Phrase-based Statistical Machine Translation: Models, Search, Training

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der Rheinisch-Westfälischen Technischen Hochschule Aachen zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom–Informatiker Richard Zens

aus

Düren

Berichter:
Professor Dr.–Ing. Hermann Ney
Professor Dr. Francisco Casacuberta

Tag der mündlichen Prüfung: Freitag, 29. Februar 2008

Moderation is a fatal thing. Nothing succeeds like excess.

Oscar Wilde – A Woman of No Importance, Third Act, 1893

Acknowledgments

At this point, I would like to express my gratitude to all the people who supported and accompanied me during the progress of this work.

First, I would like to thank my advisor Professor Dr.-Ing. Hermann Ney, head of the Chair of Computer Science 6 at the RWTH Aachen University. This thesis would not have been possible without his advice, continuous interest and support.

I would also like to thank Professor Dr. Francisco Casacuberta from the Universidad Politecnica de Valencia for agreeing to review this thesis and for his interest in this work.

All the people at the Chair of Computer Science 6 deserve my gratitude for many fruitful discussions, helpful feedback, and for the very good working atmosphere. I want to thank all those who helped me when writing this thesis by proofreading it, pointing out bad formulations and requesting clarifications. Furthermore, I would like to thank the secretaries and the system administrators for their continuous support.

I am very thankful for the friendly atmosphere and the support I received at the Advanced Research Institute International, Kyoto, Japan during my stay in 2003. It was a very interesting and valuable experience.

I would also like to thank all the people who made the CLSP summer research workshop on the open source SMT toolkit "Moses" possible: the organizers from CLSP/JHU and all members of both teams. It was a productive and fun environment.

Grosser Dank gilt meinen Eltern, die mir das Studium der Informatik ermöglicht haben. Desweiteren möchte ich mich bei meiner Familie und Freunden für den angenehmen Ausgleich zum Arbeitsleben bedanken.

Abstract

Machine translation is the task of automatically translating a text from one natural language into another. In this work, we describe and analyze the phrase-based approach to statistical machine translation. In any statistical approach to machine translation, we have to address three problems: the modeling problem, i. e. how to structure the dependencies of source and target language sentences; the search problem, i. e. how to find the best translation candidate among all possible target language sentences; the training problem, i. e. how to estimate the free parameters of the model from the training data.

We will present improved alignment and translation *models*. We will present alignment models which improve the alignment quality significantly. We describe several phrase translation models and analyze their contribution to the overall translation quality.

We formulate the *search* problem for phrase-based statistical machine translation and present different search algorithm in detail. We analyze the search and show that it is important to focus on alternative reorderings, whereas on the other hand, already a small number of lexical alternatives are sufficient to achieve good translation quality. The reordering problem in machine translation is difficult for two reasons: first, it is computationally expensive to explore all possible permutations; second, it is hard to select a good permutation. We compare different reordering constraints to solve this problem efficiently and introduce a lexicalized reordering model to find better reorderings.

We investigate alternative *training* criteria for phrase-based statistical machine translation. In this context, we generalize the known word posterior probabilities to n-gram posterior probabilities.

The resulting machine translation system achieves state-of-the-art performance on the large scale Chinese-English NIST task. Furthermore, the system was ranked first in the official TC-Star evaluations in 2005, 2006 and 2007 for the Chinese-English broadcast news speech translation task.

Kurzfassung

Die maschinelle Übersetzung befasst sich mit dem Problem der automatischen Übersetzung eines Textes der Quellsprache in die Zielsprache. In dieser Arbeit beschreiben und analysieren wir den phrasenbasierten statistischen Ansatz in der maschinellen Übersetzung. In der statistischen Übersetzung müssen im Allgemeinen drei Probleme angegangen werden: erstens das Modellierungsproblem, d.h. wie die Abhängigkeiten zwischen einem Satz der Quellsprache und dem entsprechenden Satz der Zielsprache beschrieben werden; zweitens das Suchproblem, d.h. wie die beste Übersetzung unter allen möglichen Sätzen der Zielsprache gefunden wird; und drittens das Trainingsproblem, d.h. wie die freien Parameter des Modells bestimmt werden.

Wir beschreiben verbesserte Alignment- und Übersetzungsmodelle. Die Alignmentmodelle verbessern die Alignmentqualität signifikant. Es werden mehrere phrasenbasierte Übersetzungsmodelle beschrieben und deren Beitrag zur Übersetzungsqualität analysiert.

Wir formulieren das Suchproblem für die phrasenbasierte statistische Übersetzung und beschreiben verschiedene Suchalgorithmen im Detail. Wir analysieren die Suche und zeigen, dass es insbesondere wichtig ist alternative Umordnungen zu berücksichtigen. Andererseits ist bereits eine relativ geringe Anzahl lexikalischer Alternativen ausreichend, um gute Übersetzungen zu produzieren. Das Umordnungsproblem der maschinellen Übersetzung ist aus zwei Gründen schwierig: erstens ist es ein kombinatorisches Problem alle Permutationen zu testen und zweitens ist es schwierig eine geeignete Permutation auszuwählen. Wir vergleichen verschiedene Einschränkungen um das Umordnungsproblem effizient zu lösen. Desweiteren beschreiben wir ein lexikalisiertes Umordnungsmodell das hilft bessere Umordnungen auszuwählen.

Wir untersuchen verschiedene Trainingskriterien für die phrasenbasierte statistische Übersetzung. In diesem Kontext generalisieren wir die bekannten Wortposteriorwahrscheinlichkeiten zu n-gramm-Posteriorwahrscheinlichkeiten.

Die Übersetzungsqualität des resultierenden Übersetzungssystems entspricht dem aktuellen Stand der Technik. Desweiteren erreichte das System den ersten Rang in den offiziellen Evaluationen der Jahre 2005, 2006 und 2007 für die Chinesisch-Englische Übersetzungsaufgabe die im Rahmen des TC-Star Projektes der Europäischen Union durchgeführt wurden.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Machine Translation

Machine translation (MT) is the task of automatically translating a text from one natural language into another. MT was one of the first envisioned applications of computers back in the 1950's. Even after more than 50 years of research, MT is still an open problem.

Nowadays, the demand for MT is steadily growing. In the European Union, documents have to be translated into all official languages (currently 23). This multilinguality is considered a part of the democracy. In 2006, the European Union spend about 800 million Euros for the translation of text documents and interpretation, e. g. of the parliamentary speeches [EU 07]. According to news paper articles the estimate for 2007 is about 1.1 billion Euros[a]. The European Union is already utilizing MT system to assist the translation efforts; typically, the MT output is post-edited by human translators. Improved machine translation quality could help to reduce this post-editing work.

The United Nations also translates a large number of documents into several languages. In fact, the United Nations corpora for Chinese-English and Arabic-English are among of the largest bilingual corpora distributed via the Linguistic Data Consortium (LDC). Also in the private sector, there is a large demand for MT: technical manuals have to translated into several languages. These technical manuals have strict style guidelines and a specific terminology.

An even larger demand exists in the World Wide Web. Independent of your nationality, most web pages are available only in a foreign language. MT can be used to get an idea of the content of those foreign web sites. MT can help to reduce the language barrier and enable easier communication.

There exist different approaches to address the problem of machine translation. We will

[a] Handelsblatt 30.04.2007

now give a rough overview over these different methodologies.

- **The Rule-based Approach**.
  In rule-based system, the source language text is analyzed, e.g. using parsers and/or morphological tools, and transformed into intermediary representation. From this representation, the target language text is generated. The rules are written by human experts. As a large number of rules is required to capture the phenomena of natural language, this is a time consuming process. As the set of rules grows over time, it gets more and more complicated to extend it and ensure consistency.

- **The Data-driven Approach**.
  In the data-driven approach, bilingual and monolingual corpora are used as main knowledge source. Often, a further distinction is made between the example-based approach, where the basic idea is to do translation by analogy, and the statistical approach. In the statistical approach, MT is treated as a decision problem: given the source language sentence, we have to decide for the target language sentence that is the best translation. Then, Bayes rule and statistical decision theory are used to address this decision problem.

In this work, we will follow the statistical approach to machine translation. A general misconception about statistical machine translation (SMT) is that it implies certain types of models, e.g. word replacement models or phrase-based models. This is not the case. A SMT system is characterized by its use of statistical decision theory and Bayes decision rule to minimize the number of decision errors.

The statistical approach has several advantages. Statistical decision theory is a well-understood area which provides a sound way to combine several knowledge sources into a global decision criterion with the goal of minimizing the number of errors. It provides a sound framework to resolve the ambiguities of translating the source language text into the target language. The model parameters are estimated from training data. As more and more training data becomes available, SMT systems get better and better. There exists freely available software to build SMT systems. Also, in recent public evaluations, SMT systems performed very well.

1.2 The Statistical Approach to Machine Translation

[Weaver 55] proposed to use statistical methods and ideas from information theory for MT. Despite first successful results, the problem turned out to be more complicated than expected. The ALPAC report found in 1966 that ten years of research fell behind the expectations and, thus, the funding of MT research was largely reduced. Increased computing power and the availability of parallel corpora, the Canadian Hansards, paved the way for its revival in the late 1980's with the seminal work of the IBM group [Brown & Cocke$^+$ 88, Brown & Cocke$^+$ 90, Brown & Della Pietra$^+$ 93].

1.2.1 Bayes decision rule for machine translation

In statistical machine translation, we are given a source language sentence $f_1^J = f_1 \ldots f_j \ldots f_J$, which is to be translated into a target language sentence $e_1^I = e_1 \ldots e_i \ldots e_I$. Statistical decision theory tells us that among all possible target language sentences, we should choose the sentence which minimizes the expected loss [Duda & Hart$^+$ 00]:

$$\hat{e}_1^{\hat{I}} = \operatorname*{argmin}_{I, e_1^I} \left\{ \sum_{I', e'^{I'}_1} Pr(e'^{I'}_1 | f_1^J) \cdot L(e_1^I, e'^{I'}_1) \right\} \tag{1.1}$$

This is the Bayes decision rule for statistical machine translation. Here, $L(e_1^I, e'^{I'}_1)$ denotes the loss function under consideration. It measures the loss (or errors) of a candidate translation e_1^I assuming the correct translation is $e'^{I'}_1$. $Pr(e_1^I|f_1^J)$ denotes the posterior probability distribution over all target language sentences e_1^I given the specific source sentence f_1^J.[b]

Note that the Bayes decision rule explicitly depends on the loss function $L(\cdot, \cdot)$. In case we want to minimize the sentence or string error rate, the corresponding loss function is:

$$\begin{aligned} L_{0-1}(e_1^I, e'^{I'}_1) &= \begin{cases} 0 & \text{if } e_1^I = e'^{I'}_1 \\ 1 & \text{else} \end{cases} && (1.2) \\ &= 1 - \delta(e_1^I, e'^{I'}_1) && (1.3) \end{aligned}$$

Here, $\delta(\cdot, \cdot)$ denotes the Kronecker-function. This loss function is called 0-1 loss as it assign a loss of zero to the correct solution and a loss of 1 otherwise. Using the 0-1 loss, Bayes decision can be simplified to

$$\hat{e}_1^{\hat{I}} = \operatorname*{argmax}_{I, e_1^I} \left\{ Pr(e_1^I | f_1^J) \right\} \tag{1.4}$$

This decision rule is also called the maximum a-posteriori (MAP) decision rule. Thus, we select the hypothesis which maximizes the posterior probability $Pr(e_1^I|f_1^J)$.

It is noteworthy that virtually all MT systems use the MAP decision rule although they are usually *not* evaluated using the 0-1 loss function. The most common evaluation criterion nowadays is the Bleu score [Papineni & Roukos$^+$ 02], which is used in many public evaluations such as NIST, TC-Star, and IWSLT. This results in a mismatch between the decision rule that is used to generate a translation hypothesis and the loss function that is used to evaluate it. [Kumar & Byrne 04] presented a Bleu induced Bayes risk decoder and reported performance gains. A similar approach was taken in [Venugopal & Zollmann$^+$ 05, Ehling & Zens$^+$ 07].

[b] The notational convention will be as follows: we use the symbol $Pr(\cdot)$ to denote general probability distributions with (nearly) no specific assumptions. In contrast, for model-based probability distributions, we use the generic symbol $p(\cdot)$.

Once we specified the Bayes decision rule for statistical machine translation, we have to address three problems [Ney 01]:

- the **modeling problem**, i. e. how to structure the dependencies of source and target language sentences;
- the **search problem**, i. e. how to find the best translation candidate among all possible target language sentences;
- the **training problem**, i. e. how to estimate the free parameters of the models from the training data.

In this work, we will address these problems. We will describe the modeling problem in Chapter 3 and Chapter 4, the search problem in Chapter 5, and finally the training problem in Chapter 6.

1.2.2 Log-linear model

In the original work on statistical machine translation [Brown & Cocke+ 90], the posterior probability was decomposed:

$$Pr(e_1^I|f_1^J) \quad = \quad \frac{Pr(e_1^I) \cdot Pr(f_1^J|e_1^I)}{P(f_1^J)} \tag{1.5}$$

Note that the denominator $P(f_1^J)$ depends only on the source sentence f_1^J and, in case of the MAP decision rule, can be omitted during the search:

$$\hat{e}_1^{\hat{I}} \quad = \quad \operatorname*{argmax}_{I, e_1^I} \left\{ Pr(e_1^I) \cdot Pr(f_1^J|e_1^I) \right\} \tag{1.6}$$

This is the so-called fundamental equation of statistical machine translation [Brown & Della Pietra+ 93]. The decomposition into two knowledge sources is known as the source-channel approach to statistical machine translation [Brown & Cocke+ 90]. It allows an independent modeling of the target language model $Pr(e_1^I)$ and the translation model $Pr(f_1^J|e_1^I)$. The target language model $Pr(e_1^I)$ describes the well-formedness of the target language sentence. The translation model $Pr(f_1^J|e_1^I)$ links the source language sentence to the target language sentence.

An alternative to the classical source-channel approach is the direct modeling of the posterior probability $Pr(e_1^I|f_1^J)$. Using a log-linear model was proposed in [Papineni & Roukos+ 98, Och & Ney 02].

$$Pr(e_1^I|f_1^J) \quad = \quad p_{\lambda_1^M}(e_1^I|f_1^J) \tag{1.7}$$

$$p_{\lambda_1^M}(e_1^I|f_1^J) \quad = \quad \frac{\exp\left(\sum_{m=1}^M \lambda_m h_m(e_1^I, f_1^J)\right)}{\sum\limits_{e'^{I'}_1} \exp\left(\sum_{m=1}^M \lambda_m h_m(e'^{I'}_1, f_1^J)\right)} \tag{1.8}$$

Here, we have models $h_m(e_1^I, f_1^J)$ and model scaling factors λ_m. Again, the denominator represents a normalization factor that depends only on the source sentence f_1^J. Therefore, we can omit it during the search process in case of the MAP decision rule. The result is a linear combination of the individual models $h(\cdot,\cdot)$:

$$\begin{aligned}
\hat{e}_1^{\hat{I}} &= \operatorname*{argmax}_{I,e_1^I} \left\{ Pr(e_1^I|f_1^J) \right\} && (1.9)\\
&= \operatorname*{argmax}_{I,e_1^I} \left\{ \frac{\exp\left(\sum_{m=1}^M \lambda_m h_m(e_1^I, f_1^J)\right)}{\sum\limits_{{e'}_1^{I'}} \exp\left(\sum_{m=1}^M \lambda_m h_m({e'}_1^{I'}, f_1^J)\right)} \right\} && (1.10)\\
&= \operatorname*{argmax}_{I,e_1^I} \left\{ \sum_{m=1}^M \lambda_m h_m(e_1^I, f_1^J) \right\} && (1.11)
\end{aligned}$$

This approach is a generalization of the source-channel approach. It has the advantage that additional models $h(\cdot,\cdot)$ can be easily integrated into the overall system. The model scaling factors λ_1^M are trained according to the maximum class posterior criterion, e.g., using the GIS algorithm [Och & Ney 02]. Alternatively, one can train them with respect to the final translation quality measured by an error criterion [Och 03]. This is the so-called minimum error rate training (MERT).

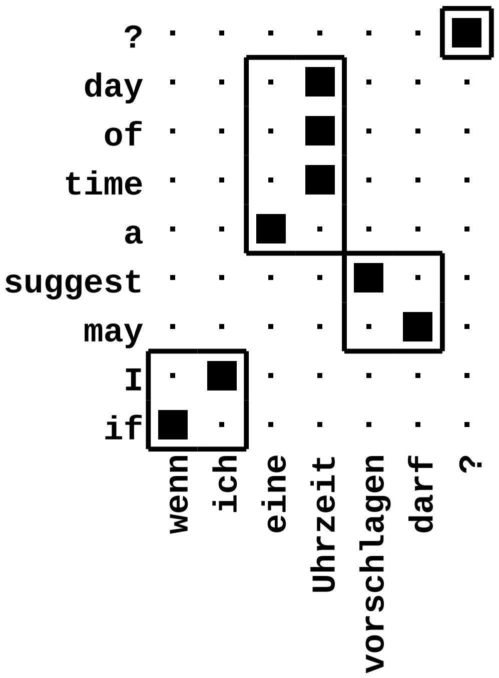
1.2.3 Phrase-based approach

The basic idea of phrase-based translation is to segment the given source sentence into phrases, then translate each phrase and finally compose the target sentence from these phrase translations. An example is shown in Figure 1.1. The source language sentence "wenn ich eine Uhrzeit vorschlagen darf?" is written along the x-axis and the English target language sentence "if I may suggest a time of day?" along the y-axis. The phrase pairs are represented as boxes and the word alignment within the phrases as black squares. The used phrase pairs are also listed in the table on the right hand side of Figure 1.1. To introduce the notation that we will use throughout this work, the idea is also illustrated in Figure 1.2. Formally, we define a segmentation of a given sentence pair (f_1^J, e_1^I) into K phrase pairs:

$$k \quad \rightarrow \quad s_k := (i_k; b_k, j_k), \text{ for } k = 1 \ldots K. \tag{1.12}$$

Here, i_k denotes the end position of the k^{th} target phrase. The pair (b_k, j_k) denotes the start and end positions of the source phrase that is aligned to the k^{th} target phrase. Phrases are defined as nonempty contiguous sequences of words. We constrain the segmentations such that all words in the source and the target sentence are covered by exactly one phrase. Thus, there are no gaps and there is no overlap. Formally:

$$\begin{aligned}
\bigcup_{k=1}^K \{b_k, \ldots, j_k\} &= \{1, \ldots, J\} && (1.13)\\
\{b_k, \ldots, j_k\} \cap \{b_{k'}, \ldots, j_{k'}\} &= \emptyset \;\; \forall k \neq k' && (1.14)
\end{aligned}$$



source phrase	target phrase
wenn ich	if I
eine Uhrzeit	a time of day
vorschlagen darf	may suggest
?	?

Figure 1.1: Example of phrase-based translation and the list of used phrase pairs.

For a given sentence pair (f_1^J, e_1^I) and a given segmentation s_1^K, we define the bilingual phrases as:

$$\tilde{e}_k := e_{i_{k-1}+1} \dots e_{i_k} \tag{1.15}$$

$$\tilde{f}_k := f_{b_k} \dots f_{j_k} \tag{1.16}$$

Let $|\tilde{e}|$ denote the length of a phrase $\tilde{e}$. Sometimes we have to refer to words within a phrase. Therefore, we define

$$\tilde{e}_k^i = e_{i_{k-1}+i} \tag{1.17}$$

As result, we have: $\tilde{e}_k = \tilde{e}_k^1 \tilde{e}_k^2 \dots \tilde{e}_k^{|\tilde{e}_k|}$. Using definition 1.17, we can not only refer to words within the k^{th} phrase, but also to words before the k^{th} phrase. For instance, $\tilde{e}_k^0$ refers to $e_{i_{k-1}}$, i. e. the last word of the $(k-1)^{\text{th}}$ phrase. As some models involve a special treatment of the sentence start and sentence end, we use the following definitions:

$$i_0 = 0 \tag{1.18}$$

$$j_0 = 0 \tag{1.19}$$

$$i_{K+1} = I + 1 \tag{1.20}$$

$$b_{K+1} = J + 1 \tag{1.21}$$

$$e_i = \text{sentence start symbol if } i \leq 0 \tag{1.22}$$

$$e_{I+1} = \text{sentence end symbol} \tag{1.23}$$

As it should be always clear from the context if the sentence start or sentence end is meant, we use the symbol '\$' for both. Note that the segmentation s_1^K contains the information about the phrase-level reordering. The segmentation s_1^K is introduced as a

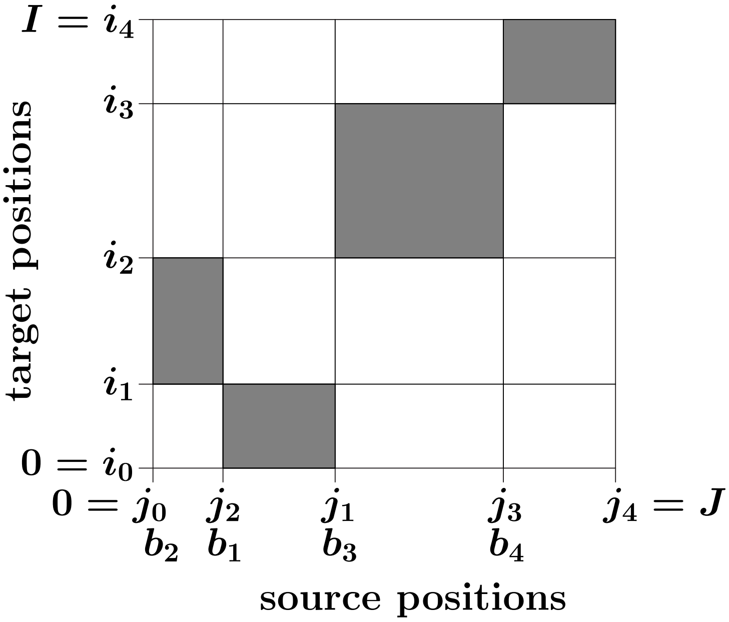


Figure 1.2: Illustration of the phrase segmentation.

hidden variable in the translation model.

$$
\begin{aligned}
Pr(e_1^I|f_1^J) &= \sum_{s_1^K} Pr(e_1^I, s_1^K|f_1^J) && (1.24)\\
&= \sum_{s_1^K} \frac{\exp\left(\sum_{m=1}^M \lambda_m h_m(e_1^I, s_1^K; f_1^J)\right)}{\sum\limits_{e'^{I'}_1, s'^{K'}_1} \exp\left(\sum_{m=1}^M \lambda_m h_m(e'^{I'}_1, s'^{K'}_1; f_1^J)\right)} && (1.25)\\
&\approx \max_{s_1^K} \frac{\exp\left(\sum_{m=1}^M \lambda_m h_m(e_1^I, s_1^K; f_1^J)\right)}{\sum\limits_{e'^{I'}_1, s'^{K'}_1} \exp\left(\sum_{m=1}^M \lambda_m h_m(e'^{I'}_1, s'^{K'}_1; f_1^J)\right)} && (1.26)
\end{aligned}
$$

Theoretically, it is correct to sum over all possible segmentations. In practice, we use the maximum approximation for this sum (Equation 1.26). As the denominator of this term is independent of the target sentence e_1^I, it can be omitted in the resulting MAP decision rule:

$$
\hat{e}_1^{\hat{I}} = \operatorname*{argmax}_{I, e_1^I} \left\{ \max_{s_1^K} \sum_{m=1}^M \lambda_m h_m(e_1^I, s_1^K; f_1^J) \right\} \qquad (1.27)
$$

As a result of the maximum approximation, the models $h(\cdot)$ depend not only on the

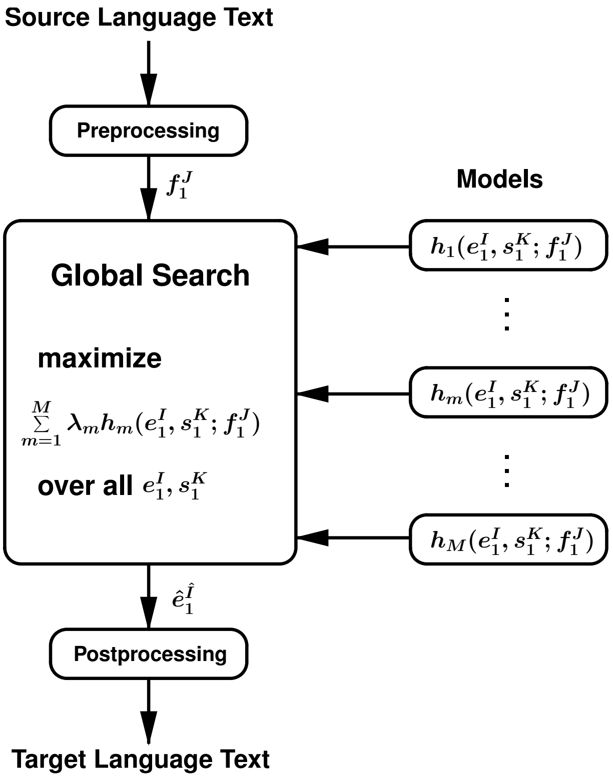


Figure 1.3: Illustration of Bayes architecture for machine translation.

sentence pair (f_1^J, e_1^I), but also on the segmentation s_1^K, i. e. we have models $h(e_1^I, s_1^K; f_1^J)$. The resulting architecture is illustrated in Figure 1.3.

During the search, the translation hypothesis is generated phrase by phrase. This can be interpreted as a sequence of K decisions where in each step, we have to decide on a triple $(\tilde{e}_k, b_k, j_k)$. In step k we translate the source positions $b_k, \ldots, j_k$, i. e. the source phrase $\tilde{f}_k$, with the target phrase $\tilde{e}_k$.

1.3 Related Work

As mentioned earlier, the revival of the statistical approach to machine translation began in the late 1980's with the work of the IBM group [Brown & Cocke+ 88, Brown & Cocke+ 90, Brown & Della Pietra+ 93]. These first approaches used a single-word based lexicon models, i. e. the translation probabilities depend just on single words. They introduced the concept of the word alignment, i. e. the correspondence between source and target words. During the JHU workshop 1999 an open-source training software for the IBM models was implemented [Al-Onaizan & Curin+ 99]. The main component was the GIZA tool for the EM training of the word alignment models, which was later extended to GIZA++ [Och & Ney 03]. The search algorithms were based on stack decoding [Berger & Brown+ 96], multi-stack decoding [Wang & Waibel 97], greedy techniques [Germann & Jahr+ 01, Germann & Jahr+ 04] or dynamic programming [Tillmann & Vogel+ 97, Tillmann & Ney 00, Tillmann & Ney 03].

Nowadays, the phrase-based translation approach is much more popular. Most phrase-based systems, including the one presented here, are derived from the alignment template approach [Och & Tillmann+ 99, Och 02, Och & Ney 04]. An alignment template is a triple which describes the alignment between a source phrase and a target phrase. These phrases are defined at the level of word classes. The alignment templates are extracted from word-aligned bilingual corpora. The alignment template approach was shown to outperform single-word based approaches in [Och & Tillmann+ 99, Och 02].

In the phrase-based approach here, we do not use word classes; the phrases are thus defined at the word level. The extraction of the phrases from word-aligned bilingual corpora uses the same algorithm as for the alignment templates.

A similar approach is described in [Tomás & Casacuberta 01]; there the phrase translation probabilities are estimated using the EM algorithm, but the phrase segmentation is constrained to be monotonic. The experiments were done for Spanish-Catalan task where this constraint might be suitable. For language pairs which are not as close, the monotonicity constraint might be inappropriate. The phrases are identified using the IBM model 1 lexicon. Later, this approach was extended to non-monotonic decoding [Tomás & Casacuberta 04].

[Marcu & Wong 02] present a joint probability model for phrase-based translation. It does not use the word alignment for extracting the phrases, but directly generates a phrase alignment. The training is done using the EM algorithm. Decoding is done using an extension of a greedy decoder for single-word based models. A problem of this approach is that it does not scale well to large data tasks. Therefore, [Birch & Callison-Burch+ 06] tried to improve the scalability, but still this approach remains applicable only for small tasks.

In [Tillmann & Xia 03], a phrase-based unigram model is described that uses the joint probability of source and target phrase estimated using relative frequencies. The phrase pairs are extracted from word-aligned bilingual corpora similar to the alignment templates; more details on the phrase extraction are presented in [Tillmann 03].

In [Koehn & Och+ 03], various aspects of phrase-based systems are compared, e. g. the phrase extraction method, the underlying word alignment model. The public Pharaoh decoder is described in more detail in [Koehn 04a]. During the Johns Hopkins University summer research workshop 2006, the decoder was re-written and made publicly available as open-source [Koehn & Hoang+ 07].

In [Vogel 03], a phrase-based system is used that allows reordering within a window of up to three words. Improvements for a Chinese–English task are reported compared to a monotonic search. Instead of using the relative frequency estimate, the phrase translation probability was computed using the IBM model 1 lexicon.

An implementation of the alignment template approach and the phrase-based approach using weighted finite state transducers was presented in [Kumar & Byrne 03, Kumar & Byrne 05]. The use of finite state techniques for MT was already proposed in [Vidal 97, Knight & Al-Onaizan 98]. This approach is appealing as there exists publicly available tools for manipulating finite state automata, for instance

[Mohri & Pereira$^+$, Kanthak & Ney 04].

Other translation systems model the translation process using tree structures. These can be monolingual tree structures on the target language side as e. g. in [Yamada & Knight 01, Yamada & Knight 02] and more recently [Galley & Hopkins$^+$ 04, Galley & Graehl$^+$ 06], or bilingual (also call synchronous) tree structures as e. g. in [Wu 96, Wu 97, Melamed 04, Chiang 05, Zollmann & Venugopal 06, Chiang 07]. Some these approaches are based on the phrase-based approach [Chiang 05, Chiang 07] or utilize methods from the phrase-based approach [DeNeefe & Knight$^+$ 07]. Although all these approaches are formally syntax-based, not all of them make use of linguistically motivated non-terminals.

2 Scientific Goals

- **Improved word alignment models.**
  Word aligned bilingual training corpora are the starting point for most phrase-based systems. We present alignment models which, compared to IBM model 4, improve the word alignment quality significantly.

- **Comparison of several phrase-based models.**
  The overall translation model is a combination of a set of models. We investigate several phrase-based translation models and their combinations as well as their contributions to the overall translation quality.

- **Detailed analysis of search algorithms.**
  We formulate the search problem for phrase-based SMT and analyze the search space. We present different search algorithm in detail and compare the quality and efficiency to a public available decoder.

- **Efficient search for ambiguous input.**
  Often machine translation is a component of a larger pipeline; then the input to the machine translation component may be ambiguous. A typical example is a speech translation system. Taking a large number of multiple input alternatives into account is computationally problematic. Here, we show how to do the phrase matching of the input lattice and the phrase table in an efficient way and we will describe search algorithms for lattice input.

- **Improved reordering.**
  The reordering problem in MT is difficult for two reasons: first, it is computational expensive to explore all possible permutations; second, it is hard to select a good permutation. We compare several reordering strategies to solve this problem efficiently. Additionally, we introduce a lexicalized reordering model that improves the translation quality significantly.

- **Efficient phrase-table representation.**
  Standard phrase-based MT systems store the whole phrase table in memory. We introduce a phrase-table representation that is loaded on-demand from disk. This enables online phrase-based SMT, i. e. the translation of arbitrary text without time-consuming pre-filtering of the phrase table.

- **Comparison of training criteria.**
  We investigate alternative training criteria for adjusting the model scaling factors of a phrase-based SMT system. In this context, we generalize the known word posterior probabilities to n-gram posterior probabilities. These can be also used in a rescoring/reranking framework.

3 Improved Word Alignment Models

3.1 Introduction

In this chapter, we will describe improvements of the standard IBM word alignment models.[a] Word-aligned bilingual corpora are an important knowledge source for many tasks in natural language processing. Obvious applications are the extraction of bilingual word or phrase lexica [Melamed 00, Och & Ney 00]. These applications depend heavily on the quality of the word alignment [Och & Ney 00]. Word alignment models were first introduced in statistical machine translation [Brown & Della Pietra$^+$ 93]. The alignment describes the mapping from source sentence words to target sentence words.

Using the IBM translation models IBM-1 to IBM-5 [Brown & Della Pietra$^+$ 93], as well as the Hidden-Markov alignment model [Vogel & Ney$^+$ 96], we can produce alignments of good quality. However, all these models are based on restricted alignments in the sense that a source word can be aligned to at most one target word. This constraint is necessary to reduce the computational complexity of the models, but it makes it impossible to align phrases in the target language (English) such as 'the day after tomorrow' to one word in the source language (German) 'übermorgen'. Another drawback of the IBM models is that no algorithm for computing the Viterbi alignment for the fertility-based alignment models IBM 3-5 is known. Therefore approximation have to be used.

We will present two theoretically well-founded techniques which at least partially overcome the negative effect of this constraint.

1. We will perform the training in both translation directions, source-to-target and target-to-source, simultaneously. The parameter estimation is done using the EM-algorithm. After each iteration, we compute the lexicon probabilities by combining the lexicon counts estimated in the two directions. Thus, the proposed model utilizes the knowledge from both training directions to improve the parameter estimation. We will analyze a linear and a log-linear combination of lexicon counts.

2. We will present a word alignment algorithm that avoids the ”one-target word per source word constraint” and produces symmetric word alignments. This algorithm considers the alignment problem as a task of finding the edge cover with minimal costs in a bipartite graph. The parameters of the IBM models and HMM, in particular the state occupation probabilities, will be used to determine the costs of aligning a specific source word to a target word. The advantage of this approach is that the global optimal solution can be found efficiently, whereas for the standard IBM model 4 an efficient algorithm for finding the global optimum is not known.

