the steps in decoding a source input using the trained SFST model. In section 10.3, we present the global lexical selection and the sentence reconstruction models. In section 10.4, we discuss our rationale for choosing the classifer to train the global lexical selection model. In section 10.5, we report the results of the two translation models on a few data sets and contrast the strengths and limitations of the two approaches.

10.2 SFST Training and Decoding

In this section, we describe each of the components of our SFST system shown in figure 10.1 in detail. The SFST approach described here is similar to the approach described in Bangalore and Riccardi (2000) which has subsequently been adopted by Banchs et al. (2005), and Kumar et al. (2006).

10.2.1 Word Alignment

The first stage in the process of training a lexical selection model is obtaining an alignment relation (f) that given a pair of source ($S = s_1 s_2 \ldots s_{|S|}$) and target ($T = t_1 t_2 \ldots t_{|T|}$) language sentences, maps source language words into target language word subsequences, as shown below. For each of the source words (at position i), there is at least one target word (at position j) that the source word is related to or it is unmapped (mapped to ϵ). Also, for the target language sentence, each target word is related to some source word or may be unmapped.

English: I need to make a collect call
Japanese: 私は　コレクト　コールを　かける　必要があります
Alignment: 1 5 0 3 0 2 4

Figure 10.3 Example of a bilingual text with alignment information.

I:私は need:必要があります to;make:コールを a;collect:コレクト call:かける

Figure 10.4 Bilanguage strings resulting from alignments shown in figure 10.3.

$$(\forall i\ \ (\exists j\ \ (s_i, t_j) \in f)\ \vee\ (s_i, \epsilon) \in f)\ \wedge\ (\forall j\ \ (\exists i\ \ (s_i, t_j) \in f)\ \vee\ (\epsilon, t_j) \in f). \quad (10.1)$$

For the work reported in this chapter, we have used the GIZA++ tool (Och and Ney, 2003) which implements a string-alignment algorithm.[1] GIZA++ alignment, however, is asymmetric in that the word mappings are different depending on the direction of alignment – source-to-target or target-to-source. Hence, in addition to the alignment relation f as shown in Eq. (10.1), we train another alignment relation g as shown in Eq. (10.2).

$$(\forall j\ \ (\exists i\ \ (t_j, s_i) \in g)\ \vee\ (t_j, \epsilon) \in g)\ \wedge\ (\forall i\ \ (\exists j\ \ (t_j, s_i) \in g)\ \vee\ (\epsilon, s_i) \in g). \quad (10.2)$$

10.2.2 Bilanguage Representation

From the alignment information (see figure 10.3), we construct a bilanguage representation of each sentence (B^f) in the bilingual corpus. The bilanguage string consists of a sequence of source-target symbol pairs $(s_i, f(s_i))$[2] as shown in Eq. (10.3). When the source word (s_{i-1}) is not aligned to any word of the target string, then the symbol pair in the bilanguage is represented as $(s_{i-1}; s_i, f(s_i))$ (see *to;make* and *a;collect* in figure 10.4).[3] Note that the tokens of a bilanguage could be either ordered according to the word order of the source language or ordered according to the word order of the target language. Here we use source word-ordered bilanguage strings as shown in figure 10.4 corresponding to the alignment shown in figure 10.3.

1. We use the default parameters that come with the GIZA++ installation.
2. If s_i is related to more than one target word, then we create a phrase by concatenating those target words with ";" as a delimiter between them.
3. Note that we could have concatenated the unmapped source word to the previous source word (s_{i-2}).

Table 10.1 Examples of acquired phrases

エイ ティー アンド ティー	A T and T
私の 家の 電話に	to my home phone
私は コレクト コールを かける 必要があります	I need to make a collect call
私は あなたを どうやって お手伝いしましょうか	how may I help you
はい あなたは いただけますか	yes could you

$$
\begin{aligned}
B^f &= b_1^f\ b_2^f\ \ldots\ b_M^f \\
b_i^f &= (s_{i-1}; s_i, f(s_i))\ \ if\ \ f(s_{i-1}) = \epsilon \\
&= (s_i, f(s_{i-1}); f(s_i))\ \ if\ \ s_{i-1} = \epsilon \\
&= (s_i, f(s_i))\ \ otherwise.
\end{aligned} \tag{10.3}
$$

We also construct a source word-ordered bilanguage using the alignment relation g similar to the bilanguage using the alignment relation f as shown in Eq. (10.3).

Thus, the bilanguage corpus obtained by combining the two alignment relations is $B = B_f \cup B_g$.

10.2.3 Bilingual Phrase Acquisition and Local Reordering

While word-to-word translation is only approximating the lexical selection process, phrase-to-phrase mapping can greatly improve the translation of collocations, recurrent strings, etc. Also, the use of phrases allows for reordering of words in the phrase to be in correct target language order, thus solving part of the lexical reordering problem. Moreover, SFSTs built on a phrase-segmented corpus could take advantage of the larger lexical contexts provided by phrases to improve the computation of the probability $P(W_S, W_T)$.

The bilanguage representation could result in multiple words of the source sentence to be mapped to multiple words of the target sentence as a consequence of some words being aligned to ϵ. In addition to these phrases, we compute substrings up to a given length k on the bilanguage string and for each bilanguage substring we reorder its target words to be in the same order as they are in the target language sentence corresponding to that bilanguage string. In Eq. (10.5), we show that tp_i^{i+k} is a substring of the target string and is a permutation of the target words that are related to the source words. For large values of k, we could have entire sentence pairs as entries in the dictionary. A sample of the resulting phrase dictionary is shown in table 10.1.

Using these phrases, we retokenize the bilanguage (B^{phr}) into tokens of source-target word or phrase pairs (Eq.(10.4)). We formulate this process below where $sp_i^{i+k} = s_i s_{i+1} \ldots s_{i+k}$ represents a phrase created by concatenating the tokens $s_i \ldots s_{i+k}$.

$$
\begin{aligned}
B^{phr} &= B^1 \cup B^2 \ldots B^k && (10.4)\\
B^k &= b_1^k\ b_2^k\ \ldots\ b_p^k \\
b_i^k &= (sp_i^{i+k}, tp_i^{i+k}) \\
tp_i^{i+k} &\in perm(f(s_i)f(s_{i+1})\ldots f(s_{i+k}))\ \wedge\ \exists l\ tp_i^{i+k} = t_l t_{l+1}\ldots t_{l+k} && (10.5)
\end{aligned}
$$

10.2.4 SFST Model

From the bilanguage corpus B^{phr}, we train an n-gram language model using language modeling tools (Goffin et al., 2005; Stolcke, 2002), in order to approximate the joint probability $P(S,T)$ as $\prod\ P(s_i, t_i|s_{i-1}, t_{i-1}\ldots s_{i-n-1}, t_{i-n-1})$. The resulting language model is represented as a quasi-deterministic weighted finite-state acceptor($\mathcal{S} \times \mathcal{T} \rightarrow [0,1]$) (Allauzen et al., 2004). The weights are the negative logarithm of the n-gram probability estimates. The symbols on the arcs of this automaton are the pairs of the bilanguage strings treated as a single token ($s_i\_t_i$). These symbols can be interpreted as input and output symbols (s_i:t_i) of a weighted finite-state transducer. Thus the resulting weighted finite-state transducer (*TransFST*) can be used to compute the most likely target T^* for a given input source sentence S. This resulting transducer is in most cases no longer deterministic on the input side.

10.2.5 Decoding

In this section, we describe the decoding process and the global reordering process in detail. During decoding, we are interested in searching for the most likely T^* for a given input source sentence S as shown in Eq.(10.6) and an n-gram approximation model in Eq.(10.7).

$$
\begin{aligned}
T^* &= \arg\min_T \log(P(S,T)) && (10.6)\\
&= \arg\min_T \sum_{i=1}^{M} \log(P(s_i, t_i|s_{i-1}, t_{i-1}\ldots s_{i-n-1}, t_{i-n-1})) && (10.7)
\end{aligned}
$$

Since we represent the translation model as a weighted finite-state transducer ($TransFST$), the decoding process of translating a source input sentence (I_s) amounts to a transducer composition ($\circ$) and search for the best probability path ($BestPath$) and projecting the target sequence (π_1) as shown in Eq.(10.8). Here I_s is a finite-state automaton representing the input sentence. However, this method of decoding allows for the input to be more general finite-state automata (e.g., word graphs or weighted lattices) such as those that are output from a speech recognizer. This decoder can also be composed with a speech recognition model represented as a finite-state transducer for a uniform speech-to-text translation model.

$$T^* = \pi_1(BestPath(I_s \circ TransFST)). \tag{10.8}$$

However, we have noticed that on the development corpus, the resulting target sentences are typically shorter than the intended target sentences. This mismatch may be due to the over/underestimation of probability for backoff and unseen events during the n-gram language model training. In order to alleviate this mismatch, we introduce a negative word insertion penalty model as a mechanism to produce more words in the target sentences.

10.2.6 Word Insertion Model

The word insertion model (WIP) is also encoded as a weighted finite-state automaton and is included in the decoding sequence as shown in Eq. (10.9). The word insertion FST has one state and $|\sum_T|$ number of arcs each weighted with a λ weight representing the word insertion cost. The weight is currently uniform for all target words, but could be selectively set differently for different word classes depending on their impact on the translation quality (for example, verbs have lower weight than determiners). On composition, as shown in Eq. (10.9), the word insertion model penalizes or rewards paths which have more words depending on whether λ is greater than or less than zero.

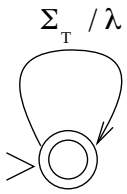

Figure 10.5 Word insertion model, with cost λ on $|\sum_T|$ arcs.

$$T^* = \pi_1(BestPath(I_s \circ TransFST \circ WIP)). \tag{10.9}$$

10.2.7 Global Reordering

Local reordering, as described in section 10.2.3, accounts only for word-order variations within phrases and is restricted by the window size k used for phrase extraction. In order to account for word reordering beyond this window size, we need to permute the words/phrases and select the most likely sentence resulting from the permutation of these phrases. Although the FST approach produces a lattice of locally reordered translations that could be permuted, due to the complexity of such a permutation we limit ourselves to permuting the words/phrases of the

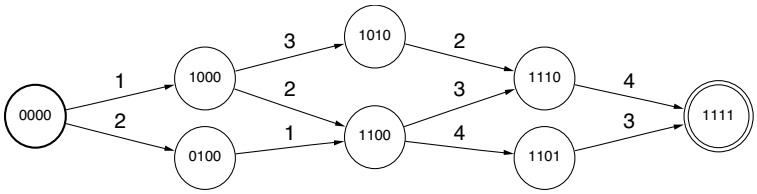

Figure 10.6 Locally constraint permutation automaton for a sentence with 4 positions and a window size of 2.

best translation (T'). In order to compute the likelihood of a string resulting from a permutation, we use an n-gram language model (LM_t) trained on the target language corpus (see also figure 10.2). The search for the most likely target language string can be the computed as shown in Eq. (10.10).

$$T^* = BestPath(perm(T') \circ LM_t). \tag{10.10}$$

Unfortunately, the number of states of the minimal permutation automaton of even a linear automaton (finite-state machine representation of a string) grows exponentially with the number of words of the string. So, instead of creating a full permutation automaton, we choose to constrain permutations to be within a local window of adjustable size (also see Kanthak et al. (2005)). The alternative is to use heuristic forward pruning to limit the explosion of the state space to be searched. Figure 10.6 shows the resulting minimal permutation automaton for an input sequence of 4 words and a window size of 2.

We describe the construction of the constrained permutation automaton shown in figure 10.6. Each state of the automaton is indexed by a bit vector of size equal to the number of words/phrases of the target sentence. Each bit of the bit vector is set to 1 if the word/phrase in that bit position is used on any path from the initial to the current state. The next word for permutation from a given state is restricted by the window size (2 in our case) positions counting from the first yet uncovered position in that state. For example, the state indexed with vector "1000" represents the fact that the word/phrase at position 1 has been used. The next two (window=2) positions are the possible outgoing arcs from this state with labels 2 and 3 connecting to states "1100" and "1010" respectively. The bit vectors of two states connected by an arc differ only by a single bit. Note that bit vectors elegantly solve the problem of recombining paths in the automaton, as states with the same bit vectors can be merged. As a result, a fully minimized permutation automaton has only a single initial and final state.

While the SFST decoding applies naturally to lattices output by a speech recognizer, as discussed in section 10.2.5, global reordering as presented here requires us to first extract n-best lists from lattices. Decoding using global reordering is performed for each entry of the n-best list separately and the best decoded target sentence is picked from the union of the n globally reordered results. An alternative approach, which we have not explored in this chapter, is to transform a speech

recognizer lattice into a word confusion network (Mangu et al., 2000) and permute the arcs of a word confusion network in a manner similar to permuting the words of a string.

10.3 Discriminant Models for Lexical Selection

The approach presented in the previous section is a generative model for statistical machine translation relying on local (phrasal) associations between the source and target sentences. In this section, we present our approach for a *global* lexical selection model which is based on discriminatively trained classification techniques. Discriminant modeling techniques have become the dominant method for resolving ambiguity in speech and natural language processing tasks, outperforming generative models for the same task. Discriminative training for machine translation has been typically restricted to setting a few parameters (such as distortion weights, language model weight, translation model weight) using a minimum-error training method (Och and Ney, 2002). With the exception of Wellington et al. (2006) and Tillmann and Zhang (2006), discriminative training has not been used to directly train the parameters of lexical translation tables. We expect the discriminatively trained global lexical selection models to outperform generatively trained local lexical selection models, as well as provide a framework for incorporating rich morphosyntactic information.

Statistical machine translation can be formulated as a search for the best target sequence that maximizes $P(T|S)$, where S is the source sentence and T is the target sentence. Ideally, $P(T|S)$ should be estimated directly to maximize the conditional likelihood on the training data (discriminant model). However, T corresponds to a sequence with an exponentially large combination of possible labels, and traditional classification approaches cannot be used directly. Although conditional random fields (CRFs) (Lafferty et al., 2001) train an exponential model at the sequence level, in translation tasks such as ours the computational requirements of training such models are prohibitively expensive.

We investigate two approaches to approximating the string level global classification problem, using different independence assumptions. A comparison of the two approaches is summarized in table 10.2.

10.3.1 Sequential Lexical Choice Model

This approach is similar to the SFST approach in that it relies on local associations between the source and target words(phrases). As in the SFST approach, the first step is to construct a bilanguage representation based on a word alignment. We then formulate the translation problem as a sequential local classification problem as shown in Eq. (10.11). We can use a conditional model (instead of a joint model as in SFST) and the parameters are determined using discriminant training, which allows for a richer conditioning context. For the purpose of this chapter, we use

Table 10.2 A comparison of the sequential and bag-of-words lexical choice models

	Sequential lexical model	**Bag-of-words lexical model**
Output target	Target word(s) for each bilanguage position i	Target word given a source sentence
Input features	$BOgram(S, i-d, i+d)$: bag of n-grams in source side of the bilanguage in the interval $[i-d, i+d]$	$BOgram(S, 0, \|S\|)$: bag of n-grams in source sentence
Probabilities	$P(t_i \| BOgram(S, i-d, i+d))$ Independence assumption between the labels	$P(BOW(T) \| BOgram(S, 0, \|S\|))$
Number of classes	One per target word or phrase	
Training samples	One per bilanguage token	One per sentence
Preprocessing	Source/target *word* alignment	Source/target *sentence* alignment

unigrams, bigrams, and trigrams constructed from a window of d tokens around the ith bilanguage token as features ($\Phi(S, i)$, shortened as Φ in the rest of the section) to estimate the probability in Eq. (10.11). Note that although there are $|S|$ number of tokens in the resulting translation ($phrT$), the number of words in the target string (T) could be more or less than $|S|$ since some of the tokens ($phrt_i$) may be multiword phrases or may even be the empty word (ϵ).

$$P(phrT|S) = \prod_{i=1}^{|S|} P(phrt_i|\Phi(S, i-d, i+d)). \tag{10.11}$$

10.3.2 Bag-of-Words Lexical Choice Model

The sequential lexical choice model described in the previous section treats the selection of a lexical choice for a source word in the local lexical context as a classification task. The data for training such models is derived from the word-alignment corpus obtained from alignment algorithms such as GIZA++. The decoded target lexical items have to be further reordered, but for closely related languages the reordering could be incorporated into correctly ordered target phrases as discussed previously.

For pairs of languages with radically different word order (e.g., English-Japanese), there needs to be a global reordering of words similar to the case in the SFST-based translation system. Also, for such differing language pairs, the alignment algorithms such as GIZA++ perform poorly.

These observations prompted us to formulate the lexical choice problem *without* the need for word alignment information. We require a sentence-aligned corpus as before, but we treat the target sentence as a bag of words, or BOW, assigned to the source sentence. Given a source sentence, the goal is to estimate the probability of the presence of a given target word in the target sentence. This is why, instead of producing a target sentence, what we initially obtain is a target bag of words. Each word in the target vocabulary is detected independently, so we have here a very simple use of binary static classifiers. Training sentence pairs are considered as positive examples when the word appears in the target, and negative otherwise. Thus, the number of training examples equals the number of sentence pairs, in contrast to the sequential lexical choice model which has one training example for each token in the bilingual training corpus. The classifier is trained with n-gram features ($BOgrams(S, 0, |S|)$) from the source sentence. During decoding the words with conditional probability greater than a threshold θ are considered as the result of lexical choice decoding.

$$BOW_T^* = \{t \mid P(t|BOgrams(S, 0, |S|)) > \theta\}. \tag{10.12}$$

In order to reconstruct the proper order of words in the target sentence we consider all permutations of words in BOW_T^* and weight them by a target language model. This step is similar to the one described in section 10.2.7 and we indeed use the same implementation here.

The bag-of-words approach can also be modified to allow for length adjustments of target sentences, if we add optional deletions in the final step of permutation decoding. The parameter θ and an additional word deletion penalty can then be used to adjust the length of translated outputs.

In section 10.6, we discuss several issues regarding this model.

10.4 Choosing the Classifier

This section addresses the choice of the classification technique, and argues that one technique that yields excellent performance while scaling well is *binary Maxent* with *L1-regularization*.

10.4.1 Multiclass vs. Binary Classification

The sequential and BOW models represent two different classification problems. In the sequential model, we have a *multiclass* problem where each class t_i is exclusive; therefore, all the classifier outputs $P(t_i|\Phi)$ must be jointly optimized such that $\sum_i P(t_i|\Phi) = 1$. This can be problematic: with one classifier per word in the vocabulary, even allocating the memory during training may exceed the memory capacity of current computers.

On the other hand, in the BOW model, each class can be detected independently, and two different classes can be detected at the same time. This is known as the one-vs.-other scheme: each class is trained with a separate binary classifier where the positive examples belong to the class, and all the other examples are considered as negative. The key advantage over the multiclass scheme is that not all classifiers have to reside in the memory at the same time during training. This also allows for parallelization. Fortunately for the sequential model, we can decompose a multiclass classification problem into separate one-vs.-other problems.

Despite the approximation, the one-vs.-other scheme has been shown to perform as well as the multiclass scheme (Rifkin and Klautau, 2004). As a consequence, we use the same type of binary classifier for the sequential and the BOW models.

The excellent results recently obtained with the SEARN algorithm (Daumé et al., 2006) also suggest that binary classifiers, when properly trained and combined, seem to be capable of matching more complex *structured* output approaches.

10.4.2 Geometric vs. Probabilistic Interpretation

We separate the most popular classification techniques into two broad categories:

- Geometric approaches maximize the width of a separation margin between the classes. The most popular method is the support vector machine (SVM) (Vapnik, 1998).
- Probabilistic approaches maximize the conditional likelihood of the output class given the input features. This logistic regression is also called Maxent as it finds the distribution with *maximum entropy* that properly estimates the average of each feature over the training data (Berger et al., 1996).

In a systematic study on n-gram-based sentence classification (Haffner, 2005), we found that the best accuracy is achieved with nonlinear (or kernel) SVMs, at the expense of a high test time complexity, which is unacceptable for machine translation. On the other hand, linear SVMs and regularized Maxent yield similar performance. In theory, Maxent training, which scales linearly with the number of examples, is faster than SVM training, which scales quadratically with the number of examples. Note that with recent SVM training algorithms that are linear (Joachims, 2006) or optimized for sparse data (Haffner, 2006), SVM training time may no longer be a major issue, even with millions of examples. In our first experiments with lexical choice models, we observed that Maxent slightly outperformed SVMs. Using a single threshold with SVMs, some classes of words were overdetected. This suggests that, as theory predicts, SVMs do not properly approximate the posterior probability. We therefore chose to use Maxent as the best probability approximator.

10.4.3 L1 vs. L2 Regularization

Traditionally, Maxent is regularized by imposing a Gaussian prior on each weight: this L2 regularization finds the solution with the smallest possible weights. However, on tasks like machine translation with a very large number of input features, a Laplacian L1 regularization is highly desirable, as the l_1-norm encourages sparse solutions where a large number of weights are zero.

A new L1-regularized Maxent algorithm was proposed for density estimation (Dudik et al., 2004) and we adapted it to classification. We found this algorithm to converge faster than the most commonly used algorithm in Maxent training, which is L2-regularized L-BFGS (Malouf, 2002). The only type of algorithms with a better training time are based on stochastic gradient descent [4] and critically depend on the use of an L2-regularizer. In the final model, we found the number of parameters using an L1 regularizer to be at least one order of magnitude smaller than the number of parameters using an L2 regularizer.

A detailed derivation of the algorithm is beyond the scope of this chapter, and can be found in Haffner et al. (2005). The most interesting characteristic of this algorithm is that it is entirely *featurecentric*, which is very desirable for lexical selection, as the learning process can be described as the selection of the few source features (among hundreds of thousands) that have an impact on the target word.

While the Maxent framework starts with a featurecentric description, its most common implementations encourage a nonsparse solution with a large number of features, not only because of the L2-regularizer but also because the weight vector is updated with a gradient descent procedure that will modify a large proportion of the weights at each iteration. In our case, both the definition of the regularizer in our loss function, and the greedy optimization technique we use are entirely featurecentric and encourage sparsity.

- Our regularizer is the *weighted* l_1-norm of the weight vector $R = \sum_k \beta_k |w_k|$. This formula can be derived through convex duality from a slack added to each feature and comes with a principled method (Dudik et al., 2004) to estimate the regularization metaparameters β_k for each feature, so that no cross-validation is needed. This is in contrast with other Lasso-like sparse logistic regression algorithms, which have a single constant regularization metaparameter for all features (Krishnapuram et al., 2005).
- Our optimization algorithm only modifies one weight at each iteration. The greedy procedure, inspired by Adaboost (Freund and Schapire, 1996), selects the weight w_j whose modification causes the largest drop in the loss function. For a hypothetical new weight such that $w'_j = w_j + \delta$ and $w'_k = w_k$ for $k \neq j$, this requires computing the variation of the loss function $\mathcal{L}(\mathbf{w}') - \mathcal{L}(\mathbf{w})$ as a function of δ, such that the value of δ minimizing this function can be obtained. In Haffner et al. (2005), we

4. Code available in `http://leon.bottou.org/projects/sgd`

Table 10.3 Statistics about the training and development data 05 and 06.

	Training		Dev 2005		Dev 2006	
	Chinese	English	Chinese	English	Chinese	English
Sentences	46,311		506		489	
Running words	351,060	376,615	3826	3897	5214	6362*
Vocabulary	11,178	11,232	931	898	1136	1134*
Singletons	4348	4866	600	538	619	574*
OOVs [%]	-	-	0.6	0.3	0.9	1.0
ASR WER [%]	-	-	-	-	25.2	-
Perplexity	-	-	33	-	86	-
# references	-	-	16		7	

* First of multiple reference translations only.

show that, in the case of conditional Maxent, one can only minimize a bound on this loss variation.

We also face a search over all features j which can be costly, as we compute the optimal δ for each j. We restrict this search to a subset updated at each iteration by discarding the features with no impact on the loss, and adding the features which have not been examined for the longest time.

10.5 Data and Experiments

We have performed experiments on the IWSLT06 Chinese-English training and development sets from 2005 and 2006 (Paul, 2006). The data are traveler task expressions such as seeking directions, expressions in restaurants, and travel reservations. Table 10.3 presents some statistics on the data sets. It must be noted that while the 2005 development set matches the training data closely, the 2006 development set has been collected separately and shows slightly different statistics for average sentence length, vocabulary size, and out-of-vocabulary words. Also, the 2006 development set contains no punctuation marks in Chinese, but the corresponding English translations have punctuation marks. We also evaluated our models on the speech recognition output and we report results on the 1-best output of the speech recognizer. The 1-best Chinese speech recognition word error rate is 25.2%.

For the experiments, we tokenized the Chinese sentences into character strings and trained the models discussed in the previous sections. Also, we trained a punctuation prediction model using the Maxent framework on the Chinese character strings in order to insert punctuation marks into the 2006 development data set. The resulting character string with punctuation marks is used as input to the translation decoder. For the 2005 development set, punctuation insertion was not needed since the Chinese sentences already had the true punctuation marks.

Table 10.4 Results (BLEU) scores for the three different models on the transcriptions for development sets 2005 and 2006 and ASR 1-best for development set 2006

	Dev 2005	Dev 2006	
	Text	Text	ASR 1-best
SFST	51.8	19.5	16.5
SeqMaxEnt	53.5	19.4	16.3
BOWMaxEnt	59.9	19.3	16.6

In table 10.4 we present the results of the three different translation models – SFST, sequential Maxent and bag-of-words Maxent – on the data described above using the BLEU metric (Papineni et al., 2002). The test set has multiple reference translations as indicated in table 10.3 and the BLEU metric takes this into account. There are a few interesting observations that can be made based on these results. First, on the 2005 development set, the sequential Maxent model outperforms the SFST model, even though the two models were trained starting from the same GIZA++ alignment. The difference, however, is due to the fact that Maxent models can cope with increased lexical context[5] and the parameters of the model are discriminatively trained. The more surprising result is that the bag-of-words Maxent model significantly outperforms the sequence Maxent model. The reason is that the sequence Maxent model relies on the word alignment, which, if erroneous, results in incorrect predictions by the sequential Maxent model. The bag-of-words model, on the other hand does not rely on the word-level alignment and can be interpreted as a discriminatively trained model of dictionary lookup for a target word in the context of a source sentence.

The second set of observations relates to the difference in performance between the 2005 development set and the 2006 development set. As indicated in the data release document, the 2006 set was collected in a very different manner compared to the 2005 set. As a consequence, the mismatch between the training set and the 2006 development set in terms of lexical and syntactic difference can be seen precipitating the lower performance. Due to this mismatch, the performance of the Maxent models is not very different from that of the SFST model, indicating the lack of good generalization across different genres. However, we believe that the Maxent framework allows for incorporation of linguistic features such as morphological and syntactic information that could potentially help in generalization across genres. For translation of ASR 1-best, we see a systematic degradation of about 3% in BLEU score compared to translating the transcription.

In order to compensate for this mismatch between the 2005 and 2006 data sets, we computed a ten-fold average BLEU score by including 90% of the 2006 development set into the training set and using 10% of the 2006 development set for testing,

5. We use six words to the left and right of a source word for sequential Maxent, but only two words preceding source and target words for the SFST approach.

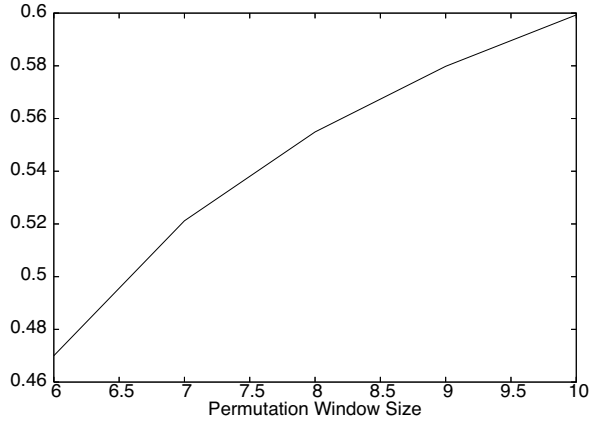

Figure 10.7 Improvement in BLEU score with the increase in size of the permutation window.

each time. The average BLEU score across these ten runs increased to 22.8 for the BOWMaxEnt model.

In figure 10.7, we show the improvement of the BLEU score with the increase in the permutation window sizes. We had to limit ourselves to a permute window size of 10 due to memory limitations, even though the curve has not plateaued. We anticipate using pruning techniques so that we can increase the window size further.

10.5.1 UN and Hansard Corpora

In order to test the scalability of the global lexical selection approach, we also performed lexical selection experiments on the United Nations (Arabic-English) corpus and the Hansard (French-English) corpus using the SFST-based model and the BOW Maxent model. We used one million training sentence pairs and tested on 994 test sentences for the UN corpus. We used the same training and test split for the Hansard corpus as in Zens and Ney (2004) (1.4 million training sentence pairs and 5432 test sentences). The vocabulary sizes for the two corpora are shown in table 10.5. Also in table 10.5 are the results in terms of F-measure between the words in the reference sentence and the decoded sentences. We can see that the BOW model outperforms the SFST-MT model on both corpora significantly. This is due to a systematic 10% relative improvement for open-class words, as they benefit from a much wider context. BOW performance on closed-class words is higher for the UN corpus but lower for the Hansard corpus.

However, the BLEU scores for the BOW models after permutation is far lower than those for the SFST-based model for both the corpora. The reason for the poor performance is that the search space of strings created by a permutation window of 10 is woefully inadequate given that the average lengths of the sentences in the UN and Hansard corpora are 26.8 and 16.0 words respectively. In the case of the IWSLT

Table 10.5 Lexical selection results (F-measure) and BLEU scores on the Arabic-English UN corpus and the French-English Hansard corpus. In parenthesis are the F-measures for open-class and closed-class lexical items.

Corpus	Vocabulary		SFST-MT		BOW	
	Source	Target	F-measure	BLEU score	F-measure	BLEU score
UN	252,571	53,005	64.6 (60.5/69.1)	30.9	69.5 (66.2/72.6)	15.6
Hansard	100,270	78,333	57.4 (50.6/67.7)	27.0	60.8 (56.5/63.4)	14.6

data, the permutation window is sufficient to create a full search space given that the average length of the sentences is less than ten words.

One can imagine segmenting the target language sentences into smaller units (of less than ten words) such as clauses, and reconstructing the sentences by restricting permutations to within clauses. However, this approach would entail aligning the target language clauses with the source language clauses which in turn requires a consistent syntactic parse of the source and target sentences. Training of robust syntactic parsers relies on large parse-annotated sentence corpora which are typically insufficient or absent for many languages. Hence, in this chapter, we explore the limits of techniques that do not rely on syntactic information. However, we believe that judicious use of syntactic information in our models should further improve overall performance.

10.6 Discussion

The bag-of-words approach is very promising because it performs reasonably well despite considerable and easy-to-identify losses in the transfer of information between the source and the target. The first and most obvious loss is about word position. The only information we currently use to restore the target word position is the target language model. The information about the grammatical role of a word in the source sentence is completely lost. The language model might fortuitously recover this information if the sentence with the correct grammatical role for the word happens to be the maximum likelihood sentence in the permutation automaton.

We are currently working toward incorporating syntactic information on the target words so as to be able to recover some of the grammatical role information lost in the classification process. In preliminary experiments, we have associated the target lexical items with supertag information (Bangalore and Joshi, 1999). Supertags are labels that provide linear ordering constraints as well as grammatical relation information. Although associating supertags to target words increases the class set for the classifier, we have noticed that the degradation in the F-score is on the order of 3% across different corpora. The supertag information can then be

exploited in the sentence construction process. The use of supertags in phrase-based SMT systems has been shown to improve results (Hassan et al., 2006).

A less obvious loss is the number of times a word or concept appears in the target sentence. *Function words* like "the" and "of" can appear many times in an English sentence. In the model discussed in this chapter, we index each occurrence of the function word with a counter. In order to improve this method, we are currently exploring a technique where the function words serve as attributes (e.g., definiteness, tense, case) on the contentful lexical items, thus enriching the lexical item with morphosyntactic information.