[a] The experiments in this chapter have been performed in cooperation with Evgeny Matusov.

Many words in the bilingual sentence-aligned texts are singletons, i. e. they occur only once. This is especially true for the highly inflected languages such as German. As a (partial) solution to this problem, we will smooth the lexicon probabilities of the full-form words using a probability distribution that is estimated using the word stems. Thus, we exploit that multiple full-form words share the same word stem and have similar meanings and translations.

We will evaluate these methods on the German–English Verbmobil task and the French–English Canadian Hansards task. We will show statistically significant improvements compared to state-of-the-art results in [Och & Ney 03].

3.2 Review of Statistical Word Alignment Models

In this section, we will give an overview of the commonly used statistical word alignment models [Brown & Della Pietra$^+$ 93]. We are given a source language sentence $\mathbf{f} := f_1^J$ which has to be translated into a target language sentence $\mathbf{e} := e_1^I$. According to the classical source-channel approach, we will choose the sentence with the highest probability among all possible target language sentences:

$$\hat{\mathbf{e}} = \operatorname*{argmax}_{\mathbf{e}} \Big\{ Pr(\mathbf{e}|\mathbf{f}) \Big\} \tag{3.1}$$

$$= \operatorname*{argmax}_{\mathbf{e}} \Big\{ Pr(\mathbf{e}) \cdot Pr(\mathbf{f}|\mathbf{e}) \Big\} \tag{3.2}$$

This decomposition into two knowledge sources allows for an independent modeling of target language model $Pr(\mathbf{e})$ and translation model $Pr(\mathbf{f}|\mathbf{e})$. The word alignment $\mathbf{a}$ is introduced as into the translation model a hidden variable:

$$Pr(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} Pr(\mathbf{f},\mathbf{a}|\mathbf{e}) \tag{3.3}$$

Usually, we use restricted alignments in the sense that each source word is aligned to at most one target word. Thus, an alignment $\mathbf{a}$ is a mapping from source sentence positions to target sentence positions $\mathbf{a} := a_1^J = a_1 ... a_j ... a_J$, $a_j \in \{0, \ldots, I\}$. The alignment may contain alignments $a_j = 0$ with the 'empty' word e_0 to account for source sentence words that are not aligned to any target word. A detailed description of the popular translation models IBM-1 to IBM-5 [Brown & Della Pietra$^+$ 93], as well as the Hidden-Markov alignment model (HMM) [Vogel & Ney$^+$ 96] can be found in [Och & Ney 03]. All these models include parameters $p(f|e)$ for the single-word based lexicon. They differ in the alignment model. Now, we sketch the structure of the seven models:

- In IBM-1 all alignments have the same probability, i.e. the distribution is uniform with value $\frac{1}{I+1}$.
- IBM-2 uses a zero-order alignment model $p(a_j|j, I, J)$ where different alignment positions are independent from each other.

- The HMM uses a first-order model $p(a_j|a_{j-1}, I)$ where the alignment position a_j depends on the previous alignment position a_{j-1}. In the homogenous version, the distance (or distortion) of the positions is modeled, i. e. $p(\ |a_j - a_{j-1}|\ |\ I)$.
- In IBM-3, we have an (inverted) zero-order alignment model $p(j|a_j, I, J)$ with an additional fertility model $p(\phi|e)$ which describes the number of words ϕ aligned to an English word e.
- In IBM-4, we have an (inverted) first-order alignment model $p(j|j', I, J)$ and a fertility model $p(\phi|e)$.
- The models IBM-3 and IBM-4 are deficient as they waste probability mass on non-strings. IBM-5 is a reformulation of IBM-4 with a suitably refined alignment model to avoid deficiency.
- Model 6 is a log-linear combination of the previously described alignment models [Och & Ney 03].

A *Viterbi alignment* $\hat{\mathbf{a}}$ of a specific model is an alignment for which the following equation holds:

$$\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmax}} \Big\{ Pr(\mathbf{a}|\mathbf{f}, \mathbf{e}) \Big\} \tag{3.4}$$

$$= \underset{\mathbf{a}}{\operatorname{argmax}} \Big\{ Pr(\mathbf{f}, \mathbf{a}|\mathbf{e}) \Big\} \tag{3.5}$$

We measure the quality of an alignment model using the quality of the Viterbi alignment compared to a manually produced reference alignment.

In Section 3.4, we will apply the lexicon symmetrization methods to the models described previously. Therefore, we will now briefly review the standard training procedure for the lexicon model. The EM-algorithm [Dempster & Laird$^+$ 77] is used to train the free lexicon parameters $p(f|e)$.

In the E-step, the lexical counts for each sentence pair $(\mathbf{f}, \mathbf{e})$ are calculated and then summed over all sentence pairs:

$$N(f, e) = \sum_s \sum_{\mathbf{a}} p(\mathbf{a}|\mathbf{f}^s, \mathbf{e}^s) \sum_{i,j} \delta(f, f_j^s) \delta(e, e_i^s) \tag{3.6}$$

with

$$p(\mathbf{a}|\mathbf{f}^s, \mathbf{e}^s) = \frac{p(\mathbf{a}, \mathbf{f}^s|\mathbf{e}^s)}{\sum_{\mathbf{a}} p(\mathbf{a}, \mathbf{f}^s|\mathbf{e}^s)} \tag{3.7}$$

In the M-step the lexicon probabilities are:

$$p(f|e) = \frac{N(f, e)}{\sum\limits_{f'} N(f', e)} \tag{3.8}$$

3.3 Related Work

The popular IBM models for statistical machine translation are described in [Brown & Della Pietra+ 93]. The HMM-based alignment model was introduced in [Vogel & Ney+ 96]. A good overview of these models is given in [Och & Ney 03]. In that article Model 6 is introduced as the log-linear interpolation of the other models. Additionally, state-of-the-art results are presented for the Verbmobil task and the Canadian Hansards task for various configurations. Therefore, we chose them as baseline. Compared to our work, these publications kept the training of the two translation directions strictly separate, whereas we integrate both directions into one combined training. Additional linguistic knowledge sources such as dependeny trees or parse trees were used in [Cherry & Lin 03, Gildea 03]. Bilingual bracketing methods were used to produce a word alignment in [Wu 97]. [Melamed 00] uses an alignment model that enforces one-to-one alignments. In [Toutanova & Ilhan+ 02], extensions to the HMM-based alignment model are presented.

3.4 Symmetrized Lexicon Model

During the standard training procedure, the lexicon parameters $p(f|e)$ and $p(e|f)$ were estimated independent of each other in strictly separate trainings. In this section, we present two symmetrization methods for the lexicon model. The motivation is that a combination of the two training directions should result in better and more reliable parameter estimates for the lexicon model than each of the individual directions. As a starting point, we use the *joint* lexicon probability $p(f, e)$ and determine the conditional probabilities for the source-to-target direction $p(f|e)$ and the target-to-source direction $p(e|f)$ as follows:

$$p(f|e) = \frac{p(f,e)}{\sum\limits_{f'} p(f',e)} \tag{3.9}$$

$$p(e|f) = \frac{p(f,e)}{\sum\limits_{e'} p(f,e')} \tag{3.10}$$

The nonsymmetric auxiliary Q-functions for reestimating the lexicon probabilities with the EM-algorithm can be represented as follows. Here, $N_{ST}(f, e)$ and $N_{TS}(f, e)$ denote the collected lexicon counts for the source-to-target (ST) direction and the target-to-source (TS) direction, respectively. Hence, $N_{ST}(f, e)$ is the result of the E-step of the EM-algorithm in the source-to-target direction and $N_{TS}(f, e)$ is the result of the E-step in the target-to-source direction.

$$Q_{ST}(\{p(f|e)\}) = \sum_{f,e} N_{ST}(f,e) \cdot \log \frac{p(f,e)}{\sum\limits_{f'} p(f',e)} \tag{3.11}$$

$$Q_{TS}(\{p(e|f)\}) = \sum_{f,e} N_{TS}(f,e) \cdot \log \frac{p(f,e)}{\sum\limits_{e'} p(f,e')} \tag{3.12}$$

3.4.1 Linear interpolation

To estimate this joint probability with the EM-algorithm, we define the auxiliary Q-function as a linear interpolation of the Q-functions for the source-to-target and target-to-source direction:

$$Q_\alpha(\{p(f,e)\}) \;=\; \alpha \cdot Q_{ST}(\{p(f|e)\}) + (1-\alpha) \cdot Q_{TS}(\{p(e|f)\}) \tag{3.13}$$

$$\begin{aligned} =\;& \alpha \cdot \sum_{f,e} N_{ST}(f,e) \cdot \log p(f,e) + (1-\alpha) \cdot \sum_{f,e} N_{TS}(f,e) \cdot \log p(f,e) \\ & -\alpha \cdot \sum_{e} N_{ST}(e) \cdot \log \sum_{f'} p(f',e) - (1-\alpha) \cdot \sum_{f} N_{TS}(f) \cdot \log \sum_{e'} p(f,e') \end{aligned} \tag{3.14}$$

The unigram counts $N(e)$ and $N(f)$ are determined, for each of the two translation directions, by taking a sum of $N(f,e)$ over f and over e, respectively:

$$N(f) = \sum_{e} N(f,e) \qquad\qquad N(e) = \sum_{f} N(f,e) \tag{3.15}$$

We define the *combined lexicon count* $N_\alpha(f,e)$:

$$N_\alpha(f,e) := \alpha \cdot N_{ST}(f,e) + (1-\alpha) \cdot N_{TS}(f,e) \tag{3.16}$$

Now, we derive the symmetrized Q-function over $p(f,e)$ for a certain word pair (f,e). Then, we set this derivative to zero to determine the reestimation formula for $p(f,e)$ and obtain the following equation:

$$\frac{N_\alpha(f,e)}{p(f,e)} = \alpha \cdot \frac{N_{ST}(e)}{\sum\limits_{f'} p(f',e)} + (1-\alpha) \cdot \frac{N_{TS}(f)}{\sum\limits_{e'} p(f,e')} \tag{3.17}$$

We do not know a closed form solution for this equation. As approximation, we use the following term:

$$\hat{p}(f,e) \;=\; \frac{N_\alpha(f,e)}{\sum\limits_{f',e'} N_\alpha(f',e')} \tag{3.18}$$

This estimate is an exact solution, if the unigram counts for f and e are independent of the translation direction, i. e. $N_{ST}(f) = N_{TS}(f)$ and $N_{ST}(e) = N_{TS}(e)$. We make this approximation and, thus, linearly combine the lexicon counts after each iteration of the EM-algorithm. Then, we normalize these counts (according to 3.9 and 3.10) to determine the lexicon probabilities for each of the two translation directions.

3.4.2 Log-linear interpolation

We will show in Section 3.8.3 that the linear interpolation results in significant improvements over the baseline system. Motivated by these experiments, we investigated also a log-linear interpolation of the lexicon counts of the two translation directions. The combined lexicon count $N_\alpha(f,e)$ is now defined as:

$$N_\alpha(f,e) = N_{ST}(f,e)^\alpha \cdot N_{TS}(f,e)^{1-\alpha} \tag{3.19}$$

The normalization is done in the same way as for the linear interpolation. The linear interpolation resembles more a union of the two lexica whereas the log-linear interpolation is more similar to an intersection of both lexica. Thus, for the linear interpolation, a word pair (f,e) obtains a large combined count, if the count in at least one direction is large. For the log-linear interpolation, the combined count is large only if both lexicon counts are large.

In the experiments, we will use the interpolation weight $\alpha = 0.5$ for both the linear and the log-linear interpolation, i. e. the two translation directions are weighted equally.

3.4.3 Evidence trimming

Initially, the lexicon contains all word pairs that cooccur in the bilingual training corpus. The majority of these word pairs are not translations of each other. Therefore, we would like to remove those lexicon entries. Evidence trimming is one way to do this. The *evidence* of a word pair (f,e) is the estimated count $N(f,e)$. Now, we discard a word pair if its evidence is below a certain threshold τ. In the case of the symmetric lexicon, we can further refine this method. For estimating the lexicon in the source-to-target direction $\hat{p}(f|e)$, the idea is to keep all entries from this direction and to boost the entries that have a high evidence in the target-to-source direction $N_{TS}(f,e)$. We use the following formula:

$$\bar{N}_{ST}(f,e) = \begin{cases} \alpha N_{ST}(f,e) + (1-\alpha) N_{TS}(f,e) & \text{if } N_{ST}(f,e) > \tau \\ 0 & \text{else} \end{cases} \tag{3.20}$$

The count $\bar{N}_{ST}(f,e)$ is now used to estimate the source-to-target lexicon $\hat{p}(f|e)$. With this method, we do not keep entries in the source-to-target lexicon $\hat{p}(f|e)$ if their evidence is low, even if their evidence in the target-to-source direction $N_{TS}(f,e)$ is high. In the experiments, we will use $\tau = 0$. For the target-to-source direction, we apply this method in a similar way.

3.4.4 Improved training initialization

We can transform the symmetrized lexicon parameters $p(f|e)$ and $p(e|f)$ from one translation direction into the other one using the Bayes theorem:

$$p(e) \cdot p(f|e) \quad = \quad p(f) \cdot p(e|f) \tag{3.21}$$

Using Equation 3.21, we can “swap” the parameters between the two translation directions. Thus, we initialize a more difficult translation direction with the better estimated lexicon probabilities from the easier direction. Experimentally, it turned out to be sufficient to apply formula 3.21 only after the first training iteration.

The prior probabilities $p(e)$ and $p(f)$ appearing in the previous equation can be estimated in two different ways. One way is to determine the "internal" marginal distributions using the combined lexicon counts $N_\alpha(f, e)$. Another possibility is to estimate the priors "externally" as relative word (unigram) frequencies in monolingual corpora of the source and the target language, respectively. Using external priors may be benefial because they can be estimated on larger monolingual corpora which are more widely available than bilingual corpora.

3.5 Lexicon Smoothing

The lexicon model described so far is based on full-form words. For highly inflected languages such as German this may cause problems, because compared to the word stem, the full-form words occur less frequent in the training corpus. For instance, the token-type ratio for German is usually much lower than for English, e.g., in Verbmobil it is 99.4 for English and 56.3 for German. The information that multiple full-form words share the same word stem is not used in the lexicon model. To take this information into account, we smooth the lexicon model with a backing-off lexicon that is based on word stems. The smoothing method we apply is absolute discounting with interpolation:

$$p(f|e) = \frac{\max\{N(f,e) - d, 0\}}{N(e)} + \alpha(e) \cdot \beta(f, \bar{e}) \tag{3.22}$$

This method is well known from language modeling [Ney & Martin$^+$ 97]. Here, $\bar{e}$ denotes the generalization, i.e. the word stem, of the word e. The nonnegative value d is the discounting parameter, $\alpha(e)$ is a normalization constant and $\beta(f, \bar{e})$ is the normalized backing-off distribution.

The formula for $\alpha(e)$ is:

$$\begin{aligned} \alpha(e) &= \frac{1}{N(e)} \left(\sum_{f:N(f,e)>d} d + \sum_{f:N(f,e)\le d} N(f,e) \right) && (3.23) \\ &= \frac{1}{N(e)} \sum_{f} \min\{d, N(f,e)\} && (3.24) \end{aligned}$$

This formula is a generalization of the one typically used in publications on language modeling. This generalization is necessary, because the lexicon counts may be fractional whereas in language modeling typically integer counts are used. Additionally, we want to allow for discounting values d greater than one. One effect of the discounting parameter d is that all lexicon entries with a count less than d are discarded and the freed probability mass is redistributed among the other entries. The backing-off distribution $\beta(f, \bar{e})$ is estimated as:

$$\beta(f, \bar{e}) = \frac{N(f, \bar{e})}{\sum\limits_{f'} N(f', \bar{e})} \tag{3.25}$$

Here, $N(f,\bar{e})$ denotes the count of the event that the source language word f and the target language word stem $\bar{e}$ occur together. These counts are computed by summing the lexicon counts $N(f,e)$ over all full-form words e which share the same word stem $\bar{e}$: In the experiments, we use the discounting parameter $d = 0.05$.

3.6 State Occupation Probabilities

The training of all alignment models is done using the EM-algorithm. We will now have a closer look at the E-step. In the E-step, the counts for each sentence pair $(\mathbf{f},\mathbf{e})$ are calculated. Here, we present this calculation on the example of the HMM. For its lexicon parameters, the marginal probability of a target word e_i to occur at the target sentence position i as the translation of the source word f_j at the source sentence position j is estimated with the following sum:

$$p_j(i,\mathbf{f}|\mathbf{e}) \;=\; \sum_{\mathbf{a}:a_j=i} Pr(\mathbf{f},\mathbf{a}|\mathbf{e}) \tag{3.26}$$

This value represents the likelihood of aligning f_j to e_i via every possible alignment $\mathbf{a} = a_1^J$ that includes the alignment connection $a_j = i$. By normalizing over the target sentence positions, we arrive at the *state occupation probability*:

$$p_j(i|\mathbf{f},\mathbf{e}) = \frac{p_j(i,\mathbf{f}|\mathbf{e})}{\sum\limits_{i'=0}^{I} p_j(i',\mathbf{f}|\mathbf{e})} \tag{3.27}$$

In the M-step of the EM training, the state occupation probabilities are aggregated for all words in the source and target vocabularies by taking the sum over all training sentence pairs. The lexicon probabilities $p(f|e)$ are determined by renormalization.

Similarly, the training can be performed in the inverse (target-to-source) direction, yielding the state occupation probabilities $p_i(j|\mathbf{e},\mathbf{f})$.

The negated logarithms of the state occupation probabilities

$$w(i,j;\mathbf{f},\mathbf{e}) := -\log p_j(i|\mathbf{f},\mathbf{e}) \tag{3.28}$$

can be interpreted as the *costs* of aligning a source word f_j with a target word e_i. An alternative formulation of the word alignment task is the following: we have to find a mapping between the source and the target words, so that each source and each target position is covered and the total costs of the alignment are minimal.

For the models IBM-1, IBM-2 and HMM, there exist efficient algorithms to compute the exact state occupation probabilities. For the HMM this is done using the Baum-Welch algorithm [Baum 72]. So far, an efficient algorithm to compute the sum over all alignments in the fertility-based models IBM-3 to IBM-5 is not known. Therefore, this sum is approximated using a subset of promising alignments [Brown & Della Pietra$^+$ 93, Och & Ney 00]. In both cases, the resulting estimates are more precise than the ones obtained by the maximum approximation, i. e. by considering only the Viterbi alignment.

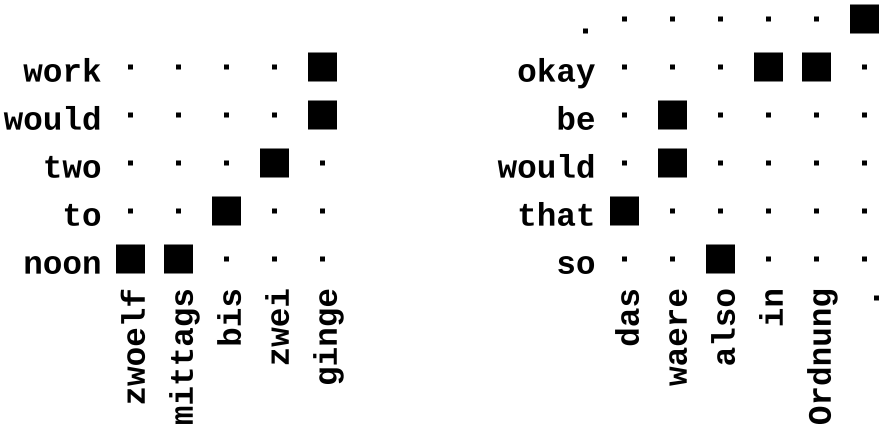


Figure 3.1: Examples of symmetric alignments with one-to-many and many-to-one connections (Verbmobil task, spontaneous speech).

Instead of using the state occupation probabilities from only one training direction as costs (Equation 3.28), we can interpolate the state occupation probabilities from the source-to-target and the target-to-source training for each pair (i,j) of positions in a sentence pair $(\mathbf{f}, \mathbf{e})$. This will improve the estimation of the local alignment costs. Having such symmetrized costs, we can employ graph alignment algorithms (cf. Section 3.7) to produce reliable alignment connections which include many-to-one and one-to-many alignment relationships. The presence of both relationship types characterizes a symmetric alignment that can potentially improve the translation results. In Figure 3.1, we show two examples of symmetric alignments.

Another important advantage is the efficiency of the graph algorithms used to determine the final symmetric alignment. They will be discussed in the following section.

3.7 Alignment Algorithms

In this section, we describe the alignment extraction algorithms. We assume that for each sentence pair $(\mathbf{f}, \mathbf{e})$ we are given a cost matrix $C \in \mathbb{R}^{J \times I}$.[b] The elements of this matrix c_{ji} are the local costs that result from aligning source word f_j to target word e_i. For a given alignment $A \subseteq J \times I$, we define the costs of this alignment $c(A)$ as the sum of the local costs of all aligned word pairs:

$$c(A) \quad = \sum_{(j,i) \in A} c_{ji} \tag{3.29}$$

Note that A is an unconstrained alignment with arbitrary alignment links. Now, our task is to find the alignment with the minimum costs. Obviously, the empty alignment has always costs of zero and would be optimal. To avoid this, we introduce additional

[b] For notational convenience, we omit the dependency on the sentence pair $(\mathbf{f}, \mathbf{e})$ in this section.

constraints. The first constraint enforces that each source word has to be aligned to at least one target word or alternatively to the empty word; this constraint will be called source sentence coverage constraint. Similarly, the second constraint enforces that each target word has to be aligned to at least one source word or the empty word. This will be called target sentence coverage constraint.

Enforcing only the source sentence coverage, the minimum cost alignment is a mapping from source positions j to target positions a_j, including zero for the empty word. Each target position a_j can be computed as:

$$a_j = \operatorname*{argmin}_{i \in \{0,\ldots,I\}} \{c_{ji}\} \tag{3.30}$$

Thus, in each column we choose the row with the minimum costs. This method resembles the common IBM models in the sense that the IBM models are also a mapping from source positions to target positions. Therefore, this method is comparable to the IBM models for the source-to-target direction. Similarly, if we enforce only the target sentence coverage, the minimum cost alignment is a mapping from target positions i to source positions b_i. Here, we have to choose in each row the column with the minimum costs. The complexity of these algorithms is in $O(J \cdot I)$.

The algorithms for determining such a non-symmetric alignment are rather simple. A more interesting case arises, if we enforce both constraints, i.e. each source word as well as each target word has to be aligned at least once. Even in this case, we can find the global optimum in polynomial time.

The task is to find a symmetric alignment A, for which the costs $c(A)$ are minimal. This task is equivalent to finding a *minimum-weight edge cover* (MWEC) in a complete bipartite graph. An edge cover of G is a set of edges E' such that each node of G is incident to at least one edge in E'. The two node sets of this bipartite graph correspond to the source sentence positions and the target sentence positions, respectively. The costs of an edge are the elements of the cost matrix C.

The correspondence between word alignment and an edge cover in a complete bipartite graph is illustrated in Figure 3.2. The two sets of nodes of the graph correspond to the source and target sentence positions, respectively. The edges correspond to potential alignment links. In this illustration, the red edges correspond to the edge cover that represents the alignment in the left part of the figure.

To solve the minimum-weight edge cover problem, we reduce it to the maximum-weight bipartite matching problem. As described in [Keijsper & Pendavingh 98], this reduction is linear in the graph size. For the maximum-weight bipartite matching problem, well-known algorithm exist, e.g. based on the Hungarian method. The complexity of this algorithm is in $O((I + J) \cdot I \cdot J)$. We will call the solution of the minimum-weight edge cover problem the "MWEC algorithm". The algorithm enforcing either source sentence coverage or target sentence coverage will be referred to as the *one-sided* minimum-weight edge cover algorithm (o-MWEC).

The elements of the cost matrix c_{ji} of a sentence pair $(\mathbf{f}, \mathbf{e})$ can be computed as a weighted

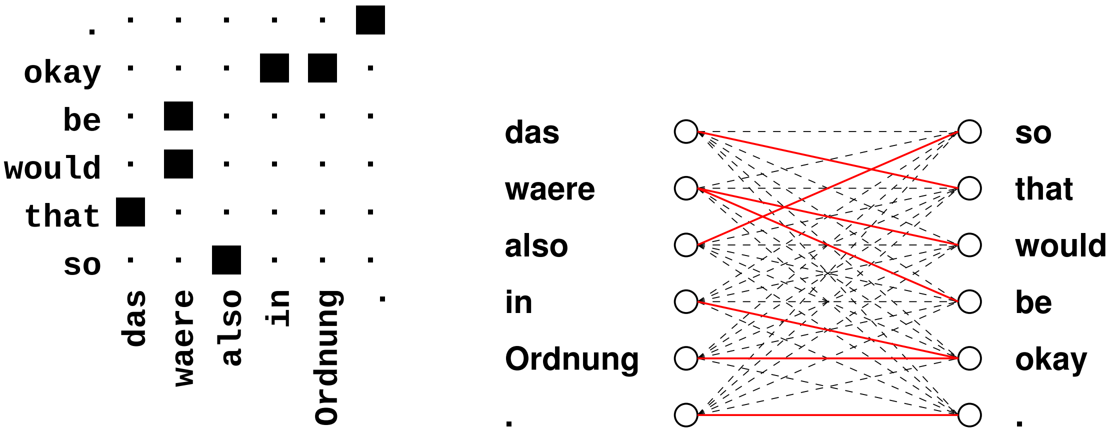


Figure 3.2: Illustration: alignment as edge cover. Left: word aligned sentence pair. Right: the same alignment as edge cover in a complete bipartite graph. The nodes correspond to the positions in the source and target sentence, respectively. The red edges correspond to the edge cover of the bipartite graph that is equivalent to the alignment on the left hand side.

linear interpolation of various cost types h_m:

$$c_{ji} = \sum_{m=1}^{M} \lambda_m \cdot h_m(j,i) \tag{3.31}$$

In our experiments, we will use the negated logarithm of the state occupation probabilities as described in Section 3.6. To obtain a more symmetric estimate of the costs, we will interpolate both the source-to-target direction and the target-to-source direction (thus the state occupation probabilities are interpolated log-linearly). Because the alignments determined in the source-to-target training may substantially differ in quality from those produced in the target-to-source training, we will use an interpolation weight α:

$$c_{ji} = \alpha \cdot w(i,j;\mathbf{f},\mathbf{e}) + (1-\alpha) \cdot w(j,i;\mathbf{e},\mathbf{f}) \tag{3.32}$$

Additional feature functions can be included to compute c_{ji}; for example, one could make use of bilingual word or phrase dictionaries.

To apply the methods described in this section, we made two assumptions: first, the costs of an alignment can be computed as the sum of *local* costs. Second, the features have to be *static* in the sense that we have to fix the costs before aligning any word. Therefore, we cannot directly apply dynamic features such as the IBM-4 distortion model in a straightforward way. One way to overcome these restrictions is the use of state occupation probabilities; e.g. for IBM-4, they contain the distortion model.

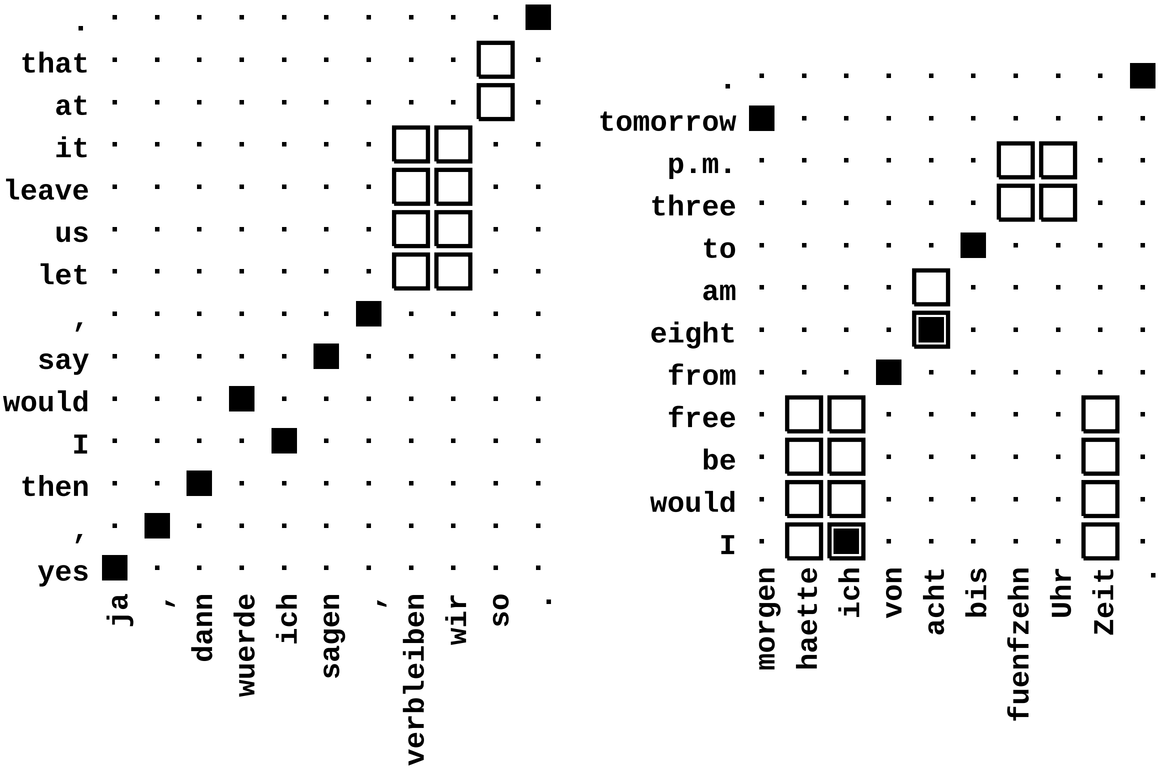


Figure 3.3: Manual alignment examples with *S(ure)* (filled dots) and *P(ossible)* connections for the Verbmobil task.

3.8 Word Alignment Experiments

In this section, we will present the word alignment experiments. We used the German–English Verbmobil task and the French–English Canadian Hansards task. First, we will describe the evaluation criterion, the alignment error rate, in Section 3.8.1. Then, we will explain the experimental setup including the corpus statistics in Section 3.8.2. Afterwards, we will present word alignment results for the different approaches.

3.8.1 Evaluation criterion

We use the same evaluation criterion as described in [Och & Ney 00, Och & Ney 03]. The generated word alignment is compared to a reference alignment which is produced by human experts. The annotation scheme explicitly takes the ambiguity of the word alignment into account. There are two different kinds of alignments: sure alignments (S) which are used for alignments that are unambiguous and possible alignments (P) which are used for alignments that might or might not exist. The P relation is used especially to align words within idiomatic expressions, free translations, and missing function words. It is guaranteed that the sure alignments are a subset of the possible alignments ($S \subseteq P$). The obtained reference alignment may contain many-to-one, one-to-many and many-to-many relationships. Figure 3.3 shows an example of a manually aligned sentence with S

Table 3.1: Corpus statistics of the Verbmobil and Canadian Hansards task.

		German	English	French	English
Train	Sentences	34 K		128 K	
	Words	329 625	343 076	2.12 M	1.93 M
	Vocabulary	5 936	3 505	37 542	29 414
	Singletons	2 600	1 305	12 986	9 572
Dictionary	Entries	4 404		28 701	
Test	Sentences	354		500	
	Words	3 233	3 109	8 749	7 946
	S relations	2 559		4 443	
	P relations	4 596		19 779	

and P relations.

The quality of an alignment A is computed as appropriately redefined precision and recall measures. Additionally, we use the alignment error rate (AER), which is derived from the well-known F-measure.

$$\text{recall} = \frac{|A \cap S|}{|S|}, \quad \text{precision} = \frac{|A \cap P|}{|A|} \tag{3.33}$$

$$\text{AER}(S, P; A) = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|} \tag{3.34}$$

With these definitions a recall error can only occur if a S(ure) alignment is not found and a precision error can only occur if a found alignment is not even P(ossible).

3.8.2 Experimental setup

We evaluated the presented word alignment methods on the Verbmobil and the Canadian Hansards task. The German–English Verbmobil task [Wahlster 00] is a speech translation task in the domain of appointment scheduling, travel planning and hotel reservations. The French–English Canadian Hansards task consists of the debates in the Canadian Parliament.

The corpus statistics for the two tasks are shown in Table 3.1. The number of running words and the vocabularies are based on full-form words including punctuation marks. To investigate the performance of our methods on sparse data, we also used smaller training corpora of only 500 sentences. As in [Och & Ney 03], the first 100 sentences of the test corpus are used as a development corpus to optimize model parameters that are not trained via the EM algorithm, e.g. the discounting parameter for lexicon smoothing. The statistics for the number of sure and possible connections in the manual reference alignments is also given in Table 3.1.

We use the same training schemes (model sequences) as presented in [Och & Ney 03]: $1^5 H^5 3^3 4^3 6^3$ for the Verbmobil task, i.e. 5 iteration of IBM-1, 5 iterations of the HMM, 3 iteration of IBM-3, . . . ; for the Canadian Hansards task, we use $1^5 H^{10} 3^3 4^3 6^3$. If not

Table 3.2: AER[%] of lexicon symmetrization methods for the Verbmobil and the Canadian Hansards (S→T: source-to-target direction, T→S: target-to-source direction).

	Verbmobil				Canadian Hansards			
Corpus size:	0.5 k		34 k		0.5 k		128 k	
	S→T	T→S	S→T	T→S	S→T	T→S	S→T	T→S
Baseline	16.4	21.5	5.7	10.0	25.9	27.5	12.5	12.4
Lin.	15.7	18.1	5.2	8.0	26.7	26.2	10.9	10.1
+ refinements	13.9	17.8	4.5	8.2	21.5	22.9	9.9	9.8
+ priors init.	13.8	17.5	4.6	8.3	21.4	22.9	9.6	9.9
Log-lin.	16.1	17.8	9.4	11.9	22.6	25.5	12.9	14.6
+ refinements	13.3	15.3	5.1	7.7	19.9	19.7	8.6	8.4
+ priors init.	12.8	15.2	5.2	7.7	19.9	19.9	8.6	8.3

stated otherwise, the conventional dictionary is not used. As we use the same training and testing conditions as [Och & Ney 03], we will refer to the results presented in that article as the baseline results.

3.8.3 Lexicon symmetrization

In Table 3.2, we present the following experiments performed for both the Verbmobil task and the Canadian Hansards task:

- **Lin.:** linear interpolation of the lexicon counts after each training iteration as described in Section 3.4.1.
- **Log-lin.:** log-linear interpolation of the lexicon counts as described in Section 3.4.2.
- **refinements** of both interpolation methods using evidence trimming as described in Section 3.4.3.
- **priors init.:** training initialization with external priors trained on the full monolingual corpora as described in Section 3.4.4.

We see that the refined evidence trimming results in an improvement of the AER in most cases, especially for the log-linear interpolation. As expected, the initialization with external priors improves the alignment quality on the small Verbmobil corpus. In the following, we will compare our experimental results to the state-of-the-art system of [Och & Ney 03]. From the different variants in Table 3.2, we chose for both tasks the systems with the best AER on the development corpus, i.e. on the small corpora the systems with improved initialization using external priors and on the large corpora the systems with the refined evidence trimming. For notational convenience, we will refer to these systems in the tables from now on with Lin. and Log-lin., respectively.

In Table 3.2, we also compare both interpolation variants for the Verbmobil task to [Och & Ney 03]. We observe notable improvements in the AER using the linear interpolation. A reduction of the AER is achieved for both the small and the full corpus. For the translation direction German → English (S→T) on the full corpus, an improvement of more than 20% relative is achieved from 5.7% for the baseline system to 4.5% using the linear interpolation. Using the log-linear interpolation, we also observe a substantial reduction of the AER, especially on the small corpus, e.g., a relative improvement of 22% for the German → English (S→T) direction.

The linear interpolation outperforms the log-linear interpolation on the full Verbmobil corpus. This can be explained as follows. The reference alignments for this task contain a large number of sure (S) alignment points, averaging at 0.79 points per German word[c]. Thus, the automatical alignment methods producing many (reliable) alignment points are more favorable for this task. This is exactly the case for the linear interpolation, where reliable counts are boosted by the count symmetrization. Generally, for the linear interpolation holds (*): *if two values which differ by a high order of a magnitude are interpolated, the result is of the same magnitude as the larger of the two values.* In our case, this causes a substantial number of symmetric lexicon counts significantly larger than 0, and leads to a final Viterbi alignment with more reliable alignment points.

Investigating the effect of the log-linear interpolation, for which the statement (*) does not hold, we arrive at the conclusion that the log-linear count interpolation results in fewer lexicon counts which are significantly larger than zero. Only a few very reliable alignment points “survive” the training, resulting in higher precision and lower recall error rates. In case of the Verbmobil task, where the reference alignments are not sparse, this is not the best strategy.

However, the lexicon model symmetrization with log-linear interpolation is appropriate for the training on the small corpus. Many alignment points produced by this training are not reliable, causing a low alignment precision. The log-linear interpolation reduces the number of large lexicon counts significantly. So, in the end only a few very reliable alignment points are produced. This happens at the expense of the recall, but still improves the AER.

We also performed word alignment experiments on Verbmobil (sub)corpora of different sizes. The results support the previous observations. With increasing training corpus size the linear interpolation outperforms the log-linear interpolation. This is illustrated in Figure 3.4. We observe that both symmetrization variants result in improvements for all corpus sizes over the baseline model.

In Table 3.2, we also compare the symmetrization methods with the baseline system for the Canadian Hansards task. Here, the log-linear interpolation with evidence trimming performs best. For the full corpus, we achieve a relative improvement over the baseline of more than 30% for both translation directions.

For the Canadian Hansards task, the log-linear interpolation performs better than the linear interpolation even for the large corpus. We explain this as follows. In Table 3.1,

[c] This may be attributed to the differences between German and English which demand frequent one-to-many and/or many-to-one alignment mappings.

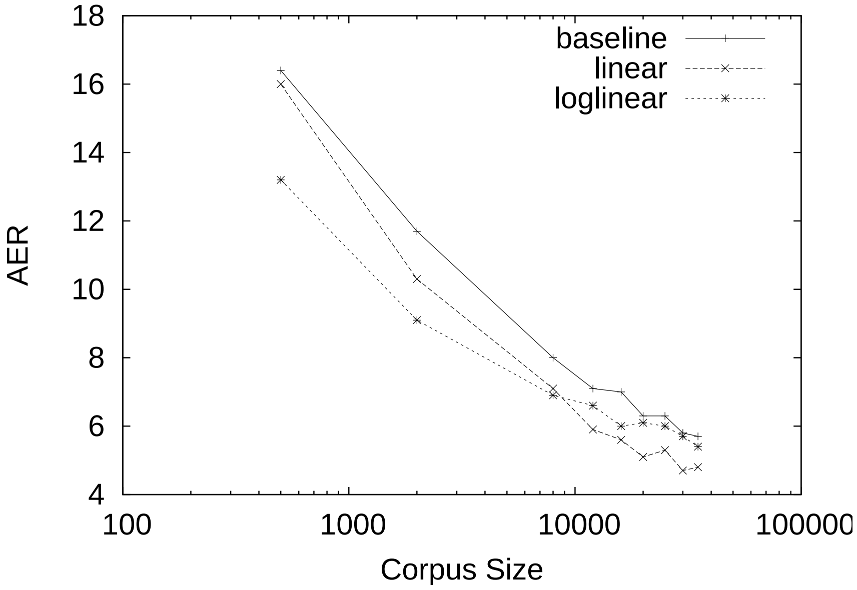


Figure 3.4: AER[%] of different alignment methods as a function of the training corpus size for the Verbmobil task (source-to-target direction).

we see that the reference alignments for the test corpus contain many possible (P) points and only a few sure (S) alignment points. There is an average of 0.5 alignment points per French word. Thus, reducing the number of automatically produced alignment points is more often favorable than not, because missing a possible point does not count as an error.

The improvements over the baseline in Table 3.2 are statistically significant at the 99% level. This was checked with a pairwise test using bootstrap resampling [Bisani & Ney 04].

Alignment examples for the standard training and the training with the symmetrized lexicon are shown in Figure 3.5. We see that in the standard training (on the left hand side), the German word 'Dias' has multiple incorrect alignment links, whereas using the symmetrized lexicon, there is only the correct alignment link to the English word 'slides'.

3.8.4 Generalized alignments

In [Och & Ney 03] generalized alignments are used, thus, the final Viterbi alignments of both translation directions are combined using some heuristic. Experimentally, the best heuristic for the Canadian Hansards task is the intersection. This again confirms that producing fewer alignment points will result in a reduced AER on the Canadian Hansards task. For the Verbmobil task, the refined method of [Och & Ney 03] is used. To obtain comparable results to [Och & Ney 03], we added a conventional dictionary to the bilingual training corpus. The results are summarized in Table 3.3. We see that both lexicon symmetrization methods yield an improvement with respect to the AER. Additionally, we observe that precision and recall are more balanced for the symmetrized lexicon variants, especially for the Canadian Hansards task.

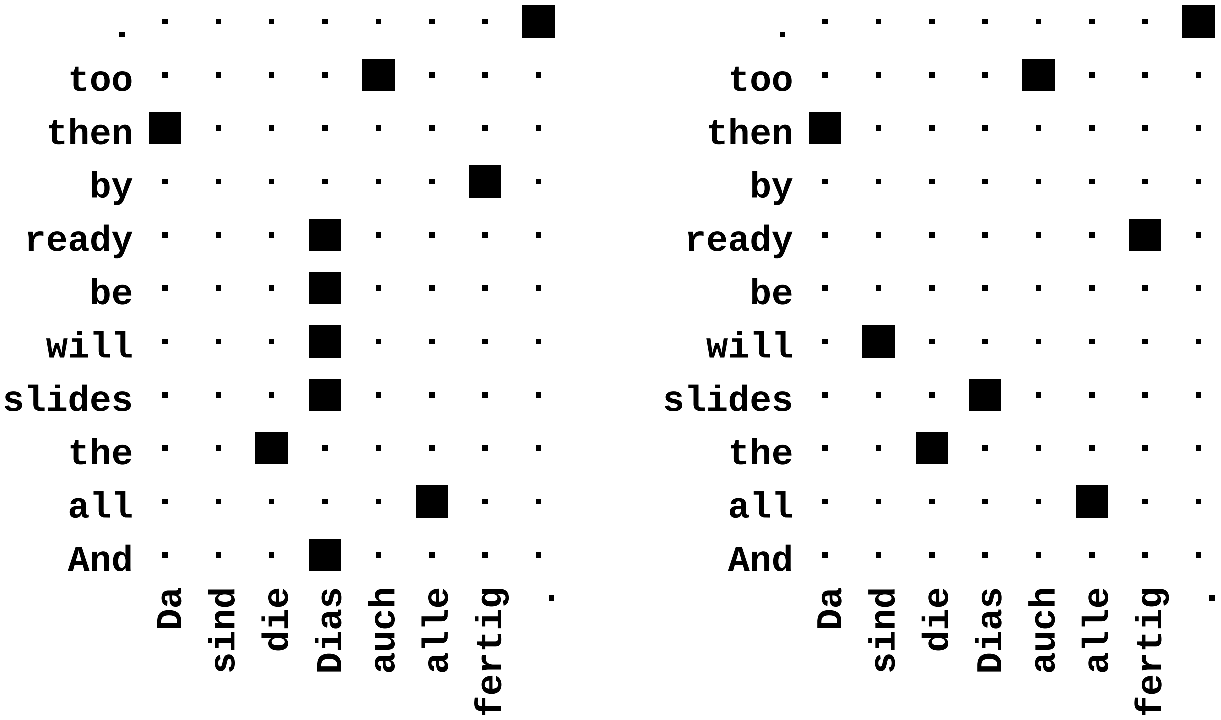


Figure 3.5: Verbmobil task: alignment examples for the standard training (left) and the training with symmetrized lexicon (right).

Table 3.3: Effect of different lexicon symmetrization methods on precision, recall and the AER for the generalized alignments for the Verbmobil task (VM) and the Canadian Hansards task (CH).

Corpus:	VM 0.5 k + dictionary			CH 0.5 k + dictionary		
Combination:	refined			intersection		
	Precision[%]	Recall[%]	AER[%]	Precision[%]	Recall[%]	AER[%]
Baseline	87.8	92.9	9.9	91.5	71.3	18.7
Lin.	90.9	92.8	8.2	86.4	82.7	15.2
Log-lin.	90.2	92.3	8.8	84.5	84.2	15.6

3.8.5 Lexicon smoothing

In Table 3.4, we present the results for the lexicon smoothing as described in Section 3.5 on the large Verbmobil corpus[d]. As expected, a notable improvement in the AER is reached if the lexicon smoothing is performed for German, because many full-form words with the same stem are present in this language. These improvements are statistically significant at the 95% level. As expected the effect of smoothing is small or there is no

[d] The word stems were determined using LingSoft tools.

Table 3.4: Effect of smoothing the lexicon probabilities on the AER[%] for the Verbmobil task (S→T: source-to-target direction, smooth English; T→S: target-to-source direction, smooth German).

	Verbmobil			
	34 k		34 k + dictionary	
	S→T	T→S	S→T	T→S
Baseline	5.7	10.0	4.6	8.8
+ Smoothing	5.2	9.3	4.6	8.0

effect at all if it is applied to the English part.

3.8.6 Non-symmetric alignments

In this section, we will compare the different non-symmetric alignment algorithms. We use the state occupation probabilities from only one translation direction to determine the word alignment. This allows for a fair comparison with the Viterbi alignment computed as the result of the standard training procedure. In the source-to-target translation direction, we cannot estimate the probability for the target words with fertility zero and choose to set it to 0. In this case, the minimum weight edge cover problem is solved by the one-sided MWEC algorithm. Like the Viterbi alignments of the standard models, the alignments produced by this algorithm satisfy the constraint that multiple source (target) words can only be aligned to one target (source) word.

In Table 3.5. we show the performance of the one-sided MWEC algorithm in comparison with the experiment reported by [Och & Ney 03]. We report not only the final alignment error rates of Model 6, but also the intermediate results for the HMM and IBM-4 training schemes.

Consequently, we observe similar alignment quality when comparing the Viterbi and the one-sided MWEC alignments.

As we use the state occupation probabilities of the corresponding model, we observe similar alignment quality when comparing the Viterbi and the one-sided MWEC alignments.

We also evaluated the alignment quality after applying alignment generalization methods, i. e. we combine the alignment of both translation directions. Experimentally, the best generalization heuristic for the Canadian Hansards task is the intersection of the source-to-target and the target-to-source alignments. For the Verbmobil task, the refined method of [Och & Ney 03] is used. Again, we observed similar alignment error rates when merging either the Viterbi alignments or the o-MWEC alignments.

3.8.7 Symmetric alignments

The heuristically generalized Viterbi alignments presented in the previous section can potentially avoid the alignment constraints. However, the choice of the optimal gener-

Table 3.5: AER [%] for non-symmetric alignment methods and for various models (HMM, IBM-4, Model 6) for the Verbmobil and Canadian Hansards tasks (generalization: refined for Verbmobil and intersection for Canadian Hansards).

		Verbmobil			Canadian Hansards		
Alignment method		HMM	IBM4	M6	HMM	IBM4	M6
T→S	Baseline	7.6	4.8	4.6	14.1	12.9	11.9
	o-MWEC	7.3	4.8	4.5	14.0	13.1	11.9
S→T	Baseline	12.1	9.3	8.8	14.4	12.8	11.7
	o-MWEC	12.0	9.3	8.5	14.3	13.0	11.7
Generalized	Baseline	7.1	4.7	4.7	8.4	6.9	7.8
	o-MWEC	6.7	4.6	4.6	8.2	7.1	7.8

alization heuristic may depend on a particular language pair and may require extensive manual optimization. In contrast, the symmetric MWEC algorithm is a systematic and theoretically well-founded approach to the task of producing a symmetric alignment. We will use the best heuristic of [Och & Ney 03] as baseline for the experiments with symmetric alignments.

In the experiments with the symmetric MWEC algorithm, the optimal interpolation parameter α (see Equation 3.32) for the Verbmobil corpus was empirically determined as 0.8. This shows that the model parameters can be estimated more reliably in the direction from German to English. In the inverse English-to-German training direction, the mappings of many English words to one German word are not allowed by the modeling constraints. This is problematic as such alignment mappings are significantly more frequent than mappings of many German words to one English word.

The experimentally best interpolation parameter for the Canadian Hansards corpus was $\alpha = 0.5$. Thus, the model parameters estimated in the translation direction from French to English are as reliable as the ones estimated in the direction from English to French.

Lines A (2) and B (2) of Table 3.6 show the performance of the MWEC algorithm. The alignment error rates are slightly lower if the HMM or the full Model 6 training scheme is used to train the state occupation probabilities on the Canadian Hansards task. On the Verbmobil task, the improvement is more significant, yielding an alignment error rate of 4.1%.

Additional experiments were performed with a log-linear combination of the state occupation probabilities of the HMM model and IBM model 4 (HMM+IBM4) as well as with Model 6 (HMM+M6). We set the interpolation parameters for the two translation directions proportional to the optimal values determined in the previous experiments. On the Verbmobil task, we obtain a further improvement of 19% relative over the baseline result reported in [Och & Ney 03], reaching an AER of 3.8%.

The improvements of the alignment quality on the Canadian Hansards task are less significant. The manual reference alignments for this task contain many possible connections

Table 3.6: AER[%] for different alignment symmetrization methods and for various alignment models on the Canadian Hansards and the Verbmobil tasks (MWEC: minimum weight edge cover, EW: empty word).

Canadian Hansards 128 K corpus + dictionary							
Symmetrization Method			Models				
			HMM	IBM4	M6	HMM + IBM4	HMM + M6
A	1	Baseline (intersection)	8.4	6.9	7.8	–	–
	2	MWEC	7.9	9.3	7.5	8.2	7.4
	3	+global EW costs	6.6	7.4	6.9	6.4	6.4
	4	o-MWEC T→S	7.3	7.9	7.4	6.7	7.0
	5	S→T	7.7	7.6	7.2	6.9	6.9
	6	S↔T (intersection)	7.2	6.6	7.6	6.0	7.1

Verbmobil 34 K corpus + dictionary							
Symmetrization Method			Models				
			HMM	IBM4	M6	HMM+IBM4	HMM+M6
B	1	Baseline (refined)	7.1	4.7	4.7	–	–
	2	MWEC	6.4	4.4	4.1	4.3	3.8
	3	+global EW costs	5.8	5.8	6.6	6.0	6.7
	4	o-MWEC T→S	6.8	4.4	4.1	4.5	3.7
	5	S→T	9.3	7.2	6.8	7.5	6.9
	6	S↔T (refined)	6.7	4.3	4.1	4.6	3.7

and only a few sure connections (cf. Table 3.1). Thus, automatic alignments consisting of only a few reliable alignment points are favored. Because the differences in the number of words and word order between French and English are not as dramatic as, e.g., between German and English, the probability of the empty word alignment is not very high. Therefore, plenty of alignment points are produced by the MWEC algorithm, resulting in a high recall and lower precision. To increase the precision, we replaced the empty word connection costs (previously trained as state occupation probabilities using the EM algorithm) by a global, word- and position-independent costs depending only on one of the involved languages. The alignment error rates for these experiments are given in lines A (3) and B (3) of Table 3.6. The global empty word probability for the Canadian Hansards task was empirically set to 0.45 for French and for English, and, for the Verbmobil task, to 0.6 for German and 0.1 for English. On the Canadian Hansards task, we achieved further significant reduction of the AER. In particular, we reached an AER of 6.6% by performing only the HMM training. In this case the effectiveness of the MWEC algorithm is combined with the efficiency of the HMM training, resulting in a fast and robust alignment training procedure.

We also tested the simpler one-sided MWEC algorithm. In contrast to the experiments presented in Section 3.8.6, we used the log-linear interpolated state occupation probabilities (given by the Equation 3.32) as costs. Thus, although the algorithm is not able to produce a symmetric alignment, it operates with symmetrized costs. In addition, we used a combination heuristic to obtain a symmetric alignment. The results of these experiments are presented in Table 3.6, lines A/B (4)-(6).

The performance of the one-sided MWEC algorithm turned out to be quite robust on both tasks. However, the o-MWEC alignments are not symmetric and the achieved low AER depends heavily on the differences between the involved languages, which may favor many-to-one alignments in one translation direction only. For example, the alignment quality deteriorates for the English to German Verbmobil task, because the algorithm cannot produce alignments which map several English words to one German word, see line B (5) of Table 3.6.

Applying the generalization heuristics (line A/B (6) of Table 3.6), we achieve an AER of 6.0% on the Canadian Hansards task when interpolating the state occupation probabilities trained with the HMM and with the IBM-4 schemes. On the Verbmobil task, the interpolation of the HMM and the Model 6 schemes yields the best result of 3.7% AER. In the latter experiment, we reached 97.3% precision and 95.2% recall.

3.9 Conclusions

We have addressed the task of automatically generating word alignments for bilingual corpora. This problem is of great importance for many tasks in natural language processing, especially in the field of machine translation.

We have presented lexicon symmetrization methods for the five IBM and the HMM translation models. We have evaluated these methods on the Verbmobil task and the Canadian Hansards task and compared our results to the state-of-the-art system of [Och & Ney 03]. We have shown that both the linear and the log-linear interpolation of the lexicon counts after each iteration of the EM-algorithm result in statistically significant improvements of the alignment quality. For the Canadian Hansards task, the AER improved by about 30% relative.

Additionally, we presented an algorithm for directly generating symmetric alignments. We exploited state occupation probabilities derived from the IBM and HMM translation models. We used the negated logarithms of these probabilities as local alignment costs and reduced the word alignment problem to the graph problem of finding an minimum weight edge cover. We presented efficient algorithms to solve this problem and evaluated their performance. An advantage of this alignment algorithm is that it is straightforward to include additional features. Furthermore it is guaranteed to find the optimal solution. Thus, we do not have to worry about approximations such as the hill climbing for the fertility-based IBM models. We showed that interpolating the alignment costs of the source-to-target and the target-to-source translation directions can result in a significant improvement of the alignment quality.

We also performed translation experiments using the improved word alignment models. Although we achieved significant improvements in terms of alignment error rate, the effect on translation quality is minimal. Similar results were obtained by [Vilar & Popović$^{+}$ 06, Ayan & Dorr 06, Fraser & Marcu 07]. One can conclude that the correlation between

AER and MT quality, e. g. measured with the Bleu score, is not strong enough. One reason might be that the phrase-based approach can compensate alignment errors to some extend. So, in the future, we will either need better metrics for measuring alignment quality or we will have to tune directly w.r.t. translation quality. The latter is of course very time consuming. An attempt for a better metric is the consistent phrase error rate of [Ayan & Dorr 06], which measures the overlap of phrases in the generated alignment and the reference alignment. As these phrases are the elementary units of a phrase-based SMT system, this idea seems to be promising. Unfortunately, it turned out that even this metric does not correlate well enough with MT quality.

4 Phrase-based Translation

As described in Section 1.2.1, we have to address three problems [Ney 01]:

- the ***modeling problem***, i. e. how to structure the dependencies of source and target language sentences;

- the search problem, i. e. how to find the best translation candidate among all possible target language sentences;

- the training problem, i. e. how to estimate the free parameters of the models from the training data.

In this chapter, the main focus is on the modeling problem.

As mentioned in the introduction, one core component of a phrase-based SMT system are the bilingual phrase-pairs and their associated translation probability. In this chapter, we will describe several models to compute the phrase translation score. We will distinguish two types of models.

First, there are the phrase models. These depend on a single phrase-pair and do not take the context outside the phrase pair into account. Thus, there are no dependencies across phrase boundaries. These models will be described in Section 4.3. The second type are the reordering models. These models do have dependencies across phrase boundaries and, thus, take the context into account. These models will be described in Section 4.4.

4.1 Motivation

One major disadvantage of single-word based (SWB) approaches is that contextual information is not taken into account. The lexicon probabilities are based only on single words. For many words, the translation depends heavily on the surrounding words. In the single-word based translation approach, this disambiguation is addressed by the language model only, which is often not capable of doing this. An example is shown in Figure 4.1.

One way to incorporate the context into the translation model is to learn translations for whole phrases instead of single words. Here, a phrase is simply a sequence of words. So, the basic idea of phrase-based (PB) translation is to segment the given source sentence into phrases, then translate each phrase and finally compose the target sentence from these phrase translations as illustrated in Figure 4.2. As seen in the last phrase pair of the example, punctuation marks are treated as normal words.

Figure 4.1: Translation example.

Source	was halten Sie vom Hotel Gewandhaus ?
Reference	what do you think about the hotel Gewandhaus ?
SWB	what do you from the hotel Gewandhaus ?
PB	what do you think of hotel Gewandhaus ?

Figure 4.2: Example for phrase-based translation.

SOURCE: abends würde ich gerne entspannen und vielleicht in die Sauna gehen .

source segmentation	translation
abends	in the evening
würde ich gerne entspannen	I would like to relax
und	and
vielleicht in die Sauna gehen	maybe go to the sauna
.	.

TARGET: in the evening I would like to relax and maybe go to the sauna .

4.2 Phrase Extraction

The system somehow has to learn which phrases are translations of each other. Therefore, we use the following approach: first, we train statistical alignment models using GIZA++ and compute the Viterbi word alignment of the training corpus. This is done for both translation directions. We combine both alignments using a refined heuristic to obtain a symmetrized word alignment matrix. This alignment matrix is the starting point for the phrase extraction. The following criterion defines the set of bilingual phrases $\mathcal{BP}$ of the sentence pair $(f_1^J; e_1^I)$ and the alignment matrix $A \subseteq J \times I$ that is used in the translation system.

$$\begin{aligned}\mathcal{BP}(f_1^J, e_1^I, A) = \Big\{ \left(f_{j_1}^{j_2}, e_{i_1}^{i_2}\right) &: \forall (j,i) \in A : j_1 \leq j \leq j_2 \leftrightarrow i_1 \leq i \leq i_2 \\ &\wedge \exists (j,i) \in A : j_1 \leq j \leq j_2 \wedge i_1 \leq i \leq i_2 \Big\} \end{aligned} \tag{4.1}$$

This criterion is identical to the alignment template criterion described in [Och & Tillmann$^+$ 99]. It means that two phrases are considered to be translations of each other, if the words are aligned only within the phrase pair and not to words outside. The phrases have to be contiguous. Figure 4.3 is an example of a word aligned sentence pair. Figure 4.4 shows the bilingual phrases extracted from this sentence pair according to the defined criterion. More details on the phrase extraction can be found in [Och 02].

Figure 4.3: Word aligned sentence pair.

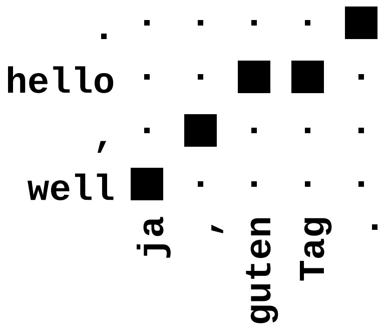


Figure 4.4: Extracted bilingual phrases.

source phrase	target phrase
ja	well
ja,	well,
ja, guten Tag	well, hello
ja, guten Tag.	well, hello.
,	,
, guten Tag	, hello
, guten Tag.	, hello.
guten Tag	hello
guten Tag.	hello.
.	.

4.3 Phrase Models

As described in Section 1.2.2, we use a log-linear combination of several models (also called feature functions). In this section, we will describe the models that are used in the first pass, i. e. during the search. Not all these models are proper probability distributions. Some are just real valued features which seem (and maybe are) useless on their own. Nevertheless, they can be an important ingredient of the overall system.

As we will use the notation that was introduced in Section 1.2.3, we will repeat the key elements as a reminder. We have a segmentation s_1^K of a sentence pair (f_1^J, e_1^I) into K phrase pairs. Each $s_k = (i_k; b_k, j_k)$ is a triple consisting of the last position i_k of the k^{th} target phrase $\tilde{e}_k$ and the start and end positions of the k^{th} source phrase $\tilde{f}_k$ are b_k and j_k, respectively:

$$\tilde{e}_k := e_{i_{k-1}+1} \dots e_{i_k} \tag{4.2}$$

$$\tilde{f}_k := f_{b_k} \dots f_{j_k} \tag{4.3}$$

4.3.1 Phrase-based model

The phrase-based translation model is the main component of our translation system. The hypotheses are generated by concatenating target language phrases. Here, a phrase is simply a contiguous sequence of words. The pairs of source and corresponding target phrases are extracted from the word-aligned bilingual training corpus. The phrase extraction algorithm is described in Section 4.2. The main idea is to extract phrase pairs that are consistent with the word alignment. Thus, the words of the source phrase are aligned only to words in the target phrase and vice versa.

We use relative frequencies to estimate the phrase translation probabilities:

$$p(\tilde{f}|\tilde{e}) = \frac{N(\tilde{f}, \tilde{e})}{N(\tilde{e})} \tag{4.4}$$

Here, the number of co-occurrences of a phrase pair $(\tilde{f}, \tilde{e})$ that are consistent with the word alignment is denoted as $N(\tilde{f}, \tilde{e})$. If one occurrence of a target phrase $\tilde{e}$ has $N > 1$ possible translations, each of them contributes to $N(\tilde{f}, \tilde{e})$ with $1/N$. A target phrase can have multiple consistent source phrases if there are non-aligned words at the boundary of the source phrase. The reason for using the fractional count $1/N$ is to penalize these ambiguous occurrences. The marginal count $N(\tilde{e})$ is the number of occurrences of the target phrase $\tilde{e}$ in the training corpus. Note that $N(\tilde{e}) \geq \sum_{\tilde{f}} N(\tilde{f}, \tilde{e})$, because there might be occurrences of the target phrase with no consistent source phrase. These contribute to the marginal count $N(\tilde{e})$, but not to the joint count $N(\tilde{f}, \tilde{e})$. The resulting feature function is:

$$h_{\text{Phr}}(e_1^I, s_1^K; f_1^J) \quad = \quad \sum_{k=1}^{K} \log p(\tilde{f}_k | \tilde{e}_k) \tag{4.5}$$

To obtain a more symmetric model, we use the phrase-based model in both directions $p(\tilde{f}|\tilde{e})$ and $p(\tilde{e}|\tilde{f})$. The inverse feature function is:

$$h_{\text{iPhr}}(e_1^I, s_1^K; f_1^J) \quad = \quad \sum_{k=1}^{K} \log p(\tilde{e}_k | \tilde{f}_k) \tag{4.6}$$

We use the same algorithm as for the standard direction except that source and target side are switched. Note that because of the treatment of non-aligned words, the joint counts for the two directions are not necessarily identical.

The relative frequency estimates typically overestimate the probabilities of rare events as most of the longer phrases occur only once in the training corpus. To overcome this problem, we will use additional models to smooth the phrase translation probabilities. These will be described in the following sections.

4.3.2 Phrase-count model

The phrase-count model can explicitly penalize rare phrase pairs. The idea is to add a certain penalty, if the joint count of a phrase pair $N(\tilde{f}, \tilde{e})$ is below a threshold τ:

$$h_{\text{C},\tau}(e_1^I, s_1^K; f_1^J) \quad = \quad \sum_{k=1}^{K} [N(\tilde{f}_k, \tilde{e}_k) \leq \tau] \tag{4.7}$$

The corresponding model scaling factor acts as a penalty for each rare phrase pair. We use $[\cdot]$ to denote a true or false statement [Graham & Knuth+ 94], i. e. the result is 1 if the statement is true, and 0 otherwise. In general, we use the following convention:

$$[\,\mathcal{C}\,] = \begin{cases} 1, & \text{if condition } \mathcal{C} \text{ is true} \\ 0, & \text{if condition } \mathcal{C} \text{ is false} \end{cases} \tag{4.8}$$

Note that this model can be used with different thresholds. In the experiments, we typically use this model with thresholds $1, 2$ and 3, each with its own model scaling factor.

4.3.3 Word-based lexicon model

In this section and the following ones, we use a word-based lexicon to compute the phrase translation probabilities. For the standard direction, these have the form:

$$h(e_1^I, s_1^K; f_1^J) = \sum_{k=1}^{K} \sum_{j=b_k}^{j_k} \log p(f_j|\tilde{e}_k) \tag{4.9}$$

The models differ in the definition of $p(f|\tilde{e})$. The inverse model $p(e|\tilde{f})$ is defined in a similar way.

In the first variant, the word-based lexicon is used to compute a score of a phrase pair similar to the IBM model 1. But here, we are summing only within a phrase pair and not over the whole target language sentence. Furthermore, we just use the lexical probabilities and omit the (uniform) alignment probabilities.

$$h_{\text{Lex}}(e_1^I, s_1^K; f_1^J) \quad = \quad \sum_{k=1}^{K} \sum_{j=b_k}^{j_k} \log \left(p(f_j|e_0) + \sum_{i=i_{k-1}+1}^{i_k} p(f_j|e_i) \right) \tag{4.10}$$

Here, e_0 denotes the empty word. We use the IBM model 4 lexicon trained with GIZA++ as word translation probabilities $p(f|e)$. The word-based lexicon model is also used in both directions $p(f|e)$ and $p(e|f)$. The inverse feature function is:

$$h_{\text{iLex}}(e_1^I, s_1^K; f_1^J) \quad = \quad \sum_{k=1}^{K} \sum_{i=i_{k-1}+1}^{i_k} \log \left(p(e_i|f_0) + \sum_{j=b_k}^{j_k} p(e_i|f_j) \right) \tag{4.11}$$

4.3.4 Word-based noisy-or model

The word-based noisy-or models a disjunctive interaction, also called noisy-OR gate [Pearl 88]. The idea is that there are multiple independent causes, in our case the target words e, that can generate an event f. It can be easily integrated into the search algorithm. The corresponding feature function is:

$$h_{\text{Nor}}(e_1^I, s_1^K; f_1^J) \quad = \quad \sum_{k=1}^{K} \sum_{j=b_k}^{j_k} \log \left(1 - \prod_{i=i_{k-1}+1}^{i_k} (1 - p(f_j|e_i)) \right) \tag{4.12}$$

This model can be used in both directions $p(f|e)$ and $p(e|f)$. The inverse feature function is:

$$h_{\text{iNor}}(e_1^I, s_1^K; f_1^J) \quad = \quad \sum_{k=1}^{K} \sum_{i=i_{k-1}+1}^{i_k} \log \left(1 - \prod_{j=b_k}^{j_k} (1 - p(e_i|f_j)) \right) \tag{4.13}$$

4.3.5 Deletion model

[Och & Gildea+ 03] presented a deletion model in a rescoring/reranking framework. It is designed to penalize hypotheses that miss the translation of a word. For each source word, we check if a target word with a probability higher than a given threshold τ exists. If not, this word is considered a deletion. The model simply counts the number of deletions.