A third issue concerning the bag-of-words model is the problem of *synonyms* – target words which translate the same source word. Suppose that in the training data, target words t_1 and t_2 are, with equal probability, translations of the same source word. Then, in the presence of this source word, the probability of detecting the corresponding target word, which is normally 0.8 (we assume some noise), will be, because of discriminant learning, split equally between t_1 and t_2, that is, 0.4 and 0.4. Because of this synonym problem, we immediately see that the threshold has to be set lower than 0.5, which is observed experimentally. However, if we set the threshold to 0.3, both t_1 and t_2 will be detected in the target sentence, and we found this to be a major source of undesirable insertions.

The BOW approach is different from the parsing-based approaches (Melamed, 2004; Zhang and Gildea, 2005; Cowan et al., 2006) where the translation model tightly couples the syntactic and lexical items of the two languages. The decoupling of the two steps in our model has the potential to generate paraphrased sentences not necessarily isomorphic to the structure of the source sentence.

Our approach is also different from Carpuat and Wu (2007b), where lexical choice is modeled as a word-sense disambiguation problem. Their approach relies on local associations being computed by first using an alignment step and then using position-specific, syntactic, and collocational features for disambiguating the word (or phrase) senses. Their approach is similar to the sequential lexical choice model presented in this chapter. The results of the word-sense disambiguation model are then incorporated into the decoder of a phrase-based machine translation model during the translation of a sentence.

10.7 Conclusions

We view machine translation as consisting of lexical selection and lexical reordering steps. These two steps need not necessarily be sequential and could be tightly integrated. We have presented the weighted finite-state transducer model of machine translation where lexical choice and a limited amount of lexical reordering are tightly integrated into a single transduction. We have also presented a novel approach to translation where these two steps are loosely coupled and the parameters of the lexical choice model are discriminatively trained using a maximum entropy model. The lexical reordering model in this approach is achieved using a permu-

tation automaton. We have evaluated these two approaches on the 2005 and 2006 IWSLT development sets and shown that the techniques scale well to the Hansard and UN corpora.

11 Discriminative Phrase Selection for SMT

Jesús Giménez
Lluís Màrquez

This chapter explores the application of discriminative learning to the problem of phrase selection in statistical machine translation. Instead of relying on maximum likelihood estimates for the construction of translation models, we suggest using local classifiers which are able to take further advantage of contextual information. Local predictions are softly integrated into a factored phrase-based statistical machine translation (MT) system leading to a significantly improved lexical choice, according to a heterogeneous set of metrics operating at different linguistic levels. However, automatic evaluation has also revealed that improvements in lexical selection do not necessarily imply an improved sentence grammaticality. This fact evinces that the integration of dedicated discriminative phrase translation models into the statistical framework requires further study. Besides, the lack of agreement between metrics based on different similarity assumptions indicates that more attention should be paid to the role of automatic evaluation in the context of MT system development.

11.1 Introduction

Traditional statistical machine translation (SMT) architectures, like the one implemented in this chapter, address the translation task as a search problem (Brown et al., 1990). Given an input string in the source language, the goal is to find the output string in the target language which maximizes the product of a series of probability models over the search space defined by all possible partitions of the source string and all possible reorderings of the translated units. This search process implicitly decomposes the translation problem into two separate but interrelated subproblems:

Word selection, also referred to as *lexical choice*, is the problem of deciding, given a word (or phrase) f in the source sentence, which word (or phrase) e in the target language is the most appropriate translation. This problem is mainly

addressed by translation models, which serve as probabilistic bilingual dictionaries, typically accounting for $P(f|e)$, $P(e|f)$ of $P(e, f)$. Translation models provide, for each word (or phrase) in the source vocabulary, a list of translation candidates with associated translation probabilities. During the search there is another component which addresses word selection, the language model. This component helps the decoder to move toward translations which are more appropriate, in terms of grammaticality, in the context of what is known so far about the target sentence being generated.

Word ordering refers to the problem of deciding which position the translation candidate e must occupy in the target sentence. This problem is mainly addressed by the reordering model which allows for certain word movement inside the sentence. Again, the language model helps the decoder, in this case, to move toward translations which preserve a better word ordering according to the rules of the target language.

In standard phrase-based SMT systems, like that described by Koehn et al. (2003), the estimation of these models is fairly simple. For instance, translation models are built on the basis of relative frequency counts, i.e., maximum likelihood estimates (MLEs). Thus, all the occurrences of the same source phrase are assigned, no matter what the context is, the same set of translation probabilities. For that reason, recently, there is a growing interest in the application of discriminative learning, both for word ordering (Chang and Toutanova, 2007; Cowan et al., 2006) and, especially, for word selection (Bangalore et al., 2007; Carpuat and Wu, 2007b; Giménez and Màrquez, 2007a; Stroppa et al., 2007; Vickrey et al., 2005).

Interest in discriminative word selection has also been motivated by recent results in word sense disambiguation (WSD). The reason is that SMT systems perform an implicit kind of WSD, except that instead of working with word senses, SMT systems operate directly on their potential translations. Indeed, recent semantic evaluation campaigns have treated word selection as a separate task, under the name of *multilingual lexical sample* (Chklovski et al., 2004; Jin et al., 2007). Therefore, the same discriminative approaches that have been successfully applied to WSD should be also applicable to SMT. In that spirit, instead of relying on MLE for the construction of the translation models, approaches to discriminative word selection suggest building dedicated discriminative translation models which are able to take a wider feature context into account. Lexical selection is, therefore, addressed as a classification task. For each possible source word (or phrase) according to a given bilingual lexical inventory (e.g., the translation model), a distinct classifier is trained to predict lexical correspondences based on local context. Thus, during decoding, for every distinct instance of every source phrase, a distinct context-aware translation probability distribution is potentially available.

In this chapter, we extend the work presented in Giménez and Màrquez (2007a). First, in section 11.2, we describe previous and current approaches to dedicated word selection. Then, in section 11.3, our approach to discriminative phrase translation (DPT) is fully described. We present experimental results on the application of DPT models to the Spanish-to-English translation of European Parliament pro-

ceedings. In section 11.4, prior to considering the full translation task, we measure the local accuracy of DPT classifiers at the isolated *phrase translation* task in which the goal is not to translate the whole sentence but only individual phrases without having to integrate their translations in the context of the target sentence. We present a comparative study on the performance of four different classification settings based on two different learning paradigms, namely support vector machines and maximum entropy models.

In section 11.5, we tackle the full translation task. We have built a state-of-the-art factored phrase-based SMT system based on linguistic data views at the level of shallow parsing (Giménez and Màrquez, 2005, 2006). We compare the performance of DPT- and MLE-based translation models built on the same parallel corpus and phrase alignments. DPT predictions are integrated into the SMT system in a *soft* manner, by making them available to the decoder as an additional log-linear feature so they can fully interact with other models (e.g., language, distortion, word penalty, and additional translation models) during the search. We separately study the effects of using DPT predictions for all phrases as compared to focusing on a small set of very frequent phrases.

This chapter has also served to study the problem of machine translation evaluation. We have applied a novel methodology for *heterogeneous* automatic MT evaluation which allows for separately analyzing quality aspects at different linguistic levels, e.g., lexical, syntactic, and semantic (Giménez and Màrquez, 2007b). This methodology also offers a robust mechanism to combine different similarity metrics into a single measure of quality based on *human likeness* (Giménez and Màrquez, 2008). We have complemented automatic evaluation results through error analysis and by conducting a number of manual evaluations. Our main conclusions are summarized in section 11.6.

11.2 Approaches to Dedicated Word Selection

Brown et al. (1991a,b) were the first to suggest using dedicated WSD models in SMT. In a pilot experiment, they integrated a WSD system based on mutual information into their French-to-English word-based SMT system. Results were limited to the case of binary disambiguation, i.e., deciding between only two possible translation candidates, and to a reduced set of very common words. A significantly improved translation quality was reported according to a process of manual evaluation. However, apparently, they abandoned this line of research.

Some years passed until these ideas were recovered by Carpuat and Wu (2005b), who suggested integrating WSD predictions into a phrase-based SMT system. In a first approach, they did so in a *hard* manner, either for decoding, by constraining the set of acceptable word translation candidates, or for postprocessing the SMT system output, by directly replacing the translation of each selected word with the WSD system prediction. However, they did not manage to improve MT quality. They encountered several problems inherent in the SMT architecture. In particular,

they described what they called the *language model effect* in SMT: "The lexical choices are made in a way that heavily prefers phrasal cohesion in the output target sentence, as scored by the language model." This problem is a direct consequence of the hard interaction between their WSD and SMT systems. WSD predictions cannot adapt to the surrounding target context. In a later work, Carpuat and Wu (2005a) analyzed the converse question, i.e., they measured the WSD performance of SMT systems. They showed that dedicated WSD models significantly outperform the WSD ability of current state-of-the-art SMT models. Consequently, SMT should benefit from WSD predictions.

Simultaneously, Vickrey et al. (2005) studied the application of *context-aware* discriminative word selection models based on WSD to SMT. Similarly to Brown et al. (1991b), they worked with translation candidates instead of word senses, although their models were based on maximum entropy and dealt with a larger set of source words and higher levels of ambiguity. However, they did not approach the full translation task but limited themselves to the *blank-filling* task, a simplified version of the translation task, in which the target context surrounding the word translation is available. They did not encounter the language model effect because (i) the target context was fixed a priori, and (ii) they approached the task in a soft way, i.e., allowing WSD-based models to interact with other models during decoding.

Following similar approaches to that of Vickrey et al. (2005), Cabezas and Resnik (2005) and Carpuat et al. (2006) used WSD-based models in the context of the full translation task to aid a phrase-based SMT system. They reported a small improvement in terms of BLEU score, possibly because they did not work with phrases but limited their work to single words. Besides, they did not allow WSD-based predictions to interact with other translation probabilities. More recently, a number of authors, including us, have extended these works by moving from words to phrases and allowing discriminative models to fully cooperate with other phrase translation models. Moderately improved MT quality results have been obtained (Bangalore et al., 2007; Carpuat and Wu, 2007a,b; Giménez and Màrquez, 2007a; Stroppa et al., 2007; Venkatapathy and Bangalore, 2007). All these works were being elaborated at the same time, and were presented in very near dates with very similar conclusions. We further discuss the differences between them in section 11.6.

In a different approach, Chan et al. (2007) used a WSD system to provide additional features for the hierarchical phrase-based SMT system based on bilingual parsing developed by Chiang (2005, 2007). These features were intended to give a bigger weight to the application of rules that are consistent with WSD predictions. A moderate but significant improvement in terms of BLEU was reported.

As another alternative direction, Specia et al. (2007) has suggested applying inductive logic programming techniques to the problem of word selection. They have presented very promising results on a small set of words from different grammatical categories. They have not yet approached the full translation task.

Overall, apart from evincing that this is a very active research topic, some of the works listed in this section show clear evidence that dedicated word selection models might be useful for the purpose of MT.

11.3 Discriminative Phrase Translation

Instead of relying on MLE estimation to score the phrase pairs (f_i, e_j) in the translation table, DPT models deal with the translation of every source phrase f_i as a multiclass classification problem, in which every possible translation of f_i is a class. As an illustration, in figure 11.1 we show a real example of Spanish-to-English phrase translation, in which the source phrase "creo que," in this case translated as "I believe that," has several possible candidate translations.

11.3.1 Problem Setting

Training examples are extracted from the same training data as in the case of conventional MLE-based models, i.e., a phrase-aligned parallel corpus (see section 11.5.1). We use each occurrence of each source phrase f_i to generate a positive training example for the class corresponding to the actual translation e_j of f_i in the given sentence, according to the automatic phrase alignment. Let us note that phrase translation is indeed a multilabel problem. Since word alignments allow words both in the source and the target sentence to remain unaligned, the phrase extraction algorithm employed allows each source phrase to be aligned with more than one target phrase, and vice versa, with the particularity that all possible

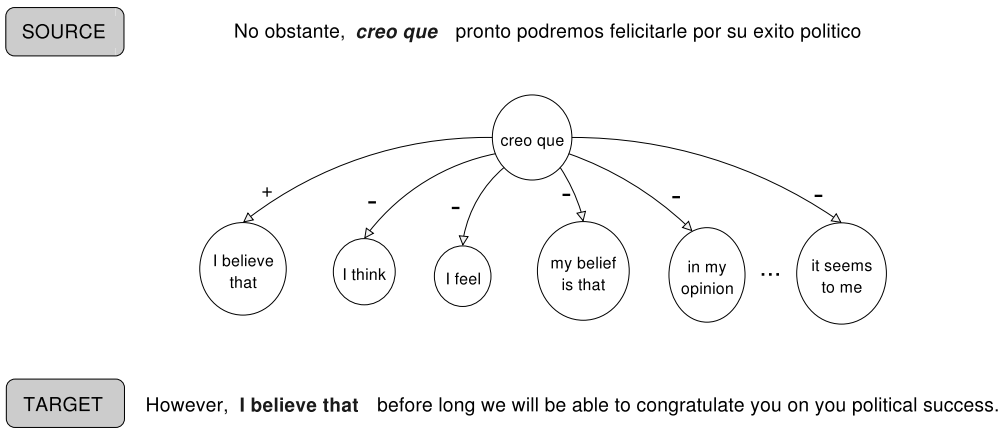

Figure 11.1 An example of phrase translation.

phrase translations are embedded or overlap. However, since the final goal of DPT classifiers is not to perform local classification but to provide a larger system with translation probabilities, in our current approach no special treatment of multilabel cases has been performed.

11.3.2 Learning

There exist a wide variety of learning algorithms which can be applied to the multiclass classification scenario defined. In this work we have focused on two families, namely support vector machines (SVMs)[1] (Vapnik, 1995; Cristianini and Shawe-Taylor, 2000), and maximum entropy (ME)[2] (Jaynes, 1957; Berger et al., 1996). We have tried four different learning settings based, respectively, on linear binary SVMs (SVMlinear), degree-2 polynomial binary SVMs (SVMpoly2), linear multiclass SVMs (SVMmc), and multiclass ME models (MaxEnt).

Binary vs. Multiclass Classification

While approaches 3 and 4 implement by definition a multiclass classification scheme, approaches 1 and 2 are based on binary classifiers, and, therefore, the multiclass problem must be binarized. We have applied *one-vs.-all* binarization, i.e., a binary classifier is learned for every possible translation candidate e_j in order to distinguish between examples of this class and all the rest. Each occurrence of each source phrase f_i is used to generate a positive example for the actual class (or classes) corresponding to the aligned target phrase (or phrases), and a negative example for the classes corresponding to the other possible translations of f_i. At classification time, given a source phrase f_i, SVMs associated to each possible candidate translation e_j of f_i will be applied, and the most confident candidate translation will be selected as the phrase translation.

Support Vector Machines vs. Maximum Entropy

The SVM and ME algorithms are based on different principles. While the SVM algorithm is a linear separator which relies on margin maximization, i.e., on finding the hyperplane which is more distant to the closest positive and negative examples, ME is a probabilistic method aiming at finding the least biased probability distribution that encodes certain given information by maximizing its entropy. An additional interest in comparing the behavior of SVM and ME classifiers is motivated by the nature of the global MT system architecture. While the outcomes of ME classifiers are probabilities which can be easily integrated into the SMT frame-

1. SVMs have been learned using the SVM^{light} and SVM^{struct} packages by Thorsten Joachims, which are freely available at `http://svmlight.joachims.org` (Joachims, 1999).
2. ME models have been learned using the `MaxEnt` package by Zhang Le, which is freely available at `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`.

work, SVM predictions are unbounded real numbers. This issue will be further discussed in section 11.5.2.

Linear vs. Polynomial Kernels

Although SVMs allow for a great variety of kernel functions (e.g., polynomial, Gaussian, sigmoid, etc.), in this work, based on results published in recent WSD literature (Lee and Ng, 2002; Màrquez et al., 2006), we have focused on linear and polynomial kernels of degree 2 (see section 11.4). The main advantage of using linear kernels, over other kernel types, is that this allows for working in the primal formulation of the SVM algorithm and, thus, to take advantage of the extreme sparsity of example feature vectors. This is a key factor, in terms of efficiency, since it permits a considerable speedup of both the training and classification processes (Giménez and Màrquez, 2004a). The use of linear kernels requires, however, the definition of a rich feature set.

11.3.3 Feature Engineering

We have built a feature set which considers different kinds of information, always from the source sentence. Each example has been encoded on the basis of the *local context* of the phrase to be disambiguated and the *global context* represented by the whole source sentence.

As for the local context, we use n-grams ($n \in \{1, 2, 3\}$) of word forms, parts of speech (POS), lemmas, and base phrase chunking IOB labels[3] in a window of five tokens to the left and to the right of the phrase to disambiguate. We also exploit parts of speech, lemmas, and chunk information inside the source phrase, because, in contrast to word forms, these may vary and thus report very useful information. The text has been automatically annotated using the following tools: SVMTool for POS tagging (Giménez and Màrquez, 2004b), Freeling for lemmatization (Carreras et al., 2004), and Phreco for base phrase chunking (Carreras et al., 2005). These tools have been trained on the WSJ Penn Treebank (Marcus et al., 1993) for the case of English, and on the 3LB Treebank (Navarro et al., 2003) for Spanish, and, therefore, rely on their tag sets. However, in the case of parts of speech, because tag sets take into account fine morphological distinctions, we have additionally defined several coarser classes grouping morphological variations of nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, determiners, and punctuation marks.

As for the global context, we collect topical information by considering content words (i.e., nouns, verbs, adjectives, and adverbs) in the source sentence as a bag of lemmas. We distinguish between lemmas at the left and right of the source phrase being disambiguated.

3. IOB labels are used to denote the position (*i*nside, *o*utside, or *b*eginning of a chunk) and, if applicable, the type of chunk.

Table 11.1 An example of phrase translation features

Source Sentence creo$_{[creer:VMI:B-VP]}$ que$_{[que:CS:B-CONJP]}$ pronto$_{[pronto,AQ,O]}$ podremos$_{[podremos,VMS,B-VP]}$ felicitarle$_{[felicitarle,VMN,I-VP]}$ por$_{[por,SP,B-PP]}$ su$_{[su,DP,B-NP]}$ éxito$_{[exito,NC,I-NP]}$ político$_{[politico,AQ,I-NP]}$ ·$_{[.,Fp,O]}$

Source phrase features

Lemma n-grams	(creer)$_1$, (que)$_2$, (creer,que)$_1$
POS n-grams	(VMI)$_1$, (CS)$_2$, (VMI,CS)$_1$
Coarse POS n-grams	(V)$_1$, (C)$_2$, (V,C)$_1$
Chunk n-grams	(B-VP)$_1$, (B-CONJP)$_2$, (B-VP,B-CONJP)$_1$

Source sentence features

Word n-grams	(pronto)$_1$, (podremos)$_2$, (felicitarle)$_3$, (por)$_4$, (su)$_5$, (_,pronto)$_{-1}$, (pronto,podremos)$_1$, (podremos,felicitarle)$_2$, (felicitarle,por)$_3$, (por,su)$_4$, (_,_,pronto)$_{-2}$, (_,pronto,podremos)$_{-1}$, (pronto,podremos,felicitarle)$_1$, (podremos,felicitarle,por)$_2$, (felicitarle,por,su)$_3$
Lemma n-grams	(pronto)$_1$, (poder)$_2$, (felicitar)$_3$, (por)$_4$, (su)$_5$, (_,pronto)$_{-1}$, (pronto,poder)$_1$, (poder,felicitar)$_2$, (felicitar,por)$_3$, (por,su)$_4$, (_,_,pronto)$_{-2}$, (_,pronto,poder)$_{-1}$, (pronto,poder,felicitar)$_1$, (poder,felicitar,por)$_2$, (felicitar,por,su)$_3$
POS n-grams	(AQ)$_1$, (VMS)$_2$, (VMN)$_3$, (SP)$_4$, (DP)$_5$, (_,AQ)$_{-1}$, (AQ,VMS)$_1$, (VMS,VMN)$_2$, (VMN,SP)$_3$, (SP,DP)$_4$, (_,_,AQ)$_{-2}$, (_,AQ,VMS)$_{-1}$, (AQ,VMS,VMN)$_1$, (VMS,VMN,SP)$_2$, (VMN,SP,DP)$_3$
Coarse POS n-grams	(A)$_1$, (V)$_2$, (V)$_3$, (S)$_4$, (D)$_5$ (_,A)$_{-1}$, (A,V)$_1$, (V,V)$_2$, (V,S)$_3$, (S,D)$_4$ (_,A,V)-1, (_,_,A)$_{-2}$, (A,V,V)$_1$, (V,V,S)$_2$, (V,S,D)$_3$
Chunk n-grams	(O)$_1$, (B-VP)$_2$, (I-VP)$_3$, (B-PP)$_4$, (B-NP)$_5$, (_,O)$_{-1}$, (O,B-VP)$_1$, (B-VP,I-VP)$_2$, (I-VP,B-PP)$_3$, (B-PP,B-NP)$_4$, (_,_,O)$_{-2}$, (_,O,B-VP)$_{-1}$, (O,B-VP,I-VP)$_1$, (B-VP,I-VP,B-PP)$_2$, (I-VP,B-PP,B-NP)$_3$
Bag-of-lemmas	left = ∅ right = { pronto, poder, felicitar, éxito, político }

As an illustration, table 11.1 shows the feature representation for the example depicted in figure 11.1. At the top, the reader may find the sentence annotated at the level of shallow syntax (following a "word$_{[lemma:PoS:IOB]}$" format). The corresponding source phrase and source sentence features are shown below. We have not extracted any feature from the target phrase, nor the target sentence, nor the correspondence (i.e., word alignments) between source and target phrases. The reason is that, in this work, the final purpose of DPT models is to aid an existing SMT system to make better lexical choices during decoding, and using these types of features would have forced us to build a more complex decoder.

11.4 Local Phrase Translation

Analogously to the *word translation* task definition by Vickrey et al. (2005), rather than predicting the sense of a word according to a given sense inventory, in *phrase translation* the goal is to predict the correct translation of a *phrase*, for a given target language, in the context of a sentence. This task is simpler than the full translation task in that phrase translations of different source phrases do not have

Table 11.2 Numerical description of the set of "all" phrases

#occurrences	#phrases	Phrase length	#phrases	Phrase entropy	#phrases
(100, 500]	23,578	**1**	7004	**[0, 1)**	6154
(500, 1000]	3340	**2**	12,976	**[1, 2)**	11,648
(1000, 5000]	2997	**3**	7314	**[2, 3)**	8615
(5000, 10,000]	417	**4**	2556	**[3, 4)**	3557
(10,000, 100,000]	295	**5**	799	**[4, 5)**	657
> 100,000	22			**[5, 6)**	18

to interact in the context of the target sentence. However, it provides an insight into the gain potential.

11.4.1 Data Sets and Settings

We have used the data from the Openlab 2006 Initiative[4] promoted by the TC-STAR Consortium.[5] This test suite is entirely based on European Parliament proceedings[6] covering April 1996 to May 2005. We have focused on the Spanish-to-English task. The training set consists of 1,281,427 parallel sentences. After performing phrase extraction over the training data (see details in section 11.5.1), also discarding source phrases occurring only once (around 90%), translation candidates for 1,729,191 source phrases were obtained. In principle, we could have built classifiers for all these source phrases. However, in many cases learning could be either unfruitful or not necessary at all. For instance, 27% of these phrases are not ambiguous (i.e., have only one associated possible translation), and most phrases count on few training examples. Based on these facts, we decided to build classifiers only for those source phrases with more than one possible translation and 100 or more occurrences. Besides, due to the fact that phrase alignments have been obtained automatically and, therefore, include many errors, source phrases may have a large number of associated phrase translations. Most are wrong and occur very few times. We have discarded many of them by considering only as possible phrase translations those which are selected more than 0.5% of the time as the actual translation.[7] The resulting training set consists of 30,649 Spanish source phrases. Table 11.2 presents a brief numerical description of the phrase set. For instance, it can be observed that most phrases are trained on less than 5000 examples. Most of them are length-2 phrases and most have an entropy lower than 3.

4. `http://tc-star.itc.it/openlab2006/`
5. `http://www.tc-star.org/`
6. `http://www.europarl.eu.int/`
7. This value was empirically selected so as to maximize the local accuracy of classifiers on a small set of phrases of a varying number of examples.

Table 11.3 Evaluation scheme for the local phrase translation task

	Evaluation scheme	
#examples	**Development and test**	**Test only**
2–9	leave-one-out	
10–99	10-fold cross-validation	
100–499	5-fold cross-validation	
500–999	3-fold cross-validation	
1000–4999	train(80%)–dev(10%)–test(10%)	train(90%)–test(10%)
5000–9999	train(70%)–dev(15%)–test(15%)	train(80%)–test(20%)
> 10,000	train(60%)–dev(20%)–test(20%)	train(75%)–test(25%)

As to feature selection, we discarded features occurring only once in the training data, and constrained the maximum number of dimensions of the feature space to 100,000, by discarding the less frequent features.

11.4.2 Evaluation

Local DPT classifiers are evaluated in terms of accuracy against automatic phrase alignments, which are used as the gold standard. Let us note that, in the case of multilabel examples, we count the prediction by the classifier as a hit if it matches any of the classes in the solution. Moreover, in order to maintain the evaluation feasible, a heterogeneous evaluation scheme has been applied (see table 11.3). Basically, when there are few examples available we apply cross-validation, and the more examples available, the fewer folds are used. Besides, because cross-validation is costly, when there are more than 1000 examples available we simply split them into training, development, and test sets, keeping most of the examples for training and a similar proportion of examples for development and test. Also, as the number of examples increases, the smaller proportion is used for training and the bigger proportion is held out for development and test. In all cases, we have preserved, when possible, the proportion of samples of each phrase translation so folders do not get biased.

11.4.3 Adjustment of Parameters

Supervised learning algorithms are potentially prone to overfit training data. There are, however, several alternatives to fight this problem. In the case of the SVM algorithm, the contribution of training errors to the objective function of margin maximization is balanced through the C regularization parameter of the soft margin approach (Cortes and Vapnik, 1995). In the case of the ME algorithm, the most popular method is based on the use of a Gaussian prior on the parameters of the model, whose variance, σ^2, may be balanced (Chen and Rosenfeld, 1999). Learning parameters are typically adjusted so as to maximize the accuracy of local classifiers over held-out data. In our case, a greedy iterative strategy has been followed. In the

Table 11.4 Numerical description of the representative set of 1000 phrases selected

#occurrences	#phrases	Phrase length	#phrases	Phrase entropy	#phrases
(100, 500]	790	**1**	213	**[1, 2)**	467
(500, 1000]	100	**2**	447	**[2, 3)**	362
(1000, 5000]	92	**3**	240	**[3, 4)**	139
(5000, 10,000]	11	**4**	78	**[4, 5)**	31
(10,000, 50,000]	7	**5**	22	**[5, 6)**	1

first iteration several values are tried. In each following iteration, n values around the top scoring value of the previous iteration are explored at a resolution of $1/n$ times the resolution of the previous iteration, and so on, until a maximum number of iterations I is reached.[8]

11.4.4 Comparative Performance

We present a comparative study of the four learning schemes described in section 11.3.2. The C and σ^2 parameters have been adjusted. However, because parameter optimization is costly, taking into account the large number of classifiers involved, we have focused on a randomly selected set of 1000 representative source phrases with a number of examples in the [100, 50,000] interval. Phrases with a translation entropy lower than 1 have also been discarded. A brief numerical description of this set is available in table 11.4.

Table 11.5 shows comparative results, in terms of accuracy. The local accuracy for each source phrase is evaluated according to the number of examples available, as described in table 11.3. DPT classifiers are also compared to the *most frequent translation* baseline (MFT), which is equivalent to selecting the translation candidate with highest probability according to MLE. The macro column shows macroaveraged results over all phrases, i.e., the accuracy for each phrase counts equally toward the average. The micro column shows microaveraged accuracy, where each test example counts equally.[9] The Optimal columns correspond to the accuracy computed on optimal parameter values, whereas the Default columns correspond to the accuracy computed on default C and σ^2 parameter values. In the case of SVMs, we have used the SVM^{light} default value for the C Parameter.[10] In the case

8. In our case, $n = 2$ and $I = 3$. In the case of the C parameter of SVMs, first iteration values are set to 10^i (for $i \in [-4, +4]$), while for the σ^2 of ME prior Gaussians, values are $\{0, 1, 2, 3, 4, 5\}$.

9. The contribution of each phrase to microaveraged accuracy has been conveniently weighted so as to avoid the extra weight conferred to phrases evaluated via cross-validation.

10. The C parameter for each binary classifier is set to $\frac{\sum (\vec{x_i}\vec{x_i})^{-1}}{N}$, where $\vec{x_i}$ is a sample vector and N corresponds to the number of samples. In the case of multiclass SVMs, the default value is 0.01.

Table 11.5 Phrase translation accuracy over a selected set of 1000 phrases based on different learning types vs. the MFT baseline

	Optimal		Default	
Model	**macro (%)**	**micro (%)**	**macro (%)**	**micro (%)**
MFT	64.72	67.63	64.72	67.63
SVMlinear	70.59	74.12	69.31	73.51
SVMpoly2	**71.10**	**74.70**	**69.79**	**73.86**
SVMmc	69.93	73.39	57.56	63.15
MaxEnt	71.08	74.34	67.38	70.69

of ME, we have set σ^2 to 1 for all classifiers. The reason is that this was the most common return value, with a frequency over 50% of the cases of the parameter tuning process on the selected set of 1000 phrases.

When the C and σ^2 are properly optimized, all learning schemes, except linear multiclass SVMs, exhibit a similar performance, with a slight advantage in favor of polynomial SVMs. The increase with respect to the MFT baseline is comparable to that described by Vickrey et al. (2005). These results are, taking into account the differences between both tasks, also coherent with results attained in WSD (Agirre et al., 2007). However, when default values are used, ME models suffer a significant decrease, and multiclass SVMs fall even below the MFT baseline. Therefore, in these cases, a different parameter adjustment process for every phrase is required.

11.4.5 Overall Performance

The aim of this subsection is to analyze which factors have a bigger impact on the performance of DPT classifiers applied to the set of *all* phrases. In this scenario, no matter how greedy the process is, the adjustment of the C and σ^2 becomes impractical. For that reason we have used fixed default values. In the case of SVMs, for the sake of efficiency, we have limited our study to the use of linear kernels.

Phrase translation results are shown in table 11.6. Again, phrases are evaluated according to the number of examples available, as described in table 11.3. We distinguish between the case of using *all* the 30,649 phrases counting on 100 or more examples (columns 1 and 2), and the case of considering only a small subset of 317 very *frequent* phrases occurring more than 10,000 times (columns 3 and 4). The first observation is that both DPT learning schemes outperform the MFT baseline when default learning parameters are used, linear SVMs clearly being the best. The second observation is that the difference, in terms of microaveraged accuracy gain with respect to the MFT baseline, between using all phrases and focusing on a set of very frequent ones is very small. The reason is that the set of frequent phrases dominates the evaluation with 51.65% of the total number of test cases. In contrast, macroaveraged results confer a significantly wider advantage to DPT models applied to the set of frequent phrases, especially in the case of linear SVMs. This result is significant, taking into account the high results of the MFT baseline

Table 11.6 Overall phrase translation accuracy

Model	All		Frequent	
	macro (%)	**micro (%)**	**macro (%)**	**micro (%)**
MFT	70.51	80.49	79.77	86.12
SVMlinear	74.52	**85.48**	86.32	**91.33**
MaxEnt	72.73	82.53	82.31	87.94

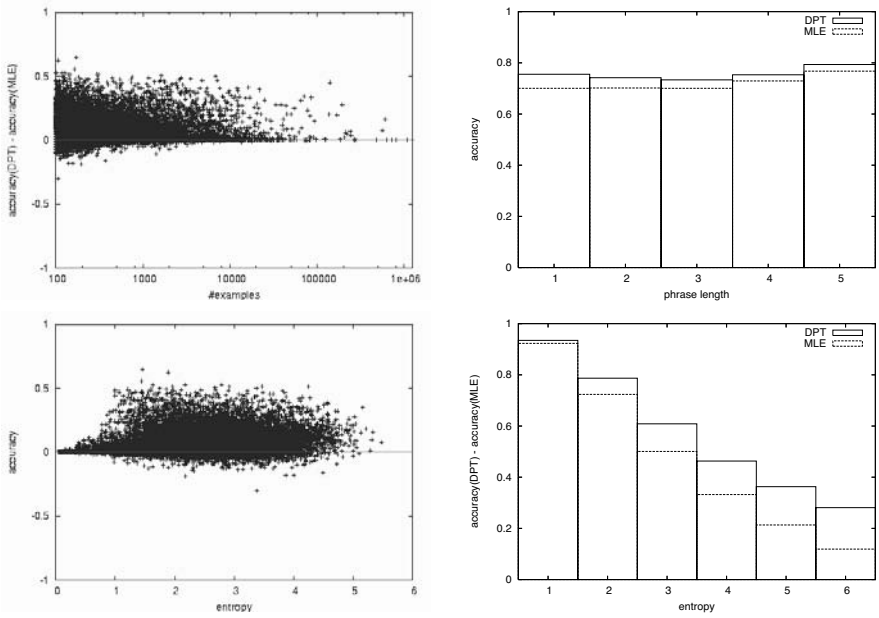

Figure 11.2 Analysis of phrase translation results

on this set. A third, marginal, observation is that frequent phrases are easier to disambiguate, presumably because of their lower entropy (see MFT performance).

In figure 11.2 we analyze several factors which have a direct influence on the behavior of DPT classifiers. All plots correspond to the case of linear SVMs. For instance, the top-left plot shows the relationship between the local accuracy gain and the number of training examples for all source phrases. As expected, DPT classifiers trained on fewer examples exhibit the most unstable behavior, yielding a maximum accuracy gain of 0.65 and a maximum decrease of 0.30. However, in general, with a sufficient number of examples (over 10,000), DPT classifiers outperform the MFT baseline. It can also be observed that for most of the phrases trained on more than around 200,000 examples, the accuracy gain is very low. The reason, however, is in the fact that these are phrases with very low translation entropy, mostly stop words, such as punctuation marks (*“.”*, *“,”*), determiners (*el*, *la*, *los*, *las*, *un*, *una*), or conjunctions and prepositions (*y*, *de*, *en*, *a*). There is a very interesting positive case, that of *que*, which acts mostly as a conjunction or relative

pronoun, and that most often gets translated into "that" or "which." This phrase, which appears more than 600,000 times in the data with a translation entropy of 1.68, attains an accuracy gain of 0.16.

The top-right plot shows the relationship between microaveraged accuracy and source phrase length. There is improvement across all phrase lengths, but, in general, the shorter the phrase, the larger the improvement. This plot also indicates that phrases up to length 3 are on average harder to disambiguate than longer phrases. Thus, there seems to be a tradeoff between phrase length, level of ambiguity (i.e., translation entropy), and number of examples. Shorter phrases are harder because they exhibit higher ambiguity. DPT is a better model for these phrases because it is able to properly take advantage of the large number of training examples. Longer phrases are easier to model because they present a lower ambiguity. Midlength phrases are hardest because they present a high ambiguity and not many examples.

We further investigate this issue in the two bottom plots. The bottom-left plot shows the relationship between the local accuracy gain and translation entropy for all source phrases. It can be observed that for phrases with entropy lower than 1 the gain is very small, while for higher entropy levels the behavior varies. In order to clarify this scenario, we analyze the relationship between microaveraged accuracy and phrase translation entropy at different intervals (bottom-right plot). As expected, the lower the entropy, the higher the accuracy. Interestingly, it can also be observed that as the entropy increases, the accuracy gain in favor of DPT models increases as well.

11.5 Exploiting Local DPT Models for the Global Task

In this section, we analyze the impact of DPT models when the goal is to translate the whole sentence. First, we describe our phrase-based SMT baseline system and how DPT models are integrated into the system. Then, some aspects of evaluation are discussed, with special focus on the adjustment of the parameters governing the search process. Finally, MT results are evaluated and analyzed, and several concrete cases are commented on.

11.5.1 Baseline System

Our system follows a standard phrase-based SMT architecture. This involves three main components:

- *Translation model.* For translation modeling, we follow the approach by Koehn et al. (2003), in which phrase pairs are automatically induced from word alignments. In our case, however, we have built richer word alignments by working with *linguistic data views* up to the level of shallow syntax, as described in Giménez and Màrquez (2005, 2006). This can be seen as a particular case of the recently emerged factored

MT models (Koehn and Hoang, 2007). Phrase alignments are extracted from a word-aligned parallel corpus linguistically enriched with part-of-speech information, lemmas, and base phrase chunk labels. Text has been automatically annotated using the tools described in section 11.3.3. We have used the *GIZA++ SMT Toolkit*[11] to generate word, POS, lemma, and chunk label alignments (Och and Ney, 2003). We have followed the *global phrase extraction* strategy described in Giménez and Màrquez (2005), i.e., a single translation table is built on the union of alignments corresponding to different linguistic data views. Phrase extraction is performed following the *phrase-extract* algorithm described by Och (2002). This algorithm takes as input a word-aligned parallel corpus and returns, for each sentence, a set of phrase pairs that are *consistent* with word alignments. A phrase pair is said to be consistent with the word alignment if all the words within the source phrase are only aligned with words within the target phrase, and vice versa. We have worked with the union of source-to-target and target-to-source alignments, with no heuristic refinement. Only phrases up to length 5 are considered. Also, phrase pairs appearing only once are discarded, and phrase pairs in which the source/target phrase is more than three times longer than the target/source phrase are ignored. Phrase pairs are scored on the basis of relative frequency (i.e., maximum likelihood estimates, MLEs).

- *Language model.* We use the *SRI language modeling toolkit* (Stolcke, 2002). Language models are based on word trigrams. Linear interpolation and Kneser-Ney discounting have been applied for smoothing.
- *Search algorithm.* We use the *Pharaoh* stack-based beam search decoder (Koehn, 2004a), which naturally fits with the previous tools. Keeping with usual practice, in order to speed up the translation process, we have fixed several of the decoder parameters. In particular, we have limited the number of candidate translations to 30, the maximum beam size (i.e., stack size) to 100, and used a beam threshold of 10^{-5} for pruning the search space. We have also set a distortion limit of 4 positions.

This architecture was extended by Och and Ney (2002) for considering additional *feature functions* further than the language and translation probability models. Formally:

$$\hat{e} = \arg\max_{e} \{\log P(e|f)\} \approx \arg\max_{e} \left\{ \sum_{m=1}^{M} \lambda_m h_m(e, f) \right\}.$$

The integration of DPT predictions into this scheme is straightforward. Models are combined in a log-linear fashion:

11. `http://www.fjoch.com/GIZA++.html`

$$\begin{aligned}\log P(e|f) \approx\ & \lambda_{lm} \log P(e) + \lambda_g \log P_{\text{MLE}}(f|e) + \lambda_d \log P_{\text{MLE}}(e|f) \\ & + \lambda_{\text{DPT}} \log P_{\text{DPT}}(e|f) + \lambda_d \log P_d(e,f) + \lambda_w \log w(e).\end{aligned}$$

$P(e)$ stands for the language model probability. $P_{\text{MLE}}(f|e)$ corresponds to the MLE-based generative translation model, whereas $P_{\text{MLE}}(e|f)$ corresponds to the analogous discriminative model. $P_{\text{DPT}}(e|f)$ corresponds to the DPT model which uses DPT predictions in a wider feature context. Finally, $P_d(e,f)$ and $w(e)$ correspond to the distortion and word penalty models.[12] The λ parameters controlling the relative importance of each model during the search must be adjusted. We further discuss this issue in subsection 11.5.5.

11.5.2 Soft Integration of DPT Predictions

We consider every instance of f_i as a separate classification problem. In each case, we collect the classifier outcome for all possible phrase translations e_j of f_i. In the case of ME classifiers, outcomes are directly probabilities. However, in the case of SVMs, outcomes are unbounded real numbers. We transform them into probabilities by applying the *softmax function* described by Bishop (1995):

$$P(e_j|f_i) = \frac{e^{\gamma\ score_{ij}}}{\sum_{k=1}^{K} e^{\gamma\ score_{ik}}},$$

where K denotes the number of possible target phrase translations for a given source phrase f_i, and score_{ij} denotes the outcome for target phrase e_j according to the SVM classifier trained for f_i. In order to verify the suitability of this procedure, we computed rejection curves for the estimated output probabilities with respect to classification accuracy. For that purpose, we have used the representative set of 1000 phrases from subsection 11.4.4. This set offers almost 300,000 predictions. In order to calculate rejection curves, the probability estimates for these predictions are sorted in decreasing order. At a certain level of rejection (n%), the curve plots the classifier accuracy when the lowest scoring n% subset is rejected. We have collected values for 100 rejection levels at a resolution of 1%. We tested different values for the γ parameter of the softmax function. The selected final value is $\gamma = 1$. In figure 11.3 (left plot) we plot the rejection curve for linear SVMs. For the sake of comparison, the rejection curve for ME classifiers is also provided (right plot). It can be observed that both rejection curves are increasing and smooth, indicating a good correlation between probability estimates and classification accuracy.[13]

12. We have used default *Pharaoh*'s word penalty and distortion models.
13. Other transformation techniques can be found in the recent literature. For instance, Platt (2000) suggested using a sigmoid function.

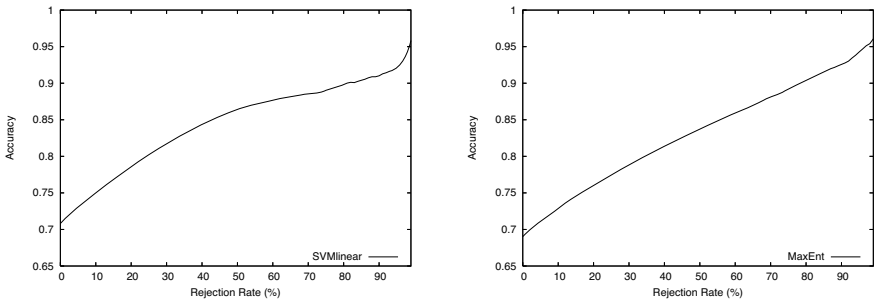

Figure 11.3 Rejection curves. Linear SVMs + softmax (left) vs. ME (right).

At translation time, we do not constrain the decoder to use the translation e_j with highest probability. Instead, we make all predictions available and let the decoder choose. We have precomputed all DPT predictions for all possible translations of all source phrases appearing in the test set. The input text is conveniently transformed into a sequence of identifiers,[14] which allows us to uniquely refer to every distinct instance of every distinct word and phrase in the test set. Translation tables are accordingly modified so that each distinct occurrence of every single source phrase has a distinct list of phrase translation candidates with their corresponding DPT predictions. Let us note that, as described in section 11.4.1, for each source phrase, not all associated target translations which have an MLE-based prediction also have a DPT prediction, but only those with a sufficient number of training examples. In order to provide equal opportunities for both models, we have incorporated translation probabilities for these phrases into the DPT model by applying linear discounting.

As an illustration, table 11.7 shows a fragment of the translation table corresponding to the phrase *creo que* in the running example. Notice how this concrete instance has been properly identified by indexing the words inside the phrase (creo que $\rightarrow$ creo$_{14}$ que$_{441}$). We show MLE-based and DPT predictions (columns 3 and 4, respectively) for several phrase candidate translations sorted in decreasing MLE probability order. The first observation is that both methods agree on the top-scoring candidate translation, "I believe that." However, the distribution of the probability mass is significantly different. While in the case of the MLE-based model, there are three candidate translations clearly outscoring the rest, concentrating more than 70% of the probability mass, in the case of the DPT model predictions give a clear advantage to the top-scoring candidate although with less probability, and the rest of the candidate translations obtain a very similar score.

14. In our case a sequence of w_i tokens, where w is a word and i corresponds to the number of occurrences of word w seen in the test set before the current occurrence number. For instance, the source sentence in the example depicted in figure 11.1 is transformed into *creo*$_{14}$ *que*$_{441}$ *pronto*$_0$ *podremos*$_0$ *felicitarle*$_0$ *por*$_{109}$ *su*$_0$ *éxito*$_3$ *político*$_4$.$_{366}$.

Table 11.7 An example of a translation table

f_i	e_j	$P_{MLE}(e\|f)$	$P_{DPT}(e\|f)$
...			
creo$_{14}$ que$_{441}$	i believe that	0.3624	0.2405
creo$_{14}$ que$_{441}$	i think that	0.1975	0.0506
creo$_{14}$ que$_{441}$	i think	0.1540	0.0475
creo$_{14}$ que$_{441}$	i feel that	0.0336	0.0511
creo$_{14}$ que$_{441}$	i think it	0.0287	0.0584
creo$_{14}$ que$_{441}$	i believe that it	0.0191	0.0487
creo$_{14}$ que$_{441}$	i think that it	0.0114	0.0498
creo$_{14}$ que$_{441}$	believe that	0.0108	0.0438
creo$_{14}$ que$_{441}$	i believe that this	0.0077	0.0482
creo$_{14}$ que$_{441}$	i believe it	0.0060	0.0439
...			

Using this technique for integrating DPT predictions into the system, we have avoided having to implement a new decoder. However, because translation tables may become very large, it involves a severe extra cost in terms of memory and disk consumption. Besides, it imposes a serious limitation on the kind of features the DPT system may use. In particular, features from the target sentence under construction and from the correspondence between source and target (i.e., alignments) cannot be used.

11.5.3 Settings

We use the data sets described in section 11.4.1. Besides, for evaluation purposes, we count on a separate set of 1008 sentences. Three human references per sentence are available. We have randomly split this set into two halves, which are respectively used for development and test.

11.5.4 Evaluation

Evaluating the effects of using DPT predictions in the full translation task presents two serious difficulties. In the first place, the actual room for improvement caused by better translation modeling is smaller than estimated in section 11.4. This is mainly due to the SMT architecture itself which relies on a search over a probability space in which several models cooperate. For instance, in many cases errors caused by a poor translation modeling may be corrected by the language model. In a recent study over the same data set (Spanish-to-English translation of the Openlab 2006 corpus), Vilar et al. (2006) found that only around 28% of the errors committed by their SMT system were related to word selection. In half of these cases errors are caused by a wrong word-sense disambiguation, and in the other half the word sense is correct but the lexical choice is wrong. In the second place, most conventional

automatic evaluation metrics have not been designed for this purpose and may, therefore, not be able to reflect possible improvements attained due to a better word selection. For instance, n-gram-based metrics such as BLEU (Papineni et al., 2002) tend to favor longer string matchings, and are, thus, biased toward word ordering. In order to cope with evaluation difficulties we have applied several complementary actions, which are described below.

Heterogeneous Automatic MT Evaluation

Most existing metrics limit their scope to the lexical dimension. However, recently, there have been several attempts to take into account deeper linguistic levels. For instance, ROUGE (Lin and Och, 2004a) and METEOR (Banerjee and Lavie, 2005) may consider stemming. Additionally, METEOR may perform a lookup for synonymy in WordNet (Fellbaum, 1998). We may find as well several syntax-based metrics (Liu and Gildea, 2005; Amigó et al., 2006; Owczarzak et al., 2007; Mehay and Brew, 2007), and even metrics operating at the level of shallow semantics (Giménez and Màrquez, 2007b) and semantics (Giménez, 2007). For the purpose of performing heterogeneous automatic MT evaluations, we use the IQ$_{\text{MT}}$ package (Giménez and Amigó, 2006), which provides a rich set of more than 500 metrics at different linguistic levels.[15] For our experiments, we have selected a representative set of around 50 metrics, based on different similarity criteria:

- *Lexical similarity*
  - **BLEU-n | BLEUi-n:** Accumulated and individual BLEU scores for several n-gram levels ($n = 1...4$) (Papineni et al., 2002).
  - **NIST-n | NISTi-n:** Accumulated and individual NIST scores for several n-gram levels ($n = 1...5$) (Doddington, 2002).
  - **GTM-e:** General text matching F-measure, for several values of the e parameter controlling the reward for longer matchings ($e = 1...3$) (Melamed et al., 2003).
  - **METEOR:** F-measure based on unigram alignment (Banerjee and Lavie, 2005):
    - **METEOR$_{exact}$:** only "exact" module.
    - **METEOR$_{porter}$:** "exact" and porter_stem".
    - **METEOR$_{wnstm}$:** "exact", "porter_stem" and "wn_stem".
    - **METEOR$_{wnsyn}$:** "exact", "porter_stem", "wn_stem" and "wn_synonymy".
  - **ROUGE:** Recall-oriented measure (Lin and Och, 2004a):
    - **ROUGE$_n$:** for several n-grams ($n = 1...4$).
    - **ROUGE$_L$:** longest common subsequence.

15. The IQ$_{\text{MT}}$ software is available at `http://www.lsi.upc.edu/~nlp/IQMT`

        - **ROUGE$_{w_1.2}$:** weighted longest common subsequence ($w = 1.2$).
        - **ROUGE$_{S*}$:** skip bigrams with no max-gap-length.
        - **ROUGE$_{SU*}$:** skip bigrams with no max-gap-length, including unigrams.
    - **WER:** Word error rate (Nießen et al., 2000). We use 1-WER.
    - **PER:** Position-independent word error rate (Tillmann et al., 1997b). We use 1-PER.
    - **TER:** Translation edit rate (Snover et al., 2006). We use 1-TER.
- *Shallow syntactic similarity (SP)*
    - **SP-O_p-$\star$** Average lexical overlapping over *p*arts-of-speech.
    - **SP-O_c-$\star$** Average lexical overlapping over base phrase *c*hunk types.

At a more abstract level, we use the NIST metric to compute accumulated/individual scores over sequences of:

- **SP-NIST$_l$** Lemmas.
- **SP-NIST$_p$** Parts of speech.
- **SP-NIST$_c$** Base phrase chunks.
- **SP-NIST$_{iob}$** Chunk IOB labels.

- *Syntactic similarity*
    - *On dependency parsing (DP)*
        - **DP-HWC** These metrics correspond to variants of the headword chain matching (HWCM) metric presented by Liu and Gildea (2005), slightly modified so as to consider different headword chain types:
            - **DP-HWC$_w$** *w*ords.
            - **DP-HWC$_c$** grammatical *c*ategories.
            - **DP-HWC$_r$** grammatical *r*elations.

          In all cases only chains up to length 4 are considered.
        - **DP-$O_l|O_c|O_r$** These metrics correspond exactly to the LEVEL, GRAM, and TREE metrics introduced by Amigó et al. (2006):
            - **DP-O_l-$\star$** Average overlapping among words according to the *l*evel of the dependency tree they hang at.
            - **DP-O_c-$\star$** Average overlapping among words *directly hanging* from terminal nodes (i.e., grammatical *c*ategories) of the same type.
            - **DP-O_r-$\star$** Average overlapping among words ruled by nonterminal nodes (i.e., grammatical *r*elationships) of the same type.
    - *On constituency parsing (CP)*
        - **CP-STM** These metrics correspond to a variant of the syntactic tree matching (STM) metric presented by Liu and Gildea (2005), which considers subtrees up to length 9.

* **CP-O_p-⋆** Similarly to *"SP-O_p-⋆"*, this metric computes average lexical overlapping over *p*arts-of-speech, which are now consistent with the full parsing.

* **CP-O_c-⋆** Analogously, this metric computes average lexical overlapping over base phrase *c*hunk types.

- *Shallow-semantic similarity*
  - *On named entities (NEs)*
    * **NE-O_e-⋆** Average lexical overlapping among NEs of the same type. This metric includes the NE type "O" (i.e., not-an-NE). We introduce another variant, **"NE-O_e-⋆⋆"**, which considers only actual NEs.
    * **NE-M_e-⋆** Average lexical matching among NEs of the same type.
  - *On semantic roles (SR)*
    * **SR-O_r-⋆** Average lexical overlapping among SRs of the same type.
    * **SR-M_r-⋆** Average lexical matching among SRs of the same type.
    * **SR-O_r** This metric reflects " role overlapping ", i.e., overlapping among semantic roles independently of their lexical realization.
- *Semantic similarity*
  - *On discourse representations (DRs)*
    * **DR-STM** This metric is similar to the *CP-STM* variant referred to above, in this case applied to discourse representation structures, i.e., discourse referents and discourse conditions (Kamp, 1981), instead of constituent trees.
    * **DR-O_r-⋆** Average lexical overlapping among discourse *r*epresentation structures of the same type.
    * **DR-O_{rp}-⋆** Average morphosyntactic overlapping (i.e., among grammatical categories –*p*arts-of-speech– associated to lexical items) among discourse *r*epresentation structures of the same type.

A detailed description of these metrics can be found in the IQ_{MT} technical manual (Giménez, 2007). Let us only explicitly note that most of these metrics rely on automatic linguistic processors, which are not equally available for all languages, and which exhibit different levels of performance (i.e., effectiveness and efficiency). This implies several important limitations on their applicability.

MT Evaluation Based on Human Likeness

Heterogeneous MT evaluations might be very informative. However, a new question arises. Since metrics are based on different similarity criteria, and, therefore, biased toward different aspects of quality, scores conferred by different metrics may be controversial. Thus, as system developers we require an additional tool, a metaevaluation criterion, which allows us to select the most appropriate metric

or set of metrics for the task at hand. Most often, metrics are evaluated on the basis of *human acceptability*, i.e., according to their ability to capture the degree of acceptability to humans of automatic translations, usually measured in terms of correlation with human assessments. However, because human assessments are expensive to acquire, a prominent alternative metaevaluation criterion, referred to as *human likeness*, has been recently suggested. Metrics are evaluated according to their ability to capture the features that distinguish human translations from automatic ones. This can be measured in terms of discriminative power (Corston-Oliver et al., 2001; Lin and Och, 2004b; Kulesza and Shieber, 2004; Amigó et al., 2005; Gamon et al., 2005). The underlying assumption is that, given that human translations are the gold standard, a *good* metric should never rank automatic translations higher than human translations. Then, when a system receives a high score according to such a metric, we can ensure that the system is able to emulate the behavior of human translators.

We follow the approach suggested by Amigó et al. (2005) in *QARLA*, a probabilistic framework originally designed for the case of automatic summarization, but adapted by Giménez and Amigó (2006) to the MT scenario. Following their methodology, we use *QARLA* in two complementary steps. First, we determine the set of metrics with highest discriminative power by maximizing over the KING measure. Second, we use QUEEN to measure overall MT quality according to the optimal metric set.[16] Given a set of test cases A, a set of similarity metrics X, and sets of human references R:

- **QUEEN$_{X,R}(A)$** operates under the *unanimity* principle, i.e., the assumption that a "good" translation must be similar to all human references according to all metrics. QUEEN is defined as the probability, over $R \times R \times R$, that, for every metric in X, the automatic translation a is more similar to a human reference r than two other references, r' and r'', to each other. Formally:

$$\text{QUEEN}_{X,R}(a) = Prob(\forall x \in X : x(a,r) \geq x(r',r'')).$$

where $x(a,r)$ stands for the similarity between $a \in A$ and $r \in R$ according to the metric $x \in X$. Thus, QUEEN is able to capture the features that are common to *all* human references, and accordingly reward those automatic translations which share them, and penalize those which do not. Besides, QUEEN exhibits several properties which make it really practical for the purpose of our task. First, since QUEEN focuses on unanimously supported quality distinctions, it is is a measure of high precision. Second, QUEEN provides a robust means of combining several metrics into a single measure of quality; it is robust against metric redundancy, i.e., metrics devoted to very similar quality aspects, and with respect to metric scale properties.

16. The KING and QUEEN measures are available inside IQ$_{\text{MT}}$.

- **KING$_{A,R}(X)$** represents the probability that, for a given set of human references R, and a set of metrics X, the QUEEN quality of a human reference is not lower than the QUEEN quality of *any* automatic translation in A. Formally:

$$\text{KING}_{A,R}(X) = Prob(\forall a \in A : \text{QUEEN}_{X,R-\{r\}}(r) \geq \text{QUEEN}_{X,R-\{r\}}(a)).$$

Thus, KING accounts for the proportion of cases in which a set of metrics has been able to fully distinguish between automatic and manual translations.

MT evaluation based on human likeness has been successfully applied to the optimization of SMT system parameters (Lambert et al., 2006). Besides, it has been shown to be a robust means of integrating different linguistic quality dimensions (Giménez and Màrquez, 2008).

A Measure of Phrase Translation Accuracy

For the purpose of evaluating the changes related only to a specific set of phrases (e.g., "all" vs. "frequent" sets), we introduce a new measure, **A$_{pt}$**, which computes p*hrase* t*ranslation* a*ccuracy* for a given list of source phrases. For every test case, A$_{pt}$ counts the proportion of phrases from the list appearing in the source sentence which have a valid[17] translation both in the target sentence and in at least one reference translation. Cases in which no valid translation is available in any reference translation are not taken into account. Moreover, in order to avoid using the same target phrase more than once for the same translation case, when a phrase translation is used, source and target phrases are discarded. In fact, because in general source-to-target alignments are either unknown or automatically acquired, A$_{pt}$ calculates an approximate solution. Current A$_{pt}$ implementation inspects phrases from left to right in decreasing length order.

11.5.5 Adjustment of Parameters

As we have seen in section 11.4, DPT models provide translation candidates only for specific subsets of phrases. Therefore, in order to translate the whole test set, alternative translation probabilities for all the source phrases in the vocabulary which do not have a DPT prediction must be provided. We have used MLE-based predictions to complete DPT tables. However, interaction between DPT and MLE models is problematic. Problems arise when, for a given source phrase, f_i, DPT predictions must compete with MLE predictions for larger source phrases f_j overlapping with or containing f_i (see section 11.5.6). We have alleviated these problems by splitting DPT tables into three subtables: (1) phrases with DPT

17. Valid translations are provided by the translation table.

prediction, (2) phrases with DPT prediction only for subphrases of it, and (3) phrases with no DPT prediction for any subphrase. Formally:

$$P_{\text{DPT}'}(e|f) = \begin{cases} \lambda'_d P_{\text{DPT}}(e|f) & \text{if } \exists\ P_{\text{DPT}}(e|f) \\ \lambda_o P_{\text{MLE}}(e|f) & \text{if } (\neg\exists\ P_{\text{DPT}}(e|f)) \wedge\ (\exists\ P_{\text{DPT}}(e'|f') \wedge\ (f' \cap\ f \neq \emptyset)) \\ \lambda_\neg P_{\text{MLE}}(e|f) & \textit{otherwise} \end{cases} .$$

In order to perform fair comparisons, all λ parameters governing the search must be adjusted. We have simultaneously adjusted these parameters following a greedy iterative strategy similar to that applied for the optimization of the C and σ^2 parameters of local DPT classifiers (see subsection 11.4.3).[18] The parameter configuration yielding the highest score, according to a given automatic evaluation measure x, over the translation of the development set will be used to translate the test set. Let us remark that, since metrics are based on different similarity assumptions, optimal parameter configurations may vary very significantly depending on the metric used to guide the optimization process. Most commonly, the BLEU metric, widely accepted as a de facto standard, is selected. However, in this work, we additionally study the system behavior when λ parameters are optimized on the basis of human likeness, i.e, by maximizing translation quality according to the QUEEN measure over the metric combination X^+ of highest discriminative power according to KING. This type of tuning has proved to lead to more robust system configurations than tuning processes based on BLEU alone (Lambert et al., 2006).

For the sake of efficiency, we have limited our study to the set of lexical metrics provided by IQ_{MT}. Metrics at deeper linguistic levels have not been used because their computation is currently too slow to allow for massive evaluation processes as it is the case of parameter adjustment. Moreover, due to the fact that exploring all possible metric combinations was not viable, for the KING optimization we have followed a simple algorithm which performs an *approximate suboptimal* search. The algorithm proceeds as follows. First, individual metrics are ranked according to their KING quality.[19] Then, following that order, metrics are individually added to the set of optimal metrics only if the global KING increases.

The resulting optimal set is: $X^+ = \{\ \text{METEOR}_{wnsyn},\ \text{ROUGE}_{w\_1.2}\ \}$, which includes variants of METEOR and ROUGE, metrics which, interestingly, share a common ability to capture lexical and morphological variations (use of stemming, and dictionary lookup).

18. In order to keep the optimization process feasible in terms of time, the search space is pruned as described in section 11.5.1.
19. KING has been computed over a representative set of baseline systems based on different nonoptimized parameter configurations.

11.5.6 Results

We compare the performance of DPT- and MLE-based models in the full translation task. Since the adjustment of internal parameters (C and σ^2) is impractical, based on the results from section 11.4, we have limited our study to test the behavior of binary SVMs. Also, for the sake of efficiency, we have limited our study to linear kernels.

We have used a system which relies on MLE for the estimation of translation models *(MLE)* as a baseline. We separately study the case of (i) using DPT for the set of *all* phrases and that of (ii) using DPT predictions for the reduced set of *frequent* phrases. This latter set exhibits a higher local accuracy. However, most phrases in this set are single words.[20] Thus, it constitutes an excellent material to analyze the interaction between DPT- and MLE-based probabilities in the context of the global task. Besides, this set covers 67% of the words in the test, whereas the "all" set covers up to 95% of the words. In both cases, DPT predictions for uncovered words are provided by the MLE model.

Table 11.8 shows automatic evaluation results according to different metrics, including BLEU and QUEEN. For the sake of informativeness, METEOR_{wnsyn} and $\text{ROUGE}_{w\_1.2}$ scores used in QUEEN computations are provided as well. Phrase translation accuracy is evaluated by means of the A_{pt} measure, both over the set of "all" and "frequent" phrases. We have separately studied the cases of parameter optimizations based on BLEU (rows 1 to 3) and QUEEN (rows 4 to 6). The first observation is that in the two cases DPT models yield an improved lexical choice according to the respective evaluation metric guiding the adjustment of parameters. However, for the rest of metrics there is not necessarily improvement. Interestingly, in the case of BLEU-based optimizations, DPT predictions as an additional feature report a significant BLEU improvement over the MLE baseline only when all phrases are used (see rows 2 and 3). In contrast, in the case of QUEEN-based optimizations, improvements take place in both cases, although with less significance. It is also interesting to note that the significant increase in phrase translation accuracy (A_{pt}) only reports a very modest improvement in the rest of the metrics (see rows 5 and 6). This could be actually revealing a problem of interaction between DPT predictions and other models.

BLEU vs QUEEN

Table 11.8 illustrates the enormous influence of the metric selected to guide the optimization process. A system adjusted so as to maximize the score of a specific metric does not necessarily maximize the scores conferred by other metrics. In that respect, BLEU and QUEEN exhibit completely opposite behaviors. Improvements

20. The "frequent" set consists of 240 length-1 phrases, 64 length-2 phrases, 12 length-3 phrases, and 1 length-4 phrase.

Table 11.8 Automatic evaluation of MT results (DPT predictions as an additional feature)

System Config.	QUEEN (lexical)	METEOR (wnsyn)	ROUGE (w_1.2)	$\mathbf{A}_{pt}$ (all)	$\mathbf{A}_{pt}$ (frq)	BLEU
BLEU-based optimization						
MLE	0.4826	0.7894	0.4385	0.7099	0.7915	0.6331
$\mathbf{DPT}_{all}$	0.4717	0.7841	0.4383	0.7055	0.7823	**0.6429**
$\mathbf{DPT}_{frq}$	0.4809	0.7863	0.4386	0.7102	0.7941	0.6338
QUEEN-based optimization						
MLE	0.4872	0.7924	0.4384	0.7158	0.8097	0.6149
$\mathbf{DPT}_{all}$	0.4907	**0.7949**	0.4391	0.7229	0.8115	0.6048
$\mathbf{DPT}_{frq}$	**0.4913**	0.7934	**0.4404**	**0.7245**	**0.8251**	0.6038

in BLEU do not necessarily imply improvements in QUEEN, and vice versa. We have further analyzed this controversial relationship by comparing optimal parameter configurations, and observed that λs are in a very similar range, except for the weight of the word penalty model (λ_w), close to 0 in the case of BLEU, whereas in the case of QUEEN, it takes negative values around -1, thus favoring longer translations. This seems to indicate that the heuristically motivated brevity penalty factor of BLEU could be responsible for the BLEU vs. QUEEN puzzle observed. We have verified this hypothesis by inspecting BLEU values before applying the penalty factor. These are on average 0.02 BLEU points higher (0.605 $\rightarrow$ 0.625), which explains part of the puzzle. The other part must be found in the fact that, while BLEU is based on n-gram precision, QUEEN is a metametric which combines different quality aspects, in this case borrowed from ROUGE and METEOR.