Here, we will use a within-phrase variant of the deletion model and we will use it already during the search.

$$h_{\mathrm{Del}}(e_1^I, s_1^K; f_1^J) \quad = \quad \sum_{k=1}^{K} \sum_{j=b_k}^{j_k} \prod_{i=i_{k-1}+1}^{i_k} \left[\, p(f_j|e_i) < \tau \,\right] \tag{4.14}$$

The word translation probabilities $p(f|e)$ are the same as for the word-based lexicon model, i. e. the GIZA++ IBM model 4 lexicon. Here, $[\cdot]$ denotes a true or false statement as in Equation 4.8. The deletion model is used in both directions. The inverse feature function is:

$$h_{\mathrm{iDel}}(e_1^I, s_1^K; f_1^J) \quad = \quad \sum_{k=1}^{K} \sum_{i=i_{k-1}+1}^{i_k} \prod_{j=b_k}^{j_k} \left[\, p(f_j|e_i) < \tau \,\right] \tag{4.15}$$

4.3.6 Word and phrase penalty model

In addition, we use two simple heuristics, namely a word penalty and a phrase penalty:

$$h_{\mathrm{WP}}(e_1^I, s_1^K; f_1^J) \quad = \quad I \tag{4.16}$$
$$h_{\mathrm{PP}}(e_1^I, s_1^K; f_1^J) \quad = \quad K \tag{4.17}$$

These two models affect the average sentence and phrase lengths. The model scaling factors can be adjusted to prefer longer sentences and longer phrases.

The word penalty is simply the target phrase length. In combination with the scaling factor this results in a constant cost per produced target language word. With this feature, we are able to adjust the sentence length. If we set a negative scaling factor, longer sentences are more penalized than shorter ones, and the system will favor shorter translations. Alternatively, by using a positive scaling factor, the system will favor longer translations.

Similar to the word penalty, the phrase penalty results in a constant cost per produced phrase. The phrase penalty is used to prefer either fewer and thus longer phrases or more and thus shorter phrases.

In practice, the word ”penalty” is typically a bonus per word to prefer longer translations. As the sentence length is important for the Bleu measure, these two models are, despite their simplicity, quite important to achieve good Bleu scores.

The word penalty was already proposed in [Brown & Della Pietra+ 93] to counteract the general preference for shorter translations.

4.4 Reordering Models

In this section, we will describe the models that take dependencies across phrase boundaries into account. These models are especially important for choosing good reorderings.

4.4.1 Distortion penalty model

The reordering model of the baseline system is distance-based, i. e. it assigns costs based on the distance (also called distortion) from the end position of a phrase to the start position of the next phrase. This very simple reordering model is widely used, for instance in [Och & Tillmann$^+$ 99, Koehn 04a, Och & Ney 04].

$$h_{\text{Dist}}(e_1^I, s_1^K; f_1^J) \quad = \quad \sum_{k=1}^{K+1} q_{\text{Dist}}(b_k, j_{k-1}) \tag{4.18}$$

with

$$q_{\text{Dist}}(j, j') = |j - j' + 1| \tag{4.19}$$

The distortion penalty model assigns costs of zero to a translation which is monotonic at the phrase level. The more phrases are reordered, the higher the distortion penalty. As we have defined $b_{K+1} = J + 1$ in Equation 1.21, the sum includes a jump from the last position of the final phrase to a position "one after the sentence end".

Often it is combined with a limit D on the jump width:

$$q_{\text{Dist}}(j, j') = \begin{cases} |j - j' - 1| & \text{if} |j - j' - 1| < D \\ \infty & \text{else} \end{cases} \tag{4.20}$$

4.4.2 Language model

In addition to the distortion penalty model, a standard n-gram language model (LM) is used. The purpose of the language model is to ensure the well-formedness of the target sentence. Despite that an n-gram LM takes only local context into account, it turns out to be quite powerful and hard to improve over. The language model requires only monolingual training data; thus, compared to the translation models, it can be trained on significantly larger volumes of data. We use the SRILM toolkit library [Stolcke 02]. Smoothing is done using modified Kneser-Ney discounting [Kneser & Ney 95, Chen & Goodman 98].

$$h_{\text{LM}}(e_1^I, s_1^K; f_1^J) \quad = \quad \sum_{i=1}^{I+1} \log p(e_i | e_{i-n+1}^{i-1}) \tag{4.21}$$

The language model includes a sentence end probability, which is the $(I + 1)^{\text{th}}$ term in the sum as we have defined e_{I+1} as the sentence boundary marker.

4.4.3 Phrase orientation model

4.4.3.1 Introduction

In [Och & Gildea$^+$ 04], a lexicalized reordering model was shown to improve translation quality. Recently, in [Tillmann & Zhang 05, Koehn & Axelrod$^+$ 05], a variant of this reordering model has been described that tries to predict the orientation of a phrase, i. e. it answers the question 'should the next phrase be to the left or to the right of the current phrase?' This phrase orientation probability is conditioned on the current source and target phrase and relative frequencies are used to estimate the probabilities, i. e. the model is lexicalized.

We adopt the idea of predicting the orientation, but we propose to use a log-linear model. The relative-frequency based approach may suffer from the data sparseness problem, because most of the phrases occur only once in the training corpus. Our approach circumvents this problem by using a combination of phrase-level and word-level features and by using word-classes or part-of-speech information. Maximum entropy is a suitable framework for combining these different features with a well-defined training criterion.

In [Koehn & Axelrod$^+$ 05] several variants of the orientation model have been tried. It turned out that for different tasks, different models show the best performance. Here, we let the maximum entropy training decide which features are important and which features can be neglected. We will see that additional features do not hurt performance and can be safely added to the model.

To make use of word level information, we need the word alignment within the phrase pairs. This can be easily stored during the extraction of the phrase pairs from the bilingual training corpus. If there are multiple possible alignments for a phrase pair, we use the most frequent one.

The notation is introduced using the illustration in Figure 4.5. There is an example of a left and a right phrase orientation. We assume that we have already produced the three-word phrase in the lower part. Now, the model has to predict if the start position of the next phrase j' is to the left or to the right of the current phrase. The reordering model is applied only at the phrase boundaries. We assume that the reordering within the phrases is correct. Note that in our approach the phrases do not have to be adjacent. If we limit the phrases to be adjacent, the resulting model is equivalent to the BTG model of [Wu 97]. Such an approach was used in [Xiong & Liu$^+$ 06].

The approach of [Ohashi & Yamamoto$^+$ 05] and the follow-up work [Nagata & Yamamoto$^+$ 06] is similar to our work as it predicts four types of distortion. This is equivalent to a setting of $D = 2$ in the following Section 4.4.3.2. The model were conditioned on the POS-level phrases and used relative frequencies to estimate the distortion probabilities. In contrast, we use a richer feature set and discriminative training to estimate the parameters.

In the remaining part of this section, we will describe the details of this reordering model. The classes our model predicts will be defined in Section 4.4.3.2. Then, the feature functions will be defined in Section 4.4.3.3. The training criterion and the training events of the log-linear model will be described in Section 4.4.3.4.

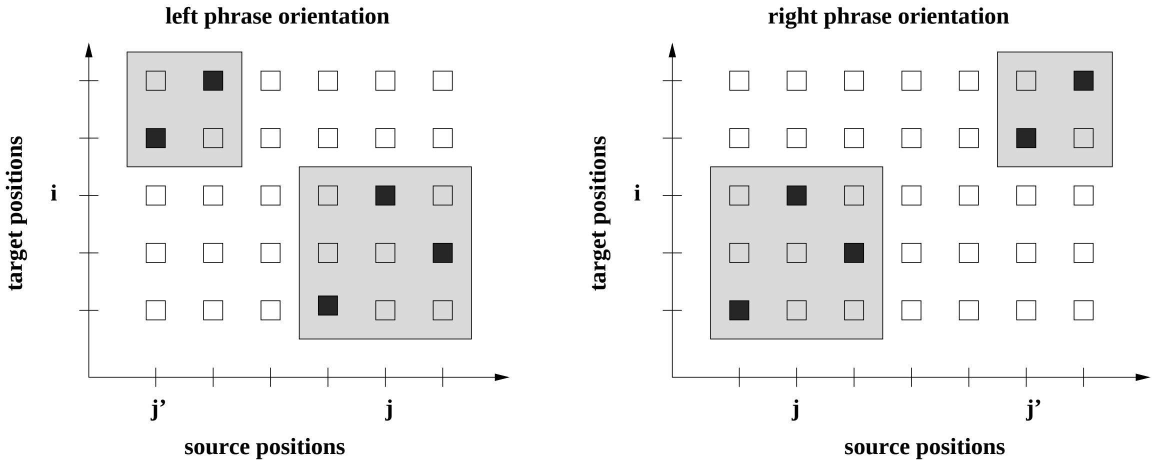


Figure 4.5: Illustration of the phrase orientation.

4.4.3.2 Class definition

Ideally, this model predicts the start position of the next phrase. But as predicting the exact position is rather difficult, we group the possible start positions into classes. In the simplest case, we use only two classes. One class for the positions to the left and one class for the positions to the right. As a refinement, we can use four classes instead of two: 1) one position to the left, 2) more than one positions to the left, 3) one position to the right, 4) more than one positions to the right.

In general, we use a parameter D to specify $2 \cdot D$ classes of the types:

- exactly d positions to the left, $d = 1, ..., D - 1$
- at least D positions to the left
- exactly d positions to the right, $d = 1, ..., D - 1$
- at least D positions to the right

Note that the case $d = 0$, i. e. to remain in the same position, is not possible as the model is applied only at phrase boundaries and the phrases do not overlap.

Let $c_{j,j'}$ denote the orientation class for a movement from source position j to source position j' as illustrated in Figure 4.5. In the case of two orientation classes, $c_{j,j'}$ is defined as:

$$c_{j,j'} = \begin{cases} \text{left}, & \text{if } j' < j \\ \text{right}, & \text{if } j' > j \end{cases} \tag{4.22}$$

Then, the reordering model has the form

$$p(c_{j,j'} | f_1^J, e_1^I, i, j)$$

A well-founded framework for directly modeling the probability $p(c_{j,j'} | f_1^J, e_1^I, i, j)$ is maximum entropy [Berger & Della Pietra+ 96]. In this framework, we have a set of N feature functions $h_n(f_1^J, e_1^I, i, j, c_{j,j'}), n = 1, \ldots, N$. Each feature function h_n is weighted with a factor θ_n. The resulting model is:

$$p_{\theta_1^N}(c_{j,j'}|f_1^J, e_1^I, i, j) \quad = \quad \frac{\exp\left(\sum\limits_{n=1}^{N} \theta_n h_n(f_1^J, e_1^I, i, j, c_{j,j'})\right)}{\sum\limits_{c'} \exp\left(\sum\limits_{n=1}^{N} \theta_n h_n(f_1^J, e_1^I, i, j, c')\right)} \tag{4.23}$$

The functional form is identical to Equation 1.8, but here we will use a large number of binary features, whereas in Equation 1.8 usually only a very small number of real-valued features is used. More precisely, the resulting reordering model $p_{\theta_1^N}(c_{j,j'}|f_1^J, e_1^I, i, j)$ is used as an additional component in the log-linear combination of Equation 1.8.

4.4.3.3 Feature definition

The feature functions of the reordering model depend on the last alignment link (j, i) of a phrase. Note that the source position j is not necessarily the end position of the source phrase. We use the source position j which is aligned to the last word of the target phrase in target position i. The illustration in Figure 4.5 contains such an example. In the three-word phrase, the second source word is aligned to the last target word.

To introduce generalization capabilities, some of the features will depend on word classes or part-of-speech information. Let $\mathrm{C}(f)$ and $\mathrm{C}(e)$ denote the word class of a source or target word, respectively. Then, the feature functions are of the form $h_n(f_1^J, e_1^I, i, j, j')$. We consider the following binary features:

1. source words within a window around the current source position j

$$h_{f,d,c}(f_1^J, e_1^I, i, j, j') \quad = \quad \delta(f_{j+d}, f) \cdot \delta(c, c_{j,j'})$$

2. target words within a window around the current target position i

$$h_{e,d,c}(f_1^J, e_1^I, i, j, j') \quad = \quad \delta(e_{i+d}, e) \cdot \delta(c, c_{j,j'})$$

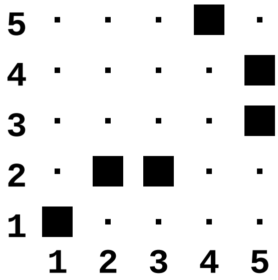
3. word classes or part-of-speech within a window around the current source position j

$$h_{F,d,c}(f_1^J, e_1^I, i, j, j') \quad = \quad \delta(\mathrm{C}(f_{j+d}), F) \cdot \delta(c, c_{j,j'})$$

4. word classes or part-of-speech within a window around the current target position i

$$h_{E,d,c}(f_1^J, e_1^I, i, j, j') \quad = \quad \delta(\mathrm{C}(e_{i+d}), E) \cdot \delta(c, c_{j,j'})$$

Here, $\delta(\cdot, \cdot)$ denotes the Kronecker-function. In the experiments, we will use $d \in \{-1, 0, 1\}$. Many other feature functions are possible, e. g. combinations of the described feature functions, n-gram or multi-word features, joint source and target language feature functions.



target position	source positions		class
i	j	j'	$c_{j,j'}$
0	0	1	right
1	1	2	right
2	3	5	right
4	5	4	left
5	4	6	right

Figure 4.6: Training of the lexicalized reordering model: illustration for the two-class case (positions 0 and 6 denote sentence boundaries).

4.4.3.4 Training

To train the log-linear parameters θ_1^N, we need labeled training data. We generate this from the word-aligned bilingual training corpus. We train IBM model 4 with GIZA++ in both translation directions and symmetrize the alignments using the refined heuristic as described in [Och & Ney 03]. Each alignment link defines an event for the maximum entropy training. An exception are the one-to-many alignments, i. e. one source word is aligned to multiple target words. In this case, only the topmost alignment link is considered because the other ones cannot occur at a phrase boundary. Many-to-one and many-to-many alignments are handled in a similar way. An example is shown in Figure 4.6, with the word alignment on the left hand side and the source and target positions used in the feature definitions in the previous section on the right hand side. Here, the positions 0 and 6 denote the sentence boundaries. For instance, if we consider the target position $i = 2$, we have a jump from source position $j = 3$ to source position $j' = 5$, i. e. we assume that a phrase ends with the alignment link (3,2) and the next phrase starts with the alignment link (5,3). Given this information, we can list the feature functions that are active for this pair. Note that per definition the feature functions are independent of the phrase boundaries on the source side. In the example, it would not matter if the next phrase pair is 4-5 on the source side and 3-5 on the target side or the phrase pair 5-5 on the source side and 3-4 on the target side. Therefore, we do *not* have to segment the training sentence pairs into phrases for training the orientation model

As training criterion, we use the maximum class posterior probability.

$$\hat{\theta}_1^N = \operatorname*{argmax}_{\theta_1^N} \left\{ \sum_{s=1}^{S} \log p_{\theta_1^N}(c_{j,j'}|f_1^J, e_1^I, i, j) \right\} \tag{4.24}$$

This corresponds to maximizing the likelihood of the log-linear model. Because the optimization criterion is convex, there is only a single optimum and no convergence problems occur. To train the model parameters θ_1^N, we use the Generalized Iterative Scaling (GIS) algorithm [Darroch & Ratcliff 72] as implemented in the YASMET toolkit [Och 01]. Alternative algorithms for tuning the parameters could be used instead; a comparison of suitable algorithms is given in [Malouf 02].

In practice, the training procedure often tends to result in an overfitted model. To avoid this, [Chen & Rosenfeld 99] have suggested a smoothing method where a Gaussian prior distribution of the parameters θ_1^N is assumed. This method tries to avoid very large lambda values and prevents features that occur only once for a specific class from getting a value of infinity.

5 Search

As described in Section 1.2.1, we have to address three problems [Ney 01]:

- the modeling problem, i. e. how to structure the dependencies of source and target language sentences;
- the ***search problem***, i. e. how to find the best translation candidate among all possible target language sentences;
- the training problem, i. e. how to estimate the free parameters of the models from the training data.

In this chapter, the main focus is on the search problem.

5.1 Introduction

The goal of the search is to find the maximizing argument in the Bayes decision rule. This is also referred to as generation or decoding[a]. Here, we are considering the MAP decision rule for the log-linear model (Equation 1.27):

$$\hat{e}_1^{\hat{I}} = \operatorname*{argmax}_{I,e_1^I} \left\{ \max_{s_1^K} \sum_{m=1}^{M} \lambda_m h_m(e_1^I, s_1^K; f_1^J) \right\} \tag{5.1}$$

We have to carry out a maximization over all possible target language sentences e_1^I and over all possible phrase segmentations s_1^K which includes the reordering of the phrases. As the models $h_m(e_1^I, s_1^K; f_1^J)$ can depend on the complete target sentence e_1^I and enumerating all target language sentences is infeasible, we are facing a hard optimization problem.

We can exploit the structure of the models described in Chapter 4.

We have to select

- the number of phrases K
- the segmentation of the source sentence into phrases and the permutation of the phrases
- the phrase translation $\tilde{e}$ of each source phrase $\tilde{f}$

[a] We will use these terms interchangeably.

We can interpret the search as a sequence of decisions $(\tilde{e}_k, b_k, j_k)$ for $k = 1, \ldots, K$. At each step we decide on a source phrase $\tilde{f}_k$ identified by its start and end positions (b_k, j_k) and the corresponding translation $\tilde{e}_k$. To ensure the constraints of the phrase segmentation, namely that there are no gaps and no overlap, we keep track of the set of source positions that are already translated ('covered'). We will call this the *coverage* set $C \subseteq \{1, \ldots, J\}$. We can represent the set of possible segmentations and translations as a graph where the arcs are labeled with the decisions $(\tilde{e}_k, b_k, j_k)$ and the states are labeled with the coverage sets C. The initial state of the graph is labeled with the empty coverage set $C = \emptyset$ meaning that no source word is yet translated. The goal state is labeled with the full coverage $C = \{1, \ldots, J\}$. Each path through this graph represents a possible translation of the source sentence, which is obtained by concatenating the target phrases $\tilde{e}$ along the path. Note that there are multiple paths representing the same translation with different phrase segmentations. The size of this graph is exponential in the length of the source sentence.

So far, we have not considered the model scores of a decision sequence. As mentioned earlier, the phrase models described in Section 4.3 do not have dependencies across phrase boundaries and thus can be computed for each phrase pair without context. In other words, these model scores do not depend on the decisions taken so far, but only on a single arc in the graph. We will use $q_{\text{TM}}(\tilde{e}_k, b_k, j_k)$ to denote the weighted sum of all phrase model scores of an arc $(\tilde{e}_k, b_k, j_k)$. In a typical setting, consisting of the phrase-based model in both directions, the lexicon model in both directions and the word and phrase penalty, this score is computed as

$$\begin{aligned} q_{\text{TM}}(\tilde{e}_k, b_k, j_k) &= \lambda_{\text{Phr}} \cdot \log p(\tilde{f}_k|\tilde{e}_k) + \lambda_{\text{iPhr}} \cdot \log p(\tilde{e}_k|\tilde{f}_k) \\ &\quad + \lambda_{\text{Lex}} \cdot \sum_{j=b_k}^{j_k} \log p(f_j|\tilde{e}_k) + \lambda_{\text{iLex}} \cdot \sum_{i=i_{k-1}+1}^{i_k} \log p(e_i|\tilde{f}_k) \\ &\quad + \lambda_{\text{WP}} \cdot (i_k - i_{k-1}) + \lambda_{\text{PP}} \end{aligned} \tag{5.2}$$

The reordering models, on the other hand, take the context outside the phrase pair into account. Thus, their model scores do depend on the decisions taken so far. Although in principal, these models could depend on the whole decision sequence, in practice, the models depend only on a small subset of the information. The n-gram language model, for example, depends only on the last $(n-1)$ words of the target sentence. The distortion penalty model depends only on the end position of the previous source phrase.

As it is a desirable property that the score of an arc should depend only on the arc itself and the adjacent states, we introduce copies of the states.

As we want to assign the costs to individual arcs, the score of an arc should depend only on the arc itself and the adjacent states. Therefore, we introduce copies of the states. We distinguish them according to the language model history and the end position of the last phrase. The language model history is required to compute the LM score of the expansions; the end position of the last phrase is required to compute the distortion model score of the expansions. We will use $\tilde{e}' \oplus \tilde{e}$ to denote the language model history after expanding the given history $\tilde{e}'$ with the phrase $\tilde{e}$. The score of this language model expansion weighted with language model scaling factor is denoted as $q_{\text{LM}}(\tilde{e}|\tilde{e}')$:

$$q_{\text{LM}}(\tilde{e}|\tilde{e}') = \lambda_{\text{LM}} \cdot \sum_{i=1}^{|\tilde{e}|} \log p(\tilde{e}^i|\tilde{e}^{i-1}, \ldots, \tilde{e}^1, \tilde{e}') \tag{5.3}$$

Here, $\tilde{e}^i$ denotes the i^{th} word of the phrase $\tilde{e}$. We define a similar auxiliary function for the distortion penalty model. We use $q_{\text{DM}}(j, j')$ to denote the weighted score of a jump from source position j to source position j':

$$q_{\text{DM}}(j, j') = \lambda_{\text{Dist}} \cdot |j - j' + 1| \tag{5.4}$$

An edge $(\tilde{e}, j, j')$ can occur multiple times in the search graph. To avoid repeated computations, we determine the set of possible target phrases for all source phrases before the actual search and store the target phrases along with their scores in a table $E(j, j')$.

The states in the search space can be identified by a triple $(C, \tilde{e}, j)$, where C denotes the coverage set, $\tilde{e}$ denotes the language model history, and j denotes the end position of the last source phrase. The language model history $\tilde{e}$ consists typically of the last $(n-1)$ words of the target hypothesis. As described in [Tillmann 01, Ueffing & Och$^+$ 02], we can exploit the backing-off structure of the language model and reduce this to the longest *observed* history. Thus, if the $(n-1)$-gram does not occur in the LM training data, we only have to keep the $(n-2)$-gram. If that does not occur as well, we can further reduce the history to the $(n-3)$-gram and so on. This will lead to an improved recombination and is especially helpful when high order n-gram language models are used. This method is supported by the SRILM toolkit with the `contextID` function.

Note that the language model history is usually *not* identical with the target phrase $\tilde{e}_k$ of the last decision $(\tilde{e}_k, b_k, j_k)$. It may be shorter if the phrase $\tilde{e}_k$ is longer than $n-1$ words. It may also include words from predecessor phrases $\tilde{e}_{k-1}, \tilde{e}_{k-2}, \ldots$ if $\tilde{e}_k$ is shorter than $n-1$ words.

The computation of the successor states of a given state $(C, \tilde{e}, j)$ is call hypothesis expansion. The score of such an expansion with a phrase pair $(\tilde{e}', j'', j')$ is computed as

$$q_{\text{TM}}(\tilde{e}', j'', j') + q_{\text{LM}}(\tilde{e}'|\tilde{e}) + q_{\text{DM}}(j, j'') \tag{5.5}$$

The successor state is $(C \cup \{j'', \ldots, j'\}, \tilde{e} \oplus \tilde{e}', j')$. Of course, we have to ensure that there is no overlap, i. e. that $C \cap \{j'', \ldots, j'\} = \emptyset$.

The search problem can be reformulated as finding the optimum path through this search graph. The size of the search graph is exponential in the source sentence length. It has been shown in [Knight 99, Udupa & Maji 06] that the search problem is NP-hard. Thus, we cannot expect to find the optimum solution efficiently in all cases. Therefore, we have to use approximations to find a solution efficiently.

The search algorithm should find a good solution, i. e. one that is close to the optimum, with limited computation time. Additionally, it should be possible to adjust the tradeoff between speed and quality. Thus, the should be no principal limitation of the search algorithm and if we spend enough computation time it should find the optimum solution.

Here, we use two techniques to achieve these goals:

- dynamic programming [Bellman 57]
- beam search [Jelinek 98]

The idea of dynamic programming is to first solve small subproblems and then use these solutions of the subproblems to compute a solution for the whole problem. Using dynamic programming, we can reduce the number of path that we have to explore in the search graph without giving up optimality. The idea of beam search is that at each step, we expand only the promising partial hypotheses and discard the hypotheses that are unlikely to lead to the optimum solution. In contrast to dynamic programming, beam search involves approximations and thus may result in suboptimal solutions.

We will use the following terms:

- **Lexical hypothesis** $(C, \tilde{e}, j)$. A lexical hypothesis is identified by a coverage C, a language model history $\tilde{e}$ and a source sentence position j.
- **Coverage hypothesis** C. We will use the term coverage hypothesis to refer to the set of all lexical hypotheses with the same coverage C.

The number of coverage hypotheses indicates how many alternative reorderings are investigated during the search. The number of lexical hypotheses is the total number of hypotheses. The number of lexical hypotheses per coverage hypothesis indicates the lexical alternatives that are taken into account.

As we will show in Section 5.2.1, the monotonic search problem can be solved quite efficiently. The drawback is that reordering can happen only within the phrases. If we want to permit reordering of phrases, the search problem becomes NP-hard, as shown for single-word based models in [Knight 99, Udupa & Maji 06]. The reordering problem in MT is similar to the traveling salesman problem and can therefore be solved with a variant of the Held-Karp algorithm [Held & Karp 62] as shown in [Tillmann & Ney 00]. The disadvantage is that the computational complexity is exponential in the sentence length. Therefore, an unconstrained search is often impractical. An alternative is to limit the reordering according to some reordering constraints. We will describe and compare several reordering constraints in Section 5.5.

The remaining part of this chapter is structured as follows: first we will describe search algorithms for text input and lattice input in Section 5.2 and Section 5.3, respectively. Afterward, we will describe the generation of word graphs and N-best lists in Section 5.4. Then, we will analyze different reordering constraints in Section 5.5. Finally, we will describe efficient data structures for the phrase table and a phrase matching algorithm for lattice input in Section 5.6 and Section 5.7, respectively.

5.2 Search for Text Input

5.2.1 Monotonic search

The monotonic search problem can be efficiently solved using dynamic programming. The resulting complexity is linear in the source sentence length. The segmentation is constrained such that both, the source sentence and the target sentence, are processed in monotonic order. As a result, we obtain:

$$b_k = j_{k-1} + 1 \ , \ k = 1, ..., K. \tag{5.6}$$

As there is no reordering of phrases, the distortion penalty model is not used for monotonic translation. Furthermore, the possible coverage sets are limited and of the form $\{1, \ldots, j\}$ which means that the source sentence has been translated up to position j. For the maximization problem in Equation 5.1, we define the quantity $Q(j, \tilde{e})$ as the maximum score of a phrase sequence that ends with the language model history $\tilde{e}$ and covers positions 1 to j of the source sentence. $\hat{Q}$ is the score of the optimum translation. The \$ symbol denotes the sentence boundary marker. We obtain the following dynamic programming recursion:

$$\begin{aligned}
Q(0, \$) &= 0 && (5.7)\\
Q(j, \tilde{e}) &= \max_{\substack{j' \,:\, j - L_s \leq j' < j \\ \tilde{e}'', \tilde{e}' \,:\, \tilde{e}' \oplus \tilde{e}'' = \tilde{e}}} \Big\{ Q(j', \tilde{e}') + q_{\text{TM}}(\tilde{e}'', j'+1, j) + q_{\text{LM}}(\tilde{e}'' | \tilde{e}') \Big\} && (5.8)\\
\hat{Q} &= \max_{\tilde{e}} \Big\{ Q(J, \tilde{e}) + q_{\text{LM}}(\$; \tilde{e}) \Big\} && (5.9)
\end{aligned}$$

Here, L_s denotes the maximum phrase length in the source language and $E(j, j')$ denotes the set of possible phrase translations of the source phrase $\tilde{f} = f_j, ..., f_{j'}$. During the search, we store back-pointers to the previous best decision $B(\cdot, \cdot)$ and to the maximizing arguments $A(\cdot, \cdot)$. These are used to trace back the best decisions and generate the translation after performing the search.

The pseudo code of the algorithm is shown in Figure 5.1. Let V_e denote the vocabulary size of the target language and, thus, ${V_e}^{n-1}$ is the maximum number of language model histories $\tilde{e}$ of an n-gram language model. Let E denote the maximum number of phrase translation candidates for a source language phrase. Then, the resulting algorithm has a worst-case complexity of $\mathcal{O}(J \cdot L_s \cdot {V_e}^{n-1} \cdot E)$. Using efficient data structures and taking into account that not all possible target language phrases can occur in translating a specific source language sentence, we can perform a very efficient search.

Note that only the sequence of phrases is monotonic. Within the phrase pairs arbitrary reorderings are possible. This monotonic algorithm is especially useful for language pairs that have a similar word order, e. g. Spanish-English or French-English.

5.2.2 Non-monotonic search

For language pairs with different sentence structures, e. g. Japanese-English, monotonic search is not adequate. Therefore, we will describe an algorithm for non-monotonic

```
INPUT: source sentence $f_1^J$, translation options $E(j,j')$ for $1 \leq j \leq j' \leq J$,
       models $q_{\mathrm{TM}}(\cdot)$ and $q_{\mathrm{LM}}(\cdot)$
$Q(0,\$) = 0$ ; all other $Q(\cdot,\cdot)$ entries are initialized to $-\infty$
FOR $j = 1$ TO $J$ DO
    FOR $j' = \max\{0, j - L_s\}$ TO $j-1$ DO
        FOR ALL LM histories $\tilde{e}'$ DO
            FOR ALL phrase translations $\tilde{e}'' \in E(j+1, j')$ DO
                score = $Q(j', \tilde{e}') + q_{\mathrm{TM}}(\tilde{e}'', j+1, j') + q_{\mathrm{LM}}(\tilde{e}''|\tilde{e}')$
                $\tilde{e} = \tilde{e}' \oplus \tilde{e}''$
                IF score $> Q(j, \tilde{e})$
                THEN $Q(j,\tilde{e})$ = score
                     $B(j,\tilde{e}) = (j', \tilde{e}')$
                     $A(j,\tilde{e}) = \tilde{e}''$
```

Figure 5.1: Monotonic search algorithm for text input.

search in this section. For single-word based models, this search strategy is described in [Tillmann & Ney 03]. The idea is that the search proceeds synchronously with the number of the already translated source positions. Here, we use a phrase-based version of this idea.

During the search, we generate the translation phrase by phrase, i. e. the search is monotonic in the target language. In the case of monotonic decoding, the processing on the source side was monotonic as well. To permit reordering, the processing on the source language side may now be non-monotonic. Thus, we can jump forth and back within the source sentence. As mentioned in Section 1.2.3, each source phrase is translated by exactly one target phrase. To avoid that multiple phrases translate the same source position, we keep track of the source positions which are already translated. As mentioned earlier, this is the coverage $C \subseteq \{1, ..., J\}$. If a position j is in the coverage, thus the source word at that position f_j is already translated; it has been 'covered' by a phrase. Internally, the coverage is represented as a bit vector of the size of the source sentence.

As mentioned earlier, the states of the search graph can be identified by a triple $(C, \tilde{e}, j)$ with the coverage set C, the language model history $\tilde{e}$ and the end position of the last source phrase j. We use the auxiliary quantity $Q(C, \tilde{e}, j)$ to denote the maximum score of a path leading from the initial state to the state $(C, \tilde{e}, j)$.

The dynamic programming recursion equations are:

$$Q(\emptyset, \$, 0) \;=\; 0 \tag{5.10}$$

$$Q(C, \tilde{e}, j) \;=\; \max_{\substack{j'', j' : j' \leq j < j' + L_s \wedge \{j', ..., j\} \subseteq C \\ \tilde{e}', \tilde{e}'' : \tilde{e}' \oplus \tilde{e}'' = \tilde{e}}} \Big\{ Q(C \setminus \{j', ..., j\}, \tilde{e}', j'') + q_{\mathrm{TM}}(\tilde{e}'', j', j) + q_{\mathrm{LM}}(\tilde{e}''|\tilde{e}') + q_{\mathrm{DM}}(j'', j') \Big\} \tag{5.11}$$

$$\hat{Q} \;=\; \max_{\tilde{e}, j} \Big\{ Q(\{1, ..., J\}, \tilde{e}, j) + q_{\mathrm{LM}}(\$|\tilde{e}) + q_{\mathrm{DM}}(j, J+1) \Big\} \tag{5.12}$$

Again, L_s denotes the maximum phrase length in the source language. To avoid repeated computations we have to traverse the search graph in a topological order. Thus, before we process a node, i. e. expand the hypothesis, we have to make sure that we have visited all predecessor nodes. We call the number of covered source positions of a partial hypothesis its source cardinality c. We can easily guarantee the topological order by processing the nodes according to their source cardinality. Therefore this processing scheme is also called source cardinality synchronous search.

There are multiple ways to implement a solution to the dynamic programming equations presented in the previous section. The pseudo-code for the non-monotonic search algorithm is presented in Figure 5.2. Let $E(j', j)$ denote the set of possible phrase translations of the source phrase $\tilde{f} = f_{j'}, ..., f_j$. The input is the source sentence f_1^J, the translation options $E(j, j')$ for all source phrases, i. e. $1 \leq j \leq j' \leq J$. The auxiliary quantity $Q(C, \tilde{e}, j)$ is used as in the dynamic programming recursion, i. e. the maximum score of a partial translation with coverage C, language model history $\tilde{e}$ and end position of the last source phrase j. In addition, we store backpointers $B(\cdot)$ to the previous best decision as well as the maximizing arguments $A(\cdot)$, i. e. the best target phrases. For each cardinality c, we have a loop over all possible source phrase lengths l. Then, we have a loop over the possible predecessor coverages C' with cardinality $c - l$. The next loop goes over all possible start position j, thus effectively we select a source phrase $\tilde{f} = f_j, ..., f_{j+l}$. We also check the "no overlap" constraint in line 4. The coverage after the expansion is $C = C' \cup \{j, ..., j + l\}$. Then, we loop over all existing predecessor states $\tilde{e}', j'$ and all translation options $\tilde{e}'' \in E(j, j + l)$. Eventually, we compute the score of the expansion in line 8. If this score is better than the existing one, we update this as well as the backpointer and the pointer to the maximizing argument.

Phrase pairs $(\tilde{f}, \tilde{e})$ are usually used multiple times in line 8. To avoid repeated computations, we generate the set of possible translations $E(j, j')$ for each phrase in the source sentence before the search along with their phrase model scores $q_{\text{TM}}(\tilde{e}, j, j')$.