Beyond Lexical Similarity

In order to analyze other quality aspects beyond the lexical dimension, in table 11.9 we provide automatic evaluation results according to several metric representatives from different linguistic levels. Metrics are grouped according to the level at which they operate (i.e., lexical, shallow-syntactic, syntactic, shallow-semantic, and semantic). We have also computed two different QUEEN values, namely QUEEN(X^+) and QUEEN(X^+_{LF}). The first value corresponds to the application of QUEEN to the optimal metric combination based on lexical features only, whereas the second value corresponds to QUEEN applied to the optimal metric combination considering linguistic features at different levels. In this latter case, the optimal metric combination, obtained following the procedure described in subsection 11.5.5, is X^+_{LF} = { METEOR$_{wnsyn}$, SP-NIST$_p$, SR-M_r-$\star$ }, which includes metrics at the lexical, morphosyntactic, and shallow-semantic levels, respectively, based on

Table 11.9 Automatic evaluation of MT results. Linguistic features and metaevaluation

		BLEU-based optim.			QUEEN-based optim.		
Metric	**KING**	**MLE**	**DPT$_{all}$**	**DPT$_{frq}$**	**MLE**	**DPT$_{all}$**	**DPT$_{frq}$**
1-WER	0.1521	0.6798	**0.6908**	0.6842	0.6652	0.6504	0.6542
1-PER	0.1422	0.7679	**0.7764**	0.7701	0.7571	0.7397	0.7481
1-TER	0.1508	0.7031	**0.7135**	0.7062	0.6882	0.6737	0.6777
BLEU	0.1164	0.6331	**0.6429**	0.6338	0.6149	0.6048	0.6038
NIST	0.1488	11.2205	**11.3403**	11.2398	10.9525	10.7508	10.8012
GTM.e1	0.1151	0.8971	0.8954	0.8977	0.8988	0.8948	**0.9013**
GTM.e2	0.1257	0.4364	**0.4391**	0.4348	0.4321	0.4296	0.4259
ROUGE$_L$	0.1270	0.6958	**0.6984**	0.6962	0.6914	0.6887	0.6907
ROUGE$_W$	**0.1594**	0.4385	0.4383	0.4386	0.4384	**0.4391**	**0.4404**
MTR$_{exact}$	0.1601	0.7324	0.7278	0.7306	0.7332	**0.7376**	**0.7381**
MTR$_{wnsyn}$	**0.1786**	0.7894	0.7841	0.7863	0.7924	**0.7949**	**0.7934**
QUEEN(X^+)	0.1806	0.4826	0.4717	0.4809	0.4872	**0.4907**	**0.4913**
SP-O_p-$\star$	0.1217	**0.6915**	0.6904	**0.6914**	0.6901	0.6834	0.6868
SP-O_c-$\star$	0.1263	0.6878	**0.6895**	0.6883	0.6884	0.6855	0.6857
SP-NIST$_l$	0.1455	11.3156	**11.4405**	11.3335	11.0455	10.8469	10.9004
SP-NIST$_p$	**0.1865**	9.9739	**10.0359**	9.9399	9.7361	9.5766	9.5174
SP-NIST$_{iob}$	0.1772	7.6315	**7.6583**	7.6174	7.4876	7.3850	7.3525
SP-NIST$_c$	0.1680	6.9357	**7.0100**	6.9507	6.8205	6.6931	6.7000
DP-HWC$_w$	0.1071	0.2755	**0.2823**	0.2712	0.2778	0.2742	0.2743
DP-HWC$_c$	0.1475	**0.5048**	0.4980	0.4994	**0.5051**	0.4848	0.5014
DP-HWC$_r$	0.1481	**0.4491**	0.4443	0.4433	**0.4492**	0.4292	0.4435
DP-O_l-$\star$	0.1382	0.5100	0.5089	0.5073	**0.5149**	0.5034	0.5063
DP-O_c-$\star$	0.1561	**0.6078**	0.6032	0.6034	0.6043	0.6053	0.6039
DP-O_r-$\star$	0.1693	**0.4672**	0.4642	0.4627	0.4653	0.4597	0.4610
CP-O_p-$\star$	0.1217	**0.6883**	**0.6893**	**0.6886**	0.6872	0.6807	0.6839
CP-O_c-$\star$	0.1349	0.6530	**0.6559**	0.6541	0.6520	0.6444	0.6485
CP-STM	0.1224	**0.4723**	0.4686	0.4674	**0.4712**	0.4606	0.4573
NE-O_e-$\star\star$	0.1131	0.7067	**0.7100**	0.7073	0.7046	0.6954	0.7020
NE-O_e-$\star$	0.0079	0.2049	0.2047	**0.2069**	0.2017	0.2004	0.2042
NE-M_e-$\star$	0.0079	0.2001	0.1991	**0.2021**	0.1964	0.1984	0.1994
SR-O_r-$\star$	0.1138	0.4347	0.4361	**0.4400**	0.4324	0.4158	0.4234
SR-M_r-$\star$	**0.1323**	**0.2892**	0.2861	**0.2892**	**0.2901**	**0.2895**	0.2846
SR-O_r	0.1230	0.6359	0.6416	**0.6482**	0.6378	0.6219	0.6329
DR-O_r-$\star$	0.1369	0.4766	0.4788	0.4758	**0.4813**	**0.4819**	0.4781
DR-O_{rp}-$\star$	0.1501	0.5840	0.5847	0.5814	0.5877	**0.5923**	0.5825
DR-STM	0.0688	0.3526	0.3510	0.3539	**0.3633**	0.3570	0.3543
QUEEN(X^+_{LF})	0.2011	0.3018	0.2989	**0.3058**	0.3008	0.2937	0.3005

unigram alignment precision, part-of-speech n-gram matching, and average lexical matching over semantic roles.

First of all, metrics are evaluated according to their ability to distinguish between manual and automatic translations, as computed by KING. Broadly speaking, KING is a measure of discriminative power. For instance, if a metric obtains a KING of 0.5, it means that in 50% of the test cases, it is able to explain by itself the differences in quality between manual and automatic translations. Thus, KING serves as an estimate of the impact of specific quality aspects on the overall system performance. In that respect, it can be observed that highest KING values are obtained by metrics based on lexical, shallow-syntactic, and syntactic similarities.

As to system evaluation, quality aspects are diverse, and thus, it is not always the case that all aspects improve together. For instance, at the lexical and shallow-syntactic levels, most metrics prefer the DPT$_{all}$ system optimized over BLEU. Only some ROUGE and METEOR variants prefer the DPT systems optimized over QUEEN. After all, the X^+ set, used in the QUEEN computation, consists of these metrics, so this result was expected. In any case, the fact that all metrics based on lexical similarities consistently prefer DPT over MLE confirms that DPT predictions yield an improved lexical choice.

At the syntactic level, however, most metrics prefer the MLE systems. Only the shallowest metrics, e.g., DP-HWC$_w$ (i.e., lexical headword matching over dependency trees), CP-O_p-$\star$ and CP-O_c-$\star$ (i.e., lexical overlapping over parts of speech and phrase constituents) seem to prefer DPT systems, always optimized over BLEU. This is a very interesting result since it reveals that an improved lexical similarity does not necessarily lead to an improved syntactic structure.

At the shallow-semantic level, while NE metrics are not very informative,[21] SR metrics seem to prefer the DPT$_{frq}$ system optimized over BLEU, whereas at the properly semantic level, metrics based on discourse representations prefer the DPT$_{all}$ and MLE systems optimized over QUEEN. Therefore, no clear conclusions can be made as to which model or optimization strategy leads to a better semantic structure.

Finally, combining metrics from all linguistic levels on the basis of human likeness, i.e., QUEEN(X^+_{LF}), the best system is DPT$_{frq}$ optimized over BLEU. This would indicate that focusing on a set of frequent phrases is more productive in terms of overall quality.

Several conclusions must be drawn from these results. First, the fact that an improved lexical and semantic similarity does not necessarily lead to an improved sentence grammaticality might be revealing problems of interaction between DPT predictions and the other models in the SMT system. We have verified this hypothesis through a number of manual evaluations. These have revealed that gains are mainly related to the adequacy dimension, whereas for fluency there is no sig-

21. Observe the low KING values attained, close to zero, except for the case of the 'NE-O_e-$\star\star$' metric, which also considers overlapping among tokens which are not named entities.

Table 11.10 Case of analysis #1. DPT models help

Source	yo quisiera que el incumplimiento institucional del consejo **fuera sancionado**
Ref 1	i would like the council 's institutional infringement **to be penalised**
Ref 2	i would like the council 's institutional non-fulfilment of its obligations **to be sanctioned**
Ref 3	i would like to see the institutional non-compliance of the council **punished**
	BLEU-based optimizations
MLE	i would like to see the failure to comply with institutional **outside** of the council **sanctioned**
DPT$_{all}$	i would like to see the institutional breach of the council **was sanctioned**
DPT$_{frq}$	i would like to see the institutional breach of the council **outside sanctioned**
	QUEEN-based optimizations
MLE	i would like to see the failure to comply with the institutional councils **outside sanctioned**
DPT$_{all}$	i would like to see the failure to comply with the institutions of the council **were to be sanctioned**
DPT$_{frq}$	i would like to see the failure to comply with the institutional councils **outside sanctioned**

nificant improvement. See, for instance, manual evaluations reported in Giménez and Màrquez (2007a). These correspond to the case of pairwise system comparisons. Two different judges evaluated a subset of translation test cases in terms of adequacy and fluency. Results reported an adequacy improvement in 39% of the cases, while in 17% there was a decrement. In the case of fluency, results reported improvement in 30% of the cases, while in 37% there was a decrement.

Second, the lack of consensus between metrics based on different similarity criteria reinforces the need for evaluation methodologies which allow system developers to take into account a heterogeneous set of quality aspects.

Error Analysis

Tables 11.10, 11.11, and 11.12 show three sentence fragments illustrating the different behavior of the system configurations evaluated. We start, in table 11.10, by showing a positive case in which the DPT predictions help the system to find a better translation for *fuera sancionado*. Observe how baseline SMT systems, whose translation models are based on MLE, all wrongfully translate *fuera* as *"outside"* instead of as an auxiliary verb form (e.g., *"was"* or *"were"*) or past form of the accompanying verb *sancionado* (e.g., *"sanctioned"* or *"penalised"*) . In contrast, DPT$_{all}$ systems are able to provide more appropriate translations for this phrase, regardless of the metric guiding the parameter optimization process. Observe also, how DPT$_{frq}$ systems, which, unfortunately, do not count on DPT predictions for this not-frequent-enough phrase, commit all the same mistakes as MLE-based systems.

Tables 11.11 and 11.12 present two cases in which the metric guiding the optimizations has a stronger influence. In table 11.11, all MLE baseline systems

Table 11.11 Case of analysis #2. DPT models may help

Source	aquel diputado **cuyo nombre** no conozco
Ref 1	the member **whose name** i do not know
Ref 2	the honourable member , **whose name** i can not recall
Ref 3	that member **whose name** i ignore
	BLEU-based optimizations
MLE	that member **whose behalf** i do not know
DPT$_{all}$	that member **whose name** i do not know
DPT$_{frq}$	that member **whose behalf** i do not know
	QUEEN-based optimizations
MLE	that member **on whose behalf** i am not familiar with
DPT$_{all}$	that member **on whose behalf** i am not familiar with
DPT$_{frq}$	that mep **whose behalf** i am not familiar with

Table 11.12 Case of analysis #3. DPT models may not help

Source	poco más del 40 % de los fondos **van a parar a** esos países .
Ref 1	only slightly more than 40 % of the money **ends up in** those countries .
Ref 2	little more than 40 % of these funds **end up in** these countries .
Ref 3	little more than 40 % of the funds **are going to** those countries .
	BLEU-based optimizations
MLE	little more than 40 % of the funds **go to them** .
DPT$_{all}$	little more than 40 % of the funds **will stop to these countries** .
DPT$_{frq}$	little more than 40 % of the funds **go to these countries** .
	QUEEN-based optimizations
MLE	just a little more than 40 % of the money **goes to those countries** .
DPT$_{all}$	little more than 40 % of the funds **are going to stop to these countries** .
DPT$_{frq}$	little more than 40 % of the funds **are going to stop to these countries** .

wrongfully translate *cuyo nombre* into *"whose behalf"*. Only the 'DPT$_{all}$' system optimized over BLEU is able to find a correct translation, *"whose name"*. In table 11.12, while MLE-based systems provide all fairly correct translations of *van a parar a* into *"go to"*, DPT predictions may cause the system to wrongfully translate *van a parar a* into *"are going to stop to"*. Only the DPT$_{frq}$ system optimized over BLEU is able to find a correct translation. The underlying cause behind these two cases is that there is no DPT prediction for *cuyo nombre* and *van a parar a*, two phrases of very high cohesion, but only for subphrases of it (e.g., *cuyo*, *nombre*, *van*, *a*, *parar*, *van a* , *a parar*). DPT predictions for these subphrases must compete with MLE-based predictions for larger phrases, which may cause problems of interaction.

11.6 Conclusions

In this chapter, we have shown that discriminative phrase translation may be successfully applied to SMT. Despite the fact that measuring improvements in word selection is a very delicate issue, experimental results, according to several well-

known metrics based on lexical similarity, show that dedicated DPT models yield a significantly improved lexical choice over traditional MLE-based ones. However, by evaluating linguistic aspects of quality beyond the lexical level (e.g., syntactic, and semantic), we have found that an improved lexical choice and semantic structure does not necessarily lead to improved grammaticality. This result has been verified through a number of manual evaluations, which have revealed that gains are mainly related to the adequacy dimension, whereas for fluency there is no significant improvement.

As we have seen in section 11.2, other authors have recently conducted similar experiments. Although closely related, there exist several important differences between the work of Carpuat and Wu (2007b), Bangalore et al. (2007), Stroppa et al. (2007), and ours. Some are related to the context of the translation task, i.e., language-pair and task domain. For instance, while we work in the Spanish-to-English translation of European Parliament proceedings, Carpuat and Wu (2007b) and Bangalore et al. (2007) work on the Chinese-to-English translation of basic travel expressions and newswire articles, and Stroppa et al. (2007) work on the Chinese-to-English and Italian-to-English translation of basic travel expressions. Additionally, Bangalore et al. (2007) present results on Arabic-to-English translation of proceedings of the United Nations and on French-to-English translation of proceedings of the Canadian Parliament.

Other differences are related to the disambiguation system itself. While we rely on SVM predictions, Carpuat and Wu (2007b) use an ensemble of four combined models (naïve Bayes, maximum entropy, boosting, and kernel PCA-based models), Stroppa et al. (2007) rely on memory-based learning, and Bangalore et al. (2007) use maximum entropy. Besides, Bangalore et al. (2007) employ a slightly different SMT architecture based on stochastic finite-state transducers which addresses the translation task as two separate processes: (i) global lexical selection, i.e., dedicated word selection; and (ii) sentence reconstruction. Moreover, their translation models are indeed bilingual language models. They also deal with reordering in a different manner. Prior to translation, the source sentence is reordered so as to approximate the right order of the target language. This allows them to perform a monotonic decoding.

There are also significant differences in the evaluation process. Bangalore et al. (2007) rely on BLEU as the only measure of evaluation, Stroppa et al. (2007) additionally rely on NIST, and Carpuat and Wu (2007b) show results according to eight different standard evaluation metrics based on lexical similarity, including BLEU and NIST. In contrast, in this study, we have used a set of evaluation metrics operating at deeper linguistic levels. We have also relied on the QUEEN measure, which allows for nonparametric combinations of different metrics into a single measure of quality.

Besides, this study has also served to start a discussion on the role of automatic metrics in the development cycle of MT systems and the importance of metaevaluation. We have shown that basing evaluations and parameter optimizations on different metrics may lead to very different system behaviors. For system comparison,

this may be solved by conducting manual evaluations. However, this is impractical for the adjustment of parameters, where hundreds of different configurations are tried. Thus, we argue that more attention should be paid to the metaevaluation process. In our case, metrics have been evaluated on the basis of human likeness. Other solutions exist. The main point, in our opinion, is that system development is *metricwise*. In other words, for the sake of robustness, it is crucial that the metric (or set of metrics) guiding the development process be able to capture the possible quality variations induced by system modifications. This is especially important given the fact that, most often, system improvements focus on partial aspects of quality, such as word selection or word ordering, which can not always be expected to improve together.

Finally, the fact that improvements in adequacy do not lead to an improved fluency evinces that the integration of local DPT probabilities into the statistical framework requires further study. We believe that if DPT models considered features from the target side under generation and from the correspondence between source and target, phrase translation accuracy would improve and cooperation with the decoder would be even softer. Nevertheless, predictions based on local training may not always be well suited for being integrated in the target translation. Thus, we also argue that if phrase translation classifiers were trained in the context of the global task, their integration would be more robust and translation quality could further improve. The possibility of moving toward a new global DPT architecture in the fashion, for instance, of those suggested by Tillmann and Zhang (2006) or Liang et al. (2006) should be considered.

Acknowledgments

This research has been funded by the Spanish Ministry of Education and Science, project OpenMT (TIN2006-15307-C03-02). We are recognized as a Quality Research Group (2005 SGR-00130) by DURSI, the Research Department of the Catalan Government. We are thankful to the TC-STAR Consortium for providing such very valuable data sets. We are also grateful to David Farwell and Sandra Fontanals who helped us as judges in the numerous processes of manual evaluation. We also thank the anonymous reviewers for their valuable comments and suggestions.

12 Semisupervised Learning for Machine Translation

Nicola Ueffing
Gholamreza Haffari
Anoop Sarkar

Statistical machine translation systems are usually trained on large amounts of bilingual text, used to learn a translation model, and also on large amounts of monolingual text in the target language, used to train a language model. In this chapter we explore the use of semisupervised methods for the effective use of monolingual data from the source language in order to improve translation quality. In particular, we use monolingual source language data from the same domain as the test set (without directly using the test set itself) and use semisupervised methods for model adaptation to the test set domain. We propose several algorithms with this aim, and present the strengths and weaknesses of each one. We present detailed experimental evaluations using French–English and Chinese–English data and show that under some settings translation quality can be improved.

12.1 Introduction

In statistical machine translation (SMT), translation is modeled as a decision process. The goal is to find the translation $\mathbf{t}$ of source sentence $\mathbf{s}$ which maximizes the posterior probability:

$$\arg\max_{\mathbf{t}} p(\mathbf{t} \mid \mathbf{s}) = \arg\max_{\mathbf{t}} p(\mathbf{s} \mid \mathbf{t}) \cdot p(\mathbf{t}). \tag{12.1}$$

This decomposition of the probability yields two different statistical models which can be trained independently of each other: the translation model $p(\mathbf{s} \mid \mathbf{t})$ and the target language model $p(\mathbf{t})$.

State-of-the-art SMT systems are trained on large collections of text which consist of bilingual corpora (to learn the parameters of the translation model), and of monolingual target language corpora (for the target language model). It

has been shown, e.g., in Brants et al. (2007), that adding large amounts of target language text improves translation quality considerably, as improved language model estimates about potential output translations can be used by the decoder in order to improve translation quality. However, the availability of monolingual corpora in the source language has not been shown to help improve the system's performance. We will show how such corpora can be used to achieve higher translation quality.

Even if large amounts of bilingual text are given, the training of the statistical models usually suffers from sparse data. The number of possible events, i.e., phrase pairs in the two languages, is too big to reliably estimate a probability distribution over such pairs. Another problem is that for many language pairs the amount of available bilingual text is very limited. In this chapter, we address this problem and propose a general framework to solve it. Our hypothesis is that adding information from source language text can also provide improvements. Unlike adding target language text, this hypothesis is a natural semisupervised learning problem.

To tackle this problem, we propose algorithms for semisupervised learning. We translate sentences from the source language and use them to retrain the SMT system with the hope of getting a better translation system. The evaluation is done just once at the end of the learning process. Note the difference between this approach and the transductive approach in Ueffing et al. (2007a), where the latter treats the test set as the additional monolingual source data. In the work presented here, the additional monolingual source data is drawn from the same domain as the test set. In particular, we *filter* the monolingual source language sentences based on their similarity to the development set, as explained in section 12.3.3. Semisupervised learning can be seen as a means to adapt the SMT system to a new domain or style that is different from the bilingual training data. For instance, a system trained on newswire could be used to translate weblog texts. The method proposed here adapts the trained models to the style and domain of the new domain without requiring bilingual data from this domain.

We present detailed experimental evaluations using French–English and Chinese–English data. In the French–English translation task we use bilingual data from the Europarl corpus, and use monolingual data from the same domain as our test set which is drawn from the Canadian Hansard corpus. In the Chinese–English task we use bilingual data from the NIST large-data track and use monolingual data from the Chinese Gigaword corpus.

12.2 Baseline MT System

The SMT system we applied in our experiments is PORTAGE. This is a state-of-the-art phrase-based translation system developed at the National Research Council Canada which has been made available to Canadian universities for research and education purposes. We provide a basic description here; for a detailed description, see Ueffing et al. (2007b).

The PORTAGE system determines the translation of a given source sentence $\mathbf{s}$ by maximizing the posterior probability over all possible translations $\mathbf{t}$ as shown in Eq. (12.1). This posterior probability is approximated by the log-linear combination of models $g_i(\mathbf{s}, \mathbf{t}), i = 1, \ldots, I$, taking both languages into account (such as the translation model in Eq. (12.1)), and target-language-based models $h_j(\mathbf{t}), j = 1, \ldots, J$ (such as the language model in Eq. (12.1)). The decoder solves the following equation:

$$\arg\max_{\mathbf{t}} p(\mathbf{t}, \mathbf{s}) = \arg\max_{\mathbf{t}} \prod_{i=1}^{I} g_i(\mathbf{s}, \mathbf{t})^{\alpha_i} \cdot \prod_{j=1}^{J} h_j(\mathbf{t})^{\beta_j}. \tag{12.2}$$

The models (or features) which are employed by the decoder are

- one or several phrase table(s), which model the translation direction $p(\mathbf{s} \mid \mathbf{t})$; they are smoothed using the methods described in Foster et al. (2006);
- one or several n-gram language model(s) trained with the SRILM toolkit described in Stolcke (2002); in the experiments reported here, we used several 4-gram models on the Chinese–English data, and a trigram model on French-English;
- a distortion model which assigns a penalty based on the number of source words which are skipped when generating a new target phrase;
- a word penalty assigning constant cost to each generated target word. This constitutes a way to control the length of the generated translation.

These different models are combined log-linearly as shown in Eq. (12.2). Their weights $\alpha_i, i = 1, \ldots, I, \beta_j, j = 1, \ldots, J$ are optimized with respect to BLEU score (Papineni et al. (2002)) using the algorithm described in Och (2003). This optimization is done on a development corpus.

The search algorithm implemented in the decoder is a dynamic-programming beam-search algorithm. After the main decoding step, rescoring with additional models is performed. The system generates a 5000-best list of alternative translations for each source sentence. These lists are rescored with the following models:

- The different models used in the decoder which are described above.
- Two different features based on IBM Model 1 from Brown et al. (1993): a Model 1 probability calculated over the whole sentence, and a feature estimating the number of source words which have a reliable translation. Both features are determined for both translation directions.
- Several higher-order n-gram language models.
- Posterior probabilities for words, phrases, n-grams, and sentence length (Zens and Ney, 2006; Ueffing and Ney, 2007). All of these posterior probabilities are calculated over the n-best list and using the sentence probabilities which the baseline system assigns to the translation hypotheses. More details on the calculation of posterior probabilities will be given in section 12.3.5.

Algorithm 12.1 Bootstrapping algorithm: classifier version

Input: each example x is either labeled $L(x)$ in some annotated data, or unlabeled as $U^0(x) := \perp$.
$t := 0$
repeat
  for each example x **do**
    Training step: Estimate θ for $\Pr(j \mid x, \theta)$ using L and $U^t(x)$
    Labeling step: $U^{t+1}(x) = \begin{cases} \arg\max_{j \in \mathcal{L}} \Pr(j \mid x, \theta) & \text{if } \Pr(j \mid x, \theta) > \text{ threshold } \zeta \\ \perp & \text{otherwise} \end{cases}$
  $t := t + 1$
until for all x: $U^{t+1}(x) = U^t(x)$

The weights of these additional models and of the decoder models are again optimized to maximize BLEU score. This is performed on a second development corpus.

12.3 The Framework

12.3.1 The Yarowsky Algorithm

The original Yarowsky algorithm (Yarowsky (1995)) was proposed in the context of a word-sense disambiguation task. The model was a simple decision list classifier $\Pr(j \mid x, \theta)$ where the class label $j \in \mathcal{L}$ is predicted based on the single most likely feature extracted from the input x. The pseudocode for this algorithm is shown in algorithm 12.1. The key idea is to use an initial classifier which was built from the seed data in such a way so as to have high precision but low recall on the unlabeled set since it could not predict any label for most examples. This is because if a feature extracted from input x has not been observed with any class label, this event is not assigned a smoothed probability estimate (unlike the common strategy in supervised learning). Hence, for examples where all the features are events of this type, the classifier labels it as $\perp$. However, even if a single feature extracted from x is observed with a class label in the labeling step, this information can be used to make a prediction for future examples by the decision list classifier (which only uses the single most likely feature to predict the class label). This classifier was then used to label the unlabeled data and those examples labeled with high confidence (above a threshold) were used along with the labeled data to train a new classifier. This process was repeated iteratively until no new labels could be found for the unlabeled data set. Each iteration could be seen as improving the recall of the classifier at the expense of its precision. In each iteration the classifier is able to provide labels for larger and larger subsets of the unlabeled data.

There are several different types of iterative self-training semisupervised algorithms that have been proposed in the literature. Haffari and Sarkar (2007) and

Algorithm 12.2 Semisupervised learning algorithm for statistical machine translation

Input: training set L of parallel sentence pairs. ▷ Bilingual training data.
Input: unlabeled set U of source text. ▷ Monolingual source language data.
Input: dev corpus C.
Input: number of iterations R, and size of n-best list N.
$T_{-1} := \{\}$. ▷ Additional bilingual training data.
$i := 0$. ▷ Iteration counter.
repeat
  Training step: $\pi^{(i)} := \textbf{Estimate}(L, T_{i-1})$.
  $X_i := \{\}$. ▷ The set of generated translations for this iteration.
  $U_i := \textbf{Filter}(U, C, i)$ ▷ The ith chunk of unlabeled sentences.
  for sentence $\mathbf{s} \in U_i$ **do**
    Labeling step: Decode $\mathbf{s}$ using $\pi^{(i)}$ to obtain N best sentence pairs with their scores
    $X_i := X_i \cup \{(\mathbf{t}_n, \mathbf{s}, \pi^{(i)}(\mathbf{t}_n \mid \mathbf{s}))_{n=1}^{N}\}$
  Scoring step: $S_i := \textbf{Score}(X_i)$ ▷ Assign a score to sentence pairs $(\mathbf{t}, \mathbf{s})$ from X_i.
  Selection step: $T_i := T_i \cup \textbf{Select}(X_i, S_i)$ ▷ Choose a subset of *good* sentence pairs $(\mathbf{t}, \mathbf{s})$ from X_i.
  $i := i + 1$.
until $i > R$

Abney (2004) provide a more detailed discussion on the relationship between these algorithms and the Yarowsky algorithm.

12.3.2 Semisupervised Learning Algorithm for SMT

Our semisupervised learning algorithm, algorithm 12.2, is inspired by the Yarowsky algorithm described in section 12.3.1. We will describe it here for (re-)training of the translation model. However, the same algorithm can be used to (re-)train other SMT models, such as the language model, as investigated in Ueffing et al. (2008).

The algorithm works as follows: First, the translation model is estimated based on the sentence pairs in the bilingual training data L. The set of source language sentences, U, is sorted according to the relevance with respect to the development corpus C, and a chunk of sentences, U_i, is filtered. These sentences are then translated based on the current model. The SMT system generates an n-best list of translation alternatives for each source sentence. This set X_i of translations is then scored, and a subset of good translations and their sources, T_i, is selected from X_i in each iteration and added to the training data. The process of generating sentence pairs, scoring them, selecting a subset of good sentence pairs, and updating the model is continued until a stopping condition is met. In the experiments presented here, this stopping criterion is either a fixed number of iterations, or we run for a fixed number of iterations on a development set, and pick the translation model based on the iteration that provides the highest improvement on the development set. Note that here we use the first for the French–English experiments and both for the Chinese–English experiments.

Note that algorithm 12.2 is a variation of the original Yarowsky algorithm in which the same unlabeled data is used in each iteration.

It has been shown by Abney (2004) that algorithm 12.2 minimizes the entropy of the probability distribution $p(\mathbf{t} \mid \mathbf{s})$ over translations of the unlabeled data set U. However, this is true only when the functions Estimate, Score, and Select have very prescribed definitions. Rather than analyzing the convergence of algorithm 12.2, we will use definitions for Estimate, Score, and Select that have been experimentally shown to improve MT performance. Following Ueffing et al. (2007a), these are different versions of the algorithm for the two different translation tasks we work on. We will present the different variants of the functions Filter, Estimate, Score, and Select in the following subsections. The exact experimental settings for the two translation tasks will be described in section 12.4.1.

In Ueffing et al. (2007a), a transductive variant of algorithm 12.2 was introduced which uses the development or test corpus as unlabeled data U. That is, this corpus is translated and reliable translations are selected and used in (re-)training to improve the performance of the SMT system. This approach generates a very small amount of new bilingual data of high relevance. Unlike the approach described in Ueffing et al. (2007a), we explore much larger amounts of monolingual source-language data. In order to identify the relevant parts of the data, we filter as explained in the following.

12.3.3 The Filter Function

In general, having more training data improves the quality of the trained models. However, when it comes to the translation of a particular test set, the question is whether *all* of the available training data is relevant to the translation task or not. Moreover, working with large amounts of training data requires more computational power. So if we can identify a subset of training data which is relevant to the current task and use only this to (re-)train the models, we can reduce the computational complexity significantly.

We propose to filter the additional monolingual source data to identify the parts which are relevant with respect to the development set. This filtering is based on n-gram coverage. For a source sentence $\mathbf{s}$ in the monolingual data, its n-gram coverage over the sentences in the development set is computed. The average over several n-gram lengths is used as a measure of relevance of this training sentence with respect to the development corpus. In the experiments presented here, this is the average over 1- to 6-gram coverage. We sort the source sentences by their coverage and successively add them as unlabeled data U_i in algorithm 12.2.

12.3.4 The Estimate Function

The Estimate function estimates a phrase translation model from the sets of bilingual data, L and T_{i-1}. Out of the three different versions of this function presented in Ueffing et al. (2007a), we use the one which performed best for

our experiments here: training an additional phrase translation model on the new bilingual data T_{i-1}. That is, the phrase translation model learned on the original bilingual data L is kept fixed, and a new model is learned on T_{i-1} only and then added as a new component in the log-linear SMT model presented in Eq. (12.2). This additional model is relatively small and specific to the test corpus C. We have to learn a weight for this new phrase translation model. To this end, the weight optimization is carried out again on the development set. After the first iteration, we reoptimize the decoder and rescoring weights for all original models and this new phrase translation model. These weights are then kept fixed throughout the following iterations. This reoptimization process is computationally expensive, so we carry it out only once.

12.3.5 The Scoring Function

In algorithm 12.2, the Score function assigns a score to each translation hypothesis $\mathbf{t}$. We used the following scoring functions in our experiments:
Length-normalized score: Each translated sentence pair $(\mathbf{t}, \mathbf{s})$ is scored according to the model probability $p(\mathbf{t} \mid \mathbf{s})$ (assigned by the SMT system) normalized by the length $|\mathbf{t}|$ of the target sentence:

$$\mathbf{Score}(\mathbf{t}, \mathbf{s}) \quad = \quad p(\mathbf{t} \mid \mathbf{s})^{\frac{1}{|\mathbf{t}|}}. \tag{12.3}$$

Confidence estimation: The goal of confidence estimation is to estimate how reliable a translation $\mathbf{t}$ is, given the corresponding source sentence $\mathbf{s}$. The confidence estimation which we implemented follows the approaches suggested in Blatz et al. (2003) and Ueffing and Ney (2007): The confidence score of a target sentence $\mathbf{t}$ is calculated as a log-linear combination of several different sentence scores. These scores are Levenshtein-based word posterior probabilities, phrase posterior probabilities, and a target language model score. The posterior probabilities are determined over the n-best list generated by the SMT system.