The computational complexity of the described algorithm in Figure 5.2 is in $\mathcal{O}\left(\sum_{c=1}^{J} L_s \cdot \left(\begin{array}{c} J \\ c \end{array}\right) \cdot J \cdot V_e^{n-1} \cdot E\right)$ which can be simplified to $\mathcal{O}(J \cdot L_s \cdot E \cdot V_e^{n-1} \cdot 2^J)$ by taking into account that $\sum_{c=0}^{J} \binom{J}{c} = 2^J$.

In Figure 5.3, we show an illustration of the search. For each cardinality, we have a list of coverage hypotheses, here represented as boxes. For each coverage hypothesis, we have a list of lexical hypotheses, here represented as circles. We generate a specific lexical hypothesis (the black circle) with cardinality c by expanding shorter hypotheses. The hypotheses with cardinality $c - 1$ are expanded with one-word phrases, the hypotheses with cardinality $c - 2$ are expanded with two-word phrases etc..

5.2.3 Pruning

The algorithm in Figure 5.2 solves the maximization problem, but it has an exponential complexity. To speed up the translation process, we use a beam search strategy [Jelinek 98] and apply pruning at several levels. We use two variants for each of the levels: threshold pruning (also called beam pruning) and histogram pruning

```
INPUT: source sentence $f_1^J$, translation options $E(j, j')$ for $1 \leq j \leq j' \leq J$,
          models $q_{\mathrm{TM}}(\cdot)$, $q_{\mathrm{LM}}(\cdot)$ and $q_{\mathrm{DM}}(\cdot)$
$Q(\emptyset, \$, 0) = 0$ ; all other $Q(\cdot, \cdot, \cdot)$ entries are initialized to $-\infty$
FOR cardinality $c = 1$ TO $J$ DO
    FOR source phrase length $l = 1$ TO $\min\{L_s, c\}$ DO
        FOR ALL coverages $C' \subset \{1, ..., J\} : |C'| = c - l$ DO
            FOR ALL start positions $j \in \{1, ..., J\} : C' \cap \{j, ..., j + l\} = \emptyset$ DO
                coverage $C = C' \cup \{j, ..., j + l\}$
                FOR ALL states $\tilde{e}', j' \in Q(C', \cdot, \cdot)$ DO
                    FOR ALL phrase translations $\tilde{e}'' \in E(j, j + l)$ DO
                        score $= Q(C', \tilde{e}', j') + q_{\mathrm{TM}}(\tilde{e}'', j, j + l) + q_{\mathrm{LM}}(\tilde{e}''|\tilde{e}') + q_{\mathrm{DM}}(j', j)$
                        language model state $\tilde{e} = \tilde{e}' \oplus \tilde{e}''$
                        IF score $> Q(C, \tilde{e}, j + l)$
                        THEN $Q(C, \tilde{e}, j + l) =$ score
                                $B(C, \tilde{e}, j + l) = (C', \tilde{e}', j')$
                                $A(C, \tilde{e}, j + l) = \tilde{e}''$
```

Figure 5.2: Non-monotonic search algorithm for text input (without pruning).

[Steinbiss & Tran+ 94]. Histogram pruning means that we keep the best N hypotheses, e. g. we could limit the total number of hypotheses to 10 000. Threshold pruning means that we keep a hypothesis only if its score is close to the best one. An advantage of threshold pruning is that it adjusts itself to the amount of ambiguity. If the ambiguity is high, i. e. many hypotheses have a similar score, we keep many hypotheses in the beam. Whereas if the ambiguity is low, i. e. there is one or a few dominant hypotheses, we keep only a few hypotheses in the beam. A drawback of threshold pruning is that there is no upper limit on the number of hypotheses in the beam. Thus, in principle, the beam could get arbitrarily large. Histogram pruning is a simple way of limiting the beam size.

We use the following pruning strategies:

1. **Observation Pruning.** Here, we limit the number of translation options per source phrase. This is done before the actual search starts. Let τ_o denote the observation pruning threshold and let $q(j, j')$ denote the maximum score of any phrase translation $\tilde{e}$ of the source phrase $f_j, \ldots, f_{j'}$:

$$q(j, j') = \max_{\tilde{e}} \left\{ q_{\mathrm{TM}}(\tilde{e}, j, j') + q_{\mathrm{LM}}(\tilde{e}) \right\} \tag{5.13}$$

   Here, $q_{\mathrm{LM}}(\tilde{e})$ denotes the weighted LM score of target phrase $\tilde{e}$ without any given history, i. e. we use the unigram score of the first word, the bigram score of the second word and so on. We keep a target phrase $\tilde{e}$ as possible phrase translation of the source phrase $f_j, \ldots, f_{j'}$ if:

$$q_{\mathrm{TM}}(\tilde{e}, j, j') + q_{\mathrm{LM}}(\tilde{e}) + \tau_o \geq q(j, j') \tag{5.14}$$

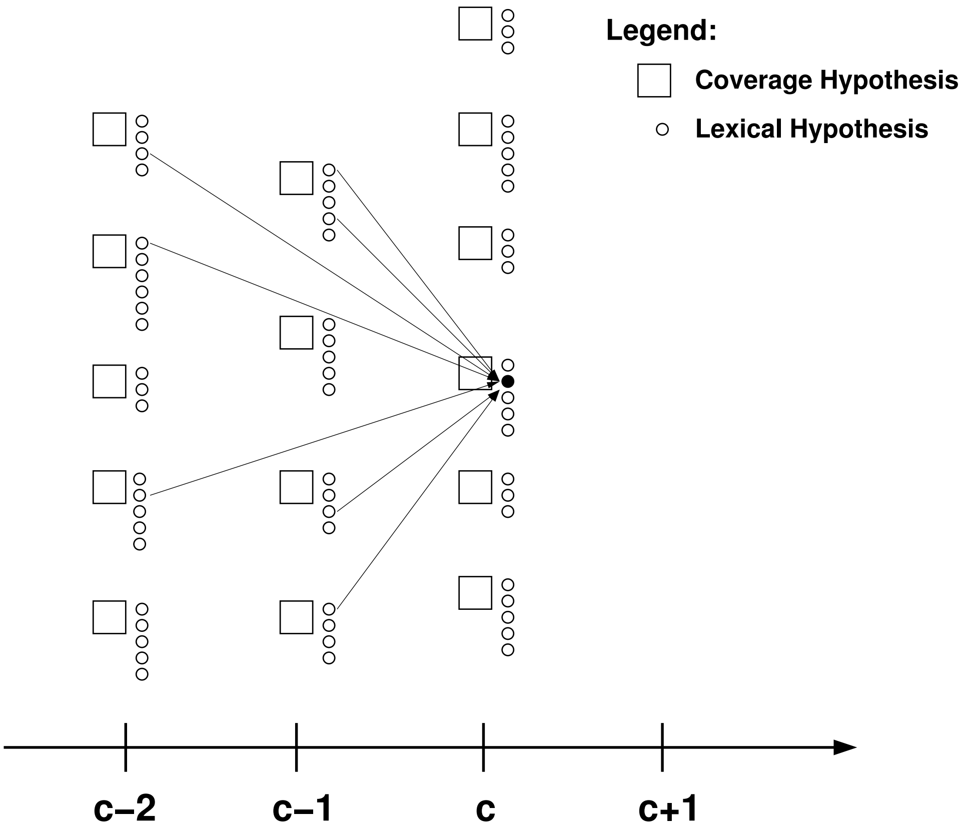


Figure 5.3: Illustration of the search. For each cardinality, we have a list of coverage hypotheses (boxes). For each coverage hypothesis, we have a list of lexical hypotheses (circles). A hypothesis with cardinality c can be generated by expanding a hypothesis of cardinality $c-1$ with a one-word phrase, by expanding a hypothesis of cardinality $c-2$ with a two-word phrase etc..

Additionally, we apply observation histogram pruning with parameter N_o. Thus, if there are more than N_o target phrases for a particular source phrase, then we keep only the top N_o candidates.

2. **Lexical Pruning per Coverage.** Here, we consider all lexical hypotheses that have the same coverage C. The hypotheses may differ, for instance, in their language model history $\tilde{e}$ or the end position of the last phrase j. Here, we include a rest score estimate $R(C, j)$ which is a estimate for completing the hypothesis (mainly for the distortion model which depends on j). A detailed description of the rest score estimate will be given in Section 5.2.4. Let τ_L denote the pruning threshold and let $Q(C)$ denote the maximum score of any hypothesis with coverage C:

$$Q(C) = \max_{\tilde{e},j} \left\{ Q(C, \tilde{e}, j) + R(C, j) \right\} \tag{5.15}$$

Then, we keep a hypothesis with score $Q(C, \tilde{e}, j)$ if:

$$Q(C, \tilde{e}, j) + R(C, j) + \tau_L \geq Q(C) \tag{5.16}$$

Additionally, we apply histogram pruning with parameter N_L. Thus, if there are more than N_L hypotheses for a particular coverage C, then we keep only the top N_L candidates.

3. **Lexical Pruning per Cardinality.** Here, we consider all lexical hypotheses with a given cardinality c. Thus, we also compare hypotheses with different coverages. Here, we include a rest score estimate $R(C, j)$ which is a estimate for completing the hypothesis. A detailed description of the rest score estimate will be given in Section 5.2.4. Let τ_c denote the pruning threshold and let $Q(c)$ denote the maximum score of any hypothesis with cardinality c:

   $$Q(c) = \max_{\substack{C:|C|=c,\\ \tilde{e},j}} \left\{ Q(C, \tilde{e}, j) + R(C, j) \right\} \tag{5.17}$$

   Then, we keep a hypothesis with score $Q(C, \tilde{e}, j)$ if:

   $$Q(C, \tilde{e}, j) + R(C, j) + \tau_c \geq Q(c) \tag{5.18}$$

   Additionally, we apply histogram pruning with parameter N_c. Thus, if there are more than N_c hypotheses for a particular cardinality c, then we keep only the top N_c candidates.

4. **Coverage Pruning per Cardinality.** Here, we consider all coverage hypotheses with a given cardinality c. As defined in Equation 5.15, $Q(C)$ is the maximum score of any hypothesis with coverage C. We will use this value as score of the coverage hypothesis C. Let τ_c denote the pruning threshold, then we keep a coverage hypothesis with score $Q(C)$ if:

   $$Q(C) + \tau_C \geq Q(c) \tag{5.19}$$

   Here, $Q(c)$ denotes the maximum score of any hypothesis with cardinality c as defined in Equation 5.17. Additionally, we apply histogram pruning with parameter N_C. Thus, if there are more than N_C coverage hypotheses for a particular cardinality c, we keep only the top N_C candidates. Note that if we prune a coverage hypothesis C, we remove all lexical hypotheses with coverage C.

Using these pruning techniques the computational costs can be reduced significantly with almost no loss in translation quality. Note that the threshold pruning parameters depend on the model scaling factors λ_1^M. The effect of doubling all lambdas is the same as dividing the threshold pruning parameters by two. Thus, if we modify the model scaling factor, this may affect the search space. This will be important during the training of the model scaling factors. One remedy is to normalize the model scaling factors, e. g. using the L_1 norm. Alternatively, we could rely on histogram pruning and do no or only very conservative threshold pruning. A detailed analysis of these effects is presented in [Cettolo & Federico 04]. Typical values for the histogram pruning are 50 to 200 for the observation pruning, 5 to 50 for the coverage pruning and 500 to 20 000 for the cardinality pruning. Note that the pruning is somewhat more fine-grained than in Pharaoh which does not use coverage pruning. We will see later in the result section that coverage

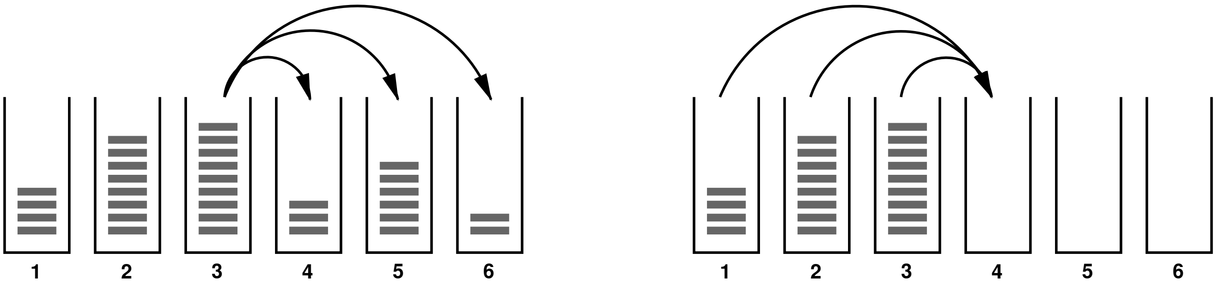


Figure 5.4: Illustration of hypotheses expansion. Left: Pharaoh-style. Right: this work.

pruning is important to adjust the relation between hypotheses with different reorderings (coverage hypotheses) and hypotheses with different lexical choice (lexical hypotheses).

Because of the histogram pruning, the computational complexity of the search is now linear in the sentence length, though with a rather large constant factor. This comes at the expense that we can no longer guarantee to find the global optimum. The experimental results seem to indicate that this is not a serious problem and the translation quality suffers only marginally.

Note that in outermost loop over the cardinalities c, in contrast to the implementation in Pharaoh [Koehn 04a] or [Tillmann 06], we do not expand hypotheses with cardinality c, but generate hypotheses with cardinality c by expanding shorter hypotheses. As a result, all generated hypotheses have the same cardinality c. This has advantages for pruning. Additionally, we avoid the problem of the Pharaoh implementation that the stacks for higher cardinalities can grow quite large and may have to be pruned multiple times. The two types of hypotheses expansion are illustrated in Figure 5.4.

An illustration of the search graph is shown in Figure 5.5. This graph is generated from left to right, i. e. with increasing cardinality. The nodes represent the different coverage hypotheses. Here, we have represented the coverage as a bitvector. For each coverage hypothesis, we have a list of lexical hypotheses identified by the end position of the current phrase in the left part of the nodes and the language model state in the right part of the nodes. The edges are labeled with the target phrases. Dashed edges indicate recombination. Each lexical hypothesis, except for the root node, has exactly one non-dashed inbound edge. This edge is used as back pointer $B(\cdot)$ and the label is the maximizing argument $A(\cdot)$. Pruning is applied for each cardinality. The pruned nodes are not expanded and, thus, do not have outbound edges.

5.2.4 Rest score estimation

During the pruning, we compare hypotheses which cover different parts of the source sentence. Here, it is important to use a rest score estimate for completing the hypothesis. Without such a rest score estimate, the search would first focus on the easy-to-translate part of the source sentence. This is of course undesirable. As rest score estimation, we use the heuristic functions described in [Och 02].

For the observation pruning, we used the score $q(j, j')$ of the best phrase translation of

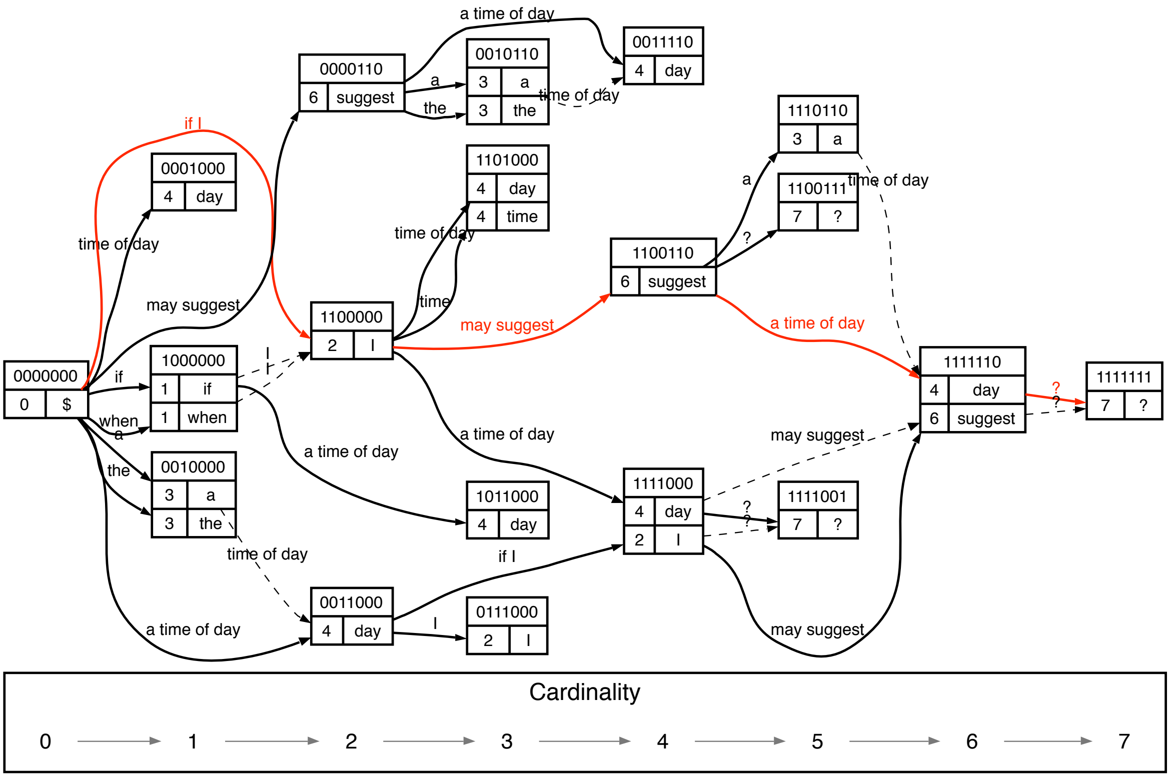


Figure 5.5: Illustration of the search. German input sentence: 'Wenn ich eine Uhrzeit vorschlagen darf?'. English translation: 'If I may suggest a time of day?'. In each node, we store the coverage (here as bitvector), the end position of the current phrase and the language model history (here: bigram). Dashed edges are recombined. Best path marked in red. Scores are omitted.

a source phrase $f_j, \ldots, f_{j'}$ (defined in Equation 5.13)[b]. We implemented two variants of using this quantity to estimate the rest score of a coverage $C \subset \{1 \ldots J\}$. The first variant follows [Koehn 03] and estimates the rest score for sequences of uncovered source positions. The second variant follows [Och 02] and estimates the rest score for individual uncovered source positions. We define $q^*(j, j')$ as the maximum score for translating source positions $j \ldots j'$:

$$q^*(j, j') = \max \left\{ q(j, j'),\ \max_{j \leq k < j'} \{q^*(j, k) + q^*(k+1, j')\} \right\} \tag{5.20}$$

This recursive definition takes alternative phrase segmentations into account. The values of $q^*(j, j')$ can computed using dynamic programming in a straightforward way. To

[b] Note the the language model score is not necessarily an optimistic rest score estimate. Nevertheless, the experiments show that including the LM score estimate is helpful.

use this auxiliary function for the rest score estimation of a hypothesis with coverage $C \subset \{1 \ldots J\}$, we sum up the rest score estimates for the sequences of uncovered source positions:

$$R_{\text{Seq}}(C) = \sum_{(j,j') \in \bar{C}} q^*(j,j') \tag{5.21}$$

Here, $\bar{C}$ denotes the set of sequences of uncovered source positions. A sequence is only contained in $\bar{C}$ if it is of maximum length, formally:

$$\begin{aligned} \bar{C} \quad = \quad & \big\{(j,j')|j \leq j' \wedge \{j \ldots j'\} \subseteq \{1 \ldots J\} \setminus C \\ & \wedge (j = 1 \vee j-1 \in C) \wedge (j' = J \vee j'+1 \in C)\big\} \end{aligned} \tag{5.22}$$

For the second variant, we define $q^*(j)$ as the maximum score for translating the source position j:

$$q^*(j) = \max_{j' \leq j \leq j''} \frac{q(j',j'')}{j''-j'+1} \tag{5.23}$$

To estimate the rest score of a hypothesis, we have to sum up this quantity over the uncovered source positions:

$$R_{\text{Pos}}(C) = \sum_{j \in \{1 \ldots J\} \setminus C} q^*(j) \tag{5.24}$$

In addition, we use a rest score estimation for the distortion model as described in [Och 02]. Thus, we compute the minimum number of jumps to complete the hypotheses. The idea is that we jump from the end position of the current phrase j back to the first uncovered position. Then, we process the remaining part of the sentence from left to right and jump over all covered sequences. The rest score estimate for the distortion penalty model is:

$$j_0(C) \quad = \quad \min\{j \mid 1 \leq j \leq J \wedge j \notin C\} \tag{5.25}$$

$$R_{\text{Dist}}(C,j) \quad = \quad \lambda_{\text{Dist}} \cdot \big(|j - j_0(C) + 1| + |C| - j_0(C) + 1\big) \tag{5.26}$$

Here, $j_0(C)$ denotes the first uncovered position in the coverage set C. The jump distance from the current position j to the first uncovered position $j_0(C)$ is $|j - j_0(C) + 1|$. The number of covered positions to the right of position $j_0(C)$ is $|C| - j_0(C) + 1$. This is the number of source positions that we have to jump over to reach the sentence end. An algorithm for computing this value for a coverage represented as a bit vector is described in [Och 02].

The overall rest score estimate $R(C,j)$ is obtained as the sum of the distortion and translation/language model rest score estimate:

$$R(C,j) = R_{\text{Dist}}(C,j) + R_{\text{Seq}}(C) \tag{5.27}$$

5.2.5 Detailed search algorithm

The detailed search algorithm including pruning is shown in Figure 5.7. The notation and functions used in this algorithm are shown in Figure 5.6. We use 'CONTINUE' and

$Q(C, \tilde{e}, j)$	score of hypothesis $(C, \tilde{e}, j)$, i. e. the hypothesis with coverage C, LM history $\tilde{e}$ and source sentence position j
$B(C, \tilde{e}, j)$	back pointer of hypothesis $(C, \tilde{e}, j)$
$A(C, \tilde{e}, j)$	maximizing argument of hypothesis $(C, \tilde{e}, j)$
L_s	maximum source phrase length
$R(C, j)$	rest score estimate (Eq. 5.27)
$q_{\text{TM}}(\tilde{e}, j, j')$	weighted translation model score for translating source phrase $f_j, \ldots, f_{j'}$ with target phrase $\tilde{e}$ (Eq. 5.2)
$q_{\text{TM}}(j, j')$	best translation model score for translating source phrase $f_j, \ldots, f_{j'}$, i. e. $q_{\text{TM}}(j, j') = \max_{\tilde{e}} q_{\text{TM}}(\tilde{e}, j, j')$
$q_{\text{LM}}(\tilde{e}\|\tilde{e}')$	weighted LM score of phrase $\tilde{e}$ given LM history $\tilde{e}'$ (Eq. 5.3)
$q_{\text{DM}}(j, j')$	weighted distortion score for a jump from source position j to source position j' (Eq. 5.4)
purgeCardinality c	free memory of cardinality c except trace back information
pruneCardinality c	apply coverage and cardinality pruning to all hypotheses with cardinality c
x isTooBadForCoverage C	check if score x cannot survive coverage or cardinality pruning

Figure 5.6: Notation and functions used in the search algorithm in Figure 5.7.

'BREAK' with the usual C/C++ semantics, i. e. 'CONTINUE' will stop the current loop iteration and continue with the next iteration; 'BREAK' will stop the whole loop.

The function 'pruneCardinality c' applies coverage and cardinality pruning to all hypotheses with cardinality c. In the function 'purgeCardinality c', we free the memory (except trace back information) of all hypotheses with cardinality c. For example, the coverage sets and the LM histories are not needed anymore and thus the memory can be reused.

The basic concept of the algorithm is the same as in Figure 5.2. An important aspect is to move as many computations as possible outside of the inner loops. This way, we can avoid redundant computations. For instance, the rest score estimate can already be computed when we know the new coverage C and the source position $j + l$ in line 8.

We also included pruning of hypotheses. Coverage and cardinality pruning is applied after we generated all hypotheses of the current cardinality c in line 22. Additionally, we apply pruning at earlier stages. We stop the expansion if we know that the resulting hypotheses would be pruned anyway. This is done in the function 'x isTooBadForCoverage C'. This way, we can avoid unnecessary computations and speed up the search significantly. Therefore, we store for each coverage C the score of its best lexical hypothesis. This score is updated whenever we generate a lexical hypothesis with coverage C and a better score. For this on-the-fly pruning to be effective, it is important that promising candidates are processed first. Therefore, we process the coverage hypotheses in line 4 and the lexical hypotheses in line 7 in order of their scores, i. e. the best ones first. As already mentioned, the translation options $E(\cdot, \cdot)$ are sorted once before the search.

In particular, we check the partial scores at the following points:

- Line 9: at this point, we have accumulated the score of the predecessor hypothesis $Q(C', \tilde{e}', j')$, the distortion model score $q_{\text{DM}}(j', j)$, the rest score estimate of the new

```
   INPUT: source sentence $f_1^J$, translation options $E(j,j')$ for $1 \leq j \leq j' \leq J$,
          models $q_{\text{TM}}(\cdot)$, $q_{\text{LM}}(\cdot)$ and $q_{\text{DM}}(\cdot)$
 0 $Q(\emptyset, \$, 0) = 0$ ; all other $Q(\cdot,\cdot,\cdot)$ entries are initialized to $-\infty$
 1 FOR cardinality $c = 1$ TO $J$ DO
 2   IF $c > L_s$ THEN purgeCardinality $c - L_s - 1$
 3   FOR source phrase length $l = 1$ TO min$\{L_s, c\}$ DO
 4     FOR ALL coverages $C' \subset \{1,...,J\} : |C'| = c - l$ DO
 5       FOR ALL start positions $j \in \{1,...,J\} : C' \cap \{j,...,j+l\} = \emptyset$ DO
 6         coverage $C = C' \cup \{j,...,j+l\}$
 7         FOR ALL states $\tilde{e}', j' \in Q(C',\cdot,\cdot)$ DO
 8           partial score $q = Q(C', \tilde{e}', j') + q_{\text{DM}}(j', j)$
 9           IF $q + R(C, j+l) + q_{\text{TM}}(j, j+l)$ isTooBadForCoverage $C$
10           THEN CONTINUE
11           FOR ALL phrase translations $\tilde{e}'' \in E(j, j+l)$ DO
12             partial score $q' = q + R(C, j+l) + q_{\text{TM}}(\tilde{e}'', j, j+l)$
13             IF $q'$ isTooBadForCoverage $C$ THEN BREAK
14             score $= q + q_{\text{TM}}(\tilde{e}'', j, j+l) + q_{\text{LM}}(\tilde{e}''|\tilde{e}')$
15             IF score$+R(C, j+l)$ isTooBadForCoverage $C$ THEN CONTINUE
16             language model state $\tilde{e} = \tilde{e}' \oplus \tilde{e}''$
17             IF score $> Q(C, \tilde{e}, j+l)$
18             THEN $Q(C, \tilde{e}, j+l) =$ score
19                  $B(C, \tilde{e}, j+l) = (C', \tilde{e}', j')$
20                  $A(C, \tilde{e}, j+l) = \tilde{e}$
21   pruneCardinality $c$
```

Figure 5.7: Detailed non-monotonic search algorithm for text input. Lines that changed compared to the algorithm in Figure 5.2 have line numbers in boldface.

hypothesis and an optimistic estimate of the translation model score $q_{\text{TM}}(j, j+l)$. If this score is too bad for the coverage hypothesis C, there is no need to process any of the possible phrase translations.

- Line 13: at this point, we have computed the score except for the language model. If this score is already too bad, there is no need to compute the LM score.

- Line 15: at this point, the full score of the expansion is known. If we can prune a hypothesis here, we can directly reuse the memory and there is no need to check for recombination.

5.3 Search for Lattice Input

5.3.1 Introduction

Machine translation systems are often used in a pipeline, i. e. the input to the MT system is the output of another *imperfect* natural language processing system. A typical example is spoken language translation, where the output of an automatic speech recognition (ASR) system is used as the input to a MT system. The traditional approach is to ignore the problem and translate only the 1-best output. This is a simple and efficient solution, but has the disadvantage that the MT system cannot recover from errors of the ASR system. A tighter integration was already proposed in e. g. [Ney 99]. More recently, e. g. in [Bertoldi & Federico 05], it was shown that taking alternative transcriptions of the ASR system into account can improve the translation quality. Therefore, we would like to take the ambiguity of the input explicitly into account.

Despite spoken language translation, there are several other interesting problems which could benefit from such an approach.

- Reordering. Alternative reorderings are represented as a lattice; this reordering lattice is then translated. This allows for a decoupling of the reordering from the core search algorithm. This way it is easy to exchange the reordering strategy without modifying the core search algorithm.

- Chinese word segmentation. In Chinese, the words are not separated by white-space and the segmentation of the Chinese character sequence into words is ambiguous. It has been shown in previous work, [Xu & Matusov$^+$ 05], that it is beneficial for MT quality to consider alternative word segmentation.

- Named entity recognition and translation. Assuming we know which parts of the input sentence are named entities, we could use a special named entity translation/transliteration module for them. On the other hand, it would be very undesirable to translate a non-named-entity with this dedicated translation module. As current named entity recognition systems are not perfect, taking hard decision before the translation process is suboptimal.

Other possible applications include punctuation insertion, spelling correction, paraphrasing, part-of-speech tagging.

The basic idea is to represent the alternative input possibilities in form of a lattice. This is a representation that can encode arbitrary symbol sequences in a compact and efficient way. It allows for a tight coupling of the MT module with the upstream natural language processing applications.

The input lattice is a directed acyclic graph with one start node and one goal node. The nodes are numbered from 1 (initial node) to J (final node). The numbering must be consistent with a topological order of the graph. The edges are labeled with source language words and optionally with costs, e. g. the acoustic and language model costs in case of an ASR lattice.

5.3.2 Monotonic search

Monotonic search on the input lattice can be solved similar to the monotonic search problem for text input. Instead of traversing the source positions, we traverse the nodes of the input lattice from 1 to J. When visiting a node j the topological order guarantees that all its predecessors have already been processed. The quantity $Q(j, \tilde{e})$ is defined as the maximum score of a phrase sequence ending with language model history $\tilde{e}$ and translating a path from the initial node 1 to node j.

We obtain the following dynamic programming recursion:

$$
\begin{aligned}
Q(1, \$) &= 0 && (5.28)\\
Q(j, \tilde{e}) &= \max_{j'<j, \tilde{e}', \tilde{e}'': \tilde{e}=\tilde{e}' \oplus \tilde{e}''} \Big\{ Q(j', \tilde{e}') + q_{\mathrm{TM}}(\tilde{e}'', j', j) + q_{\mathrm{LM}}(\tilde{e}''|\tilde{e}') \Big\} && (5.29)\\
\hat{Q} &= \max_{\tilde{e}} \Big\{ Q(J, \tilde{e}) + q_{\mathrm{LM}}(\$|\tilde{e}) \Big\} && (5.30)
\end{aligned}
$$

Here, $q_{\mathrm{TM}}(\tilde{e}'', j', j)$ denotes the translation score for translating a path from node j' to node j with the target phrase $\tilde{e}$. This is very similar to the $q_{\mathrm{TM}}(\cdot)$ function for text input, except that now, we have nodes instead of positions and we include the costs of the path in the input lattice, e. g. the acoustic and LM costs in case of an ASR lattice. Note that there may be multiple paths from node j' to node j. Thus, there may be multiple (distinct) source phrases along those paths. If the same target phrase can be generate from multiple source phrases, we choose the one with the maximum score. The set of possible translations of all paths from node j' to node j will be denoted as $E(j', j)$. In Section 5.7, we will show how we can efficiently generate these sets for all pairs (j', j).

5.3.3 Non-monotonic search

As the reordering problem for lattice input is computationally even more demanding than for text input, we implemented rather strict reordering constraints. We apply the IBM or 'skip' reordering constraints as described in [Berger & Brown$^+$ 96, Tillmann & Ney 00] for single-word based models. The idea is that we process the lattice in an almost monotonic way, but it is allowed to skip a source phrase and to insert the translation later. As we have to memorize which phrase has been skipped, the search space is increased considerably. Therefore, we typically allow only one phrase to be skipped at a time. In [Kumar & Byrne 05] local reordering constraints have been used to translate lattices in a FST framework.

We will use $\mathcal{S} \subset \{(j, j')|1 \leq j < j' \leq J\}$ to denote the set of skipped phrases. We define the auxiliary quantity $Q(j, \tilde{e}, \mathcal{S})$ as the maximum score of a hypothesis which ends in node j, has language model history $\tilde{e}$ and has skipped the blocks in $\mathcal{S}$.

$$Q(1,\$,\emptyset) \;=\; 0 \tag{5.31}$$

$$\begin{aligned} Q(j,\tilde{e},\mathcal{S}) \;=\; \max\Bigg\{ & \max_{\substack{j'<j\\ \tilde{e}',\tilde{e}'':\tilde{e}=\tilde{e}'\oplus\tilde{e}''}} \Big\{Q(j',\tilde{e}',\mathcal{S}) + q_{\mathrm{TM}}(\tilde{e}'',j',j) + q_{\mathrm{LM}}(\tilde{e}''|\tilde{e}')\Big\}, \\ & \max_{\substack{(j',j''):(j',j'')\notin\mathcal{S}\\ \tilde{e}',\tilde{e}'':\tilde{e}=\tilde{e}'\oplus\tilde{e}''}} \Big\{Q(j,\tilde{e}',\mathcal{S}\cup\{(j',j'')\}) + q_{\mathrm{TM}}(\tilde{e}'',j',j'') + q_{\mathrm{LM}}(\tilde{e}''|\tilde{e}')\Big\}\Bigg\} \end{aligned} \tag{5.32}$$

$$Q(j,\tilde{e},\mathcal{S}\cup\{(j',j)\}) = Q(j',\tilde{e},\mathcal{S}) \tag{5.33}$$

$$\hat{Q} \;=\; \max_{\tilde{e}}\Big\{Q(J,\tilde{e},\emptyset) + q_{\mathrm{LM}}(\$;\tilde{e})\Big\} \tag{5.34}$$

The first one of the two inner max operations in Equation 5.32 is the monotonic expansion similar to Equation 5.29 in monotonic search. In the second max operation, a block (j',j'') that has been skipped at an earlier stage is translated. In Equation 5.33, we skip a block (j',j).