The word posterior probabilities are calculated on basis of the *Levenshtein alignment* between the hypothesis under consideration and all other translations contained in the n-best list. The Levenshtein alignment is performed between a given hypothesis $\mathbf{t}$ and every sentence $\mathbf{t}_n$ contained in the n-best list individually. To calculate the posterior probability of target word t occurring in position i of the translation, the probabilities of all sentences containing t in position i or in a position Levenshtein-aligned to i is summed up. Let $\mathcal{L}(\mathbf{t}, \mathbf{t}_n)$ be the Levenshtein alignment between sentences $\mathbf{t}$ and $\mathbf{t}_n$, and $\mathcal{L}_i(\mathbf{t}, \mathbf{t}_n)$ that of word t in position i in $\mathbf{t}$. Consider the following example: Calculating the Levenshtein alignment between the sentences $\mathbf{t}$ =“A B C D E” and $\mathbf{t}_n$ =“B C G E F” yields

$$\mathcal{L}(\mathbf{t}, \mathbf{t}_n) = \text{“– B C G E”},$$

where "–" represents insertion of the word A into $\mathbf{t}$ and in the above alignment F is deleted from $\mathbf{t}_n$. Using this representation, the word posterior probability of word t occurring in a position Levenshtein-aligned to i is given by

$$p_{\text{lev}}(t \mid \mathbf{s}, \mathbf{t}, \mathcal{L}) = \frac{\sum_{n=1}^{N} \delta(t, \mathcal{L}_i(\mathbf{t}, \mathbf{t}_n)) \cdot p(\mathbf{s}, \mathbf{t}_n)}{\sum_{n=1}^{N} p(\mathbf{s}, \mathbf{t}_n)}. \tag{12.4}$$

The sum is normalized by the total probability mass of the n-best list. To obtain a score for the whole target sentence, the posterior probabilities of all target words are multiplied. The sentence probability is approximated by the probability $p(\mathbf{s}, \mathbf{t}_n)$ which the SMT system assigns to the sentence pair. More details on computing word posterior probabilities are available in Ueffing and Ney (2007).

The phrase posterior probabilities are determined in a similar manner by summing the sentence probabilities of all translation hypotheses in the n-best list which contain this phrase pair. The segmentation of the sentence into phrases is provided by the SMT system. Again, the single values are multiplied to obtain a score for the whole sentence.

The language model score is determined using a 5-gram model trained on the English Gigaword corpus for Chinese–English. On French–English, we used the trigram model which was provided for the NAACL 2006 shared task.

The log-linear combination of the different sentence scores into one confidence score is optimized with respect to sentence classification error rate (CER) on the development corpus. The weights in this combination are optimized using the Downhill Simplex algorithm (Press et al., 2002). In order to carry out the optimization, reference classes are needed which label a given translation as either correct or incorrect. These are created by calculating the word error rate (WER) of each translation and labeling the sentence as incorrect if the WER exceeds a certain value, and correct otherwise. Then the confidence score $c(\mathbf{t})$ of translation $\mathbf{t}$ is computed, and the sentence is classified as correct or incorrect by comparing its confidence to a threshold τ:

$$c(\mathbf{t}) \begin{cases} > \tau & \Rightarrow \mathbf{t} \text{ correct} \\ \leq \tau & \Rightarrow \mathbf{t} \text{ incorrect.} \end{cases}$$

The threshold τ is optimized to minimize CER. We then compare the assigned classes to the reference classes, determine the CER, and update the weights accordingly. This process is iterated until the CER converges.

12.3.6 The Selection Function

The Select function in algorithm 12.2 is used to create the additional training data T_i which will be used in the next iteration $i + 1$ by Estimate to augment the

information from the original bilingual training data. It has been shown in Ueffing et al. (2007a) that this selection is an important step in the algorithm and that simply keeping all generated translations yields worse results. We use the following selection functions:

Importance sampling: For each sentence $\mathbf{s}$ in the set of unlabeled sentences U_i, the *Labeling* step in algorithm 12.2 generates an n-best list of translations, and the subsequent *Scoring* step assigns a score to each translation $\mathbf{t}$ in this list. The set of generated translations for all sentences in U_i is the event space and the scores are used to put a probability distribution over this space, simply by renormalizing the scores described in section 12.3.5. We use importance sampling to select K translations from this distribution. Sampling is done with replacement, which means that the same translation may be chosen several times. Furthermore, several different translations of the same source sentence can be sampled from the n-best list. The K sampled translations and their associated source sentences make up the additional training data T_i.

Selection using a threshold: This method compares the score of each single-best translation to a threshold. The translation is considered reliable and added to the set T_i if its score exceeds the threshold. Otherwise it is discarded and not used in the additional training data. The threshold is optimized on the development beforehand. Since the scores of the translations change in each iteration, the size of T_i also changes.

Top K: This method simply keeps those translations from T_i which receive the highest scores in the scoring step.

12.4 Experimental Results

12.4.1 Setting

We ran experiments on two different corpora: one is the French–English translation task from the Europarl and Hansard corpus, and the other one is Chinese–English translation as performed in the NIST MT evaluation.[1]

The variants of algorithm 12.2 which we applied in the experiments are the ones which yielded the best results in Ueffing et al. (2007a). On the French–English task, we experimented with various settings which are described in detail in section 12.4.3. For the Chinese–English task, the application of threshold-based selection in combination with confidence scores is applied.

Table 12.1 French–English corpora

corpus	use	sentences
Europarl-training	phrase table + language model	688K
Europarl-test2006	in-domain dev1	500
Europarl-test2006	out-of-domain dev2	500
Europarl-devtest2006	dev3	2000
Hansard-training	monolingual source data	1130K
Europarl-test2006	in-domain test	1500
Europarl-test2006	out-of domain test	500

Europarl French–English

For the French–English translation task, we used the Europarl corpus as distributed for the shared task in the NAACL 2006 workshop on statistical machine translation (WMT),[2] and the Hansard corpus as distributed by ISI.[3] The corpus statistics are shown in table 12.1. The bilingual training data from the Europarl corpus is used to train translation and language models. The development sets dev1 and dev2 are used to optimize the model weights in the decoders for the baseline SMT system and the SMT system with an additional phrase table respectively. The evaluations are done on the test set provided for the NAACL 2006 French-English translation shared task, which contains 2000 in-domain sentences and 1064 out-of-domain sentences collected from news commentary. We will carry out evaluations separately for these two domains to investigate the adaptation capabilities of our methods.

Chinese–English

For the Chinese–English translation task, we used the corpora distributed for the large-data track in the 2006 NIST evaluation (see table 12.2). We used the Linguistic Data Consortium (LDC) segmenter for Chinese. A subset of the English Gigaword corpus was used as additional language model (LM) training material. Data from the Chinese Gigaword was filtered and used as additional monolingual source-language data for semisupervised learning. The multiple translation corpora multi-p3 and multi-p4 were used as development corpora. Evaluation was performed on the 2004 and 2006 test sets. The 2006 test set consists of two sections: the NIST section which was created for the NIST machine translation evaluation in 2006 and which is provided with four English references, and the GALE section which was created in the DARPA project GALE and comes with one English reference.

1. www.nist.gov/speech/tests/mt
2. www.statmt.org/wmt06/shared-task/
3. www.isi.edu/natural-language/download/hansard/

Table 12.2 Chinese–English corpora.

Corpus	Use	Sentences	Domains
non-UN	phrase table + language model	3.2M	news, magazines, laws,
UN	phrase table + language model	5.0M	UN bulletin
English Gigaword	language model	11.7M	news
Chinese Gigaword	additional source data	50K	news
multi-p3	optimize decoder	935	news
multi-p4	optimize rescoring	919	news
eval-04	test	1788	newswire, editorials, political speeches
eval-06 GALE	test	2276	broadcast conversations, broadcast news, newsgroups, newswire
eval-06 NIST	test	1664	broadcast news, newsgroups, newswire

Note that the training data consists mainly of written text, whereas the test sets comprise three and four different genres: editorials, newswire, and political speeches in the 2004 test set, and broadcast conversations, broadcast news, newsgroups, and newswire in the 2006 test set. Most of these domains have characteristics which are different from those of the training data, e.g., broadcast conversations have characteristics of spontaneous speech, and the newsgroup data is comparatively unstructured.

Evaluation Metrics

We evaluated the generated translations using three different automatic evaluation metrics. They all compare the generated translation to one or more given reference translations. The following criteria are used:

- BLEU (*bi*lingual *e*valuation *u*nderstudy)(Papineni et al., 2002): The BLEU score is based on the notion of modified n-gram precision, for which all candidate n-gram counts in the translation are collected and clipped against their corresponding maximum reference counts. These clipped candidate counts are summed and normalized by the total number of candidate n-grams.
- WER (*w*ord *e*rror *r*ate): The word error rate is based on the Levenshtein distance. It is computed as the minimum number of substitution, insertion, and deletion operations that have to be performed to convert the generated translation into the reference translation. In the case where several reference translations are provided for a source sentence, we calculate the minimal distance to this set of references as proposed in Nießen et al. (2000).

- PER (*p*osition-independent word *e*rror *r*ate) (Tillmann et al., 1997b): A shortcoming of the WER is the fact that it requires a perfect word order. In order to overcome this problem, the position-independent word can be used, comparing the words in the two sentences *without* taking the word order into account. Words that have no matching counterparts are counted as substitution errors, missing words are deletions, and additional words are insertion errors. The PER is a lower bound for the WER.

Note that BLEU score measures translation quality, whereas WER and PER measure translation errors.

We will present 95% confidence intervals for the baseline system which are calculated using bootstrap resampling. The metrics are calculated with respect to one or four English references: the French–English data comes with one reference, the Chinese–English NIST 2004 evaluation set and the NIST section of the 2006 evaluation set are provided with four references each, and the GALE section of the 2006 evaluation set comes with one reference only. This results in much lower BLEU scores and higher error rates for the translations of the GALE set (see section 12.4.2). Note that these values do not indicate lower translation quality, but are simply a result of using only one reference.

12.4.2 Chinese-English Results

On the Chinese–English translation task, we used additional source language data from the Chinese Gigaword corpus comprising newswire text for our semisupervised learning algorithm. The Chinese Gigaword sentences are sorted according to their n-gram overlap with the development corpus (see section 12.3.3). It is assumed that the test set is unknown at this point. The Chinese sentences are then divided into chunks of 5000 sentences. One of these chunks is added in each iteration as described in algorithm 12.2.

Figure 12.1 shows the BLEU score on the development set over the iterations. As is to be expected, the biggest jump occurs after the first iteration when the decoder weights are reoptimized. Up to iteration 4, which is equivalent to the use of 20,000 additional Chinese sentences in semisupervised learning, the BLEU score increases. But after that, it drops and levels off at a point which is above the baseline, but does not significantly differ from it anymore. So it seems that if the semisupervised training runs too long, it adds noise into the model rather than improving it. The overlap between the additional data and the development corpus decreases over the iterations, so that the added data might be less relevant and thus actually hurt translation quality rather than improving it.

After analyzing the results obtained on the development corpus, we evaluated three different systems on the test corpora: the system after the first iteration, which used 5000 additional Chinese sentences in semisupervised training and for which the weight optimization was carried out; the system after iteration 4 which performed best on the development set; and the system after the last iteration which

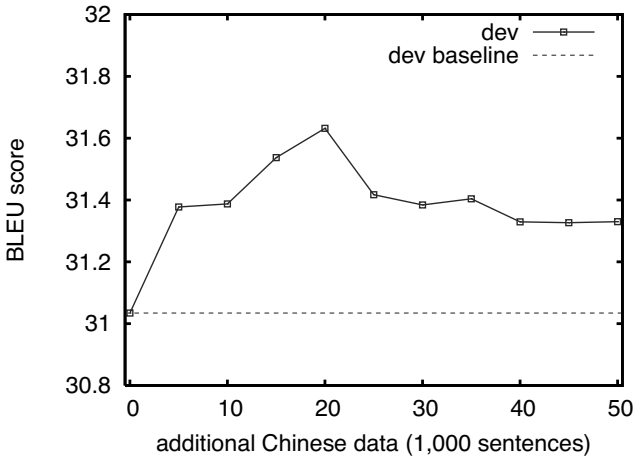

Figure 12.1 Translation quality using an additional phrase table trained on monolingual Chinese news data. Chinese–English development set.

had the highest number of additional Chinese sentences, namely 50,000, available for semisupervised learning. The last setup was tested only for comparison, as the results presented in figure 12.1 indicate already that this system might perform worse than the other two. Table 12.3 shows the translation quality achieved on the Chinese–English test sets with these three systems. The best system is clearly the one obtained after the first iteration. The translation quality is significantly better than the baseline in most cases. The system from iteration 4 (which performs best on the development set) shows a very similar performance in terms of translation quality as the first one. The error rates it achieves are slightly higher than those of the first system, but still significantly better than those of the baseline system. The third system, however, does not outperform the baseline system. As mentioned above, this was to be expected.

Table 12.4 analyzes the translation results achieved on the eval-04 test corpus separately for each genre: editorials, newswire, and political speeches. The most significant improvement in translation quality is achieved on the newswire section of the test corpus. Interestingly, all three semisupervised systems perform very similarly on this genre, whereas performance decreases on the other two genres as the number of iterations increases. This difference among the genres can be explained by the fact that the additional data is drawn from the Chinese Gigaword corpus which contains newswire data.

In Ueffing et al. (2007a), the transductive approach was evaluated on the same Chinese–English test sets. The translation quality achieved by this approach is higher than that of the method presented here (yielding a BLEU score which is 0.5 to 1.1 points higher). We see two reasons for this difference: First, transductive learning on the development or test corpus yields a model which is more focused on this corpus. It adapts the system directly to the domain and style by creating

Table 12.3 Translation quality using an additional phrase table trained on monolingual Chinese news data. Chinese–English test sets. **Bold:** best result; *italic:* significantly better than baseline.

System		BLEU[%]	WER[%]	PER[%]
eval-04 (4 refs.)				
baseline		31.8±0.7	66.8±0.7	41.5±0.5
add Chinese data	iteration 1	***32.8***	***65.7***	***40.9***
	iteration 4	*32.6*	*65.8*	***40.9***
	iteration 10	32.5	66.1	41.2
eval-06 GALE (1 ref.)				
baseline		12.7±0.5	75.8±0.6	54.6±0.6
add Chinese data	iteration 1	**13.1**	***73.9***	***53.5***
	iteration 4	13.0	*75.0*	*53.9*
	iteration 10	12.7	75.4	54.9
eval-06 NIST (4 refs.)				
baseline		27.9±0.7	67.2±0.6	44.0±0.5
add Chinese data	iteration 1	28.1	***65.8***	***43.2***
	iteration 4	**28.2**	*65.9*	*43.4*
	iteration 10	27.7	*66.4*	43.8

an additional phrase table which is specific to the development or test corpus and matches it very well. Second, the transductive approach adapts the SMT system to each of the genres. In the work presented here, the additional Chinese data came from the newswire domain only, and this yields a higher boost in translation quality for this genre than for the other ones. It would be interesting to see how the system performs if data from all domains in the test corpus is available for semisupervised learning. We also investigated a combination of the two self-training methods: using additional source language data as well as the development or test corpus for transductive learning. Unfortunately, the gains achieved by the two methods do not add up, and this system does not outperform the transductively trained one.

Table 12.5 shows how many translations were identified as confident by the scoring and selection algorithm and used to extend the additional phrase table. In the first iteration, this is approximately two thirds of the data added in the iteration. But as the semisupervised training proceeds, the number of confident sentences decreases. After ten iterations of the algorithm, less than half of the translations are kept. This confirms our assumption that noise is introduced into the procedure by running the algorithm for too long.

12.4.3 French–English Results

We ran our experiments on the French–English task to explore the behavior of the semisupervised learning algorithm with respect to the different sentence

Table 12.4 Translation quality on Chinese–English eval-04 test set, by genre. Same experimental setup as table 12.3.

System		BLEU[%]	WER[%]	PER[%]
editorials	baseline	30.7±1.2	67.0±1.1	42.3±0.9
	iteration 1	31.3	65.9	41.8
	iteration 4	30.9	66.2	42.0
	iteration 10	30.8	66.6	42.3
newswire	baseline	30.0±0.9	69.1±0.8	42.7±0.8
	iteration 1	31.1	68.1	42.0
	iteration 4	31.1	67.9	42.0
	iteration 10	31.3	68.1	42.1
speeches	baseline	36.1±1.4	62.5±1.2	38.6±0.9
	iteration 1	37.3	61.3	38.0
	iteration 4	36.8	61.5	38.0
	iteration 10	36.3	61.8	38.4

Table 12.5 Number of sentences added in each iteration. Chinese–English.

iteration	1	2	3	4	5
# confident sentences	3141	3196	3017	2889	2981
iteration	6	7	8	9	10
# confident sentences	2890	2520	2423	2324	2427

selection methods on in-domain and out-of-domain test sentences. We used 688K parallel sentence pairs from the Europarl corpus as the bilingual training data, and partitioned the NAACL 2006 WMT shared task's test set into two sets (S_{in} and S_{out}) to separate in-domain and out-of-domain test sentences. S_{in} includes the first 2000 sentences and S_{out} includes the last 1000 sentences of this test set. Then we used the first 500 sentences in S_{in} and S_{out} as the development sets dev1 and dev2, and used the rest as the test sets. As the additional monolingual source sentences, we used the French sentences in the training set of the Canadian Hansard corpus as provided by ISI.

The monolingual French sentences were sorted according to their n-gram overlap (see section 12.3.3) with the development corpora dev1 and dev2 for in-domain and out-of-domain experiments, and 5000 French sentences were added in each iteration of the semisupervised algorithm. The scoring and selection of the translations (see algorithm 12.2) were performed using the following methods:

1. a confidence score with importance sampling,
2. a length-normalized translation score with importance sampling,
3. a confidence score with a threshold, and

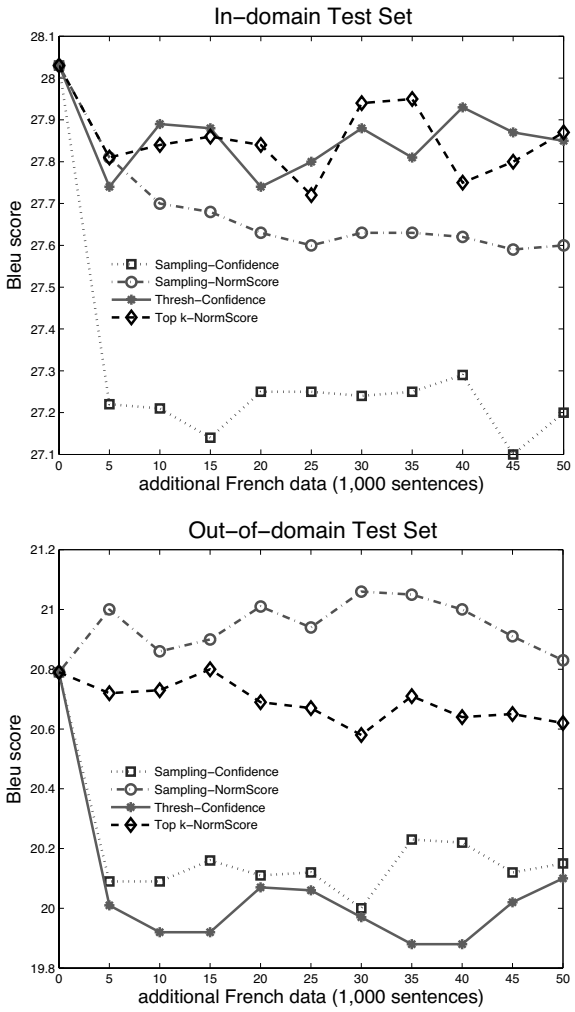

Figure 12.2 Translation quality using an additional phrase table trained on monolingual French data. The plot shows the performance of the different sentence selection and scoring schemes on in-domain and out-of-domain corpora.

4. keeping the top-K sentence pairs having the highest length-normalized translation scores.

We learn an additional phrase table on these data (and leave the original phrase tables unmodified) which is added as a new component in the log-linear model. The weight of this new component is optimized based on dev1 and dev2 for in-domain and out-of-domain experiments. Moreover, we use the development set dev3 for estimating the parameters of the confidence estimation model and the threshold in method 3.

The results of the four sentence selection methods can be seen in figure 12.2. The semisupervised algorithm deteriorates the performance of the initial SMT system for all cases except sampling with normalized translation scores. This method, however, yields an improvement on the out-of-domain data, so the system seems to adapt to this new domain. This observation is confirmed by the translation examples which will be presented in section 12.4.4. Note that the performance of top-K sentence pair selection based on the normalized scores is encouraging for both in-domain and out-of-domain experiments. It is probable that by choosing K in a more elaborate way, this method outperforms the baseline and other methods. Note that it just uses the normalized translation scores which are already generated by the decoder. Using dev1 and dev2 to train the confidence estimation models for in-domain and out-of-domain experiments may help the methods which use the confidence to boost their performance.

12.4.4 Translation Examples

Table 12.6 presents some French-English translation examples taken from out-of-domain sentences in the test set of the NAACL 2006 WMT shared task. These examples show the effect of semisupervised learning for model adaptation to a novel domain.

Table 12.7 presents some Chinese-English translation examples of the baseline and the semisupervised system using a phrase table learned on 5000 additional Chinese sentences. All examples are taken from the NIST portion of the 2006 test corpus. Except for the last example, which is taken from newswire, they come from newsgroup posts. The examples show that the semisupervised system outperforms the baseline system in terms of both adequacy and fluency.

12.5 Previous Work

Semisupervised learning has been previously applied to improve word alignments. In Callison-Burch et al. (2004), a generative model for word alignment is trained using unsupervised learning on parallel text. In addition, another model is trained on a small amount of hand-annotated word-alignment data. A mixture model provides a probability for word alignment. Experiments showed that putting a large weight on the model trained on labeled data performs best.

Along similar lines, Fraser and Marcu (2006) combine a generative model of word alignment with a log-linear discriminative model trained on a small set of hand-aligned sentences. The word alignments are used to train a standard phrase-based SMT system, resulting in increased translation quality .

In Callison-Burch (2002), co-training is applied to MT. This approach requires several source languages which are sentence-aligned with each other and all translate into the same target language. One language pair creates data for another language pair and can be naturally used in a Blum and Mitchell (1998)-style co-training

Table 12.6 Translations from the out-of-domain sentences in the NAACL 2006 French–English test corpus. The *semisupervised* examples are taken from sampling-based sentence selection with normalized scores. Lowercased output, punctuation marks tokenized.

baseline	the so-called ' grandfather bulldozer become the most of the israelis and the final asset of western diplomacy and for the americans , surprisingly , for europeans too .
semisupervised	'bulldozer' became the grandfather of most of the israelis and the final asset of western diplomacy and for the americans , surprisingly , for europeans too .
reference	the " bulldozer " had become the grandfather of most israelis and the last card of western diplomacy , for americans and , surprisingly , for europeans , too .
baseline	these are not all their exceptional periods which create bonaparte , which is probably better because leaders exceptional can provide the illusion that all problems have their solution , which is far from true .
semisupervised	these are not all periods which create their exceptional bonaparte , which is probably better because leaders exceptional can provide the illusion that all problems have their solution which is far from being true .
reference	not all exceptional periods create their bonapartes , and this is probably a good thing , for exceptional leaders may give the illusion that all problems have solutions , which is far from true .
baseline	in both cases , must be anchored in good faith moving from one to another .
semisupervised	in both cases , it must be established for faith moving from one to another .
reference	in both cases , it takes a lot of blind faith to go from one to the other .
baseline	given an initial period to experiment with growth and innovation on these fronts may prove strong paying subsequently .
semisupervised	enjoy an initial period of growth and innovation to experiment with on these fronts may prove heavily paying subsequently .
reference	using an initial period of growth to experiment and innovate on these fronts can pay high dividends later on .

algorithm. Experiments on the Europarl corpus show a decrease in WER. However, the selection algorithm applied there is actually supervised because it takes the reference translation into account.

Self-training for SMT was proposed in Ueffing et al. (2007a) where the test data was repeatedly translated and phrase pairs from the translated test set were used to improve overall translation quality. In the work presented here, the additional monolingual source data is drawn from the same domain as the test set. In particular, we *filter* the monolingual source language sentences based on their similarity to the development set as explained in section 12.3.3.

Table 12.7 Translation examples from the Chinese–English eval-06 corpus, NIST section. Lowercased output, punctuation marks tokenized.

baseline	you will continue to be arrested and beaten by villagers .
semisupervised	you continue to arrest , beat villagers ,
reference	you have continued to arrest and beat villagers .
baseline	after all , family planning is a problem for chinese characteristics .
semisupervised	after all , family planning is a difficult problem with chinese characteristics .
reference	after all , family planning is a difficult topic with chinese characteristics .
baseline	i am very disappointed in recognition of the chinese people do not deserve to enjoy democracy ! ! !
semisupervised	i am very disappointed to admit that the chinese nation do not deserve democracy ! ! !
reference	i am very disappointed to admit that the chinese people do not deserve democracy !
baseline	china has refused to talk to both sides to comment .
semisupervised	the chinese side refused to comment on both sides of the talks .
reference	china has refused to comment on the talks between the two sides .
baseline	reports said that there has been speculation that might trigger a computer in possession by the former metropolitan police chief steve vincent jazz yangguang survey of confidential information .
semisupervised	reports said that the theft triggered speculation that the computer may be in the possession of the metropolitan police chief stevenson jazz led investigation of confidential information .
reference	the report pointed out that the theft triggered speculation that the computers may contain confidential information of the probe led by former metropolitan police commissioner lord stevens .

12.6 Conclusion and Outlook

We presented a semisupervised learning algorithm for SMT which makes use of monolingual source-language data. The relevant parts of this data are identified, and then the SMT system is used to generate translations of those. The reliable translations are automatically determined and used to retrain and adapt the SMT system to a domain or style. It is not intuitively clear why the SMT system can learn something from its own output and is improved through semisupervised learning. There are two main reasons for this improvement:

First, the selection step provides important feedback for the system. The confidence estimation, for example, discards translations with low language model scores or posterior probabilities. The selection step discards bad machine translations and reinforces phrases of high quality. As a result, the probabilities of low-quality phrase pairs, such as noise in the table or overly confident singletons, degrade. The selec-

tion methods investigated here have been shown to be well suited to boost the performance of semisupervised learning for SMT.

Second, our algorithm constitutes a way of adapting the SMT system to a new domain or style without requiring bilingual training or development data. Those phrases in the existing phrase tables which are relevant for translating the new data are reinforced. The probability distribution over the phrase pairs thus gets more focused on the (reliable) parts which are relevant for the test data.

One of the key components in our approach is that translations need to be proposed for sentences in the unlabeled set (which is from the same domain as the test set), and from those translations we would like to select the ones that are useful in improving our performance in this domain. For this problem, in future work we plan to explore some alternatives in addition to the methods presented here: in translating a source sentence **f**, the difficulty in assessing the quality of a translation into the target language **e**' comes from the fact that we do not have any reference translation **e**. However, if **e**' is a good translation, then we should probably be able to *reconstruct* the input sentence **f** from it. So we can judge the quality of **e**' based on its translation **f**' (we can compute the BLEU score in this case since we *do* have access to **f**), and for this translation direction we already have the translation probability tables. This approach is an attractive alternative to the problem of selecting good translations in our algorithm.

In addition to this, it would be interesting to study the proposed methods further, using more refined filter functions, e.g., methods applied in information retrieval.

Acknowledgements

This chapter is partially based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under contract no. HR0011-06-C-0023. Any opinions, findings, and conclusions or recommendations expressed in this chapter are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA). A. S. was partially supported by NSERC, Canada (RGPIN: 264905).

13 Learning to Combine Machine Translation Systems

Evgeny Matusov
Gregor Leusch
Hermann Ney

This chapter describes how translations produced by multiple machine translation (MT) systems can be combined. We present an approach that computes a consensus translation from the outputs of several MT systems for the same sentence. Similarly to the well-established ROVER approach of Fiscus (1997) for combining speech recognition hypotheses, the consensus translation is computed by weighted majority voting on a confusion network. Faced with the problem of differences in word order between the system translations, we propose an alignment procedure that learns nonmonotone word correspondences between the individual translations using statistical modeling. The context of a whole corpus rather than a single sentence is taken into account in order to achieve high alignment quality. The confusion networks which are created from this alignment are rescored with probabilistic features such as system confidence measures and a language model. The consensus translation is extracted as the best path from the rescored lattice.

The proposed system combination approach was evaluated on well-established Chinese-to-English and Arabic-to-English large-vocabulary translation tasks. In our experiments, we combined the outputs of five state-of-the-art MT systems. Significant improvements in translation quality in comparison with the best individual MT system have been gained.

13.1 Introduction

In this chapter we will describe a method for combining the outputs of multiple machine translation systems that results in improved quality of automatic translation. We will present an algorithm that computes a consensus translation.

Currently, there exist a number of commercial and research MT systems, which are developed under different paradigms (rule-based, example-based, statistical machine translation), trained using different algorithms (e.g., hierarchical phrase-based statistical MT, syntax-augmented statistical MT) and different types and amounts of training data. When applied to the same test data, they usually exhibit different strengths and weaknesses on different sentences or sentence parts. The goal of a system combination algorithm is to produce a translation that combines the strengths of these systems (i.e., the good-quality partial translations), but leaves out the parts translated poorly by the minority of the MT engines. In this chapter, we will describe a system combination method that uses a weighted majority voting scheme coupled with a sophisticated alignment and word reordering algorithm.

In automatic speech recognition (ASR), voting schemes like the ROVER approach of Fiscus (1997) have been used widely and successfully. For ASR, the algorithm is quite straightforward, since the alignment of the hypotheses for a spoken utterance is monotone. The alignment of the recognized word sequences can be determined using the edit distance algorithm in combination with the word boundary time markers. Based on the alignment, the consensus recognition hypothesis is generated by weighted majority voting.

The same strategy is applicable to system combination in machine translation. However, the complication here is that different translation hypotheses from different systems may have different word order or sentence structure. This means that some hypotheses have to be reordered so that corresponding words can be aligned with each other. In order to perform word reordering, we have to determine a non-monotone word alignment between the hypotheses. We will show how this alignment can be effectively learned with statistical methods.

Multiengine machine translation has been studied by several research groups in recent years. Faced with the problem of word reordering, the first approaches tried to avoid it by working on the sentence level: they concentrated only on finding a way to *select* "the best" of the provided hypotheses for each sentence. Thus, the resulting translation comes from a set of already produced translations. The hypothesis selection is performed based on the scores assigned by different statistical models. The models used are n-gram language models (as in Callison-Burch and Flournoy (2001) and Nomoto (2004)), but also translation models and other features (see Paul et al., 2005). The potential of such approaches is limited, since in many cases these models come from the same family as the (weak) models used to generate the translations by one of the individual MT systems. Moreover, such approaches do not take advantage of the fact that, e.g., a part of the same sentence is translated well by the first MT system, and another part by the second MT system, etc.

The more sophisticated approaches of Bangalore et al. (2001); Jayaraman and Lavie (2005) and Rosti et al. (2007a) work on the word level. The system combination translation is created from subsentence parts (words or phrases) of the original translations. The advantage of these approaches is that a possibly new translation can be generated that includes "good" partial translations from each of the involved systems. The majority of these approaches, including the one described in

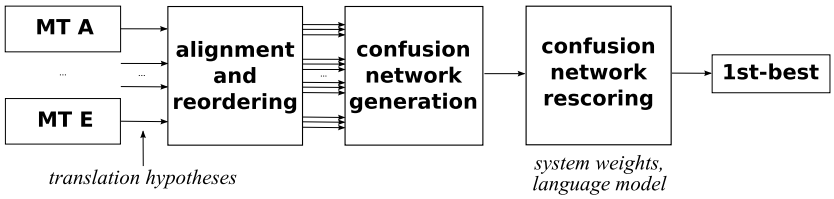

Figure 13.1 The system combination architecture.

this chapter, follow the same basic strategy for computing the system combination translation. Here is a summary of this strategy.

1. Among the individual MT system hypotheses, select the *primary* (or "skeleton") hypothesis which is assumed to have the correct word order.

2. Compute the word alignment between the primary hypothesis and each of the remaining (*secondary*) hypotheses. Generally, this alignment is nonmonotone.

3. Reorder the words in the secondary hypotheses to make the alignment with the primary translation monotone.

4. Create the *confusion network*[1] (CN) of translation alternatives to each word in the primary hypothesis, including also word insertions and deletions.

5. Score the arcs in the confusion network with different types of statistical models and select the best word sequence using weighted majority voting (i.e., as the best path in the CN).

Figure 13.1 gives an overview of this system combination architecture.

In this chapter, we will concentrate on the two most important steps in the system combination procedure: learning the word alignment and estimating the weights for scoring the confusion network. We will present the enhanced alignment algorithm of Matusov et al. (2006) and discuss its advantages in comparison with the alternative alignment approaches used in related research. We will also show which probabilistic models can improve the voting procedure.

This chapter is organized as follows. In section 13.2, we will describe the proposed word alignment approach in detail and review the related work on this subject. Section 13.3 will present the different features we used in computing the consensus translation. The experimental results, including the comparison to alternative approaches, will be presented in section 13.4. We will conclude with a summary in section 13.5.

1. A confusion network is a weighted directed acyclic graph, in which each path from the start node to the end node goes through the same sequence of all other nodes. A matrix representation of a confusion network is shown in figure 13.4.

13.2 Word Alignment

In this section we present the details of the word-alignment algorithm used in our system combination method. First, we introduce the notation in section 13.2.1. In section 13.2.2, this notation is utilized to describe the machine learning method that is used to determine the word alignment. In section 13.2.3, we show how the word order of the secondary hypotheses is changed to make the alignment monotone. We discuss alternative alignment strategies in section 13.2.4.

13.2.1 Notation

Given a single source sentence F in the test corpus, we combine M translations of this sentence $E_1, \ldots, E_m, \ldots, E_M$ coming from M MT engines. Each hypothesis E_m $(m = 1, ..., M)$ consists of I_m target language words:

$$E_m := e_{m,1}, e_{m,2}, \ldots, e_{m,i}, \ldots, e_{m,I_m}.$$

An *alignment* between two hypotheses E_n and E_m translating the same source sentence $(m, n \in \{1, ..., M\}; m \neq n)$ is generally defined as a relation $A \subseteq I_n \times I_m$ between the word positions in each of the two hypotheses. For system combination, we will consider alignments which are functions of the words in E_m, i. e. $A : \{1, \ldots, I_m\} \rightarrow \{1, \ldots, I_n\}$.

13.2.2 Estimating Word Alignment

In this section we will explain the details of the enhanced alignment approach first presented in Matusov et al. (2006). This algorithm has the following properties:

- The alignment is determined using the same machine learning methods as in statistical MT. It is an unsupervised, iterative procedure that aims at finding correspondences between words. The well-established IBM Model 1 and hidden Markov model (HMM) for alignment are trained to estimate the alignment.
- The alignment algorithm makes its decisions based on statistics learned from a whole corpus of parallel multiple translations. Thus, a correspondence between words which are not graphemically related can be learned based on their co-occurrence in the corpus. This means that, e.g., synonyms, paraphrases, and different auxiliary verbs can be matched to each other.
- The alignment algorithm takes advantage of the fact that the sentences to be aligned are in the same language. This will be done by bootstrapping the "translation" probabilities of identical and related words.
- The produced alignment is nonmonotone, which means that it can cope with translations that have significantly different syntactic structure.

- The alignment is determined using a flexible cost matrix representation. This representation allows for various extensions of the alignment which can have a positive effect on, e.g., subsequent word reordering.

Here are the details of the algorithm. For each source sentence F in the test corpus, we select one of its translations $E_n, n = 1, \ldots, M$ as the *primary* hypothesis. Then we align the *secondary* hypotheses $E_m (m = 1, ..., M; n \neq m)$ with E_n to match the word order in E_n. Since it is not clear which hypothesis should be primary, i. e., has the "best" word order, we let every hypothesis play the role of the primary translation, and align all pairs of hypotheses $(E_n, E_m);\ n \neq m$.

The word alignment is *trained* in analogy to the alignment training procedure in statistical MT, using a parallel corpus created from the multiple individual system translations for the test corpus[2] or, in addition, for any other corpus for which such translations are available. The difference is that each two sentences in the corpus that have to be aligned are in the same language. To formalize the statistical alignment training, we consider the conditional probability $p(E_m|E_n)$ of the event that, given E_n, another hypothesis E_m is generated from the E_n. Then, the alignment between the two hypotheses is introduced as a hidden variable $\mathcal{A}$:

$$p(E_m|E_n) = \sum_{\mathcal{A}} p(E_m, \mathcal{A}|E_n). \tag{13.1}$$

This probability is then decomposed into the alignment probability $p(\mathcal{A}|E_n)$ and the lexicon probability $p(E_m|\mathcal{A}, E_n)$:

$$p(E_m, \mathcal{A}|E_n) = p(\mathcal{A}|E_n) \cdot p(E_m|\mathcal{A}, E_n). \tag{13.2}$$

As in statistical machine translation, we make modeling assumptions. The alignment model is estimated using the IBM Model 1 (Brown et al., 1993) and the hidden Markov model (Vogel et al., 1996). The latter model predicts the next alignment position in dependency on the alignment of the preceding word. The lexicon probability of a sentence pair is modeled as a product of single-word–based probabilities of the aligned words:

$$p(E_m|\mathcal{A}, E_n) = \prod_{j=1}^{I_m} p(e_{m,j}|e_{n,a_j}). \tag{13.3}$$

Here, the alignment a is a function of the words in the secondary translation E_m, so that each word $e_{m,j}$ in E_m is aligned to the word $e_{n,i}$ in E_n on position $i = a_j$.

The alignment training corpus is created from a test corpus of N sentences (e. g., a few hundred) translated by the involved MT engines. However, the effective size of the training corpus is $M \cdot (M-1) \cdot N$, since all pairs of different hypotheses have to be aligned.

2. A test corpus can be used directly because the alignment training is unsupervised and only automatically produced translations are considered.

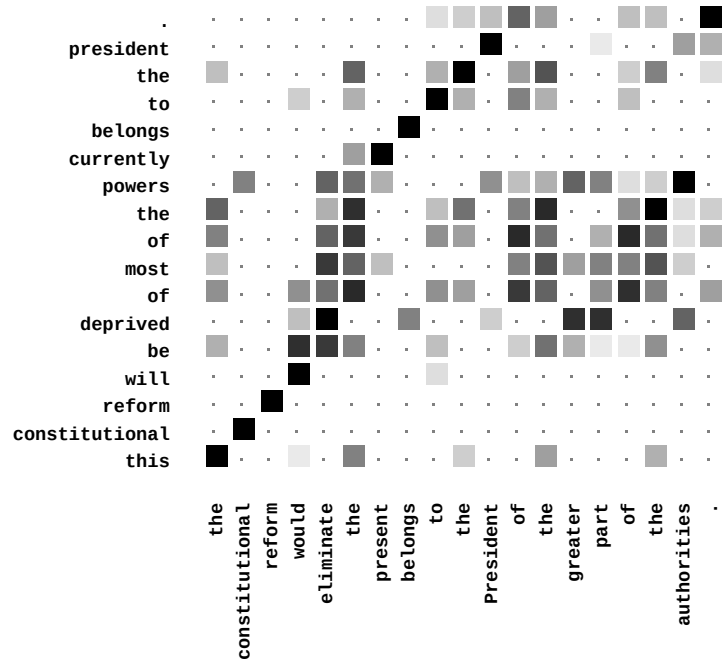

Figure 13.2 A graphical representation of an alignment cost matrix for a pair of sentences. The primary hypothesis is shown on the left, the secondary hypothesis is at the bottom. The gray color intensity of each square is inversely proportional to the alignment costs of the corresponding word pair.

The single-word–based lexicon probabilities $p(e|e')$ used in Eq.(13.3) are initialized with normalized lexicon counts collected over the sentence pairs (E_m, E_n) on this corpus. Making use of the fact that the parallel sentences are in the same language, we count co-occurring identical words, i. e., if $e_{m,j}$ is the same word as $e_{n,i}$ for some i and j. In addition, we add a fraction of a count for words with identical prefixes. We perform this initialization to "guide" the alignment algorithm so that it would always align to each other identical English words and words with the same stem, but different word form.

The model parameters – the lexicon model $p(e|e')$ and the alignment model – are trained iteratively with the EM algorithm using the GIZA++ toolkit of Och and Ney (2003). The training is performed in the directions $E_m \rightarrow E_n$ and $E_n \rightarrow E_m$. The updated lexicon tables from the two directions are interpolated after each iteration. We perform four iterations of the IBM Model 1 and five iterations of the HMM training.

The final alignments are determined using a cost matrix C for each sentence pair (E_m, E_n). The elements of this matrix are the local costs $C(i, j)$ of aligning a word $e_{m,j}$ from E_m to a word $e_{n,i}$ from E_n. Following Matusov et al. (2004), we compute these local costs by interpolating state occupation probabilities from the "source-to-target" and "target-to-source" training of the HMM model. These are marginal

probabilities of the form $p_j(i, E_m|E_n) = \sum_{a:a_j=i} Pr(E_m, \mathcal{A}|E_n)$ normalized over target positions i. The costs $C(i, j)$ are obtained by taking the negated logarithm of these probabilities. An example of a cost matrix for a sentence pair is given in figure 13.2.

For a given alignment $A \subset I_n \times I_m$, we define the costs of this alignment $C(A)$ as the sum of the local costs of all aligned word pairs. The goal is to find a minimum cost alignment fulfilling certain constraints. Two different alignments are computed using the cost matrix C: the alignment $\widetilde{a}$ used to reorder the words in each secondary translation E_m, and the alignment $\bar{a}$ used to build the confusion network.

13.2.3 Word Reordering

The alignment $\widetilde{a}$ between E_m and the primary hypothesis E_n used for reordering is determined under the constraint that it must be a function of the words in the secondary translation E_m with minimal costs. It can be easily computed from the cost matrix C as

$$\widetilde{a}_j = \arg\min_i C(i, j). \tag{13.4}$$

The word order of the secondary hypothesis E_m is changed. The words $e_{m,j}$ in E_m are sorted by the indices $i = \widetilde{a}_j$ of the words in E_n to which they are aligned. If two or more words in E_m are aligned to the same word in E_n, they are kept in the original order.

After reordering each secondary hypothesis E_m and the columns of the corresponding alignment cost matrix according to the permutation given by the alignment $\widetilde{a}$, we determine $M - 1$ monotone *one-to-one* alignments between E_n as the primary translation and $E_m, m = 1, \ldots, M; m \neq n$. This type of alignment will allow a straightforward construction of the confusion network in the next step of the algorithm. In case of many-to-one connections in $\widetilde{a}$ of words in E_m to a single word from E_n, we only keep the connection with the lowest alignment costs. This means that for each position i in E_n the unique alignment connection with a word in E_m is found with the following equation:

$$\bar{a}_i = \arg\min_{j:\ \widetilde{a}_j=i} C(i, j). \tag{13.5}$$

The use of the one-to-one alignment $\bar{a}$ implies that some words in the secondary translation will not have a correspondence in the primary translation and vice versa. We consider these words to have a null alignment with the empty word ε. In the corresponding CN, the empty word will be transformed to an ε-arc.

By using the cost matrix representation to compute the alignment (instead of the standard GIZA++ alignments) we can introduce flexibility into the alignment decisions. An extension that is effectively used in our experiments is to constrain $\widetilde{a}$ with regard to the alignment of identical words in the secondary hypothesis E_m. According to the reordering procedure, if two such words (e.g., the two words “of”

from the secondary hypothesis in the example in figure 13.2) are aligned to the same word in the primary hypothesis, they will be placed next to each other after reordering. Such word repetitions are in most cases undesirable. To avoid them, we extend the algorithm for computing the alignment $\widetilde{a}$ in Eq.(13.4) by introducing an additional constraint that identical words $e_{m,j} = e_{m,j'}$ in E_m cannot be all aligned to the same word $e_{n,i}$ in E_n. If two such connections are found, the one with the higher costs in the alignment cost matrix C is discarded (e. g., for $e_{m,j'}$) and another alignment point is determined. This is the point with the lowest costs in the same column of the matrix:

$$\widetilde{a}(j') = \underset{i':i'\neq i}{\arg\min}\, C(i', j'). \tag{13.6}$$

With this extension, the alignment of the phrase `deprived of most of the powers` from figure 13.2 with parts of the reordered secondary hypothesis improves from:

```
deprived     $     $    $  of most $  of $   the powers
eliminate greater part the $   $   of of the the authorities
```

to the more appropriate

```
deprived    $      $    $  of most of the powers
eliminate greater part the of the  of the authorities
```

13.2.4 Alternative Alignment Strategies in Related Work

Some alternative alignment procedures for MT system combination can be found in the literature. We will shortly present them here and discuss their qualities.

Sentence level combination: In the simplest case, we can consider *full* alignment of all words in one hypothesis with all words of the other hypothesis. In this case, the hypotheses are combined at the sentence level, as already mentioned in section 13.1. No majority voting can be performed at the word level anymore, but other statistical models and features described in section 13.3 can be applied to select the best translation from the given ones. In our experiments, we will compare this baseline approach with the enhanced alignment algorithm described above.

Monotone alignment based on the edit distance: Another approach, which may be suitable for translation between languages with similar sentence structure, is to ignore the reordering problem and determine a monotone alignment only. Bangalore et al. (2001) use the edit distance alignment extended to multiple sequences to construct a confusion network from several translation hypotheses. This approach would fail to align well the translation hypotheses with significantly different word order.

Local nonmonotone alignments: Recently, some researchers suggested computing nonmonotone alignments between the multiple translation hypotheses. However, the key difference from the word alignment presented in this chapter is that the proposed alignment procedures are *local*, i.e. they consider only the words in the translations for a particular sentence in order to create the alignment for this

TER alignment	HMM alignment
I think that you know # you will be aware , I believe	
$ $ I think that you know will be aware , I you believe	I think that you $ $ know I believe , you will be aware
a huge fall in average prices # a decline strong in the prices means	
a huge fall in average prices $ a decline strong in the prices means	a huge fall in $ average prices a strong decline in the means prices

Figure 13.3 Examples of the TER-based alignment in comparison with the alignment produced by the enhanced alignment and reordering algorithm of Matusov et al. (2006) (HMM alignment). In each example, the second translation is reordered to match the word order of the first one, given the alignment. The $ symbol denotes deletions/insertions in the alignment.

sentence. They do not take into account the co-occurrences of these words in the multiple translations of other sentences. Thus, the alignment is not learned statistically. In contrast, the enhanced hidden Markov model alignment algorithm presented in Matusov et al. (2006) and explained in detail in this chapter makes the alignment decisions depend on probabilities iteratively trained on a whole corpus translated by the participating MT systems. Thus, the alignment of synonyms and other related words can be learned automatically. Also, since a test corpus often contains whole documents relating to a certain topic, the word alignment can be implicitly adapted to the differences in word/phrase usage within a particular document.

Jayaraman and Lavie (2005) were the first to introduce a method that allows for local nonmonotone alignments of words in different translation hypotheses for the same sentence. Their approach uses many heuristics and is based on the alignment that is performed to calculate a specific MT error measure; performance improvements have been reported only in terms of this measure.

Rosti et al. (2007a) suggested using alignment based on the translation error rate (TER) (Snover et al., 2006). This alignment procedure computes the edit distance extended by shifts of word blocks. Only exactly matching phrases can be shifted, and the shifts are selected greedily. The costs of aligning synonyms to each other are the same as those of aligning completely unrelated words. As a result, the synonyms often will not be matched to each other, but will be considered as insertions or deletions in their original positions. This is suboptimal for CN voting, for which it is important to align as many corresponding words as possible, considering reasonable reorderings of words and phrases. Examples in figure 13.3 indicate that the TER-based alignments and word reordering used by Rosti et al. (2007a) are inferior to those computed with the machine learning technique described in section 13.2.2. Experimental results in section 13.4 support this observation.

Alignment with the source sentence: Finally, a few other system combination approaches do not perform the alignment between the hypotheses directly, but rather rely on the alignment with the source sentence. In one of the first publications

System hyps	**0.25 would your like coffee or tea** 0.35 have you tea or coffee 0.10 would like your coffee or 0.30 I have some coffee tea would you like								
Alignment and reordering	have|**would** you|**your** $|**like** coffee|**coffee** or|**or** tea|**tea** would|**would** your|**your** like|**like** coffee|**coffee** or|**or** $\|**tea** I\|**$** would|**would** you|**your** like|**like** have|**$** some\|**$** coffee|**coffee** $|**or** tea|**tea**								
Confusion network	**$**	**would**	**your**	**like**	**$**	**$**	**coffee**	**or**	**tea**
	$	have	you	$	$	$	coffee	or	tea
	$	would	your	like	$	$	coffee	or	$
	I	would	you	like	have	some	coffee	$	tea
Voting	**$**	**would**	**you**	$	**$**	**$**	**coffee**	**or**	**tea**
	0.7	0.65	0.65	0.35	0.7	0.7	1.0	0.7	0.9
	I	have	your	**like**	have	some		$	$
	0.3	0.35	0.35	0.65	0.3	0.3		0.3	0.1
Consensus translation	would you like coffee or tea								

Figure 13.4 Example of creating a confusion network from monotone one-to-one word alignments (denoted with symbol |). The words of the primary hypothesis are printed in bold. The symbol $ denotes a null alignment or an ε-arc in the corresponding part of the CN.

on system combination in MT, Nirenburg and Frederking (1994) created a chart structure where target language phrases from each system are placed according to their corresponding source phrases, together with their confidence scores. A chart-walk algorithm is used to select the best translation from the chart. More recently, Huang and Papineni (2007) and Rosti et al. (2007a) show that a system combination translation can be produced by performing a new search with one of the involved phrase-based MT systems, but using only the phrases from the translation hypotheses provided by the participating systems together with their alignment with the source sentence phrases. The advantage of this method is that good phrasal translations can be preserved. However, the sentence structure of the system combination translation is limited to the structure of the source sentence or to the word order that the new search can produce. In the second case, this introduces a bias toward the particular MT system which implements this search. This means that a good sentence structure produced by one of the other MT systems cannot be followed. Also, all of the systems either have to provide phrasal alignments with word sequences in the source sentence (they may not be available for rule-based systems), or this alignment has to be determined with an automatic algorithm that can introduce additional errors.

13.3 Confusion Network Generation and Scoring

In this section, we first show how a confusion network can be created from the word alignments learned as described in section 13.2. Then we explain how we estimate and use different probabilistic features in order to select the best translation from the CN as the system combination translation.

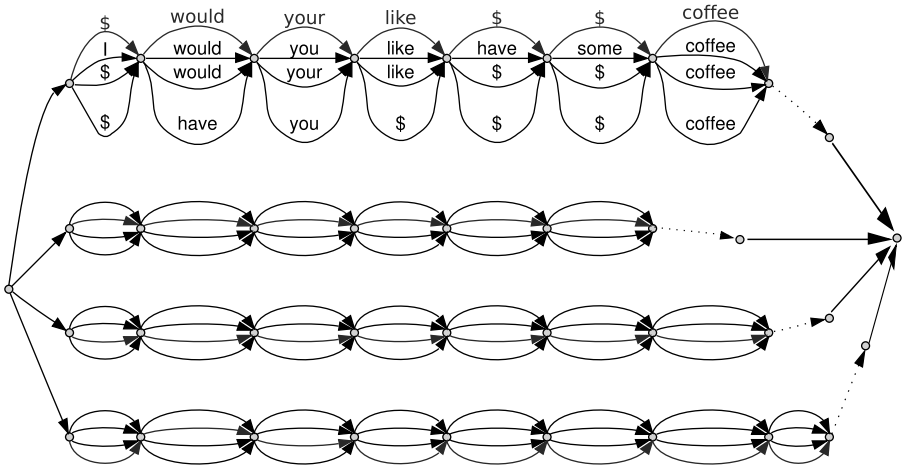

Figure 13.5 Union of several confusion networks, including the one shown in figure 13.4.

13.3.1 Building Confusion Networks

With the $M - 1$ monotone one-to-one alignments with the primary hypothesis which are obtained after reordering of the words in the secondary hypotheses (see section 13.2.3), we can follow the approach of Bangalore et al. (2001) with some extensions to construct the CN. The method is best explained by the example in figure 13.4. Here, the original $M = 4$ hypotheses are shown, followed by the alignment of the reordered secondary hypotheses 2 to 4 to the primary hypothesis 1 (shown in bold). The alignment is shown with the | symbol, where the words of the primary hypothesis are to the right of this symbol. The symbol \$ denotes a null alignment or an ε-arc in the corresponding part of the CN.

Starting from an initial state s_0, the primary hypothesis E_n is processed from left to right and a new state is produced for each word $e_{n,i}$ in E_n. Then an arc is created from the previous state to this state, for $e_{n,i}$ and for all words (or the null word) aligned to $e_{n,i}$. If there are insertions following $e_{n,i}$ (for example, "have some" in figure 13.4), the states and arcs for the inserted words are also created. When several word sequences from different secondary translations are to be inserted between two consecutive primary words $e_{n,i}$ and $e_{n,i+1}$, we compute the edit distance alignment between all the insertions to make sure that identical words are aligned to each other.

For each sentence, we obtain M CNs of the type shown in figure 13.4 by letting each individual translation hypothesis play the role of the primary hypothesis. The consensus translation can be extracted only from one of these confusion networks, i. e., from the one in which the primary hypothesis was produced by a generally

better-performing MT system. However, the word order of the resulting consensus translation will follow the word order of the primary translation, which may still be erroneous for some sentences. Because of that, a better strategy is to consider multiple primary hypotheses at once. Our experiments show that it is advantageous to unite the M CNs in a single lattice as shown in figure 13.5. Then the consensus translation can be chosen from different alignment and reordering paths in this lattice.

13.3.2 Probability Estimation

To extract a good consensus translation from the lattice shown in figure 13.5, we have to estimate the probabilities of its arcs, i.e., have to specify the weights of the words produced by each individual MT system involved in system combination. First, we will show how to estimate the weights for a single confusion network created based on the primary hypothesis E_n from the MT system n. We refer to $K = K(n)$ as the length of a path in the confusion network that was created from the alignment of the secondary hypotheses E_m with E_n. For each state $s_k, k = 0, \ldots, K$ in the CN, we denote by e_{mk} the word or the empty word which is produced at position k by MT system $m, m = 1, \ldots, M$ (i.e., an arc labeled with this word exists between the states s_{k-1} and s_k). The probability $p_k(e|F, E_n)$ of each unique word or the empty word e at position k that is used in "voting" is obtained by summing the system-specific weights over all systems $m, m = 1, \ldots, M$ which have the word e at this position:

$$p_k(e|F, E_n) = \frac{\sum_{m=1}^{M} \delta(e_{mk}, e) \cdot (\gamma_m + \beta \cdot \delta(m, n))}{\sum_{\tilde{e}} \sum_{m=1}^{M} \delta(e_{mk}, \tilde{e}) \cdot (\gamma_m + \beta \cdot \delta(m, n))}. \tag{13.7}$$

Here, γ_m are global weights that are independent of a particular word e, but give an a priori estimation of the translation quality of the MT system m. The value β is added to the system weight only when the word at the concerned arc belongs to the primary hypothesis with index n. The parameter β is used to control the deviation from the original sentence structure and is appropriate for the case of M CNs. By increasing β, we favor the words (and thus the word order) of the original hypotheses. For very large values of β, one of the original word sequences will always outweigh the alternatives at each word position, so that the system combination approach is reduced to selection on the sentence level, as described in section 13.1. A simplified voting procedure is depicted in the lower part of figure 13.4.

The set of parameters $\{\gamma_1, \ldots, \gamma_M, \beta\}$ is adjusted based on the performance of the involved MT systems on a development set in terms of an automatic MT evaluation measure (see section 13.3.7). Generally, a better consensus translation can be produced if the words hypothesized by a better-performing system get a higher probability.

13.3.3 Combining R-Best Translations from Each MT System

In a recent extension of the presented system (see also Mauser et al. (2007)), we make use of the fact that many MT systems can produce not only the single best translation but a list of R top-ranked translations for each sentence (e.g., $R = 10$). From each of the involved systems we take these R translations to better estimate the weight of each word in the confusion network.

To keep the computational costs of the algorithm low, we continue to consider only the M single best translations from the M systems as the primary hypotheses. To each primary hypothesis, we align the remaining $R - 1$ hypotheses from the "primary" system as well as the $(M - 1) \cdot R$ hypotheses from the other systems. The alignment is performed iteratively as described in section 13.2.2. The CN is created in analogy to the procedure described in section 13.3.1, but at each state in the CN we insert $M \cdot R$ arcs to the next state. To perform the weighted majority voting on this CN, we define $e_{m,r,k}$ as the word (or the empty word) at position k coming from the translation with rank[3] r of the MT system with index m. The probability for a distinct word e is then computed by summing system-specific weights over all systems and over all hypotheses of each individual system. This is formalized with the following equation, which extends Eq.(13.7).

$$p_k(e|F, E_n) = \frac{\sum_{m=1}^{M} \sum_{r=1}^{R} r^{-1} \delta(e_{m,r,k}, e) \cdot (\gamma_m + \beta \cdot \delta(m, n))}{\sum_{\tilde{e}} \sum_{m=1}^{M} \sum_{r=1}^{R} r^{-1} \delta(e_{m,r,k}, \tilde{e}) \cdot (\gamma_m + \beta \cdot \delta(m, n))}. \tag{13.8}$$

Thus, if, e.g., a word appears in the tenth best hypothesis of a single system, it has a ten times smaller weight than a word appearing in the single best hypothesis of the same system.

The idea behind using R-best lists from each system is first and foremost to improve the weight estimation. Words appearing in the majority of the R-best translations are considered more probable than those appearing in one or two translations only. By using multiple translations from each of the MT systems we also introduce additional translation alternatives at the word level, but their number is usually limited, since the R-best translations of the same MT system are often quite similar to each other.

A similar way of using R-best lists as input for MT system combination is proposed by Rosti et al. (2007b).

13.3.4 Extracting Consensus Translation

In the basic voting procedure (figure 13.4), the consensus translation is extracted as the best path from a single CN. The position-dependent probabilities $p_k(e|F, E_n)$ as

3. Rank r is the position of the translation in the list of R-best translations of a single MT system. The single best translation has the rank $r = 1$.

given by Eq.(13.7) or Eq.(13.8) are used to score each path. We define the consensus translation extracted from the CN with the primary translation of MT system n as the sequence $\hat{e}_1^K(n) := \hat{e}_{n,1}, \ldots, \hat{e}_{n,k}, \ldots, \hat{e}_{n,K(n)}$ where, at each position k in the CN, the best word $\hat{e}_k$ is selected by the following equation[4]:

$$\hat{e}_k = \arg\max_e \{p_k(e|F, E_n)\}\,. \tag{13.9}$$

The ε-arcs have to be removed from $\hat{e}_1^K$ to obtain a translation with proper words only. Note that the extracted consensus translation can be different from each of the original M (or $R \cdot M$) translations.

Due to the presence of ε-arcs in the CN, it may contain multiple identical word sequences. The number of identical sequences will increase if we extract the globally best translation from the lattice which is a union of M CNs, as described in section 13.3.1. To improve the estimation of the score for the best hypothesis, we deviate from Eq.(13.9) and sum the probabilities of identical partial paths. This is done through determinization of the lattice in the log semiring. This approach also allows us to extract n-best hypotheses without duplicates. In a subsequent step, these n-best lists could be rescored with additional statistical models just like any n-best lists from a single MT system (see, e.g., Mauser et al. (2006)).

13.3.5 Language Model Rescoring

Experimental results show that the consensus translation computed by weighted majority voting often exhibits significantly better word choice than any of the individual system translations. However, the freedom to choose any word or the empty word at each position in the CN, as well as the reordering of the secondary hypotheses, often leads to insufficient fluency of the system combination translation. To improve its word order, we employ an n-gram language model (LM). The lattice representing a union of several CNs can be directly rescored using a language model. For the LM rescoring, a transformation of the lattice is required, since the LM history has to be memorized. The ε-arcs are removed as a result of this transformation. This affects the probabilities $p_k(e|F, E_n), k = 1, \ldots, k, \ldots, K$ from Eq.(13.7) or Eq.(13.8) which are redefined in terms of proper words only. For a single full path $e_1, \ldots, e_i, \ldots, e_I$ of length I in the transformed lattice, these probabilities can be expressed with $p(e_i|F), i = 1, \ldots, i, \ldots, I$. The language model probability can then be included in the decision criterion using log-linear interpolation. In case of a trigram language model, the modified decision criterion is described with the following equation:

$$(\hat{I}, \hat{e}_1^{\hat{I}}) = \arg\max_{I, e_1^I} \{\sum_{i=1}^{I} (\log p(e_i|F) + \lambda \cdot \log p(e_i|e_{i-1}, e_{i-2}) + \alpha)\}. \tag{13.10}$$

4. Index n is omitted here for better readability.

Here, the maximization is performed over all of the paths in the LM-rescored lattice. λ is the LM scaling factor, and α is a word penalty that is used to avoid the bias toward short sentences. Note again that Eq.(13.10) is an approximation, since in practice the probabilities are summed over identical partial paths when the rescored lattice is determinized.

Whereas a general target LM can be successfully used for rescoring, as shown by Rosti et al. (2007b), in our experiments the translation fluency improves significantly by using an *adapted* LM learned from the outputs of the MT systems for the test corpus on which the system combination translation is to be determined. We attribute this to the fact that the systems we combine are all phrase-based systems. Using this special LM for lattice rescoring gives a bonus to n-grams from the original system hypotheses, in most cases from the original phrases. Presumably, many of these phrases have a correct word order, since they were extracted from the training data. Moreover, these phrases were selected as the best ones in translation, which means that a general LM has already given them high probability.

13.3.6 Preserving Word Case Information

Some MT systems produce English words with their true case (e.g., starting names with a capital letter), while others only produce lowercase translations. Whereas it is of advantage to convert all words to lowercase in order to better estimate the word alignment, it is a good idea to preserve the case information when extracting the system combination translation from the confusion network or lattice. However, we have to make sure that if a certain word wins the weighted majority when the case information is ignored, the same word should have this majority when the case information is considered, i.e., if two versions of the same word produced by different MT systems are aligned to each other. To achieve this, we perform the summation in Eq.(13.7) or Eq.(13.8) over the lowercase versions of each unique word e (i.e., we add together the probability of a lowercased and truecased version of e), but keep a separate arc in the lattice for each version of e, and assign this sum to it as its probability. In the subsequent lattice rescoring, the adapted language model will have the key role in deciding what version of the word is appropriate at this particular position.

13.3.7 Optimization of System Combination Parameters

An optimal parameter set $\{\gamma_1, \ldots, \gamma_M, \beta, \alpha, \lambda\}$ – the global system probabilities, the bonus for the primary system, the LM factor, and the word penalty – can be learned automatically using a development corpus, with the goal of improving an automatic evaluation measure.

To perform the optimization, we utilized the CONDOR optimization tool of Vanden Berghen and Bersini (2005). The optimization algorithm allows us to specify upper and lower bounds for each weight. In particular, we constrain the system probabilities γ_m to sum to 1. For the optimization, the confusion networks can be

Table 13.1 Corpus statistics for the development and test data. The Chinese word segmentation was performed using the Linguistic Data Consortium (LDC) segmentation tool.

	Chinese → English		Arabic → English	
	DEV	TEST	DEV	TEST
Sentences	554	1082	533	1056
Running source words	17,767	32,611	18,053	31,375
Avg. running target words (references)	21,437	34,567	21,931	35,648
Avg. running target words (MT hyps)	21,116	36,808	21,659	35,963
Vocabulary size (all MT hyps)	6677	8296	6128	7430

kept fixed, since the parameters involved do not affect the alignment. In each step of the optimization algorithm, the CNs for the development set are scored using a set of parameters mentioned above, and the consensus translation is extracted by finding the best path through the rescored lattice. Then, the system weights, the scaling factor for the LM, and the word penalty are updated. In our experiments, 100 to 150 iterations are necessary for convergence.

13.4 Experiments

In this section, we describe the experimental results for the presented system combination algorithm. We quantify the importance of individual algorithm features such as the enhanced word alignment and show significant improvements in translation quality in comparison with the best of the individual MT systems involved.