The search algorithms for lattice input described in this section were successfully used for speech translation [Matusov & Popović$^+$ 04, Matusov & Ney 05, Zens & Bender$^+$ 05, Matusov & Zens$^+$ 06], reordering [Zens & Och$^+$ 02, Zens & Bender$^+$ 05, Matusov & Zens$^+$ 06, Zhang & Zens$^+$ 07a, Zhang & Zens$^+$ 07b], integrated Chinese word segmentation [Xu & Matusov$^+$ 05].

5.4 Word Graph and N-best Generation

In some situations, we are not only interested in the single best MT output hypothesis, but also in alternative translations. For example in the rescoring/reranking framework, which might be the most popular usage of N-best lists, we apply additional models to improve the MT quality. Those additional models are typically hard to apply during the search, either because of the high computational demands or because they require that the translation hypothesis is fully generated. For instance, the IBM model 1 $p(f_1^J|e_1^I)$ involves a sum over the target positions, which is not applicable to partial hypotheses.

To compute the single-best or N-best translation hypotheses under the MAP decision rule (Equation 1.11), a proper normalization of the log-linear model is not required, i. e. we could discard the denominator in Equation 1.8. Nevertheless, for some applications we need to have a properly normalized probability distribution over the target language sentences. Examples are the estimation of confidence measures based on word posterior probabilities [Ueffing 05], and MBR decoding [Kumar & Byrne 04]. The sum over all possible target language sentences is approximated using the word graph (also called lattice) or the N-best translation candidates. Note that the word graph typically contains many more hypotheses than the N-best list and is thus preferable. Nevertheless, the N-best list may contain already enough translation candidates to get reliable estimates. Also, the computation of statistics on an N-best list is typically simpler than on a word graph.

The generation of word graphs and N-best lists for MT is described in [Ueffing & Och$^+$ 02] for the single-word based IBM model 4, for phrase-based models in [Koehn 03, Zens & Ney 05, Hasan & Zens$^+$ 07]. Decoding is very similar to the standard single-best search. The difference is that we keep the back-pointers to the recombined hypotheses instead of deleting them. As a result, the nodes in the search graph can have multiple incoming edges and not just one as in single-best search. Therefore, we do not generate a tree of translation hypotheses, but a graph of translation hypotheses; in fact it is a directed acyclic graph (dag).

Given this graph, we can extract the N best translation candidates, i. e. paths in the lattice, using standard finite state toolkits, e. g. [Kanthak & Ney 04]. Alternatively, we can use an extension of the A* algorithm as described in [Ueffing & Och$^+$ 02]. Also, k shortest path algorithms, e. g. [Eppstein 01]. In the experiments, we will use the A* algorithm as described in [Ueffing & Och$^+$ 02].

To ensure that the word graph contains translation candidates that are significantly better than the single-best translation hypothesis, we compute the oracle Bleu score of the lattice. Given the reference translations, we select the path that maximizes the Bleu score. This gives an upper bound for the Bleu score. We use a variant of the method described in [Dreyer & Hall$^+$ 07], where the oracle Bleu score for different permutations was computed. We traverse the word graph in topological order and keep track of the following information: the counts of the matching n-grams and the length of the hypothesis. Using this, we can compute the Bleu score. Here, we also include the brevity penalty. This was not required in [Dreyer & Hall$^+$ 07], as all permutations have the same length. When we have processed a word graph, we trace back the best path and print the target sentence. After processing all word graphs of a test set, we compute the Bleu score on the extracted sentences.

To compute the oracle Bleu score on N-best lists, we use a greedy algorithm as described in [Och & Gildea$^+$ 04]. We process the test set sentence-wise and accumulate the n-gram counts. After each sentence, we take a greedy decision and choose the n-gram counts that, if combined with the already accumulated n-gram counts, result is the largest Bleu score. This gives a conservative approximation of the true oracle Bleu score.

5.5 Reordering Contraints

5.5.1 Introduction

In this section, we will focus on the reordering problem. As the word order in source and target language may differ, the search algorithm has to allow certain reorderings. As mentioned before, the generation of a translation hypothesis is computationally expensive if arbitrary reorderings are allowed. To reduce the search space, we can apply suitable reordering constraints. Although unconstrained reordering looks perfect from a theoretical point of view, we find that in practice constrained reordering shows better performance. The possible advantages of reordering constraints are:

1. The search problem is simplified. As a result there might be fewer search errors.

2. Unconstrained reordering is only helpful if we are able to estimate the reordering probabilities reliably, which is unfortunately not the case.

In this section, we will describe two variants of reordering constraints in detail. The first constraints are based on the IBM constraints for single-word based translation models [Berger & Brown$^+$ 96]. The second constraints are based on *inversion transduction grammars* (ITG) [Wu 95, Wu 96, Wu 97, Wu & Wong 98]. We show a connection between the ITG constraints and the since 1870 known *Schröder* numbers.

In this section, we will discuss the reordering constraints from a theoretical point of view. We will answer the question of how many reorderings are permitted for the ITG constraints as well as for the IBM constraints. Since we are only interested in the number of possible reorderings, the specific word identities are of no importance here. Furthermore, we assume a one-to-one correspondence between source and target words. Thus, we are interested in the number of reorderings, i. e. permutations, that satisfy the chosen constraints. First, we will consider the ITG constraints. Afterward, we will describe the IBM constraints.

5.5.2 ITG constraints

5.5.2.1 Description

In this section, we describe the ITG constraints [Wu 95, Wu 97]. The ITG constraints were introduced in [Wu 95]. The applications were, for instance, the segmentation of Chinese character sequences into Chinese words and the bracketing of the source sentence into sub-sentential chunks. In [Wu 96] the baseline ITG constraints were used for statistical machine translation. The resulting algorithm is similar to the one presented in Section 5.5.2.4, but [Wu 96] used a single-word based lexicon model as initialization. In [Vilar 98, Vilar & Vidal 05] a model similar to Wu's method was considered.

Here, we interpret the input sentence as a sequence of blocks. In the beginning, each position is a block of its own. Then, the reordering process can be interpreted as follows: we select two consecutive blocks and merge them to a single block by choosing between two options: either keep the target phrases in monotonic order or invert the order. This idea is illustrated in Figure 5.8. The dark boxes represent the two blocks to be merged. Once two blocks are merged, they are treated as a single block and they can be only merged further as a whole. It is not allowed to merge one of the subblocks again.

An alternative description is a binary bracketing of the input sequence with two types of brackets. Dependent on the type of bracket, the two subparts are concatenated in monotonic $[\cdot]$ or in inverted $\langle\cdot\rangle$ order. An example is

$$\langle [AB][\langle CD\rangle\, E]\rangle$$

which represents the permutation $DCEAB$.

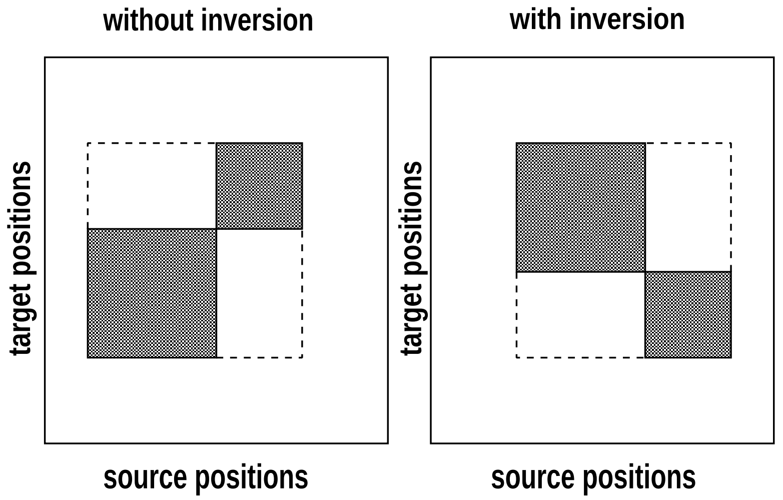


Figure 5.8: Illustration of the ITG reordering constraints: monotonic and inverted concatenation of two consecutive blocks.

5.5.2.2 Analysis

Now, we investigate, how many permutations are obtainable with this method. A permutation derived by the preceding method can be represented as a binary tree where the inner nodes are colored either black or white. At black nodes the resulting sequences of the children are inverted. At white nodes they are kept in monotonic order. This representation is equivalent to the parse trees of the simple grammar in [Wu 97].

We observe that a given permutation may be constructed in several ways by the preceding method. For instance, let us consider the identity permutation of $1, 2, ..., n$. Any binary tree with n nodes and all inner nodes colored white (monotonic order) is a possible representation of this permutation. To obtain a unique representation, we pose an additional constraint on the binary trees: if the right son of a node is an inner node, it has to be colored with the opposite color. With this constraint, each of these binary trees is unique and equivalent to a parse tree of the 'canonical-form' grammar in [Wu 97].

In [Shapiro & Stephens 91], it is shown that the number of such binary trees with n nodes is the $(n-1)th$ large *Schröder* number S_{n-1}. The (small) *Schröder* numbers have been first described by Schröder in 1870 as the number of bracketings of a given sequence, the so-called Schröder's second problem [Schröder 70]. The large *Schröder* numbers are just twice the *Schröder* numbers. Schröder remarked that the ratio between two consecutive *Schröder* numbers approaches $3 + 2\sqrt{2} = 5.8284...$. A second-order recurrence for the large *Schröder* numbers is:

$$(n+1)S_n = 3(2n-1)S_{n-1} - (n-2)S_{n-2}$$

with $n \geq 2$ and $S_0 = 1$, $S_1 = 2$.

The *Schröder* numbers have many combinatorial interpretations. Here, we will mention only two of them. The first one is another way of viewing the ITG constraints. The number

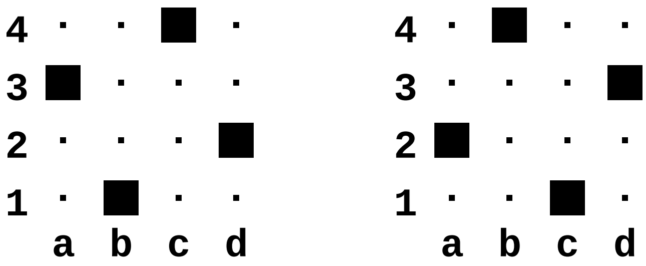


Figure 5.9: Illustration of the two reordering patterns that violate the ITG constraints: (3142) on the left and (2413) on the right.

of permutations of the sequence $1, 2, ..., n$, which avoid the subsequences (3142) and (2413), is the large *Schröder* number S_{n-1}. More details on forbidden subsequences can be found in [West 95]. The interesting point is that a search with the ITG constraints cannot generate a word-reordering that contains one of these two subsequences. In [Wu 97], these forbidden subsequences are called 'inside-out' transpositions. Thus, a reordering violates the ITG constraints if and only if it contains (3142) or (2413) as a subsequence. This means, if we select four columns and the corresponding rows from the alignment matrix and we obtain one of the two patterns illustrated in Figure 5.9, this reordering cannot be generated with the ITG constraints.

Another interpretation of the *Schröder* numbers is given in [Knuth 73]: the number of permutations that can be sorted with an output-restricted double-ended queue (deque) is exactly the large *Schröder* number. Additionally, Knuth presents an approximation for the large *Schröder* numbers:

$$S_n \quad \approx \quad c \cdot (3 + \sqrt{8})^n \cdot n^{-\frac{3}{2}} \tag{5.35}$$

where c is set to $\frac{1}{2}\sqrt{(3\sqrt{2} - 4)/\pi}$. This approximation function confirms the result of Schröder, and we obtain $S_n \in \Theta((3 + \sqrt{8})^n)$, i. e. the *Schröder* numbers grow like $(3 + \sqrt{8})^n \approx 5.83^n$.

5.5.2.3 Extended ITG constraints

In this section, we will extend the ITG constraints. This extension will go beyond basic reordering constraints. We already mentioned that the use of contiguous phrases within the ITG approach is straightforward. The only thing we have to change is the initialization of the Q-table. Now, we will extend this idea to phrases that are non-contiguous in the source language. For this purpose, we adopt the view of the ITG constraints as a bilingual grammar as, e.g., in [Wu 97]. For the baseline ITG constraints, the resulting grammar is:

$$A \ \rightarrow \ [AA] \ | \ \langle AA \rangle \ | \ f/e \ | \ f/\epsilon \ | \ \epsilon/e \tag{5.36}$$

Here, $[AA]$ denotes a monotonic concatenation and $\langle AA \rangle$ denotes an inverted concatenation. Let us now consider the case of a source phrase consisting of two parts f_1 and f_2.

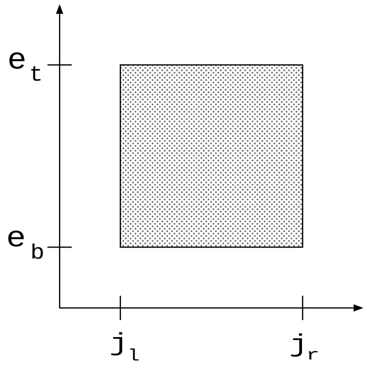


Figure 5.10: Illustration of the Q-table.

Let e denote the corresponding target phrase. We add the productions

$$A \;\rightarrow\; [e/f_1 \; A \; \epsilon/f_2] \;|\; \langle e/f_1 \; A \; \epsilon/f_2 \rangle \tag{5.37}$$

to the grammar. The probabilities of these productions are, dependent on the translation direction, $p(e|f_1, f_2)$ or $p(f_1, f_2|e)$, respectively. Obviously, these productions are not in the normal form of an ITG, but with the method described in [Wu 97], they can be normalized. These extension are especially useful for the negation in French-English (”ne ... pas”,”not”) and in case of German-English for split German verb prefixes (e. g. ”fahren ... los”,”leave”).

5.5.2.4 Dedicated search algorithm

The ITG constraints allow for a polynomial-time search algorithm, as described in [Wu 97]. It is based on the following dynamic programming recursion equations. During the search a table Q_{j_l,j_r,e_b,e_t} is constructed. Here, Q_{j_l,j_r,e_b,e_t} denotes the probability of the best hypothesis translating the source words from position j_l (left) to position j_r (right) which begins with the target language word e_b (bottom) and ends with the word e_t (top). This is illustrated in Figure 5.10.

The initialization, denoted as $Q^0_{j_l,j_r,e_b,e_t}$, is done with the phrase-based model described in Section 4.3. We introduce a new parameter p_m ($m \hat{=}$ monotonic), which denotes the probability of a monotonic combination of two partial hypotheses. Here, we formulate the recursion equation for a bigram language model, but of course, the same method can also be applied for higher order language models.

$$\begin{aligned} Q_{j_l,j_r,e_b,e_t} \;=\; \max_{\substack{j_l \le k < j_r,\\ e',e''}} \Big\{ & Q^0_{j_l,j_r,e_b,e_t}, \\ & Q_{j_l,k,e_b,e'} \cdot Q_{k+1,j_r,e'',e_t} \cdot p(e''|e') \cdot p_m, \\ & Q_{k+1,j_r,e_b,e'} \cdot Q_{j_l,k,e'',e_t} \cdot p(e''|e') \cdot (1-p_m) \Big\} \end{aligned}$$

The resulting algorithm is similar to the CYK-parsing algorithm [Kasami 65, Younger 67]. It has a worst-case complexity of $\mathcal{O}(J^3 \cdot V_e^4)$. Here, J is the length of the source sentence and V_e is the vocabulary size of the target language. Using the methods described in [Huang & Zhang$^+$ 05], it is possible to reduce this worst case complexity, but if the resulting algorithm is faster in practice, especially in the context of pruning, was left open.

5.5.2.5 Integration into beam search algorithm

For the ITG constraints a CYK-style search algorithm exists as described in the previous section. A disadvantage of that algorithm is that the language model history has to be consider at the beginning and the end of each interval.

It would be more practical with respect to language model recombination to have an algorithm that generates the target sentence phrase by phrase. The idea is to start with the beam search decoder for unconstrained search and modify it in such a way that it will produce only reorderings that do not violate the ITG constraints. Now, we describe one way to obtain such a decoder.

We have to modify the beam search decoder such that it cannot produce the two patterns that violate the ITG constraints. We implement this in the following way. We define a *span* to be a contiguous sequence of source positions. We organize the coverage in form of a stack of spans where each of them is ITG parsable. When we generate the next phrase, we put the corresponding span of source positions on the stack. If the two topmost elements of the stack are adjacent, i. e. we can combine them either in monotonic or inverted order, we do so and put the resulting span back on the stack. The combination step is repeated until the two topmost elements of the stack are no longer adjacent to each other. A permutation is ITG parsable if the final stack contains only one element, namely $\{1, ..., J\}$. The pseudo-code of the algorithm is shown in Figure 5.11. Here, the array $B[\cdot]$ denotes the start positions of the spans (B=begin) and the array $E[\cdot]$ denotes the end positions, thus the j^{th} span starts at position $B[j]$ and ends at position $E[j]$. The variable S is the current size of the arrays. This was implemented in the publicly available RWTH FSA toolkit[c] to generate permutation graphs [Kanthak & Ney 04, Kanthak & Vilar$^+$ 05]. A similar idea is used in [Zhang & Huang$^+$ 06] for synchronous binarization of grammar rules.

A drawback of this approach is that the decision if a permutation is ITG parsable or not is done at the very end. We can do better by utilizing our knowledge about the forbidden subsequences (3142,2413) to speed up the procedure and discard many partial permutation early during the generation. To avoid the forbidden patterns in Figure 5.9, we have to constrain the placement of the third phrase, because once we have placed the first three phrases we also have determined the position of the fourth phrase as the remaining uncovered position. Using the coverage and the information about the current source sentence position j_c and a candidate source position j_n to be translated next, we can determine if the permutation will eventually violate the ITG constraints. We check

[c] http://www-i6.informatik.rwth-aachen.de/~kanthak/fsa.html

```
INPUT: permutation $\pi_1, ..., \pi_J$ of $1, ..., J$
ARRAY $B, E$; INT $S = 0$;
FOR $j = 1$ TO $J$ DO
    $S = S + 1$
    $B[S] = \pi_j$;  $E[S] = \pi_j$
    WHILE $(S \geq 2 \wedge (E[S-1] = B[S] \vee B[S-1] = E[S]))$
        IF $(E[S-1] = B[S])$
        THEN $E[S-1] = E[S]$
        ELSE $B[S-1] = B[S]$
        DELETE $B[S], E[S]$
        $S = S - 1$
IF S=1 THEN ACCEPT ELSE REJECT
```

Figure 5.11: Algorithm to test if a permutation is ITG-parsable.

the following constraints:

$$\text{case a)} \quad j_n < j_c \qquad (5.38)$$
$$\forall j_n < j < j_c : j \in C \rightarrow j + 1 \in C$$
$$\text{case b)} \quad j_c < j_n \qquad (5.39)$$
$$\forall j_c < j < j_n : j \in C \rightarrow j - 1 \in C$$

The constraints in Equation 5.38 and Equation 5.39 enforce the following: imagine, we go from the current position j_c to the position to be translated next j_n. Then, it is not allowed to move from an uncovered position to a covered one.

Equation 5.38 and Equation 5.39 are necessary conditions for the ITG constraints. Thus, if the conditions are violated, the resulting permutation violates the ITG constraints. It is rather easy to see that any reordering that violates the constraint in Equation 5.38 will generate the pattern on the left-hand side in Figure 5.9. The conditions to violate Equation 5.38 are the following: the new candidate position j_n is to the left of the current position j_c, e. g. positions (a) and (d). Somewhere in between there has to be an covered position j whose successor position $j + 1$ is uncovered, e. g. (b) and (c). Therefore, any reordering that violates Equation 5.38 generates the pattern on the left-hand side in Equation 5.9, thus it violates the ITG constraints.

Using Equation 5.38 and Equation 5.39 we can discard many permutations at an early stage and therefore speed up the processing significantly.

5.5.2.6 Baxter permutations

Using the techniques from the previous section, the ITG constraints can be integrated into a standard beam search decoder. Nevertheless, we have to build the stack with the spans which results in some overhead. To avoid this overhead, we relax the ITG constraints. If we just rely on Equation 5.38 and Equation 5.39, we will generate the so-called Baxter permutations. These are a superset of the ITG permutations, e. g. (25314) is a Baxter permutation but not an ITG permutation. The Baxter permutations first arose in the context of commuting functions [Baxter 64]. They can be characterized as all permutations that avoid the barred subsequences $(25\bar{3}14, 41\bar{3}52)$ [Dulucq & Guibert 96]. Thus, the subsequences (2413) and (3142) are forbidden except if they can be expanded to (25314) or (41352), respectively.

The number of Baxter permutations B_n of a sequence $1, ..., n$ is [Chung & Graham$^+$ 78]:

$$B_n = \frac{2}{n(n+1)^2} \sum_{k=1}^{n} \binom{n+1}{k-1} \binom{n+1}{k} \binom{n+1}{k+1} \tag{5.40}$$

During the search, we can efficiently verify if the conditions in Equation 5.38 and in Equation 5.39 are violated or not. This can be done in line 7/8 of the search algorithm in Figure 5.7. Thus, the beam search decoder can be easily restricted to the Baxter permutations.

5.5.3 IBM constraints

In this section, we will describe the IBM constraints [Berger & Brown$^+$ 96] which are based on permutations with restricted displacement [Lehmer 70]. Here, we mark each position in the source sentence either as covered or uncovered. In the beginning, all source positions are uncovered. Now, we process the source positions from left to right. We are allowed to skip a position and come back to it later. According to the IBM constraints the next position has to be one of the first k uncovered positions, i. e. at any time there are no more than $k-1$ skipped positions: Let $C \subseteq \{1, ..., J\}$ denote the coverage, then it violates the IBM constraints, if:

$$|C| + k \leq \max C \tag{5.41}$$

The IBM constraints are illustrated in Figure 5.12. Here, the source sentence positions are along the x-axis. Yet uncovered positions are marked with unfilled circles, the already covered positions with filled circles. The uncovered positions that are candidates for extension are marked with unfilled squares.

For most of the target positions there are k permitted source positions. Only toward the end of the sentence this is reduced to the number of remaining uncovered source positions. Let n denote the length of the input sequence and let r_n denote the permitted number of permutations with the IBM constraints. Then, we obtain:

$$r_n = \begin{cases} k^{n-k} \cdot k! & n > k \\ n! & n \leq k \end{cases} \tag{5.42}$$

Typically, k is set to 4. In this case, we obtain an asymptotic upper and lower bound of 4^n, i. e. $r_n \in \Theta(4^n)$. Despite that the number of permutations is exponential, there exists efficient algorithms to use these constraints during the search (cf. [Tillmann & Ney 00, Tillmann & Ney 03] for single-word based models).

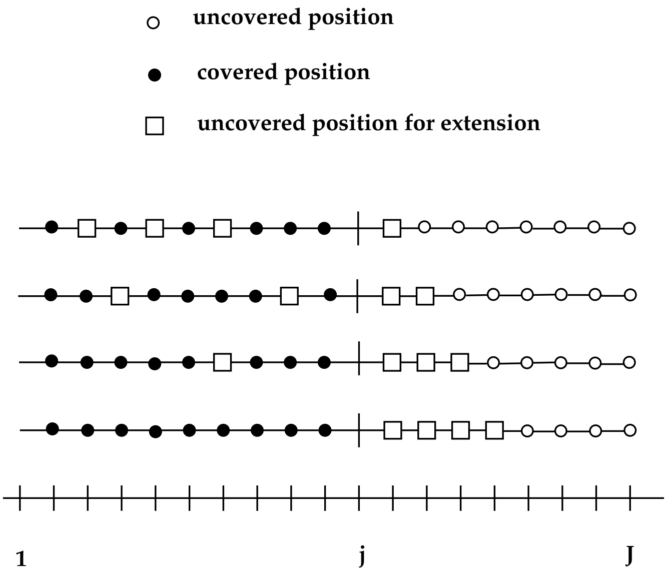


Figure 5.12: Illustration of the IBM constraints (from [Tillmann & Ney 00]).

For the phrase-based translation approach, we use the same idea. In the single-word based version, it is permitted to skip up to $k-1$ positions, here, it is permitted to skip up to $k-1$ blocks. Thus, we allow up to $k-1$ gaps in the coverage. Note that we do not have to fill these gaps with a single phrase; multiple phrases are permitted. If we set $k=1$, we obtain a search that is monotonic at the phrase level as a special case. For a coverage C, we check the following condition:

$$\left|\{j > 1 \mid j \in C \wedge j-1 \notin C\}\right| < k \tag{5.43}$$

This ensures that the number of gaps is less than k. The integration of the IBM constraints into the beam search algorithm is rather straightforward. We just have to test that the coverage C does not violate the IBM constraints, e. g. in line 6 of the search algorithm in Figure 5.7 This reduces the possible successor states in the search graph and therefore speeds up the search.

In Table 5.1, the number of permutations are listed that can be generated with different reordering constaints as a function of the length of the input sequence. A very helpful resource for the analysis of these reordering constraints was the On-Line Encyclopedia of Integer Sequences[d].

5.6 Efficient Phrase-table Representation

In phrase-based statistical machine translation, a huge number of source and target phrase pairs is memorized in the so-called phrase-table. For medium sized tasks and phrase lengths, these phrase-tables already require several GBs of memory or even do not fit at all. In Figure 5.13, we show the number of bilingual phrase pairs as a function of the

[d] `http://www.research.att.com/~njas/sequences`

Table 5.1: Number of permutations that can be generated with different reordering constraints.

n	IBM				ITG	Baxter
	$k = 2$	3	4	5		
1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	6	6	6	6	6
4	5	14	24	24	22	22
5	8	31	78	120	90	92
6	13	73	230	504	394	422
7	21	172	675	1 902	1 806	2 074
8	34	400	2 069	6 902	8 558	10 754
9	55	932	6 404	25 231	41 586	58 202
10	89	2 177	19 708	95 401	206 098	326 240
11	144	5 081	60 216	365 116	1 037 718	1 882 960
12	233	11 854	183 988	1 396 948	5 293 446	11 140 560
13	377	27 662	563 172	5 316 192	27 297 738	67 329 992
14	610	64 554	1 725 349	20 135 712	142 078 746	414 499 438
15	987	150 639	5 284 109	76 227 216	745 387 038	2 593 341 586

training corpus size for the Chinese-English NIST task. Here, we limited the length of the source phrases to five words and the number of translation candidates per source phrase to 50. We observe that the number of phrase pairs increases steadily with the training corpus size.

If the source text, which is to be translated, is known in advance, a common trick is to filter the phrase-table and keep a phrase pair only if the source phrase occurs in the text. This filtering is time-consuming as we have to go over the whole phrase-table. Furthermore, we have to repeat this filtering step whenever we want to translate a new source text.

To address these problems, we will use an efficient representation of the phrase-table with two key properties: *on-demand loading* and a *prefix tree* structure for the source phrases. The prefix tree structure exploits the redundancy among the source phrases. Using on-demand loading, we will load only the small fraction of the overall phrase-table into memory that is actually needed. The majority of the phrase-table will remain on disk.

The on-demand loading is employed on a per sentence basis, i. e. we load only the phrase pairs that are required for one sentence into memory. Therefore, the memory requirements are low, e. g. less than 25 MB for the Chinese-English NIST task. Another advantage of the on-demand loading is that we are able to translate new source sentences without filtering.

A potential problem is that this on-demand loading might be too slow. To overcome this, we use a binary format which is a memory map of the internal representation used during decoding. Additionally, we load coherent chunks of the tree structure instead of individual phrases, i. e. we have only few disk access operations. In our experiments, the

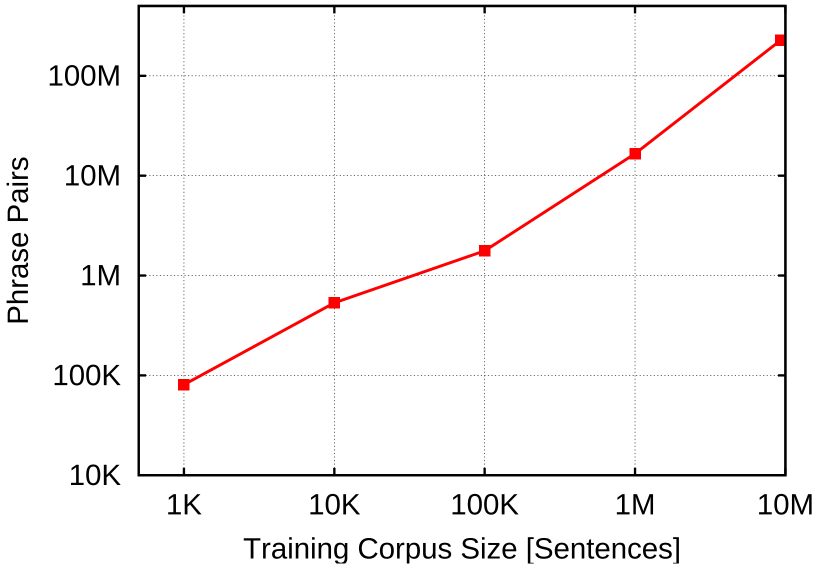


Figure 5.13: Phrase-table size as a function of the training corpus size for the Chinese-English NIST task.

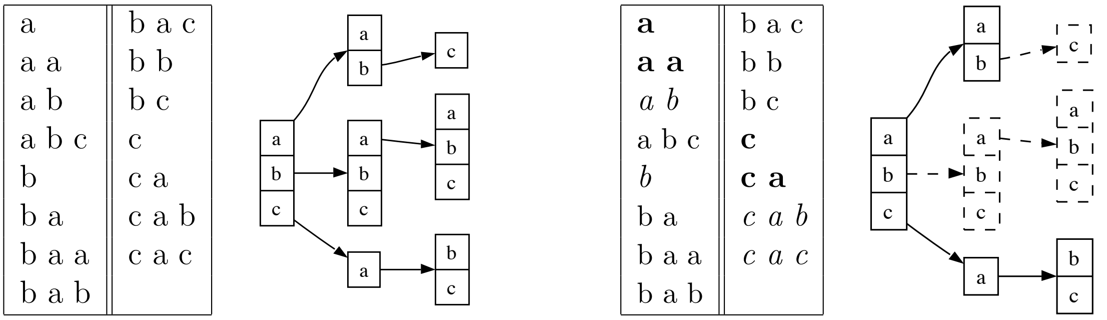
on-demand loading is *not* slower than the traditional approach.

In this section, we will describe the proposed representation of the phrase-table. A prefix tree, also called trie, is an ordered tree data structure used to store an associative array where the keys are symbol sequences. In the case of phrase-based MT, the keys are source phrases, i. e. sequences of source words and the associated values are the possible translations of these source phrases. In a prefix tree, all descendants of any node have a common prefix, namely the source phrase associated with that node. The root node is associated with the empty phrase.

The prefix tree data structure is quite common in automatic speech recognition. There, the lexicon, i. e. the mapping of phoneme sequences to words, is usually organized as a prefix tree [Ney & Haeb-Umbach$^+$ 92].

We convert the list of source phrases into a prefix tree and, thus, exploit that many of them share the same prefix. This is illustrated in Figure 5.14 (left). Within each node of the tree, we store a sorted array of possible successor words along with pointers to the corresponding successor nodes. Additionally, we store a pointer to the possible translations.

One property of the tree structure is that we can efficiently access the successor words of a given prefix. This will be a key point to achieve an efficient phrase matching algorithm in Section 5.7.2. When looking for a specific successor word, we perform a binary search in the sorted array. Alternatively, we could use hashing to speed up this lookup. We have chosen an array representation as this can be read very fast from disk. Additionally, with the exception of the root node, the branching factor of the tree is small, i. e. the potential benefit from hashing is limited. At the root node, however, the branching factor is close



a	b a c
a a	b b
a b	b c
a b c	c
b	c a
b a	c a b
b a a	c a c
b a b	

a	b a c
a a	b b
a b	b c
a b c	**c**
b	**c a**
b a	*c a b*
b a a	*c a c*
b a b	

Figure 5.14: Illustration of the prefix tree. Left: list of source phrases and the corresponding prefix tree. Right: list of matching source phrases for sentence 'c a a c' (**bold** phrases match, phrases in *italics* are loaded in memory) and the corresponding partially loaded prefix tree (the dashed part is not in memory).

to the vocabulary size of the source language, which can be large. As we store the words internally as integers and virtually all words occur as the first word of some phrase, we can use the integers directly as the position in the array of the root node. Hence, the search for the successors at the root node is a simple table lookup with direct access, i. e. in $\mathcal{O}(1)$.

If not filtered for a specific test set, the phrase-table becomes huge even for medium-sized tasks. Therefore, we store the tree structure on disk and load only the required parts into memory on-demand. This is illustrated in Figure 5.14 (right). Here, we show the matching phrases for the source sentence 'c a a c', where the matching phrases are set in **bold** and the phrases that are loaded into memory are set in *italics*. The dashed part of the tree structure is not loaded into memory. Note that some nodes of the tree are loaded even if there is no matching phrase in that node. These are required to actually verify that there is no matching phrase. An example is the 'bc' node in the lower right part of the figure. This node is loaded to check if the phrase 'c a a' occurs in the phrase-table. The translations, however, are loaded only for matching source phrases.

We made an implementation of this phrase-table representation publicly available in the Moses toolkit [Koehn & Hoang$^+$ 07][e]. In the following section, we will present a phrase matching algorithm for lattice input which utilizes the prefix-tree structure of the phrase-table described in this section.

5.7 Phrase Matching

In speech translation, the input to the MT system is not a sentence, but a lattice representing alternative ASR transcriptions. As pointed out in [Mathias & Byrne 06], one problem in speech translation is that we have to match the phrases of our phrase-table against the input lattice. This results in a combinatorial problem as the number of phrases in a lattice increases exponentially with the phrase length. We will present a phrase matching

[e] `http://www.statmt.org/moses`

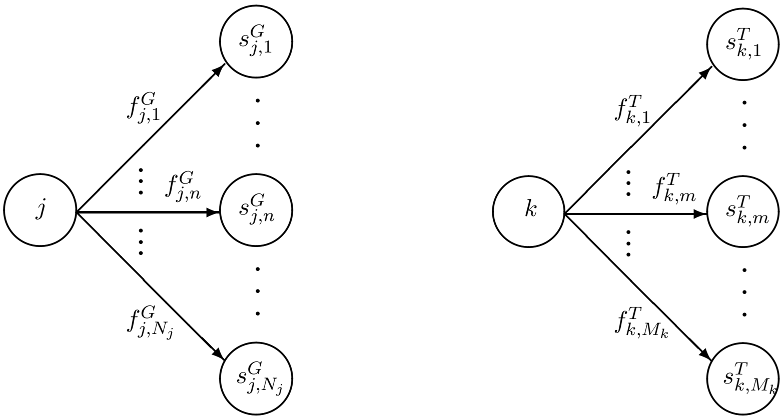


Figure 5.15: Illustration for graph G and prefix tree T. Left: graph node j with successor nodes $s^G_{j,1}, ..., s^G_{j,n} ..., s^G_{j,N_j}$ and corresponding edge labels $f^G_{j,1}, ..., f^G_{j,n}, ..., f^G_{j,N_j}$. Right: prefix tree node k with successor nodes $s^T_{k,1}, ..., s^T_{k,m}, ..., s^T_{k,M_k}$ and corresponding edge labels $f^T_{k,1}, ..., f^T_{k,m}, ..., f^T_{k,M_k}$.

algorithm that effectively solves this combinatorial problem exploiting the prefix tree data structure of the phrase-table. This algorithm enables the use of significantly larger input lattices in a more efficient way resulting in improved translation quality.

5.7.1 Problem definition

In this section, we will introduce the notation and state the problem of matching source phrases of an input graph G and the phrase-table, represented as prefix tree T. The input graph G has nodes $1, ..., j, ..., J$. The outgoing edges of a graph node j are numbered with $1, ..., n, ..., N_j$, i. e. an edge in the input graph is identified by a pair (j, n). The source word labeling the n^{th} outgoing edge of graph node j is denoted as $f^G_{j,n}$ and the successor node of this edge is denoted as $s^G_{j,n} \in \{1, ..., J\}$. This notation is illustrated in Figure 5.15.

We use a similar notation for the prefix tree T with nodes $1, ..., k, ..., K$. The outgoing edges of a tree node k are numbered with $1, ..., m, ..., M_k$, i. e. an edge in the prefix tree is identified by a pair (k, m). The source word labeling the m^{th} outgoing edge of tree node k is denoted as $f^T_{k,m}$ and the successor node of this edge is denoted as $s^T_{k,m} \in \{1, ..., K\}$. Due to the tree structure, the successor nodes of a tree node k are all distinct:

$$s^T_{k,m} = s^T_{k,m'} \quad \Leftrightarrow \quad m = m' \tag{5.44}$$

Let k_0 denote the root node of the prefix tree and let $\tilde{f}_k$ denote the prefix that leads to tree node k. Furthermore, we define $E(k)$ as the set of possible translations of the source phrase $\tilde{f}_k$. These are the entries of the phrase-table, i.e.:

$$E(k) = \left\{ \tilde{e} \,\middle|\, p(\tilde{e}|\tilde{f}_k) > 0 \right\} \tag{5.45}$$

We will need similar symbols for the input graph. Therefore, we define $F(j',j)$ as the set of source phrases of all paths from graph node j' to node j, or formally:

$$F(j',j) = \left\{ \tilde{f} \,\middle|\, \exists (j_i,n_i)_{i=1}^{I} : \tilde{f} = f^G_{j_1,n_1},...,f^G_{j_I,n_I} \wedge j_1 = j' \wedge \bigwedge_{i=1}^{I-1} s^G_{j_i,n_i} = j_{i+1} \wedge s_{j_I,n_I} = j \right\} \tag{5.46}$$

Here, the conditions ensure that the edge sequence $(j_i,n_i)_{i=1}^{I}$ is a proper path from node j' to node j in the input graph and that the corresponding source phrase is $\tilde{f} = f^G_{j_1,n_1},...,f^G_{j_I,n_I}$. This definition can be expressed in a recursive way; the idea is to extend the phrases of the predecessor nodes by one word:

$$F(j',j) = \bigcup_{(j'',n):s^G_{j'',n}=j} \left\{ \tilde{f} f^G_{j'',n} \,\middle|\, \tilde{f} \in F(j',j'') \right\} \tag{5.47}$$

Here, the set is expressed as a union over all inbound edges (j'',n) of node j. We concatenate each source phrase $\tilde{f}$ that ends at the start node of such an edge, i. e. $\tilde{f} \in F(j',j'')$, with the corresponding edge label $f^G_{j'',n}$. Additionally, we define $E(j',j)$ as the set of possible translations of all paths from graph node j' to graph node j, or formally:

$$E(j',j) = \left\{ \tilde{e} \,\middle|\, \exists \tilde{f} \in F(j',j) : p(\tilde{e}|\tilde{f}) > 0 \right\} \tag{5.48}$$

$$= \bigcup_{k:\tilde{f}_k \in F(j',j)} E(k) \tag{5.49}$$

$$= \bigcup_{(j'',n):s^G_{j'',n}=j} \; \bigcup_{\substack{k:\tilde{f}_k \in F(j',j'') \\ m:f^G_{j'',n}=f^T_{k,m}}} E(s^T_{k,m}) \tag{5.50}$$

Here, the definition was first rewritten using Equation 5.45 and then using Equation 5.47. Again, the set is expressed recursively as a union over the inbound edges. For each inbound edge (j'',n), the inner union verifies that there exists a corresponding edge (k,m) in the prefix tree with the same label, i. e. $f^G_{j'',n} = f^T_{k,m}$.

Our goal is to find all non-empty sets of translation options $E(j',j)$. The naive approach would be to enumerate all paths in the input graph from node j' to node j, then lookup the corresponding source phrase in the phrase-table and add the translations, if there are any, to the set of translation options $E(j',j)$. This solution has some obvious weaknesses: the number of paths between two nodes is typically huge and the majority of the corresponding source phrases do not occur in the phrase-table.

We omitted the probabilities for notational convenience. The extensions are straightforward. Note that we store only the target phrases $\tilde{e}$ in the set of possible translations $E(j',j)$ and not the source phrases $\tilde{f}$. This is based on the assumption that the models which are conditioned on the source phrase $\tilde{f}$ are independent of the context outside the phrase pair $(\tilde{f},\tilde{e})$. This assumption holds for the standard phrase and word translation models. Thus, we have to keep only the target phrase with the highest probability. It might be violated by lexicalized distortion models (dependent on the configuration); in that case we have to store the source phrase along with the target phrase and the probability, which is again straightforward.

Figure 5.16: Algorithm `phrase-match` for matching source phrases of input graph G and prefix tree T. Input: graph G, prefix tree T, translation options $E(k)$ for all tree nodes k; output: translation options $E(j',j)$ for all graph nodes j' and j.

```
0  FOR $j' = 1$ TO $J$ DO
1      stack.push($j', k_0$)
2      WHILE not stack.empty() DO
3          $(j, k)$ = stack.pop()
4          $E(j', j) = E(j', j) \cup E(k)$
5          FOR $n = 1$ TO $N_j$ DO
6              IF ($f^G_{j,n} = \epsilon$)
7              THEN stack.push($s^G_{j,n}, k$)
8              ELSE IF ($\exists m : f^G_{j,n} = f^T_{k,m}$)
9                      THEN stack.push($s^G_{j,n}, s^T_{k,m}$)
```

5.7.2 Algorithm

The algorithm for matching the source phrases of the input graph G and the prefix tree T is presented in Figure 5.16. Starting from a graph node j', we explore the part of the graph which corresponds to known source phrase prefixes and generate the sets $E(j',j)$ incrementally based on Equation 5.50. The intermediate states are represented as pairs (j,k) meaning that there exists a path in the input graph from node j' to node j which is labeled with the source phrase $\tilde{f}_k$, i. e. the source phrase that leads to node k in the prefix tree. These intermediate states are stored on a stack. After the initialization in line 1, the main loop starts. We take one item from the stack and update the translation options $E(j',j)$ in line 4. Then, we loop over all outgoing edges of the current graph node j. For each edge, we first check if the edge is labeled with an ϵ in line 6. In this special case, we go to the successor node in the input graph $s^G_{j,n}$, but remain in the current node k of the prefix tree. In the regular case, i. e. the graph edge label is a regular word, we check in line 8 if the current prefix tree node k has an outgoing edge labeled with that word. If such an edge is found, we put a new state on the stack with the two successor nodes in the input graph $s^G_{j,n}$ and the prefix tree $s^T_{k,m}$, respectively.

5.7.3 Computational complexity

In this section, we will analyze the computational complexity of the algorithm. The computational complexity of lines 5-9 is in $\mathcal{O}(N_j \log M_k)$, i. e. it depends on the branching factors of the input graph and the prefix tree. Both are typically small. An exception is the branching factor of the root node k_0 of the prefix tree, which can be rather large, typically it is the vocabulary size of the source language. But, as described in Section 5.6, we can access the successor nodes of the root node of the prefix tree in $\mathcal{O}(1)$, i. e. in constant time. So, if we are at the root node of the prefix tree, the computational complexity of lines 5-9 is in $\mathcal{O}(N_j)$. Using hashing at the interior nodes of the prefix tree would result in a constant time lookup at these nodes as well. Nevertheless, the sorted array implementation that we chose has the advantage of faster loading from disk which seems to be more important in practice.

An alternative interpretation of lines 5-9 is that we have to compute the intersection of the two sets f_j^G and f_k^T, with

$$\begin{aligned} f_j^G &= \left\{ f_{j,n}^G \,\middle|\, n = 1, ..., N_j \right\} & (5.51) \\ f_k^T &= \left\{ f_{k,m}^T \,\middle|\, m = 1, ..., M_k \right\}. & (5.52) \end{aligned}$$

Assuming both sets are sorted, this could be done in linear time, i. e. in $\mathcal{O}(N_j + M_k)$. In our case, only the edges in the prefix tree are sorted. Obviously, we could sort the edges in the input graph and then apply the linear algorithm, resulting in an overall complexity of $\mathcal{O}(N_j \log N_j + M_k)$. As the algorithm visits nodes multiple times, we could do even better by sorting all edges of the graph during the initialization. Then, we could always apply the linear time method. On the other hand, it is unclear if this pays off in practice and an experimental comparison has to be done which we will leave for future work.

The overall complexity of the algorithm depends on how many phrases of the input graph occur in the phrase-table. In the worst case, i. e. if all phrases occur in the phrase-table, the described algorithm is not more efficient than the naive algorithm which simply enumerates all phrases. Nevertheless, this does not happen in practice and we observe an exponential speed up compared to the naive algorithm, as will be shown in Section 7.8. We made an implementation of this algorithm for confusion networks publicly available in the Moses toolkit [Koehn & Hoang$^+$ 07] [f].

[f] http://www.statmt.org/moses

6 Training

As described in Section 1.2.1, we have to address three problems [Ney 01]:

- the modeling problem, i. e. how to structure the dependencies of source and target language sentences;
- the search problem, i. e. how to find the best translation candidate among all possible target language sentences;
- the ***training problem***, i. e. how to estimate the free parameters of the models from the training data.

In this chapter, the main focus is on the training problem.

We address the problem of training the free parameters of a statistical machine translation system. We present novel training criteria based on maximum likelihood estimation and expected loss computation.

6.1 Introduction

We will compare a variety of training criteria for statistical machine translation. In particular, we are considering criteria for the log-linear parameters or model scaling factors, i.e. the λ_1^M in Equation 1.8. We will introduce new training criteria based on maximum likelihood estimation and expected loss computation.

In the following, we will discuss the so-called training problem [Ney 01]: how do we train the free parameters λ_1^M of the model? The current state-of-the-art is to use minimum error rate training (MERT) as described in [Och 03]. The free parameters are tuned to directly optimize the evaluation criterion on a development set.

Except for the MERT, the training criteria that we will consider are additive at the sentence-level. Thus, the training problem for a development set with S sentences can be formalized as:

$$\hat{\lambda}_1^M = \operatorname*{argmax}_{\lambda_1^M} \sum_{s=1}^{S} F(\lambda_1^M, (e_1^I, f_1^J)_s) \tag{6.1}$$

Here, $F(\cdot,\cdot)$ denotes the training criterion that we would like to maximize and $(e_1^I, f_1^J)_s$ denotes a sentence pair in the development set. The optimization is done using the Nelder-Mead, or downhill simplex, algorithm [Nelder & Mead 65] from the Numerical Recipes book [Press & Teukolsky$^+$ 02]. This is a general purpose optimization procedure with the advantage that it does not require the derivative information.

In [Och & Ney 02], the log-linear weights λ_1^M were tuned to maximize the mutual information criterion (MMI). The current state-of-the-art is to optimize these parameters with respect to the final evaluation criterion; this is the so-called minimum error rate training [Och 03]. [Shen & Sarkar⁺ 04] compared different algorithms for tuning the log-linear weights in a reranking framework and achieved results comparable to the standard minimum error rate training. [Tillmann & Zhang 06] describe a perceptron style *algorithm* for training millions of features. Here, we focus on the comparison of different training *criteria*. An annealed minimum risk approach is presented in [Smith & Eisner 06] which outperforms both maximum likelihood and minimum error rate training. The parameters are estimated iteratively using an annealing technique that minimizes the risk of an expected log-BLEU approximation, which is similar to the one presented here.

We will describe the details of the different training criteria in Section 6.4 and 6.5. As the training criteria are based on the used loss function (or evaluation metric), we will first discuss evaluation metrics in the following section.

6.2 Evaluation Metrics

The automatic evaluation of machine translation is currently an active research area. There exists a variety of different metrics, e.g., word error rate, position-independent word error rate, Bleu score [Papineni & Roukos⁺ 02], NIST score [Doddington 02], METEOR [Banerjee & Lavie 05, Lavie & Agarwal 07], GTM [Turian & Shen⁺ 03]. Each of them has advantages and shortcomings.

A popular metric for evaluating machine translation quality is the Bleu score. It has certain shortcomings for comparing different machine translation systems, especially if comparing conceptually different systems, e.g. phrase-based versus rule-based systems, as shown in [Callison-Burch & Osborne⁺ 06]. On the other hand, Callison-Burch concluded that the Bleu score is reliable for comparing variants of the same machine translation system. As we are going to compare variants of a phrase-based SMT system and as Bleu is currently the most popular metric, we have chosen it as our primary evaluation metric. Nevertheless, most of the methods we will present can be easily adapted to other automatic evaluation metrics.

In the following, we will briefly review the computation of the Bleu score as some of the training criteria are motivated by this. The Bleu score is a combination of the geometric mean of n-gram precisions and a brevity penalty for too short translation hypotheses. The Bleu score for a translation hypothesis e_1^I and a reference translation $\hat{e}_1^{\hat{I}}$ is computed as:

$$\text{Bleu}(e_1^I, \hat{e}_1^{\hat{I}}) = \text{BP}(I, \hat{I}) \cdot \prod_{n=1}^{4} \text{Prec}_n(e_1^I, \hat{e}_1^{\hat{I}})^{1/4} \tag{6.2}$$

with

$$\mathrm{BP}(I,\hat{I}) = \begin{cases} 1 & \text{if } I \geq \hat{I} \\ \exp\left(1 - I/\hat{I}\right) & \text{if } I < \hat{I} \end{cases} \tag{6.3}$$

$$\mathrm{Prec}_n(e_1^I, \hat{e}_1^{\hat{I}}) = \frac{\sum\limits_{w_1^n} \min\{C(w_1^n|e_1^I), C(w_1^n|\hat{e}_1^{\hat{I}})\}}{\sum\limits_{w_1^n} C(w_1^n|e_1^I)} \tag{6.4}$$

Here, $C(w_1^n|e_1^I)$ denotes the number of occurrences, i. e. the count, of an n-gram w_1^n in a sentence e_1^I. The denominators of the n-gram precisions evaluate to the number of n-grams in the hypothesis, i.e. $I - n + 1$.

The n-gram counts for the Bleu score computation are usually collected over a whole document. For our purposes, a sentence-level computation is preferable. A problem with the sentence-level Bleu score is that the score is zero if not at least one four-gram matches. As we would like to avoid this problem, we use the smoothed sentence-level Bleu score as suggested in [Lin & Och 04]. Thus, we increase the nominator and denominator of $\mathrm{Prec}_n(\cdot,\cdot)$ by one for $n > 1$. Note that we will use the sentence-level Bleu score only during training. The evaluation on the development and test sets will be carried out using the standard Bleu score, i.e. at the corpus level. As the MERT baseline does not require the use of the sentence-level Bleu score, we use the standard Bleu score for training the baseline system.

In the following, we will describe several criteria for training the log-linear parameters λ_1^M of our model. These criteria are based on N-gram and sentence length posterior probabilities which will be introduced in the next section. For notational convenience, we assume that there is just one reference translation. Nevertheless, the methods can be easily adapted to the case of multiple references.

6.3 N-Gram and Sentence Length Posterior Probabilities

6.3.1 Introduction

Word posterior probabilities are a common approach for confidence estimation in automatic speech recognition, e.g., [Wessel 02]. This idea has been adopted to estimate confidences for machine translation, e.g. [Blatz & Fitzgerald$^+$ 03, Ueffing & Macherey$^+$ 03, Blatz & Fitzgerald$^+$ 04, Ueffing 05].

We will generalize this idea and introduce n-gram posterior probabilities. Additionally, we will introduce a sentence length model based on posterior probabilities. These are needed for some of the training criteria. Additionally, they can be used as additional models in a rescoring/reranking framework.

6.3.2 N-gram posterior probabilities

The idea is similar to the word posterior probabilities: we sum up the sentence posterior probabilities for each occurrence of an n-gram. We define the fractional count $N_{\lambda_1^M}(w_1^n, f_1^J)$ of an n-gram w_1^n for a source sentence f_1^J as its weighted frequency:

$$N_{\lambda_1^M}(w_1^n, f_1^J) = \sum_{I, e_1^I} \sum_{i=1}^{I-n+1} p_{\lambda_1^M}(e_1^I | f_1^J) \cdot \delta(e_i^{i+n-1}, w_1^n) \tag{6.5}$$

The sum over the target language sentences is limited to a lattice or an N-best list, i. e. the N best translation candidates according to the baseline model. In this equation, we use the Kronecker function $\delta(\cdot, \cdot)$, i.e. the term $\delta(e_i^{i+n-1}, w_1^n)$ evaluates to one if and only if the n-gram w_1^n occurs in the target sentence e_1^I starting at position i.

The n-gram posterior distribution is obtained by normalizing the weighted frequency counts $N_{\lambda_1^M}(w_1^n, f_1^J)$ and smoothing with a uniform distribution over all possible n-grams.

$$p_{\lambda_1^M}(w_1^n | f_1^J) \;=\; \alpha \cdot \frac{N_{\lambda_1^M}(w_1^n, f_1^J)}{\sum_{w'^n_1} N_{\lambda_1^M}(w'^n_1, f_1^J)} + \;(1-\alpha) \cdot \frac{1}{V_e^n} \tag{6.6}$$

As before, V_e denotes the vocabulary size of the target language; thus, V_e^n is the number of possible n-grams in the target language. Note that the widely used word posterior probabilities are obtained as a special case, namely if n is set to one.

To predict the number of occurrences c within a translation hypothesis, we use relative frequencies smoothed with a Poisson distribution. The mean of the Poisson distribution $\mu(w_1^n, f_1^J, \lambda_1^M)$ is chosen to be the mean of the unsmoothed distribution.

$$p_{\lambda_1^M}(c | w_1^n, f_1^J) \;=\; \beta \cdot \frac{N_{\lambda_1^M}(c, w_1^n, f_1^J)}{N_{\lambda_1^M}(w_1^n, f_1^J)} + \;(1-\beta) \cdot \frac{\mu(w_1^n, f_1^J, \lambda_1^M)^c \cdot e^{-c}}{c!} \tag{6.7}$$

with

$$\mu(w_1^n, f_1^J, \lambda_1^M) = \sum_c c \cdot \frac{N_{\lambda_1^M}(c, w_1^n, f_1^J)}{N_{\lambda_1^M}(w_1^n, f_1^J)} \tag{6.8}$$

Note that in case the mean $\mu(w_1^n, f_1^J, \lambda_1^M)$ is zero, we do not need the distribution $p_{\lambda_1^M}(c | w_1^n, f_1^J)$. The smoothing parameters α and β are both set to 0.9.

6.3.3 Sentence length posterior probability

The common phrase-based translation systems, such as [Och & Tillmann$^+$ 99, Koehn 04a], do not use an explicit sentence length model. Only the simple word penalty goes into that direction. It can be adjusted to prefer longer or shorter translations.

Here, we will use the posterior probability of a specific target sentence length I as length model:

$$p_{\lambda_1^M}(I | f_1^J) = \sum_{e_1^I} p_{\lambda_1^M}(e_1^I | f_1^J) \tag{6.9}$$

Note that the sum is carried out only over target sentences e_1^I with the specific length I. Again, the candidate target language sentences are limited to a lattice or an N-best list.

6.3.4 Rescoring/reranking

A straightforward application of the posterior probabilities is to use them as additional features in a rescoring/reranking approach. N-best lists are suitable for easily applying several rescoring techniques since the hypotheses are already fully generated. In comparison, word graph rescoring techniques need specialized tools which can traverse the graph accordingly.

The n-gram posterior probabilities can be used similar to an n-gram language model:

$$h_n(f_1^J, e_1^I) = \frac{1}{I} \log \left(\prod_{i=1}^{I} p(e_i | e_{i-n+1}^{i-1}, f_1^J) \right) \tag{6.10}$$

with:

$$p(e_i | e_{i-n+1}^{i-1}, f_1^J) = \frac{N_{\lambda_1^M}(e_{i-n+1}^{i}, f_1^J)}{N_{\lambda_1^M}(e_{i-n+1}^{i-1}, f_1^J)} \tag{6.11}$$

Note that the models do not require smoothing as long as they are applied to the same N-best list they are trained on.

If the models are used for unseen sentences, smoothing is important to avoid zero probabilities. We use a linear interpolation with weights α_n and the smoothed $(n-1)$-gram model as generalized distribution.

$$p_n(e_i | e_{i-n+1}^{i-1}, f_1^J) = \alpha_n \cdot \frac{C(e_{i-n+1}^{i}, f_1^J)}{C(e_{i-n+1}^{i-1}, f_1^J)} + (1 - \alpha_n) \cdot p_{n-1}(e_i | e_{i-n+2}^{i-1}, f_1^J) \tag{6.12}$$

An alternative usage of the n-gram posterior probabilities in rescoring is to simply accumulate the logarithm of the posterior probabilities of the n-grams of the translation candidate:

$$h_n(f_1^J, e_1^I) = \frac{1}{I - n + 1} \sum_{i=1}^{I-n+1} \log p_{\lambda_1^M}(e_i^{i+n-1} | f_1^J) \tag{6.13}$$

Experimentally, it turned out that both variants yield similar results, but the second one seems to be somewhat more stable. Therefore, we will use Equation 6.13 in the experiments.

The usage of the sentence length posterior probability for rescoring is even simpler. The resulting feature is:

$$h_L(f_1^J, e_1^I) = \log p_{\lambda_1^M}(I | f_1^J) \tag{6.14}$$

Again, the model does not require smoothing as long as it is applied to the same N-best list it is trained on. If it is applied to other sentences, smoothing becomes important. We propose smoothing the sentence length model with a Poisson distribution.

$$p_\beta(I | f_1^J) = \beta \cdot p(I | f_1^J) + (1 - \beta) \cdot \frac{\mu^I \exp(-\mu)}{I!} \tag{6.15}$$

We use a linear interpolation with weight β. The mean of the Poisson distribution μ is chosen to be the mean of the unsmoothed length model:

$$\mu = \sum_I I \cdot p_{\lambda_1^M}(I|f_1^J) \tag{6.16}$$

6.4 Maximum Likelihood

6.4.1 Sentence-level computation

A popular approach for training parameters is maximum likelihood (ML) estimation. Here, the goal is to maximize the joint likelihood of the parameters and the training data. For log-linear models, this results in a nice optimization criterion which is convex and has a single optimum. It is equivalent to the maximum mutual information (MMI) criterion. We obtain the following training criterion:

$$F_{ML-S}(\lambda_1^M, (e_1^I, f_1^J)) = \log p_{\lambda_1^M}(e_1^I|f_1^J) \tag{6.17}$$

A problem that we often face in practice is that the correct translation e_1^I might not be among the candidates that our MT system produces. Therefore, [Och & Ney 02] defined the translation candidate with the minimum word-error rate as pseudo reference translation. This has some bias towards minimizing the word-error rate. Here, we will use the translation candidate with the maximum Bleu score as pseudo reference to bias the system towards the Bleu score. However, as pointed out in [Och 03], there is no reason to believe that the resulting parameters are *optimal* with respect to translation quality measured with the Bleu score.

The goal of this sentence-level criterion is to discriminate the single correct translation against all the other "incorrect" translations. This is problematic as, even for human experts, it is very hard to define a single best translation of a sentence. Furthermore, the alternative target language sentences are not all equally bad translations. Some of them might be very close to the correct translation or even equivalent whereas other sentences may have a completely different meaning. The sentence-level ML criterion does not distinguish these cases and is therefore a rather harsh training criterion. To overcome these limitation, we propose an n-gram level computation of the log-likelihood, which will be described in the next section.

6.4.2 N-gram level computation

As an alternative to the sentence-level ML estimation, we performed experiments with an n-gram level ML estimation. Here, we limit the order of the n-grams and assume conditional independence among the n-gram probabilities. We define the following training criterion for a target language sentence e_1^I given a source language sentence f_1^J:

$$F_{ML-N}(\lambda_1^M, (e_1^I, f_1^J)) = \sum_{n=1}^{N} \sum_{i=1}^{I-n+1} \log p_{\lambda_1^M}(e_i^{i+n-1}|f_1^J) \tag{6.18}$$

Here, we use the n-gram posterior probability $p_{\lambda_1^M}(e_i^{i+n-1}|f_1^J)$ as defined in Section 6.3.2. An advantage of the n-gram level computation is that we do not have to define pseudo-references as for the sentence-level ML estimation. We can easily compute this criterion for the human reference translation. Furthermore, this criterion has the desirable property that it takes partial correctness into account, i. e. it is not as harsh as the sentence-level criterion.

6.5 Expected Bleu Score

According to statistical decision theory, one should maximize the expected gain (or equivalently minimize the expected loss). For machine translation, this means that we should optimize the expected Bleu score, or any other preferred evaluation metric.

6.5.1 Sentence-level computation

The expected Bleu score for a given source sentence f_1^J and a reference translation $\hat{e}_1^{\hat{I}}$ is defined as:

$$\mathbb{E}[\text{Bleu}|\hat{e}_1^{\hat{I}}, f_1^J] = \sum_{e_1^I} Pr(e_1^I|f_1^J) \cdot \text{Bleu}(e_1^I, \hat{e}_1^{\hat{I}}) \tag{6.19}$$

Here, $Pr(e_1^I|f_1^J)$ denotes the true probability distribution that e_1^I is a correct translation of the given source sentence f_1^J. As this probability distribution is unknown, we approximate it using the log-linear translation model $p_{\lambda_1^M}(e_1^I|f_1^J)$ from Equation 1.8. Furthermore, the computation of the expected Bleu score involves a sum over all possible translations e_1^I. This sum is approximated using an N-best list, i. e. the N best translation hypotheses of the MT system. Thus, the training criterion for the sentence-level expected Bleu computation is:

$$F_{EB-S}(\lambda_1^M, (\hat{e}_1^{\hat{I}}, f_1^J)) = \sum_{e_1^I} p_{\lambda_1^M}(e_1^I|f_1^J) \cdot \text{Bleu}(e_1^I, \hat{e}_1^{\hat{I}}) \tag{6.20}$$

An advantage of the sentence-level computation is that it is straightforward to plug in alternative evaluation metrics instead of the Bleu score. Note that the minimum error rate training [Och 03] uses only the target sentence hypothesis with the *maximum* posterior probability whereas here, the whole probability *distribution* is taken into account.

6.5.2 N-gram level computation

In this section, we describe a more fine grained computation of the expected Bleu score by exploiting its particular structure. Hence, this derivation is specific for the Bleu score but should be easily adaptable to other n-gram based metrics. We can rewrite the expected Bleu score as:

$$\mathbb{E}[\text{Bleu}|\hat{e}_1^{\hat{I}}, f_1^J] \;=\; \mathbb{E}[\text{BP}|\hat{I}, f_1^J] \cdot \prod_{n=1}^{4} \mathbb{E}[\text{Prec}_n|\hat{e}_1^{\hat{I}}, f_1^J]^{1/4} \tag{6.21}$$

We assumed conditional independence between the brevity penalty BP and the n-gram precisions Prec_n. Note that although these independence assumptions do not hold, the resulting parameters might work well for translation. In fact, we will show that this criterion is among the best performing ones in Section 7.6. This type of independence assumption is typical within the naive Bayes classifier framework [Wasserman 05]. The resulting training criterion that we will use in Equation 6.1 is then:

$$F_{EB-N}(\lambda_1^M, (\hat{e}_1^{\hat{I}}, f_1^J)) \;=\; \mathbb{E}_{\lambda_1^M}[\text{BP}|\hat{I}, f_1^J] \cdot \prod_{n=1}^{4} \mathbb{E}_{\lambda_1^M}[\text{Prec}_n|\hat{e}_1^{\hat{I}}, f_1^J]^{1/4} \tag{6.22}$$

We still have to define the estimators for the expected brevity penalty $\mathbb{E}_{\lambda_1^M}[\text{BP}|\hat{I}, f_1^J]$ as well as the expected n-gram precision $\mathbb{E}_{\lambda_1^M}[\text{Prec}_n|\hat{e}_1^{\hat{I}}, f_1^J]$:

$$\mathbb{E}_{\lambda_1^M}[\text{BP}|\hat{I}, f_1^J] \;=\; \sum_I \text{BP}(I, \hat{I}) \cdot p_{\lambda_1^M}(I|f_1^J) \tag{6.23}$$

$$\mathbb{E}_{\lambda_1^M}[\text{Prec}_n|\hat{e}_1^{\hat{I}}, f_1^J] \;=\; \frac{\sum\limits_{w_1^n} p_{\lambda_1^M}(w_1^n|f_1^J) \sum\limits_c \min\{c, C(w_1^n|\hat{e}_1^{\hat{I}})\} \cdot p_{\lambda_1^M}(c|w_1^n, f_1^J)}{\sum\limits_{w_1^n} p_{\lambda_1^M}(w_1^n|f_1^J) \sum\limits_c c \cdot p_{\lambda_1^M}(c|w_1^n, f_1^J)} \tag{6.24}$$

Here, we use the sentence length posterior probability $p_{\lambda_1^M}(I|f_1^J)$ as defined in Section 6.3.3 and the n-gram posterior probability $p_{\lambda_1^M}(w_1^n|f_1^J)$ as described in Section 6.3.2. Additionally, we predict the number of occurrences c of an n-gram. This information is necessary for the so-called clipping in the Bleu score computation, i.e. the min operator in the nominator of formulae Equation 6.4 and Equation 6.24. The denominator of Equation 6.24 is the expected number of n-grams in the target sentence, whereas the nominator denotes the expected number of correct n-grams.

7 Results

7.1 Evaluation Criteria

Recently, research has focused on MT evaluation metrics which resulted in a variety of different metrics. Nevertheless, so far, a single generally accepted criterion for the evaluation of machine translation does not exist. Therefore, we use a variety of different criteria.

- Error rates:
    - WER (word error rate):
      The WER is computed as the minimum number of substitution, insertion and deletion operations that have to be performed to convert the generated sentence into the reference sentence. This performance criterion is widely used in automatic speech recognition.
    - PER (position-independent word error rate) [Tillmann & Vogel$^+$ 97]:
      A shortcoming of the WER is that it requires a perfect word order. The word order of an acceptable sentence can be different from that of the target sentence, so that the WER measure alone could be misleading. The PER compares the words in the two sentences ignoring the word order.
    - TER (translation edit rate) [Snover & Dorr$^+$ 06]:
      The TER is an extension of the WER. In addition to the standard edit operations substitutions, insertions and deletions a new operation is introduced: shifts of whole phrases are permitted.
- Accuracy measures:
    - Bleu score [Papineni & Roukos$^+$ 02]:
      This score measures the precision of unigrams, bigrams, trigrams and fourgrams with respect to a reference translation with a penalty for too short sentences.
    - NIST score [Doddington 02]:
      This score is similar to Bleu. It is a weighted n-gram precision in combination with a penalty for too short sentences.

If available, we use multiple reference to compute these criteria. In our experiments, we use the Bleu score as primary criterion. It has been shown that the Bleu score has a high correlation with human judgement. The Bleu score is also use in many MT evaluations

Table 7.1: Corpus statistics of the BTEC task (OOV: out-of-vocabulary tokens).