13.4.1 Translation Task and Conditions

Significant improvements with the presented approach were achieved on many tasks and conditions; see Matusov et al. (2006). The approach was successfully used in evaluations within important MT projects like TC-STAR (2007) and GALE (2007). Here, we will present the results for the Chinese-to-English and Arabic-to-English translation tasks. The individual MT systems which we combine are all large vocabulary systems trained on large amounts of bilingual data for the GALE 2007 Go/NoGo MT evaluation. As the development data, we used the newswire portion of the official 2007 development set. As the evaluation data, we consider the test set of the NIST 2005 machine translation evaluation (see Doddington (2002)), which also consists of newswire documents. The corpus statistics for the development and test data are presented in table 13.1.

The five MT systems used in system combination experiments were all statistical machine translation systems. Whereas four out of five Arabic-to-English systems employed phrase-based search with a distance-based reordering model in the style of Zens and Ney (2004) and Koehn (2004a), one of the systems was a hierarchical phrase translation system. For Chinese-to-English translation, two out of five sys-

Table 13.2 Effect of the alignment and cost estimation techniques (case-insensitive error measures in %, NIST 2005 Chinese-to-English evaluation data)

System	BLEU	TER	WER	PER
Worst system (BLEU)	29.2	65.8	73.1	45.8
Best system (BLEU)	33.8	61.9	69.2	43.5
Selection	34.1	58.4	65.1	41.4
HMM alignment (single CN)	36.3	56.3	63.1	39.7
TER alignment (5 CNs)	36.2	55.3	62.1	39.5
HMM alignment (5 CNs)	36.8	55.7	62.9	39.0
+ 10-best translations as input	37.1	56.1	63.6	38.5

tems were purely phrase-based, one system used hierarchical phrases, and another one augmented the phrase-based translation with reordering of the source sentences based on syntactic chunks. The fifth system was a hybrid system combining a rule-based and a statistical translation system.

13.4.2 Evaluation Criteria

To assess the translation quality and compare the systems, we use the following well-established automatic evaluation measures: the BLEU score as in Papineni et al. (2002), the translation error rate (Snover et al., 2006), the word error rate (WER), and the position-independent word error rate (PER) (Tillmann et al., 1997b). These error measures were computed using a single reference translation for the development data and four reference translations for the test data. Punctuation marks were included in the evaluation as separate tokens.

13.4.3 Comparative Experiments

Table 13.2 presents the results of several comparative experiments performed on the Chinese-to-English task. For each experiment, the system weights and other parameters like the scaling factor for the language model were optimized automatically as described in section 13.3.7 on the development set. The results are reported on the evaluation set. As the objective function for optimization, we tested BLEU and TER, as well as the combination of both measures: $TER+(1.0-BLEU)$. Using the combination of BLEU and TER was more stable in terms of generalization from the development to the evaluation data. In addition, the resulting hypotheses had a more reasonable average sentence length than when a single evaluation measure had been used.

As the baseline experiment, we performed the selection of one of the individual system translations for each sentence. For this purpose, we created a word lattice with only five paths representing the original system translations and scored this lattice with system weights, the adapted LM, and the word penalty. The result in table 13.2 shows that this approach improves the overall translation quality in

Table 13.3 Case-insensitive/case-sensitive evaluation of the case preservation component (NIST 2005 Chinese-to-English MT evaluation data, error measures in %)

System	BLEU	TER	WER	PER
Best single system	33.8/31.0	61.9/65.3	69.2/72.2	43.5/47.3
Lowercase system combination	36.6/34.0	56.8/59.2	64.0/65.9	39.6/42.6
Truecase system combination	36.8/34.0	55.7/59.4	62.9/66.0	39.0/42.9

comparison with the best single system. However, the improvement is not significant in terms of BLEU and the approach is inferior to all variants of the presented algorithm in which a consensus translation is computed. This is an expected result, since the selection approach is not able to generate output that is different from the individual system translations.

In the next experiment, we determined a genuine consensus translation, but based only on one confusion network. As the primary hypothesis for this CN we took the translations of the system that performed best in terms of BLEU.[5] The improvements in BLEU, TER, and other measures with this simplified system combination are already significant at the 95% confidence level, as computed using bootstrap estimation. However, the translation quality can be improved even further if we unite the five CNs in a single lattice and extract the best translation from this lattice.

We also have compared the enhanced HMM algorithm presented in this work with the TER-based alignment algorithm as used by Rosti et al. (2007a). The TER-based alignment improves the TER by 0.4% absolute w.r.t. the HMM alignment. This is not surprising, since TER (and also WER, which is similar to TER) uses the same type of alignment to compute the error measure as the tested alignment algorithm. However, using the HMM alignment to create the CN improves the BLEU score and position-independent error rate.

Using 10-best translations from each system also improves BLEU and PER. This means that the system combination algorithm can make a better word choice due to better estimation of weights in the voting procedure. However, this seems to happen at the expense of the translation fluency. We plan to continue research on how to use multiple translations from each system in a more effective way.

In order to test the usefulness of the system feature that allows preservation of word case information, we performed system combination using lowercase translations only and compared the results with the truecased consensus translation using both case-insensitive and case-sensitive evaluation. For the latter type of evaluation, we had to use the `disambig` truecasing tool from the SRI LM toolkit of Stolcke (2002) to restore the case of the words in the lowercase consensus translation. Table 13.3 shows the error measures for this experiment. Interestingly, the truecase system combination performs slightly better when evaluated without considering

5. A more sophisticated strategy is followed by Sim et al. (2007), who take the minimum Bayes risk hypothesis as the primary one.

Table 13.4 Error measures (in %) for the NIST 2005 Chinese-to-English MT evaluation data

System	BLEU	TER	WER	PER
A	31.0	65.3	72.2	47.3
B	30.8	63.1	70.6	44.6
C	30.2	63.8	70.7	45.4
D	30.1	61.9	68.4	44.6
E	26.7	68.9	75.8	49.4
System combination	34.1	59.3	66.3	42.2

Table 13.5 Error measures (in %) for the NIST 2005 Arabic-to-English MT evaluation data

System	BLEU	TER	WER	PER
A	53.2	41.3	43.8	31.1
B	51.4	42.1	44.8	31.6
C	47.6	45.1	48.2	33.9
D	42.9	46.3	48.9	35.9
E	41.5	50.2	53.2	37.0
System combination	55.0	39.3	41.6	30.2

the case. This means that the probabilistic dependencies (including the language model) help to make better word choice when we differentiate between lowercase and uppercase words. On the other hand, when we perform case-sensitive evaluation, the two methods have very similar error rates. The results therefore indicate that preserving case information during system combination is a good tradeoff between the actual translation quality regarding lexical choice and the case restoration quality.

13.4.4 Final Results

In tables 13.4 and 13.5, we present the final case-sensitive results of our experiments with the best system settings (HMM alignment, five CNs combined, 10-best translations as input). These settings were used also in the GALE 2007 evaluation with the same participating systems.

The system combination is able to achieve large and significant improvements on both translation tasks. For instance, for Chinese-to-English translation BLEU improves by 3.1% absolute (w.r.t. the best system A) and TER by 2.6% absolute (w.r.t. the best system D). For Arabic-to-English, BLEU improves by 1.8% absolute and TER by 2.0% absolute (w.r.t. the best system A). Note that the relative improvement in, e.g., TER and BLEU score in comparison with the best single system is larger for Chinese. For Chinese, we combined structurally different systems (i.e., systems producing quite different translations) which nevertheless were of

similar quality in terms of automatic error measures like the BLEU score. As already observed in our previous experiments and in the work of Macherey and Och (2007), this is the ideal condition for achieving good-quality translations with a consensus-based system combination method like the one presented in this chapter.

13.5 Conclusion

In this chapter, we showed how multiple MT systems can be combined with the goal of improving translation quality. We presented a method that computes a consensus translation from the aligned MT hypotheses. The method utilizes the enhanced alignment procedure of Matusov et al. (2006), as well as its extensions.

In contrast to previous approaches to system combination in MT, the method presented here includes a machine learning component. The word alignment between the translation hypotheses is a statistical, iterative procedure. The decision on how to align two translations of a sentence takes the context of a whole corpus of automatic translations into account. This high-quality nonmonotone alignment is used for finding corresponding words and subsequent word reordering when confusion networks with translation alternatives are created. Several CNs representing alternative sentence structures are combined in a single lattice. The consensus translation is extracted from this lattice using various probabilistic features which estimate the weight of each candidate word. This translation often has a significantly better quality than the original translations and may be different from each of them. In this chapter, we have formally described the alignment algorithm, as well as the weight estimation techniques.

The quality of the produced consensus translations was evaluated on the NIST 2005 Chinese-to-English and Arabic-to-English test data, which is widely used in research. For each task, we combined the outputs of five different MT systems. Our experiments have shown that a significant improvement in all automatic evaluation measures is achieved for both translation directions. We have also shown that our approach compares favorably with alternative system combination strategies.

In the future, we would like to improve word confidence estimation for the voting procedure. We would also like to consider phrases and other syntactic and semantic structures explicitly in the alignment and rescoring steps of the system combination algorithm.

Acknowledgments

This chapter is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract no. HR0011-06-C-0023.

References

Nasreen AbdulJaleel and Leah S. Larkey. Statistical transliteration for English-Arabic cross-language information retrieval. In *Proceedings of the 2003 ACM International Conference on Information and Knowledge Management (CIKM-03)*, pages 139–146. New York, ACM, 2003.

Steven Abney. Understanding the Yarowsky algorithm. *Computational Linguistics*, 30(3), 2004.

ACE-07. *Proceedings of the International Conference on Advances in Computer Entertaiment Technology*, 2007.

Eneko Agirre, Lluís Màrquez, and Richard Wicentowski, editors. *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, June 2007. Association for Computational Linguistics.

Salah Aït-Mokhtar, Jean-Pierre Chanod, and Claude Roux. Robustness beyond shallowness: Incremental deep parsing. *Natural Language Engineering*, 8(3):121–144, 2002.

Cyril Allauzen, Mehryar Mohri, Michael Riley, and Brian Roark. A generalized construction of speech recognition transducers. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP-04)*, pages 761–764, 2004.

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. Automatic acquisition of hierarchical transduction models for machine translation. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, 1998.

Enrique Amigó, Jesús Giménez, Julio Gonzalo, and Lluís Màrquez. MT evaluation: Human-like vs. human acceptable. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, 2006.

Enrique Amigó, Julio Gonzalo, Anselmo Peñas, and Felisa Verdejo. QARLA: A framework for the evaluation of automatic sumarization. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 280–289, 2005.

George Arfken. *Mathematical Methods for Physicists*. Orlando, FL, Academic Press, 1985.

Necip Fazil Ayan. *Combining Linguistic and Machine Learning Techniques for Word Alignment Improvement*. PhD thesis, University of Maryland, College Park,

2005.

Francis R. Bach and Michael I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

Rafael E. Banchs, Josep M. Crego, Adrià Gispert, Patrick Lambert, and Jose B. Mariño. Statistical machine translation of Euparl data by using bilingual N-grams. In *ACL-05 Workshop on Building and Using Parallel Texts*, 2005.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, 2005.

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. Computing consensus translation from multiple machine translation systems. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*. Madonna di Campiglio, Italy, December 2001.

Srinivas Bangalore, Patrick Haffner, and Stephan Kanthak. Statistical machine translation through global lexical selection and sentence reconstruction. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 152–159, 2007.

Srinivas Bangalore and Aravind Joshi. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–266, 1999.

Srinivas Bangalore and Giuseppe Riccardi. Stochastic finite-state models for spoken language machine translation. In *Proceedings of the Workshop on Embedded Machine Translation Systems*, pages 52–59, 2000.

Jerome R. Bellagarda. A latent semantic analysis framework for large-span language modeling. In *Proceedings of Eurospeech'97*, pages 1451–1454, 1997.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22 (1):39–71, 1996.

Clive Best, Erik van der Goot, Ken Blackler, Teofilo Garcia, and David Horby. Europe media monitor—system description. Technical report EUR 22173 EN, Joint Research Centre - European Commission, 2005.

Daniel M. Bikel. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511, 2004.

Jeff A. Bilmes and Katrin Kirchhoff. Factored language models and generalized parallel backoff. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL03)*, pages 4–6, 2003.

Christopher M. Bishop. Modeling conditional distributions. In *Neural Networks for Pattern Recognition*, chapter 6.4. Oxford, Oxford University Press, 1995.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. Confidence estimation for machine translation. Final report, Johns Hopkins University Center for Speech and Language Processing Summer Workshop, Baltimore, 2003.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Avrim Blum. Learning Boolean functions in an infinite attribute space. *Machine Learning*, 9(4):373–386, 1992.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Annual ACM Workshop on Computational Learning Theory (COLT)*, pages 92–100, 1998.

Phil Blunsom and Trevor Cohn. Discriminative word alignment with conditional random fields. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 65–72, 2006.

Antoine Bordes, Léon Bottou, Patrick Gallinari, and Jason Weston. Solving multiclass support vector machines with LaRank. In *Proceedings of the 24th International Conference on Machine Learning (ICML-07)*, pages 89–96. New York, ACM, 2007.

Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, September 2005.

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems 20*. Cambridge, MA, MIT Press, 2008.

Béatrice Bouchou, Mickael Tran, and Denis Maurel. Towards an XML representation of proper names and their relationships. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB 2005)*, pages 44–55. Alicante, Spain, June 2005.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large language models in machine translation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning (EMNLP-CoNLL)*, June 2007.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International Conference on World Wide Web (WWW7)*, pages 107–117. Amsterdam, Elsevier, 1998.

Peter Brown, Stephen Della-Pietra, Vincent Della-Pietra, and Robert Mercer. The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2): 79–85, 1990.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. A statistical approach to sense disambiguation in machine translation. In *Proceedings of the HLT'91 Workshop on Speech and Natural Language*, pages 146–151. Association for Computational Linguistics, 1991a.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. Word-sense disambiguation using statistical methods. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 264–270, 1991b.

Peter F. Brown, P. V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

Andrea Burbank, Marine Carpuat, Stephen Clark, Markus Dreyer, Pamela Fox, Declan Groves, Keith Hall, Mary Hearne, I. Dan Melamed, Yihai Shen, Andy Way, Ben Wellington, and Dekai Wu. Statistical machine translation by parsing. Final report, Johns Hopkins University Center for Speech and Language Processing Summer Workshop, Baltimore, 2005.

Ludwig M. Busse, Peter Orbanz, and Joachim M. Buhmann. Cluster analysis of heterogeneous rank data. In *Proceedings of the 24th International Conference on Machine Learning (ICML-07)*, pages 113–120. New York, ACM, 2007.

Clara Cabezas and Philip Resnik. Using WSD techniques for lexical selection in statistical machine translation. Technical report CS-TR-4736/LAMP-TR-124/UMIACS-TR-2005-42, University of Maryland, College Park, 2005.

Chris Callison-Burch. Co-training for statistical machine translation. Master's thesis, School of Informatics, University of Edinburgh, 2002.

Chris Callison-Burch and Raymond Flournoy. A program for automatically selecting the best output from multiple machine translation engines. In *Machine Translation Summit VIII*, pages 63–66, September 2001.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluating the role of Bleu in machine translation research. In *Proceedings of the 11th European Chapter of the Association for Computational Linguistics (EACL)*, April 2006.

Chris Callison-Burch, David Talbot, and Miles Osborne. Statistical machine translation with word and sentence-aligned parallel corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 2004.

Jérôme Callut and Pierre Dupont. F_β support vector machines. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1443–1448. Montreal, 2005.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean-Michel Renders. Word-sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082, 2003.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning (ICML-07)*, pages 129–136. New York, ACM, 2007.

Andrew J. Carlson, Chad M. Cumby, Jeff L. Rosen, and Dan Roth. The SNoW learning architecture. Technical report UIUCDCS-R-99-2101, University of Illinois at Urbana-Champaign Computer Science Department, Urbana, IL, May 1999.

Marine Carpuat, Yihai Shen, Yu Xiaofeng, and Dekai Wu. Toward integrating semantic processing in statistical machine translation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT-06)*, pages 37–44, November 2006.

Marine Carpuat and Dekai Wu. Evaluating the word sense disambiguation performance of statistical machine translation. In *Proceedings of JCNLP*, 2005a.

Marine Carpuat and Dekai Wu. Word sense disambiguation vs. statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 2005b.

Marine Carpuat and Dekai Wu. How phrase sense disambiguation outperforms word sense disambiguation for statistical machine translation. In *Proceedings of the 11th Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, 2007a.

Marine Carpuat and Dekai Wu. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning (EMNLP-CoNLL)*, pages 61–72, June 2007b.

Xavier Carreras, Isaac Chao, Lluís Padró, and Muntsa Padró. Freeling: An open-source suite of language analyzers. In *Proceedings of the 4th Language Resources and Evaluation Conference (LREC)*, pages 239–242, May 2004.

Xavier Carreras, Lluís Màrquez, and Jorge Castro. Filtering-ranking perceptron learning for partial parsing. *Machine Learning*, 59:1–31, 2005.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 33–40, 2007.

Pi-Chuan Chang and Kristina Toutanova. A discriminative syntactic word order model for machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 9–16, 2007.

Olivier Chapelle, Mingmin Chi, and Alexander Zien. A continuation method for semi-supervised SVMs. In *Proceedings of the 23rd International Conference on Machine Learning (ICML-06)*, pages 185–192. New York, ACM, 2006.

Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 132–139. San Francisco, Morgan Kaufmann, 2000.

Eugene Charniak. Immediate head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 2001.

Eugene Charniak and Joshua Goodman. The state of the art in language modeling. Tutorial presented at AAAI 2002, 2002.

Ciprian Chelba and Frederick Jelinek. Exploiting syntactic structure for language modeling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, 1998.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, 1996.

Stanley F. Chen and Ronald Rosenfeld. A Gaussian prior for smoothing maximum entropy models. Technical report CMUCS-99-108, Carnegie Mellon University, Pittsburgh, 1999.

Colin Cherry and D. Lin. Soft syntactic constraints for word alignment through discriminative training. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 105–112, 2006.

David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 263–270, 2005.

David Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, 2007.

Timothy Chklovski, Rada Mihalcea, Ted Pedersen, and Amruta Purandare. The Senseval-3 multilingual English-Hindi lexical sample task. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 5–8, July 2004.

William Cohen. Learning trees and rules with set-valued features. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI)*, 1996.

William Cohen, Pradeep Ravikumar, and Stephan Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 73–78. Acapulco, Mexico, 2003.

Michael Collins and Nigel Duffy. Convolution kernels for natural language. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 625–632. Cambridge, MA, MIT Press, 2002a.

Michael Collins and Nigel Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages

263–270, 2002b.

Michael Collins, Philipp Koehn, and Ivona Kucerova. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 531–540, 2005.

Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70, 2005.

Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1999.

Hubert Comon, Max Dauchet, Remi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications. Available at `http://www.grappa.univ-lille3.fr/tata`, Release October 1, 2002.

Simon Corston-Oliver, Michael Gamon, and Chris Brockett. A machine learning approach to the automatic evaluation of machine translation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 140–147, 2001.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Rational kernels: Theory and algorithms. *Journal of Machine Learning Research*, 5:1035–1062, 2004.

Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. Cambridge, MA, MIT Press, 2004.

Corinna Cortes, Mehryar Mohri, and Jason Weston. A general regression technique for learning transductions. In *Proceedings of the 22nd International Conference on Machine Learning (ICML-05)*. New York, ACM, 2005.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

Fabrizio Costa, Sauro Menchetti, Alessio Ceroni, Andrea Passerini, and Paolo Frasconi. Decomposition kernels for natural language processing. In *Proceedings of the EACL Workshop on Learning Structured Information in Natural Language Applications*, 2006.

Brooke Cowan, Ivona Kucerova, and Michael Collins. A discriminative model for tree-to-tree translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2006.

Koby Crammer and Yoram Singer. PRanking with ranking. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14*. Cambridge, MA, MIT Press, 2002.

Josep M. Crego, José B. Mariño, and Adrià de Gispert. Reordered search and tuple unfolding for Ngram-based SMT. In *Proceedings of Machine Translation Summit X*, pages 283–89, September 2005.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge, UK, Cambridge University Press, 2000.

Silviu Cucerzan and David Yarowsky. Language independent named entity recognition combining morphological and contextual evidence. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 90–99, 1999.

Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction as classification. In *NIPS Workshop on Advances in Structured Learning for Text and Speech Processing*, 2005.

Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine Learning*, 2006. Submitted.

Daniel Déchelotte, Holger Schwenk, Hélène Bonneau-Maynard, Alexandre Allauzen, and Gilles Adda. A state-of-the-art statistical machine translation system based on Moses. In *Proceedings of Machine Translation Summit XI*, pages 127–133, September 2007.

Hervé Déjean, Éric Gaussier, and Fatia Sadat. An approach based on multilingual thesauri and model combination for bilingual lexicon extraction. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-02)*, 2002.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. Why generative phrase models underperform surface heuristics. In *Proceedings of the NAACL Workshop on Statistical Machine Translation (WMT'06)*, June 2006.

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the 2nd International Conference on Human Language Technology Research (HLT-02)*, pages 138–145, 2002.

Bonnie J. Dorr, Lisa Pearl, Rebecca Hwa, and Nizar Habash. DUSTer: A method for unraveling cross-language divergences for statistical word-level alignment. In Stephen D. Richardson, editor, *Machine Translation: From Research to Real Users, 5th Conference of the Association for Machine Translation in the Americas (AMTA-02)*, volume 2499 of *Lecture Notes in Computer Science*. Berlin, Springer-Verlag, 2002.

Miroslav Dudik, Steven Phillips, and Robert E. Schapire. Performance guarantees for regularized maximum entropy density estimation. In *Proceedings of the Annual ACM Workshop on Computational Learning Theory (COLT)*. Berlin, Springer-Verlag, 2004.

Miroslav Dudík, Steven J. Phillips, and Robert E. Schapire. Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *Journal of Machine Learning Research*, 8:1217–1260, June 2007.

Matthias Eck, Stephan Vogel, and Alex Waibel. Language model adaptation for statistical machine translation based on information retrieval. In *Proceedings of*

the 4th Language Resources and Evaluation Conference (LREC), pages 327–330, May 2004.

Marc El-Bèze and Anne-Marie Derouault. A morphological model for large vocabulary speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 1990.

Christiane Fellbaum, editor. *WordNet. An Electronic Lexical Database*. Cambridge, MA, MIT Press, 1998.

J. G. Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*. Santa Barbara, CA, 1997.

George Foster. A maximum entropy / minimum divergence translation model. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, 2000.

George Foster and Roland Kuhn. Mixture-model adaptation for SMT. In *Proceedings of the 2nd Workshop on Statistical Machine Translation (WMT'07)*, pages 128–135, 2007.

George Foster, Roland Kuhn, and J. Howard Johnson. Phrasetable smoothing for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2006.

Robert Frank, editor. *Phrase Structure Composition and Syntactic Dependencies*. Cambridge, MA, MIT Press, 2002.

Alex Fraser and Daniel Marcu. Semi-supervised training for statistical word alignment. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 769–776, 2006.

Alexander Fraser and Daniel Marcu. Measuring word alignment quality for statistical machine translation. *Computational Linguistics*, 33(3):293–303, 2007.

Andrew Freeman, Sherri Condon, and Christopher Ackerman. Cross-linguistic name matching in English and Arabic: A "one-to-many mapping" extension of the Levenshtein edit distance algorithm. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL06)*, pages 471–478, June 2006.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting alogrithm. In *Proceedings of the 13th International Conference on Machine Learning (ICML-96)*, pages 148–156, 1996.

Yoav Freund and Robert E. Shapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

Nathalie Friburger and Denis Maurel. Textual similarity based on proper names. In *ACM SIGIR 2002 Workshop on Mathematical/Formal Methods in Information Retrieval (MF/IR 2002)*, pages 155–167. Tampere, Finland, 2002.

Atsushi Fujii, Makoto Iwayama, and Noriko Kando. Overview of the patent retrieval task at the NTCIR-6 workshop. In *Proceedings of the 6th NTCIR Workshop*

Meeting, pages 359–365, 2007.

Pascale Fung and Percy Cheung. Mining very non-parallel corpora: Parallel sentence and lexicon extraction via bootstrapping and EM. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 57–63, 2004a.

Pascale Fung and Percy Cheung. Multi-level bootstrapping for extracting parallel sentences from a quasi-comparable corpus. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, pages 1051–1057, August 2004b.

Pascale Fung and Lo Yuen Yee. An IR approach for translating new words from nonparallel, comparable texts. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 414–420, 1998.

GALE. Global autonomous language exploitation (GALE), 2007. Available at `http://www.darpa.mil/ipto/programs/gale/index.htm`.

William A. Gale and Kenneth W. Church. A program for aligning sentences in bilingual corpora. *Computational Linguistics*, 19(1):75–102, 1993.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL04)*, 2004.

Michael Gamon, Anthony Aue, and Martine Smets. Sentence-level MT evaluation without reference translations: Beyond language modeling. In *Proceedings of the European Association for Machine Translation 10th Annual Conference*, pages 103–111, May 2005.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. Fast decoding and optimal decoding for machine translation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 2001.

Jesús Giménez. IQMT v 2.1. technical manual. Technical report LSI-07-29-R, LSI Department, TALP Research Center, Barcelona, Spain, 2007.

Jesús Giménez and Enrique Amigó. IQMT: A framework for automatic machine translation evaluation. In *Proceedings of the 5th Language Resources and Evaluation Conference (LREC)*, pages 685–690, 2006.

Jesús Giménez and Lluís Màrquez. Fast and accurate part-of-speech tagging: The SVM approach revisited. In Nicolas Nicolov, Kalina Bontcheva, Galia Angelova, and Ruslan Mitkov, editors, *Recent Advances in Natural Language Processing III*, Current Issues in Linguistic Theory (CILT), pages 153–162. Amsterdam, John Benjamin, 2004a.

Jesús Giménez and Lluís Màrquez. SVMTool: A general POS tagger generator based on support vector machines. In *Proceedings of the 4th Language Resources and Evaluation Conference (LREC)*, pages 43–46, May 2004b.

Jesús Giménez and Lluís Màrquez. Combining linguistic data views for phrase-based SMT. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, 2005.

Jesús Giménez and Lluís Màrquez. The LDV-COMBO system for SMT. In *Proceedings of the NAACL Workshop on Statistical Machine Translation (WMT'06)*, June 2006.

Jesús Giménez and Lluís Màrquez. Context-aware discriminative phrase selection for statistical machine translation. In *Proceedings of the 2nd Workshop on Statistical Machine Translation (WMT'07)*, pages 159–166, 2007a.

Jesús Giménez and Lluís Màrquez. Linguistic features for automatic evaluation of heterogeneous MT systems. In *Proceedings of the 2nd Workshop on Statistical Machine Translation (WMT'07)*, pages 256–264, 2007b.

Jesús Giménez and Lluís Màrquez. Heterogeneous automatic MT evaluation through non-parametric metric combinations. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP)*, 2008.

Claudio Giuliano, Alfio Gliozzo, and Carlo Strapparava. Syntagmatic kernels: A word sense disambiguation case study. In *Proceedings of the EACL Workshop on Learning Structured Information in Natural Language Applications*, 2006.

Vincent Goffin, Cyril Allauzen, Enrico Bocchieri, Dilek Hakkani-Tur, Andrej Ljolje, Sarangarajan Parthasarathy, Mazim Rahim, Giuseppe Riccardi, and Murat Saraclar. The AT&T WATSON speech recognizer. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 2005.

Sharon Goldwater, Tom Griffiths, and Mark Johnson. Interpolating between types and tokens by estimating power-law generators. In Yair Weiss, Bernhard Schölkopf, and John Platt, editors, *Advances in Neural Information Processing Systems 18*. Cambridge, MA, MIT Press, 2006.

Sharon Goldwater and D. McClosky. Improving statistical MT through morphological analysis. In *Proceedings of the Joint Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 676–683, October 2005.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*, 3rd edition. Baltimore, Johns Hopkins University Press, 1996.

Joshua T. Goodman. A bit of progress in language modeling: Extended version. Technical report MSR-TR-2001-72, Microsoft Research, Redmond, WA, 2001.

Jonathan Graehl and Kevin Knight. Training tree transducers. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL04)*, pages 105–112, 2004.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

Nizar Habash. Arabic morphological representations for machine translation. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-Based and Empirical Methods*. Berlin, Springer-Verlag, 2007.

Nizar Habash and Owen Rambow. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 573–580, 2005.

Nizar Habash and Fatiha Sadat. Arabic preprocessing schemes for statistical machine translation. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL06)*, pages 49–52, June 2006.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. On Arabic transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-Based and Empirical Methods*. Berlin, Springer-Verlag, 2007.

Gholamreza Haffari and Anoop Sarkar. Analysis of semi-supervised learning with the Yarowsky algorithm. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI2007)*. Arlington, VA, AUAI Press, 2007.

Patrick Haffner. Scaling large margin classifiers for spoken language understanding. *Speech Communication*, 2005.

Patrick Haffner. Fast transpose methods for kernel learning on sparse data. In *Proceedings of the 23rd International Conference on Machine Learning (ICML-06)*, pages 385–392. New York, ACM, 2006.

Patrick Haffner, Steven J. Phillips, and Robert E. Schapire. Efficient multiclass implementations of L1-regularized maximum entropy. Technical report abs/cs/0506101, arXiv eprint server, 2005.

Dilek Hakkani-Tür, Heng Ji, and Ralph Grishman. Using information extraction to improve cross-lingual document retrieval. In *Proceedings of the RANLP 2007 Workshop on Multi-Source, Multilingual Information Extraction and Summarization*, 2007.

Edward F. Harrington. Online ranking/collaborative filtering using the perceptron algorithm. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 250–257. New York, ACM, 2003.

Saša Hasan, Oliver Bender, and Hermann Ney. Reranking translation hypotheses using structural properties. In *Proceedings of the EACL Workshop on Learning Structured Information in Natural Language Applications*, 2006.

Ahmed Hassan, Fahmy Haytham, and Hany Hassan. Improving named entity translation by exploiting comparable and parallel corpora. In *Proceedings of the ACL 2007 Workshop on Acquisition and Management of Multilingual Lexicons (AMML)*, 2007.

Hany Hassan, Mary Hearne, Andy Way, and Khalil Sima'an. Syntactic phrase-based statistical machine translation. In *Proceedings of the IEEE 2006 Workshop on*

Spoken Language Technology, December 2006.

Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

David Haussler. Convolution kernels on discrete structures. Technical report UCSC-CRL-99-10, UC Santa Cruz, CA, 1999.

Xiaodong He. Using word-dependent transition models in HMM-based word alignment for statistical machine translation. In *Proceedings of the 2nd Workshop on Statistical Machine Translation (WMT'07)*, pages 80–87, 2007.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. In Peter J. Bartlett, Bernhard Schölkopf, Dale Schuurmans, and Alex J. Smola, editors, *Advances in Large Margin Classifiers*, pages 115–132. Cambridge, MA, MIT Press, 2000.

Magnus Lie Hetland. A survey of recent methods for efficient retrieval of similar time sequences. In *Data Mining in Time Series Databases*. Hackensack, NJ, World Scientific, 2004.

Shigeto Higuchi, Masatoshi Fukui, Atsushi Fujii, and Tetsuya Ishikawa. PRIME: A system for multi-lingual patent retrieval. In *Proceedings of Machine Translation Summit VIII*, pages 163–167, 2001.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. Adaptation of the translation model for statistical machine translation. In *Proceedings of the European Association for Machine Translation 10th Annual Conference*, May 2005.

Lynette Hirschman, Florence Reeder, John Burger, and Keith Miller. Name translation as a machine translation evaluation task. In *Proceedings of the 2nd Language Resources and Evaluation Conference (LREC)*, pages 21–28, 2000.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(3):55–67, 1970.

Fei Huang and Kishore Papineni. Hierarchical system combination for machine translation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning (EMNLP-CoNLL)*, pages 277–286, June 2007.

Liang Huang and David Chiang. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, 2007.

Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Upper Saddle River, NJ, Prentice Hall, 2001.

John Hutchins. ALPAC: The (in)famous report. In Sergei Nirenburg, Harold Somers, and Yorick Wilks, editors, *Readings in Machine Translation*, pages 131–135. Cambridge, MA, MIT Press, 2003.

Rob Hyland, Chris Clifton, and Rod Holland. GeoNODE: Visualizing news in geospatial context. In *Federal Data Mining Symposium and Exposition '99, AFCEA*, March 1999.

Camelia Ignat, Bruno Pouliquen, Ralf Steinberger, and Tomaž Erjavec. A tool set for the quick and efficient exploration of large document collections. In *Proceedings of the Symposium on Safeguards and Nuclear Material Management. 27th Annual Meeting of the European Safeguards Research and Development Association (ESARDA-2005)*. London, May 2005.

A. Ittycheriah and Salim Roukos. A maximum entropy word aligner for Arabic-English machine translation. In *Proceedings of the Joint Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 89–96, October 2005.

S. Jayaraman and Alon Lavie. Multi-engine machine translation guided by explicit word matching. In *Proceedings of the European Association for Machine Translation 10th Annual Conference*, pages 143–152, May 2005.

Edwin Thompson Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, May 1957.

Frederick Jelinek. *Statistical Methods for Speech Recognition*. Cambridge, MA, MIT Press, 1998.

Heng Ji, Cynthia Rudin, and Ralph Grishman. Re-ranking algorithms for name tagging. In *Proceedings of the HLT/NAACL 06 Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, 2006.

Peng Jin, Yunfang Wu, and Shiwen Yu. Semeval-2007 task 05: Multilingual chinese-english lexical sample. In Eneko Agirre, Lluís Màrquez, and Richard Wicentowski, editors, *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 19–23. Association for Computational Linguistics, June 2007.

Thorsten Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher J.C. Burges, and Alex J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. Cambridge, MA, MIT Press, 1999.

Thorsten Joachims. A support vector method for multivariate performance measures. In *International Conference on Machine Learning (ICML)*, pages 377–384, 2005.

Thorsten Joachims. Training linear SVMs in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. New York, ACM, 2006.

Thorsten Joachims, Hang Li, Tie-Yan Liu, and ChengXiang Zhai, editors. *Proceedings of the SIGIR-07 Workshop on Learning to Rank for Information Retrieval*, 2007.

Douglas Jones, Wade Shen, and Brian Delaney. Toward an interagency language roundtable-based assessment of speech-to-speech translation abilities. In *Pro-*

ceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA-06), 2006.

Sung Young Jung, SungLim Hong, and Eunok Paek. An English to Korean transliteration model of extended Markov window. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00)*, pages 383–389, 2000.

Béla Kálmán. *The World of Names: A study in Hungarian onomatology.* Budapest, Akadémiai Kiadó, 1978.

Hans Kamp. A theory of truth and semantic representation. In J.A.G. Groenendijk, T.M.V. Janssen, , and M.B.J. Stokhof, editors, *Formal Methods in the Study of Language*, pages 277–322. Amsterdam, Mathematisch Centrum, 1981.

Stephan Kanthak, David Vilar, Evgeny Matusov, Richard Zens, and Hermann Ney. Novel reordering approaches in phrase-based statistical machine translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 167–174, 2005.

Alexandre Klementiev and Dan Roth. Named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL06)*, June 2006a.

Alexandre Klementiev and Dan Roth. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, 2006b.

Kevin Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615, 1999.

Kevin Knight and Jonathan Graehl. Machine transliteration. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, 1997.

Philipp Koehn. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas (AMTA-04)*, 2004a.

Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395, 2004b.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X*, September 2005.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT-05)*, October 2005.

Philipp Koehn and Hieu Hoang. Factored translation models. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and*

Conference on Natural Language Learning (EMNLP-CoNLL), pages 868–876, June 2007.

Philipp Koehn and Kevin Knight. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL Workshop on Unsupervised Lexical Acquisition*, 2002.

Philipp Koehn and Christof Monz. Manual and automatic evaluation of machine translation between European languages. In *Proceedings of the NAACL Workshop on Statistical Machine Translation (WMT'06)*, pages 102–121, June 2006.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL03)*, 2003.

Stasinos Konstantopoulos. What's in a name? Quite a lot. In *Proceedings of the RANLP Workshop on Computational Phonology*. Borovets, Bulgaria, September 2007.

Balaji Krishnapuram, Lawrence Carin, Mario A.T. Figueiredo, and Alexander J. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):957–968, 2005.

R. Kuhn and R. de Mori. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583, 1990.

Alex Kulesza and Stuart M. Shieber. A learning approach to improving sentence-level MT evaluation. In *Proceedings of the 10th Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, 2004.

Shankar Kumar, Yonggang Deng, and William Byrne. A weighted finite state transducer translation template model for statistical machine translation. *Natural Language Engineering*, 12(1):35–75, 2006.

John Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*. San Francisco, Morgan Kaufmann, 2001.

Patrick Lambert and R.E. Banchs. Tuning machine translation parameters with SPSA. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT-06)*, November 2006.

Patrik Lambert, Jesús Giménez, Marta R. Costa-jussà, Enrique Amigó, Rafael E. Banchs, Lluís Màrquez, and J.A. R. Fonollosa. Machine translation system development based on human likeness. In *Proceedings of the IEEE 2006 Workshop on Spoken Language Technology*, 2006.

Gert R.G. Lanckriet, Nello Cristianini, Michael I. Jordan, and W.S. Noble. Kernel-based integration of genomic data using semidefinite programming. In B. Schölkopf, K. Tsuda, and J.P. Vert, editors, *Kernel Methods in Computational Biology*, pages 231–259. Cambridge, MA, MIT Press, 2004.

Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284, 1998.

Leah Larkey, Fangfang Feng, Margaret Connell, and Victor Lavrenko. Language-specific models in multilingual topic tracking. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR-04)*, pages 402–409. New York, ACM, 2004.

Yoong Keok Lee and Hwee Tou Ng. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 41–48, 2002.

Young-Suk Lee. Morphological analysis for statistical machine translation. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL04)*, pages 57–60, 2004.

Christina Leslie and Rui Kuang. Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 5:1435–1455, 2004.

Gregor Leusch, Nicola Ueffing, David Vilar, and Hermann Ney. Preprocessing and normalization for automatic evaluation of machine translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005.

Xin Li, Paul Morie, and Dan Roth. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 419–424, 2004.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. An end-to-end discriminative approach to machine translation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768, 2006.

Percy Liang, Ben Taskar, and Dan Klein. Alignment by agreement. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, 2007.

Chin-Yew Lin and Franz Josef Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 2004a.

Chin-Yew Lin and Franz Josef Och. ORANGE: A method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, page 501, August

2004b.

Lucian Vlad Lita, Monica Rogati, and Alon Lavie. BLANC: Learning evaluation metrics for MT. In *Proceedings of the Joint Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, October 2005.

Ding Liu and Daniel Gildea. Syntactic features for evaluation of machine translation. In *Proceedings of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, 2005.

Yang Liu, Qun Liu, and Shouxun Lin. Log-linear models for word alignment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 459–466, 2005.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

Adam Lopez and Philip Resnik. Word-based alignment, phrase-based translation: What's the link? In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA-06)*, pages 90–99, 2006.

Xiaoyi Ma and Christopher Cieri. Corpus support for machine translation at LDC. In *Proceedings of the 5th Language Resources and Evaluation Conference (LREC)*, pages 859–864, 2006.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. The Penn Arabic treebank: Building a large-scale annotated Arabic corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109, 2004.

Wolfgang Macherey and Franz Josef Och. An empirical study on computing consensus translations from multiple machine translation systems. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning (EMNLP-CoNLL)*, pages 986–995, June 2007.

Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Conference on Natural Language Learning (CoNLL-02)*, pages 49–55, 2002.

G. Maltese and F. Mancini. An automatic technique to include grammatical and morphological information in a trigram-based statistical language model. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 1992.

Lidia Mangu, Eric Brill, and Andreas Stolcke. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Computer Speech and Language*, pages 373–400, 2000.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. SPMT: Statistical machine translation with syntactified target language phrases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*

(EMNLP), pages 44–52, 2006.

Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Jose B. Mariño, Rafael E. Banchs, Josep M. Crego, Adrià de Gispert, Patrick Lambert, José A.R. Fonollosa, and Marta R. Costa-jussà. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549, 2006.

Lluís Màrquez, Gerard Escudero, David Martínez, and German Rigau. Supervised corpus-based methods for WSD. In Phil Edmonds and Eneko Agirre, editors, *Word Sense Disambiguation: Algorithms, Applications, and Trends*, chapter 7. Norwell, MA, Kluwer, 2006.

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus R. Frean. Functional gradient techniques for combining hypotheses. In Alexander J. Smola, Peter Bartlett, Bernhard Schökopf, and Dale Schuurman, editors, *Advances in Large Margin Classifiers*, chapter 12. Cambridge, MA, MIT Press, 1999.

Llew Mason, Jonathan Baxter, Peter L. Bartlett, and Marcus R. Frean. Boosting algorithms as gradient descent. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 512–518. Cambridge, MA, MIT Press, 2000.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proceedings of the 11th European Chapter of the Association for Computational Linguistics (EACL)*, pages 33–40, April 2006.

Evgeny Matusov, Richard Zens, and Hermann Ney. Symmetric word alignments for statistical machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, pages 219–225, August 2004.

Arne Mauser, David Vilar, Gregor Leusch, Yuqi Zhang, and Hermann Ney. The RWTH machine translation system for IWSLT 2007. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT-07)*, October 2007.

Arne Mauser, Richard Zens, Evgeny Matusov, Saša Hasan, and Hermann Ney. The RWTH statistical machine translation system for the IWSLT 2006 evaluation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT-06)*, pages 103–110, November 2006.

Dennis Mehay and Chris Brew. BLEUATRE: Flattening syntactic dependencies for MT evaluation. In *Proceedings of the 11th Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, 2007.

I. Dan Melamed. Statistical machine translation by parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 2004.

I. Dan Melamed, Ryan Green, and Joseph P. Turian. Precision and recall of machine translation. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL03)*, 2003.

MLNER. *ACL Workshop on Multilingual and Mixed-language Named Entity Recognition: Combining Statistical and Symbolic Models*, 2003.

Daichi Mochihashi and Yuji Matsumoto. Context as filtering. In Yair Weiss, Bernhard Schölkopf, and John Platt, editors, *Advances in Neural Information Processing Systems 18*. Cambridge, MA, MIT Press, 2006.

Robert C. Moore. Fast and accurate sentence alignment of bilingual corpora. In Stephen D. Richardson, editor, *Machine Translation: From Research to Real Users, 5th Conference of the Association for Machine Translation in the Americas (AMTA-02)*, volume 2499 of *Lecture Notes in Computer Science*, pages 135–144. Berlin, Springer-Verlag, 2002.

Robert C. Moore. A discriminative framework for bilingual word alignment. In *Proceedings of the Joint Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 81–88, October 2005.

Robert C. Moore and Chris Quirk. Faster beam-search decoding for phrasal statistical machine translation. In *Proceedings of Machine Translation Summit XI*, pages 321–327, September 2007.

MUC-6. *Message Understanding Conference (MUC-6)*, 1995. San Francisco, Morgan Kauffman.

Andrea Mulloni. Automatic prediction of cognate orthography using support vector machines. In *Proceedings of the Student Research Workshop, ACL 2007*, pages 25–30. Prague, 2007.

Dragos Stefan Munteanu, Alexander Fraser, and Daniel Marcu. Improved machine translation performance via parallel sentence extraction from comparable corpora. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL04)*, pages 265–272, 2004.

Dragos Stefan Munteanu and Daniel Marcu. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4): 477–504, 2005.

Dragos Stefan Munteanu and Daniel Marcu. Extracting parallel sub-sentential fragments from non-parallel corpora. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 81–88, 2006.

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticæ Investigationes*, 30(1), 2007. Special issue on Named

Entities: Recognition, Classification and Use.

Makoto Nagao. A framework of a mechanical translation between Japanese and English by analogy principle. In *International NATO Symposium on Artificial and Human Intelligence*, 1981. Reprinted in Sergei Nirenburg, Harold Somers, and Yorick Wilks, editors. *Readings in Machine Translation*, Cambridge, MA, MIT Press, 2003.

Masaaki Nagata, Kuniko Saito, Kazuhide Yamamoto, and Kazuteru Ohashi. A clustered global phrase reordering model for statistical machine translation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 713–720, 2006.

Borja Navarro, Montserrat Civit, M. Antonia Martí, Raquel Marcos, and Belén Fernández. Syntactic, semantic and pragmatic annotation in cast3lb. In *Proceedings of SProLaC*, pages 59–68, 2003.

John A. Nelder and Roger Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

Sonja Nießen and Hermann Ney. Statistical machine translation with scarce resources using morpho-syntactic information. *Computational Linguistics*, 30(2), 2004.

Sonja Nießen, Franz Josef Och, Gregor Leusch, and Hermann Ney. An evaluation tool for machine translation: Fast evaluation for MT research. In *Proceedings of the 2nd Language Resources and Evaluation Conference (LREC)*, 2000.

Sergei Nirenburg and Robert Frederking. Toward multi-engine machine translation. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 147–151. Plainsboro, NJ, March 1994.

Tadashi Nomoto. Multi-engine machine translation with voted language model. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 494–501, 2004.

Franz Josef Och. An efficient method for determining bilingual word classes. In *Proceedings of the 9th European Chapter of the Association for Computational Linguistics (EACL)*, pages 71–76, 1999.

Franz Josef Och. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. PhD thesis, RWTH Aachen University, Aachen, Germany, 2002.

Franz Josef Och. Minimum error rate training for statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, 2003.

Franz Josef Och, Dan Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. A smorgasbord of features for statistical machine translation. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational*

Linguistics (HLT-NAACL04), 2004.

Franz Josef Och and Hermann Ney. A comparison of alignment models for statistical machine translation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00)*, 2000a.

Franz Josef Och and Hermann Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, 2000b.

Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, 2002.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, March 2003.

Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004.

Karolina Owczarzak, Josef van Genabith, and Andy Way. Dependency-based automatic evaluation for machine translation. In *Proceedings of the NAACL-HLT/AMTA Workshop on Syntax and Structure in Statistical Translation (SSST)*, pages 80–87, 2007.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. Research report RC22176, IBM, 2001.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

M. Paul, T. Doi, Y. Hwang, K. Imamura, H. Okuma, and E. Sumita. Nobody is perfect: ATR's hybrid approach to spoken language translation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT-05)*, pages 55–62, October 2005.

Michael Paul. Overview of the IWSLT 2006 evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT-06)*, pages 1–15, November 2006.

Vern Paxson. Flex – fast lexical analyzer generator. Lawrence Berkeley Laboratory, Berkeley, CA, 1995. Available at `http://flex.sourceforge.net/`.

Simon Perkins, Kevin Lacker, and James Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356, 2003.

John C. Platt. Probabilities for SV machines. In Alexander J. Smola, Peter L. Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. Cambridge, CA, MIT Press, 2000.

Thierry Poibeau. The multilingual named entity recognition framework. In *Proceedings of the 10th European Chapter of the Association for Computational Linguistics (EACL)*, pages 155–158, 2003.

Maja Popović and Hermann Ney. Towards the use of word stems and suffixes for statistical machine translation. In *Proceedings of the 4th Language Resources and Evaluation Conference (LREC)*, pages 1585–1588, May 2004.

Bruno Pouliquen, Ralf Steinberger, and Jenya Belyaeva. Multilingual multi-document continuously updated social networks. In *Proceedings of the RANLP 2007 Workshop on Multi-Source, Multilingual Information Extraction and Summarization*. Borovets, Bulgaria, September 2007a.

Bruno Pouliquen, Ralf Steinberger, and Clive Best. Automatic detection of quotations in multilingual news. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2007)*. Borovets, Bulgaria, September 2007b.

Bruno Pouliquen, Ralf Steinberger, Camelia Ignat, Irina Temnikova, Anna Widiger, Wajdi Zaghouani, and Jan Žižka. Multilingual person name recognition and transliteration. *CORELA – Cognition, Représentation, Langage*, 2005. Numéros spéciaux sur le traitement lexicographique des noms propres.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C++*. Cambridge, UK, Cambridge University Press, 2002.

Adam Przepiórkowski. Slavic information extraction and partial parsing. In *Proceedings of the ACL Workshop on Balto-Slavonic Natural Language Processing*. Prague, June 2007.

Chris Quirk and Arul Menezes. Do we need phrases? Challenging the conventional wisdom in statistical machine translation. In *Proceedings of the 11th European Chapter of the Association for Computational Linguistics (EACL)*, April 2006.

Chris Quirk, Arul Menezes, and Colin Cherry. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 2005.

Chris Quirk, Raghavendra Udopa U., and Arul Menezes. Generative models of noisy translations with applications to parallel fragment extraction. In *Proceedings of Machine Translation Summit XI*, pages 377–384, September 2007.

Reinhard Rapp. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 320–322, 1995.

Reinhard Rapp. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 519–526, 1999.

Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In E. Brill and K. Church, editors, *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 133–142, 1996.

Adwait Ratnaparkhi. A linear observed time statistical parser based on maximal entropy models. In Claire Cardie and Ralph Weischedel, editors, *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–10, 1997.

Philip Resnik and Noah A. Smith. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380, 2003.

Stefan Riezler and John T. Maxwell III. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT*, 2005.

Stefan Riezler and Alexander Vasserman. Incremental feature selection and ℓ_1 regularization for relaxed maximum-entropy modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2004.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 47–54, 2004.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

Ronald Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8), 2000.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie J. Dorr. Combining outputs from multiple machine translation systems. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL07)*, April 2007a.

Antti-Veikko Rosti, Spyros Matsoukas, and Richard Schwartz. Improved word-level system combination for machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 312–319, 2007b.

Dan Roth. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 806–813, 1998.

Dan Roth. Learning in natural language. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 898–904, 1999.

Fatiha Sadat and Nizar Habash. Combination of Arabic preprocessing schemes for statistical machine translation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 1–8, 2006.

Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. New York, McGraw-Hill, 1986.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge, UK, Cambridge University Press, 2004.

Libin Shen and Aravind K. Joshi. An SVM-based voting algorithm with application to parse reranking. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL03)*, pages 9–16, 2003.

Libin Shen and Aravind K. Joshi. Ranking and reranking with perceptron. *Machine Learning*, 60(1-3):73–96, 2005.

Libin Shen, Anoop Sarkar, and Franz Josef Och. Discriminative reranking for machine translation. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL04)*, 2004.

Tarek Sherif and Grzegorz Kondrak. Substring-based transliteration. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 944–951, 2007.

Yusuke Shinyama and Satoshi Sekine. Named entity discovery using comparable news articles. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, pages 848–853, August 2004.

Khe Chai Sim, William Byrne, Mark J.F. Gales, Hichem Sahbi, and Phil Woodland. Consensus network decoding for statistical machine translation system combination. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages IV–105 – IV–108. Honolulu, April 2007.

Michel Simard, George F. Foster, and Pierre Isabelle. Using cognates to align sentences in bilingual corpora. In *Proceedings of the 4th Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, 1992.

Alex J. Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML-00)*, pages 911–918, June 2000.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA-06)*, pages 223–231, 2006.

Lucia Specia, Mark Stevenson, and Maria das Graças Volpe Nunes. Learning expressive models for word sense disambiguation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 41–48, 2007.

Richard Sproat, Tao Tao, and ChengXiang Zhai. Named entity transliteration with comparable corpora. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 73–80, 2006.

Ralf Steinberger and Bruno Pouliquen. Cross-lingual named entity recognition. *Lingvisticæ Investigationes*, 30(1):135–162, 2007. Special issue on Named Enti-

ties: Recognition, Classification and Use.

Ralf Steinberger, Bruno Pouliquen, and Camelia Ignat. Providing cross-lingual information access with knowledge-poor methods. *Informatica*, 28(4):415–423, 2004. Special issue on Information Society in 2004.

Ralf Steinberger, Bruno Pouliquen, and Camelia Ignat. Navigating multilingual news collections using automatically extracted information. *Journal of Computing and Information Technology – CIT*, 13(4):257–264, 2005.

Ralf Steinberger, Bruno Pouliquen, and Camelia Ignat. Using language-independent rules to achieve high multilinguality in text mining. In Françoise Soulié-Fogelman, Ralf Steinberger, Jakub Piskorski, Domenico Perrotta, and Clive Best, editors, *Mining Massive Data Sets for Security*. Amsterdam, IOS Press, Forthcoming.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaž Erjavec, Dan Tufiş, and Dániel Varga. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of the 5th Language Resources and Evaluation Conference (LREC)*, pages 24–26, 2006.

Ralph E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Application*. New York, Wiley, 1986.

Andreas Stolcke. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Speech and Language Processing (ICSLP)*, pages 901–904, 2002.

Nicolas Stroppa, Antal van den Bosch, and Andy Way. Exploiting source similarity for SMT using context-informed features. In *Proceedings of the 11th Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, pages 231–240, 2007.

Hristo Tanev. Unsupervised learning of social networks from a multiple-source news corpus. In *Proceedings of the RANLP 2007 Workshop on Multi-Source, Multilingual Information Extraction and Summarization*. Borovets, Bulgaria, 2007.

Ben Taskar. *Learning Structured Prediction Models: A Large Margin Approach*. PhD thesis, Stanford University, Stanford, CA, 2004.

Ben Taskar, Simon Lacoste-Julien, and Michael Jordan. Structured prediction via the extragradient method. In Yair Weiss, Bernhard Schölkopf, and John Platt, editors, *Advances in Neural Information Processing Systems 18*. Cambridge, MA, MIT Press, 2006.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. A discriminative matching approach to word alignment. In *Proceedings of the Joint Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 73–80, October 2005.

TC-STAR. European research project TC-STAR – technology and corpora for speech-to-speech translation, 2007.

Yee Whye Teh. A Bayesian interpretation of interpolated Kneser-Ney. Technical report TRA2/06, School of Computing, National University of Singapore, 2006.

Ambuj Tewari and Peter L. Bartlett. On the consistency of multiclass classification methods. In *Proceedings of the 18th Annual ACM Workshop on Computational Learning Theory (COLT)*, 2005.

Christoph Tillmann, Stefan Vogel, Hermann Ney, and A. Zubiaga. A DP-based search using monotone alignments in statistical translation. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 289–296, 1997a.

Christoph Tillmann, Stefan Vogel, Hermann Ney, A. Zubiaga, and H. Sawaf. Accelerated DP-based search for statistical translation. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 2667–2670. Rhodes, Greece, September 1997b.

Christoph Tillmann and Tong Zhang. A localized prediction model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 2005.

Christoph Tillmann and Tong Zhang. A discriminative global training algorithm for statistical MT. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 721–728, 2006.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Conference on Natural Language Learning (CoNLL-03)*, pages 142–147, 2003.

Kristina Toutanova and Hisami Suzuki. Generating case markers in machine translation. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL07)*, April 2007.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

Joseph Turian. *Constituent Parsing by Classification.* PhD thesis, New York University, New York, September 2007.

Joseph Turian and I. Dan Melamed. Advances in discriminative parsing. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, 2006.

Joseph Turian, Luke Shen, and I. Dan Melamed. Evaluation of machine translation and its evaluation. In *Proceedings of Machine Translation Summit IX*, 2003.

Nicola Ueffing, Gholamreza Haffari, and Anoop Sarkar. Transductive learning for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, 2007a.

Nicola Ueffing, Gholamreza Haffari, and Anoop Sarkar. Semi-supervised model adaptation for statistical machine translation. *Machine Translation*, 2008. To

appear.

Nicola Ueffing and Hermann Ney. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40, 2007.

Nicola Ueffing, Franz Josef Och, and Hermann Ney. Generation of word graphs in statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 156–163, 2002.

Nicola Ueffing, Michel Simard, Samuel Larkin, and J. Howard Johnson. NRC's PORTAGE system for WMT 2007. In *Proceedings of the 2nd Workshop on Statistical Machine Translation (WMT'07)*, 2007b.

Masao Utiyama and Hitoshi Isahara. Reliable measures for aligning Japanese-English news articles and sentences. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 72–79, 2003.

Takehito Utsuro, Hiroshi Ikeda, Masaya Yamane, Yuji Matsumoto, and Makoto Nagao. Bilingual text matching using bilingual dictionary and statistics. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, pages 1076–1082, 1994.

Frank Vanden Berghen and Hugues Bersini. CONDOR, a new parallel, constrained extension of Powell's UOBYQA algorithm: Experimental results and comparison with the DFO algorithm. *Journal of Computational and Applied Mathematics*, 181:157–175, 2005.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* New York, Springer-Verlag, 1995.

Vladimir N. Vapnik. *Statistical Learning Theory.* New York, Wiley, 1998.

Sriram Venkatapathy and Srinivas Bangalore. Three models for discriminative machine translation using global lexical selection and sentence reconstruction. In *Proceedings of the NAACL-HLT/AMTA Workshop on Syntax and Structure in Statistical Translation (SSST)*, pages 152–159, 2007.

David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. Word-sense disambiguation for machine translation. In *Proceedings of the Joint Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, October 2005.

David Vilar, Jia Xu, Luis Fernando D'Haro, and Hermann Ney. Error analysis of machine translation output. In *Proceedings of the 5th Language Resources and Evaluation Conference (LREC)*, pages 697–702, 2006.

Pascal Vincent and Yoshua Bengio. Kernel matching pursuit. *Machine Learning*, 48(1-3):169–191, 2002.

S.V.N. Vishwanathan and Alex Smola. Fast kernels for string and tree matching. In B. Schölkopf, K. Tsuda, and J.P. Vert, editors, *Kernel Methods in Computational Biology*, pages 113–130. Cambridge, CA, MIT Press, 2004.

Stefan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based word alignment in statistical translation. In *Proceedings of the 16th International Conference on*

Computational Linguistics (COLING-96), pages 836–841, August 1996.

Hannah Wallach. Topic modeling: Beyond bag-of-words. In *Proceedings of the 23rd International Conference on Machine Learning (ICML-06)*. New York, ACM, 2006.

Shaojun Wang. Exploiting syntactic, semantic and lexical regularities in language modeling via directed Markov random fields. In *Proceedings of the 22nd International Conference on Machine Learning (ICML-05)*. New York, ACM, 2005.

Zhuoran Wang, John Shawe-Taylor, and Sandor Szedmak. Kernel regression–based machine translation. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL07)*, pages 185–188, April 2007.

Ben Wellington, Joseph Turian, Chris Pike, and I. Dan Melamed. Scalable purely-discriminative training for word and tree transducers. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA-06)*, 2006.

Jason Weston, Bernhard Schölkopf, Eleazar Eskin, Christina Leslie, and William Stafford Noble. Dealing with large diagonals in kernel matrices. *Annals of the Institute of Statistical Mathematics*, 55(2):391–408, June 2003.

Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, 1997.

Fei Xia and Michael McCord. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, August 2004.

Deyi Xiong, Qun Liu, and Shouxun Lin. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 521–528, 2006.

Jia Xu, Yonggang Deng, Yuqing Gao, and Hermann Ney. Domain dependent statistical machine translation. In *Proceedings of Machine Translation Summit XI*, September 2007.

Yuehua Xu and Alan Fern. On learning linear ranking functions for beam search. In *Proceedings of the 24th International Conference on Machine Learning (ICML-07)*, pages 1047–1054. New York, ACM, 2007.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 531–538, 2001.

Kenji Yamada, Kevin Knight, and Greg Langmead. *Adapter for Allowing Both Online and Offline Training of a Text-to-Text System*. U.S. Patent application 20070094169, 2007.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, 1995.

David Yarowsky, Grace Ngai, and Richard Wicentowski. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the 1st International Conference on Human Language Technology Research (HLT-01)*, pages 161–168, 2001.

Richard Zens and Hermann Ney. Improvements in phrase-based statistical machine translation. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL04)*, pages 257–264, 2004.

Richard Zens and Hermann Ney. N-gram posterior probabilities for statistical machine translation. In *Proceedings of the NAACL Workshop on Statistical Machine Translation (WMT'06)*, June 2006.

Hao Zhang and Daniel Gildea. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 2005.

Bing Zhao, Matthias Eck, and Stephan Vogel. Language model adaptation for statistical machine translation with structured query models. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, August 2004.

Bing Zhao and Stephan Vogel. Adaptive parallel sentences mining from web bilingual news collection. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 745–748, 2002.

Justin Zobel and Philip Dart. Finding approximate matches in large lexicons. *Software – Practice and Experience*, 25:331–345, 1995.

Justin Zobel and Philip Dart. Phonetic string matching. In *Lessons from Information Retrieval. Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR-96)*. New York, ACM, 1996.

Contributors

Srinivas Bangalore
AT&T Labs-Research
Florham Park, NJ 07932, USA
srini@research.att.com

Nicola Cancedda
Xerox Research Centre Europe
38240 Meylan, France
Nicola.Cancedda@xrce.xerox.com

Josep Maria Crego
Universitat Politècnica de Catalunya
08034 - Barcelona
jmcrego@gps.tsc.upc.edu

Marc Dymetman
Xerox Research Centre Europe
38240 Meylan, France
marc.dymetman@xrce.xerox.com

Jakob Elming
Copenhagen Business School
2000 Frederiksberg, Denmark
jel.isv@cbs.dk

George Foster
Institute for Information Technology
National Research Council Canada
Gatineau, QC, J8X 3X7, Canada
George.Foster@nrc-cnrc.gc.ca

Jesús Giménez
Technical University of Catalonia

Barcelona 08034, Spain
jgimenez@lsi.upc.edu

Cyril Goutte
Institute for Information Technology
National Research Council Canada
Gatineau, QC, J8X 3X7, Canada
Cyril.Goutte@nrc-cnrc.gc.ca

Nizar Habash
Center for Computational Learning Systems
Columbia University
New York, NY 10115, USA
habash@ccls.columbia.edu

Gholamreza Haffari
Simon Fraser University
Burnaby, BC, Canada
ghaffar1@cs.sfu.ca

Patrick Haffner
AT&T Labs-Research
Florham Park, NJ 07932, USA
haffner@research.att.com

Hitoshi Isahara
National Institute of Information and Communications Technology
Keihanna Science City, Kyoto, Japan
isahara@nict.go.jp

Stephan Kanthak
NUANCE Automotive R&D
52072 Aachen, Germany
stylon@gmx.de

Alexandre Klementiev
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
klementi@uiuc.edu

Gregor Leusch
RWTH Aachen University

52056 Aachen, Germany
leusch@informatik.rwth-aachen.de

Pierre Mahé
Xerox Research Centre Europe
38240 Meylan, France
piedros@gmail.com

Lluís Màrquez
TALP Research Center
Technical University of Catalonia (UPC)
Barcelona 08034, Spain
lluism@lsi.upc.edu

Evgeny Matusov
RWTH Aachen University
52056 Aachen, Germany
matusov@informatik.rwth-aachen.de

I. Dan Melamed
New York University
New York, NY 10003, USA
{lastname}@cs.nyu.edu

Ion Muslea
Language Weaver
Marina del Rey, CA 90292, USA
imuslea@languageweaver.com

Hermann Ney
RWTH Aachen University
52056 Aachen, Germany
ney@informatik.rwth-aachen.de

Bruno Pouliquen
European Commission - Joint Research Centre
21027 Ispra (VA), Italy
Bruno.Pouliquen@jrc.it

Dan Roth
Department of Computer Science
University of Illinois at Urbana-Champaign

Urbana, IL 61801, USA
danr@uiuc.edu

Anoop Sarkar
School of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6, Canada
anoop@cs.sfu.ca

John Shawe-Taylor
Department of Computer Science
University College London
London, WC1E 6BT, United Kingdom
J.Shawe-Taylor@cs.ucl.ac.uk

Ralf Steinberger
European Commission - Joint Research Centre
21027 Ispra (VA), Italy
Ralf.Steinberger@jrc.it

Joseph Turian
Département d'Informatique et Recherche Opérationnelle (DIRO)
Université de Montréal
Montréal Québec, Canada, H3T 1J4
{lastname}@iro.umontreal.ca

Nicola Ueffing
nicola.ueffing@gmail.com

Masao Utiyama
National Institute of Information and Communications Technology
Keihanna Science City, Kyoto, Japan
mutiyama@nict.go.jp

Zhuoran Wang
Department of Computer Science
University College London
London, WC1E 6BT, United Kingdom
z.wang@cs.ucl.ac.uk

Benjamin Wellington
New York University

New York, NY, 10003
{lastname}@cs.nyu.edu

Kenji Yamada
Language Weaver
Marina del Rey, CA 90292, U.S.A.
kyamada@alumni.usc.edu

Index