		Arabic	Chinese	Japanese	English
Train	Sentence pairs	20 000			
	Running words	180 075	176 199	198 453	189 927
	Vocabulary size	15 371	8 687	9 277	6 870
	Singletons	8 319	4 006	4 431	2 888
C-Star'03	Sentences	506			
	Running Words	3 552	3 630	4 130	3 823
	OOV	133	114	61	65

as official criterion. The Bleu and NIST scores are computed using the `mteval-v11b.pl`[a] tool. Note that there is a difference in the way the brevity penalty is computed in the `mteval-v11b.pl` compared to the original implementation from IBM. In the IBM implementation, the reference length which is closest to the hypothesis length is chosen, whereas in the `mteval-v11b.pl` tool always the shortest reference length is chosen. If not mentioned otherwise, we will report all error measures *case-insensitive*.

Recently, the Bleu score has been critized, e. g. in [Callison-Burch & Osborne$^+$ 06], for favoring statistical systems over non-statistical system (or maybe better phrase-based systems over non-phrasebased system). On the other hand, they have also shown that the Bleu score is appropriate for comparing variants of the same system. As we are going to compare variants of a phrase-based SMT system, the Bleu score is well suited for our purposes.

We also conducted statistical significance tests and report statistical confidence intervals. These were computed using bootstrap resampling [Koehn 04b, Zhang & Vogel 05]. The tool for computing the confidence intervals was kindly provided by the National Research Council Canada.

7.2 Task Description and Corpus Statistics

7.2.1 BTEC

The *Basic Travel Expression Corpus* (BTEC) [Takezawa & Sumita$^+$ 02] is a multilingual speech corpus which contains tourism-related sentences similar to those that are found in phrase books. We use the Arabic-English, the Chinese-English and the Japanese-English data. The corpus statistics are shown in Table 7.1. This corpus was made available to the IWSLT 2005 evaluation participants.

As the BTEC is a rather clean corpus, the preprocessing consisted mainly of tokenization, i.e., separating punctuation marks from words. Additionally, we replaced contractions such as *it's* or *I'm* in the English corpus and we removed the case information. For

[a] http://www.nist.gov/speech/tests/mt/resources/scoring.htm

Arabic, we removed the diacritics and we split common prefixes: Al, w, f, b, l. There was no special preprocessing for the Chinese and the Japanese training corpora.

7.2.2 NIST: Chinese-English

Additional experiments were carried out on the large data track of the Chinese-English NIST task. The corpus statistics of the bilingual training corpus are shown in Table 7.2. The language model was trained on the English part of the bilingual training corpus and additional monolingual English data from the GigaWord corpus. We use modified Kneser-Ney smoothing as implemented in the SRILM toolkit [Stolcke 02]. We use the default setting for discarding low-frequency n-grams, which means that singletons are discarded for order three and higher. Each evaluation set consists of 100 news stories from different news agencies. The NIST 2004 evaluation set additionally contains 50 editorials and 50 speeches. For the four English reference translations of the evaluation sets, the accumulated statistics are presented. As larger test sets yield more reliable results, we will also report results on the combined NIST 2003-2005 evaluation sets. For these experiments, we concatenated the three sets and compute the error measures on the resulting larger corpus. We also report the number of out-of-vocabulary (OOV) words for each evaluation set. Note that the OOVs are counted after word segmentation; it may happen that, for instance, an unknown word is incorrectly segmented into two known words. This explains the very small number of OOVs. A human inspection showed that most of them are English abbreviations.

The data is preprocessed in the following way. We word segment the Chinese corpus using the LDC word segmentation tool. The English corpus is tokenized, i. e. we separate words and punctuation marks, and converted to lowercase. Then, we apply text normalization, e. g. of abbreviations, contractions. We detect numbers and dates and replace them with a special number or date token, respectively. This is done for the Chinese and the English corpus. Long sentence pairs pose a problem for the GIZA++ training and are therefore split into shorter segments using the method described in [Xu & Zens^{+} 05, Xu & Zens^{+} 06]. The original eight million sentence pairs are split into about 20 million segments.

The LDC word segmenter uses a word list with about 44 K entries. Therefore, the Chinese vocabulary size of 251 K maybe somewhat surprising. In Table 7.3, we show an analysis of the Chinese vocabulary. It turns out that many entries in the Chinese vocabulary are non-Chinese words or numbers. Nevertheless, these account only for a small fractions of the Chinese corpus. About 96.9% of the Chinese corpus are Chinese words (82.0%), punctuation marks (13.5%) or number/date categories (2.4%). For this analysis, we used the following simplifying definitions. Chinese words are defined as entries that consist of Chinese characters. Punctuation marks are all entries consisting of a single punctuation mark, e. g. ?!,();:. Non-Chinese words are all entries that start with a letter. Categories are the two labels for number and date expressions. Numbers are all entries that start with a digit, but are not recognized as a number during the preprocessing.

We generate word alignments using GIZA++ in both directions and combine the two alignments using a refined heuristic. From this word-aligned bilingual corpus, we extract the phrase pairs using the method described in Section 4.2. To increase the likelihood that

Table 7.2: Corpus statistics of the Chinese-English NIST task (OOV: out-of-vocabulary tokens).

			Chinese	English
Train	Sentence pairs		8 M	
	Segments after splitting		20 M	
	Running words		249 M	269 M
	Vocabulary size		251 K	431 K
	Singletons		110 K	161 K
Eval	2002	Sentences	878	3 512
		Running Words	25 K	105 K
		OOV	3	
	2003	Sentences	919	3 676
		Running Words	26 K	122 K
		OOV	13	
	2004	Sentences	1 788	7 152
		Running Words	52 K	245 K
		OOV	17	
	2005	Sentences	1 082	4 328
		Running Words	33 K	148 K
		OOV	7	

Table 7.3: Analysis of the Chinese vocabulary for the NIST task.

	Types		Tokens	
	absolute	relative [%]	absolute	relative [%]
Chinese words	48 K	19.1	204.1 M	82.0
Punctuation marks	26	<0.1	33.7 M	13.5
Non-Chinese words	127 K	50.6	5.9 M	2.4
Categories	2	<0.1	3.4 M	1.4
Numbers	76 K	30.3	1.7 M	0.7
Other	<1 K	0.3	0.1 M	<0.1

the number and date categories are aligned to each other, we add an artificial sentence pair per category consisting only of the category label and assign a high weight to this sentence pair. Due to alignment errors, it may still happen that an extracted phrase pair is inconsistent w.r.t the categories, e. g. the source contains a number category, but the target phrase does not. We remove those phrase pairs.

As the MT system is trained on lower case data, we have to restore the case information after translation. This is done using the SRI DISAMBIG tool and a fourgram language model (trained with case information of course).

Language Model. As mentioned before, the language model is trained on additional monolingual data from GigaWord Second Edition (LDC2005T12). As the GigaWord

Table 7.4: Language model perplexities, out-of-vocabulary words (OOV) and memory usage in MegaByte (MB) for different LM orders for the NIST task.

	NIST test set				
Order	2002	2003	2004	2005	Memory [MB]
3	94.8	97.5	90.1	101.8	594
4	82.8	86.3	79.0	88.5	2 191
5	80.5	83.8	76.9	86.3	4 898
6	79.5	83.5	76.6	86.1	8 044
OOV	1 284 (1.2%)	1 534 (1.4%)	1 527 (0.7%)	1 470 (1.1%)	

Second Edition data overlaps with the periods from which the NIST evaluation data was chosen, we excluded all data from those months. Furthermore, we found only the Xinhua and Agence-France Press parts of GigaWord useful. We apply the same preprocessing as for the English part of the bilingual training data, including the categorization of numbers and dates. The overall language model training data consists of about 29.5 million sentences and of about 653 million words.

In Table 7.4, we show the perplexities of different n-gram orders for the references of the NIST evaluation sets 2002-2005. Additionally, we show the memory usage of the language models. For this purpose, we filter the n-grams of the language model using the vocabulary of the MT system. This way, we load only n-grams that may be used during translation and reduce the memory consumption.

We observe that, compared to the fourgram, the fivegram and sixgram LM yield only small reductions of the perplexity, but increase the memory consumption considerably. Therefore, most of our experiments will be carried out using the fourgram LM.

Standard System. The majority of translation experiments was carried out on the NIST task. Therefore, we now describe the system that is used as starting point in those experiments.

- We use the following models:
  - the phrase-based model in both directions $p(\tilde{f}|\tilde{e})$ and $p(\tilde{e}|\tilde{f})$, cf. Section 4.3.1
  - the noisy-or lexicon model in the direction $p(e|\tilde{f})$, cf. Section 4.3.4
  - the word and phrase penalty, cf. Section 4.3.6
  - the distortion penalty model, cf. Section 4.4.1
  - the fourgram language model, cf. Section 4.4.2
  - the phrase orientation model, cf. Section 4.4.3
- Search: we use a distortion limit of 10, in addition we use the phrase-level IBM constraints with window 1. We will show later that rather small beam sizes are sufficient. Nevertheless, to be on the save side, we use rather conservative pruning and limit the beam to 16 lexical hypotheses per coverage hypothesis and 4096 total

Table 7.5: Corpus statistics of the Chinese-English TC-Star task (OOV: out-of-vocabulary tokens).

			Chinese	English
Train	Sentence pairs		7 M	
	Running words		197 M	238 M
	Vocabulary size		224 K	389 K
	Singletons		99 K	165 K
Dev		Sentences	1 019	2 038
		Running words	26 K	51 K
		OOV	3	
Eval	2006	Sentences	1 232	2 464
		Running words	30 K	62 K
		OOV	0	
	2007	Sentences	917	1 834
		Running words	21 K	45 K
		OOV	1	

hypotheses per source word, i. e. we use lexical pruning per coverage with a histogram size of $N_L = 16$ and lexical pruning per cardinality with a histogram size of $N_c = 4096$. We use the 50 most promising target phrases per source phrase, i. e. we use observation histogram pruning with size $N_o = 50$. No other pruning is applied in the standard system. A rest score estimate is used for the translation models, the language model and the distortion model. This estimate is computed per sequence of uncovered positions (cf. Equation 5.21).

7.2.3 TC-Star: Chinese-English

We perform translation experiments on the Chinese-English TC-Star task. This is a broadcast news speech translation task used within the European Union project TC-Star[b]. The bilingual training data consists of virtually all publicly available LDC Chinese-English corpora. The sixgram language model was trained on the English part of the bilingual training data and additional monolingual English parts from the GigaWord corpus.

Annual public evaluations were carried out for this task within the TC-Star project. We will report results on manual transcriptions, i. e. the so-called verbatim condition, of the official evaluation test sets of the years 2006 and 2007. There are two reference translations available for the development and test sets. The corpus statistics are shown in Table 7.5.

[b] http://www.tc-star.org

Table 7.6: Corpus statistics of the Spanish-English EPPS task.

Train	Spanish	English
Sentence pairs	1.2 M	
Running words	31 M	30 M
Vocabulary size	140 K	94 K

Test confusion networks	Full	Pruned
Sentences	1 071	
Avg. length	23.6	
Avg. / max. depth	2.7 / 136	1.3 / 11
Avg. number of paths	10^{75}	264 K

7.2.4 TC-Star: Spanish-English

The experiments for speech translation were conducted on the European Parliament Plenary Sessions (EPPS) task. This is a Spanish-English speech-to-speech translation task collected within the TC-Star project. The training corpus statistics are presented in Table 7.6. The phrase-tables for this task were kindly provided by FBK-IRST.

We evaluate the `phrase-match` algorithm described in Section 5.7 in the context of confusion network (CN) decoding [Bertoldi 05, Bertoldi & Federico 05, Shen & Zens+ 06, Bertoldi & Zens+ 07], which is one approach to speech translation. CNs [Mangu & Brill+ 00] are interesting for MT because the reordering can be done similar to text input. For more details on CN decoding, please refer to [Shen & Zens+ 06, Bertoldi & Zens+ 07]. Note that the `phrase-match` algorithm is not limited to CNs, but can work on arbitrary word graphs.

Statistics of the CNs are also presented in Table 7.6. We distinguish between the full CNs and pruned CNs. The pruning parameters were chosen such that the resulting CNs are similar in size to the largest ones in [Bertoldi & Federico 05]. The average depth of the full CNs, i. e. the average number of alternatives per position, is about 2.7 words whereas the maximum is as high as 136 alternatives.

7.3 Phrase-table Representation

In this section, we present an empirical analysis of the described data structure for the large data track of the Chinese-English NIST task. In Table 7.7, we present the translation quality as a function of the maximum source phrase length. We observe a large improvement when going beyond length 1, but this flattens out very fast. Using phrases of lengths larger than 4 or 5 does not result in significant improvements. Even a length limit of 3, as proposed by [Koehn & Och+ 03], would result in good translation quality. In the following experiments on this task, we will use a limit of 5 for the source phrase length.

In Table 7.8, we present statistics about the extracted phrase pairs for the Chinese–English

Table 7.7: Effect of the maximum source phrase length on the translation performance for the Chinese-English NIST task (NIST 2002 test set, 878 sentences).

max. source phrase length	BLEU[%]	TER[%]	NIST	WER[%]	PER[%]
1	29.7	64.1	8.60	73.1	47.8
2	37.4	56.2	9.89	62.3	40.2
3	38.4	55.1	9.86	60.4	39.8
4	38.7	54.9	9.85	60.1	39.6
5	38.7	54.8	9.83	60.0	39.6
∞	38.6	55.0	9.83	60.2	39.9

Table 7.8: Phrase-table statistics for the Chinese-English NIST task. The number of target phrases per source phrase is limited to 50.

src	number of distinct				avg. tgt
len	src phrases		src-tgt pairs		candidates
	total	singletons	total	singletons	
1	251 076	61 461 (24%)	3 033 385	757 953 (25%)	12.1
2	5 822 431	2 487 004 (43%)	33 125 407	10 438 214 (32%)	5.7
3	22 900 515	13 537 198 (59%)	65 418 364	25 210 538 (39%)	2.9
4	33 769 124	23 954 096 (71%)	67 645 416	34 721 727 (51%)	2.0
5	33 596 403	26 440 833 (79%)	57 472 848	36 784 830 (64%)	1.7
total	96 339 549	66 480 592 (69%)	226 695 420	107 913 262 (48%)	2.4

NIST task as a function of the source phrase length, in this case for length 1-5. The phrases are not limited to a specific test set. We show the number of distinct source phrases, the number of distinct source-target phrase pairs and the average number of target phrases (or translation candidates) per source phrase. We also show the number and percentage of singletons for each phrase length. We have limited the number of translation candidates per source phrase to 50 (observation histogram pruning). Thus, for each source phrase, we store only the 50 most promising target phrases. This explains why the number of singletons for the source-target phrase pairs is rather low. We store a total of about 96 million distinct source phrases and more than 226 million distinct source-target phrase pairs in the described data structure. Without limiting the translation candidates to the top 50, the total number of phrase pairs would be 519 million. If loaded completely, this phrase-table would require large amounts of memory. As we use a binary format that resembles the data structures in memory, we can use the file sizes as an estimate. In this case, the required memory would be about 15 GB. Although this is possible with nowadays computers, it would be waste of memory as most phrase-pairs are not required to translate a specific text. Additionally, the loading time would be quite long. On the other hand, using on-demand loading, we are able to utilize all these phrase pairs with minimal memory usage and virtually no initialization time.

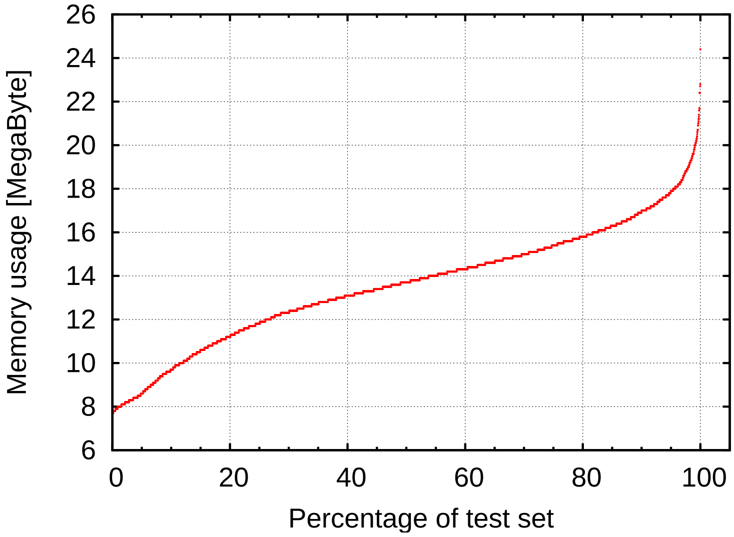


Figure 7.1: Phrase-table memory usage per sentence (sorted) for the Chinese-English NIST task (2002-2005 test sets, 4667 sentences).

In Figure 7.1, we show the memory usage of the described phrase-table data structure per sentence for the four NIST test set 2002-2005. We translated each sentence and measured the memory consumption of the phrase-table. The sentences were sorted according to the memory usage. The maximum amount of memory for the phrase-table is 24 MB. Compared to the 15 GB for storing the whole phrase-table, this is a reduction of about three orders of a magnitude. Storing all phrase pairs for the 2002 NIST test set in memory requires about 1.7 GB of memory, i. e. using the described data structures, we not only avoid the limitation to a specific test set, but we also reduce the memory requirements by about two orders of a magnitude.

Another important aspect that should be considered is translation speed. In our experiments, the translation time using the described data structure is *not* significantly slower than the traditional approach. We attribute this to the fact that we use a binary format that is a memory map of the data structure used internally and that we load the data in rather large, coherent chunks. Additionally, there is virtually no initialization time for the phrase-table which decreases the overhead of parallelization and therefore speeds up the development cycle.

7.4 Effect of different Models

To investigate the effect of different phrase models, we carried out experiments on the Chinese-English NIST task; the corpus statistics are in Table 7.2. The NIST 2002 set was used to tune the model scaling factors. The NIST evaluation sets of the years 2003-2005 are used as blind test set. The Bleu scores are reported in Table 7.9; we also report the Bleu score for the combination of all three test set ('all'). The 95% confidence interval

Table 7.9: Effect of different models on the translation quality (BLEUr4n4[%]) for the Chinese-English NIST task.

		Dev.	Test			
Search	Models	2002	2003	2004	2005	all
monotonic	4-gram LM + phrase model $p(\tilde{f}\|\tilde{e})$	31.9	30.6	29.1	29.5	29.6
	+ word penalty	32.0	30.7	31.3	30.3	30.8
	+ phrase model $p(\tilde{e}\|\tilde{f})$	33.4	32.6	31.3	31.1	31.5
	+ phrase penalty	34.0	33.1	31.7	30.9	31.8
	+ noisy-or word lexicon $p(e\|\tilde{f})$	35.4	34.4	34.1	33.3	34.0
non-monotonic	+ distortion penalty	37.4	36.1	36.9	35.1	36.3
	+ phrase orientation	38.7	37.6	38.0	36.2	37.6
	+ 6-gram instead of 4-gram	38.8	38.1	38.9	36.8	38.1
lattice	+ n-gram/length posteriors	40.6	39.8	39.9	37.8	39.5

Table 7.10: Effect of different models on the translation quality for the Chinese-English NIST task ('all' test set, 3789 sentences). The results are in percent except for the NIST score.

Search	Models	BLEU	TER	NIST	WER	PER
monotonic	4-gram LM + phrase model $p(\tilde{f}\|\tilde{e})$	29.6	61.0	8.79	64.1	44.2
	+ word penalty	30.8	61.7	9.22	66.1	44.4
	+ phrase model $p(\tilde{e}\|\tilde{f})$	31.5	59.7	9.07	63.2	43.1
	+ phrase penalty	31.8	59.0	9.07	62.6	42.6
	+ noisy-or word lexicon $p(e\|\tilde{f})$	34.0	57.7	9.64	61.9	41.0
non-monotonic	+ distortion penalty	36.3	57.1	10.04	62.0	39.8
	+ phrase orientation	37.6	56.0	10.12	60.5	39.8
	+ 6-gram instead of 4-gram	38.1	55.8	10.26	60.3	39.7
lattice	+ n-gram/length posteriors	39.5	54.9	10.36	59.4	39.2

for the development set is 1.08 % Bleu, whereas for the test set ('all') the 95% confidence interval is 0.54 % Bleu. The difference in the size of the confidence intervals is due to the different sizes of the two sets. In Table 7.10, we reported additional error measures for the combined test set ('all').

We started with a simple system using just a fourgram language model and the phrase model $p(\tilde{f}|\tilde{e})$ and then added models one by one. To reduce the computational requirements, the first set of experiments were carried out using monotonic decoding and a fourgram language model. As we are comparing only phrase models that mainly affect the lexical choice, this should not affect the conclusions. The effect of non-monotonic decoding and higher order language models is then shown in the lower part of Table 7.9 and Table 7.10.

Table 7.11: Effect of different lexicon models on the translation quality for the Chinese-English NIST task ('all' test set, 3789 sentences). The results are in percent except for the NIST score.

Model		BLEU	TER	NIST	WER	PER
IBM-1	$p(f\|\tilde{e})$	33.1	58.4	9.31	62.2	42.0
	$p(e\|\tilde{f})$	33.4	57.5	9.49	61.2	41.0
	both	32.9	57.9	9.34	61.6	41.5
Noisy-or	$p(f\|\tilde{e})$	33.2	58.0	9.48	62.0	41.4
	$p(e\|\tilde{f})$	34.0	57.7	9.64	61.9	41.0
	both	33.9	57.3	9.61	61.4	40.7

Both, the phrase translation model $p(\tilde{e}|\tilde{f})$ and the lexicon model $p(e|\tilde{f})$ result in an improvement of 1.4 Bleu points on the NIST 2002 set. We found that using the noisy-or word lexicon $p(e|\tilde{f})$ performed best among the different word-based lexicon models described in Section 4.3. Using an additional lexicon model, e. g. $p(f|\tilde{e})$, did not result in further significant improvements. It is preferable to have fewer models, as there is less chance of overfitting; therefore, we do *not* use a lexicon for the $p(f|\tilde{e})$ direction. The translation results for these alternative lexicon models on the 'all' test set are shown in Table 7.11. [Foster & Kuhn$^+$ 06] also found that the noisy-or lexicon model, which they call Zens-Ney smoothing, outperforms the IBM-style word lexicon variant.

The effect of different reordering models is shown in the lower part of Table 7.9 and Table 7.10. Using non-monotonic decoding with the distortion penalty model, we obtain a Bleu score of 36.3% on the 'all' test set, which is an improvement of about 2.3 Bleu points over the monotonic decoding. Adding the orientation model results in an additional improvement of 1.3 Bleu points on the 'all' test set. Replacing the fourgram language model with a sixgram, we obtain a Bleu score of 38.1%. Adding the n-gram and sentence length posterior probabilities, we observe an improvement of 1.4 Bleu points on the 'all' test set from 38.1% Bleu to 39.5% Bleu. Here, we used the n-gram posterior probabilities of unigram, bigram, trigrams and fourgrams, i. e. for $n = 1, 2, 3, 4$. The n-gram and sentence length posterior probabilities were computed on lattices and used during the N-best list generation to rank the candidates.

To ensure that the lattices and the N-best lists contain good translation candidates, we will present oracle Bleu scores for the lattices and the N-best lists. Thus, we select the translation candidate which results in the best Bleu score from the lattice or N-best list, respectively. Note that in these experiments we make use of the reference translations to select the best hypotheses. Thus, it is not possible to apply this to unseen test data. The purpose of these experiments is to ensure that the lattices and N-best lists contain translation candidates that are significantly better than the single-best hypotheses. This is important if the lattices and N-best lists are used in a rescoring/reranking framework as the oracle Bleu score is the theoretical upper bound that can be achieved. As pointed out in [Och & Gildea$^+$ 04], the oracle Bleu score is a *very* optimistic upper bound.

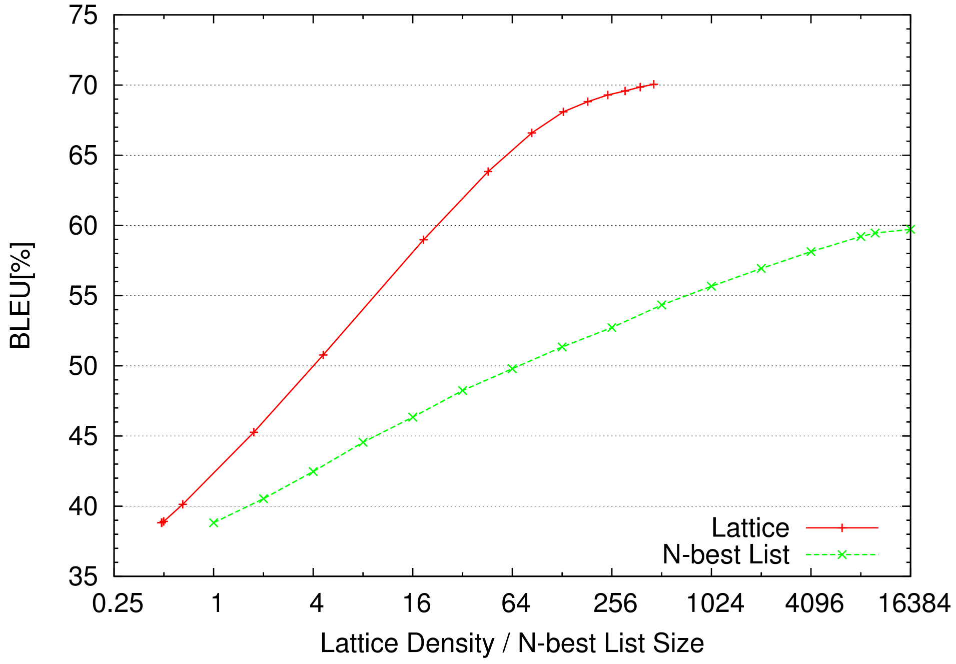


Figure 7.2: Effect of the lattice density and N-best list size on the oracle Bleu score for the Chinese-English NIST task (NIST 2002 test set, 878 sentences).

In Figure 7.2, we show the effect of the lattice density and the N-best list size on the oracle Bleu score on the Chinese-English NIST task. The lattice density is computed as the number of edges in the lattice divided by the number of source words in the input text. As the edges are labeled with phrases, the lattice density can be less than 1. The oracle Bleu score grows logarithmic with the lattice density and reaches about 70% Bleu. Also for the N-best lists, the oracle Bleu score grow logarithmic with the size, or linear in $\log N$. For the 16 K-best list an oracle Bleu score of about 60% is reached.

In all these experiments, the average phrase length of the phrases that are used to generate the best translation hypotheses ranges between 1.9 and 2.1 words per phrase, both for the source phrases and for the target phrases. The model scaling factors that were used in these experiments are reported in Table 7.12. We normalized them using the L_1 norm, i. e. the absolute values sum up to one. Each line corresponds to one setting.

In Table 7.13, we show some translation examples comparing monotonic and non-monotonic search. In case of monotonic search, the word order is often incorrect. In Table 7.14, we present some translation examples and compare the results using a trigram LM and a sixgram LM. Some translation examples showing the effect of the n-gram and sentence length posteriors are in Table 7.15.

Table 7.12: Model scaling factors for the Chinese-English NIST task. Each line represents one setting. The model scaling factors are normalized such that the absolute values sum up to one (L_1 norm).

LM	$p(\tilde{f}\|\tilde{e})$	WP	$p(\tilde{e}\|\tilde{f})$	PP	$p(e\|\tilde{f})$	Dist	Orient
0.55	0.45	–	–	–	–	–	–
0.46	0.33	-0.21	–	–	–	–	–
0.30	0.16	-0.28	0.25	–	–	–	–
0.18	0.08	-0.18	0.19	-0.38	–	–	–
0.22	0.13	-0.43	0.07	-0.07	0.08	–	–
0.21	0.13	-0.41	0.06	-0.06	0.08	0.05	–
0.13	0.08	-0.26	0.04	-0.04	0.04	0.02	0.38

Table 7.13: Translation examples showing the effect of the monotonic vs. non-monotonic search on the Chinese-English NIST task.

Reference	The Israelis prime minister’s office comdemned the shooting incident.
monotonic	Israeli prime minister’s office on the shooting incident condemned.
non-monotonic	The Israeli prime minister’s office condemned the shooting incident.

Reference	When he attends a meeting in Madrid on the 10th of this month, he said, he will use the opportunity to meet with senior officials from the United States, the EU and Russia to discuss the grave situation in the Middle East.
monotonic	He said that on 10th, he will use in Madrid in the presence of the US, EU and Russian leaders meet to discuss the Middle East is grim situation.
non-monotonic	He said that at the meeting in Madrid on 10th, he will use the opportunity to meet with senior officials of the United States, the European Union and Russia to discuss the grave situation in the Middle East region.

Reference	The explosion took place near the ”green line” that separates Israel and the West Bank.
monotonic	The explosion occurred near the separation of Israel and the West Bank’s ”green line” areas.
non-monotonic	The explosion occurred near the ”green line” separating Israel and the West Bank areas.

Reference	He said the refugees’ re-integration into society is one of the top priorities on the interim government’s agenda.
monotonic	He said that the reintegration of refugees in the interim government’s priority task.
non-monotonic	He said that the reintegration of refugees is one of the major tasks of the interim government.

Table 7.14: Translation examples showing the effect of the LM order on the Chinese-English NIST task (2002 test set).

Reference	Most U.S. allies have openly opposed any attack on Iraq.
3-gram	The United States' European allies are mostly open opposition to attack Iraq.
6-gram	Most European allies of the United States openly opposed to attack Iraq.

Reference	Annan will discuss middle east situation with U.S., Russia and European Union
3-gram	And the United States, Russia and the EU held talks with Annan will discuss the Mideast situation
6-gram	Annan will hold talks with the United States, Russia and the European Union to discuss Mideast situation

Reference	This talk may be the first step Israel has made in withdrawing its troops from the Palestinian controlled territories.
3-gram	The talks could be an Israeli withdrawal from Palestinian-controlled areas in the first step.
6-gram	The talks could be the first step towards the Israeli troop withdrawal from Palestinian-controlled areas.

Table 7.15: Translation examples showing the effect of the n-gram and sentence length posterior probabilities on the Chinese-English NIST task.

Reference	China expressed its strong displeasure with the act of the Japanese leader.
w/o posteriors	China strongly dissatisfied with the Japanese leader of this action.
w/ posteriors	The Chinese side has expressed strong dissatisfaction with this action of the Japanese leader

Reference	They are the largest refugee group since the end of the Korea war.
w/o posteriors	Since the end of Korean war fled their largest group.
w/ posteriors	They are the largest number of refugees since the end of the Korean war.

Reference	US embassy in Italy received a postal parcel with white powder in it.
w/o posteriors	The United States embassy in Italy before bags containing white powder
w/ posteriors	The US embassy in Italy received mail bags containing white powder

Table 7.16: Effect of the language model order on the translation performance for the Chinese-English NIST task ('all' test set, 3789 sentences).

LM Order	BLEU[%]	TER[%]	NIST	WER[%]	PER[%]
3	34.1	57.2	9.52	61.0	40.9
4	37.6	56.0	10.12	60.5	39.8
5	38.0	55.9	10.24	60.4	39.7
6	38.1	55.8	10.26	60.3	39.7

Table 7.17: Statistics of the training and test word alignment links for the BTEC task.

	Arabic-English	Chinese-English	Japanese-English
Training	144 K	140 K	119 K
Test	16.2 K	15.7 K	13.2 K

7.4.1 Analysis of the reordering models

7.4.1.1 Language model

In Table 7.16, we show the effect of the language model order on the translation performance. Increasing the language model order improves translation quality. There is a large improvement of 3.5 Bleu points when going from a trigram to a fourgram language model. Further improvements are obtained by increasing the language model order up to a sixgram, though the improvement is only 0.5 Bleu points. Taking into account that the memory consumption of the fourgram is only about 2 GB compared to about 8 GB for the sixgram (cf. Table 7.4), the fourgram language model provides a good tradeoff between performance and memory consumption. Therefore, we performed most of our experiments using the fourgram language model.

7.4.1.2 Phrase orientation model

Training. To train and evaluate the phrase orientation model, we use the word aligned bilingual training corpus. For evaluating the classification power of the orientation model, we partition the corpus into a training part and a test part. In our experiments on the BTEC task, we use about 10% of the corpus for testing and the remaining part for training the feature weights of the orientation model with the GIS algorithm using `YASMET` [Och 01]. The statistics of the training and test alignment links is shown in Table 7.17. The number of training events ranges from 119 K for Japanese-English to 144 K for Arabic-English. As the orientation model is independent of the phrase boundaries, there is no need to assume a phrase segmentation. The classification error rate is defined as the number of correct predictions divided by the total number of predictions.

The word classes for the class-based features are trained using the `mkcls` tool [Och 99]. In the experiments, we use 50 word classes. Alternatively, we could use part-of-speech

Table 7.18: Classification error rates [%] using two orientation classes for the BTEC task (W: words, C: classes).

		Arabic-English			Chinese-English			Japanese-English		
Baseline		6.3			12.7			26.2		
Lang.	Window	W	C	W+C	W	C	W+C	W	C	W+C
Tgt	$d = 0$	4.7	5.3	4.4	9.3	10.4	8.9	13.6	15.1	13.4
	$d \in \{0, 1\}$	4.5	5.0	4.3	8.9	9.9	8.6	13.7	14.9	13.4
	$d \in \{-1, 0, 1\}$	4.5	4.9	4.3	8.6	9.5	8.3	13.5	14.6	13.3
Src	$d = 0$	5.6	5.0	3.9	7.9	8.3	7.2	12.2	11.8	11.0
	$d \in \{0, 1\}$	3.2	3.0	2.6	4.7	4.7	4.2	10.1	9.7	9.4
	$d \in \{-1, 0, 1\}$	2.9	2.5	2.3	3.9	3.5	3.3	9.0	8.0	7.8
Src	$d = 0$	4.3	3.9	3.7	7.1	7.8	6.5	10.8	10.9	9.8
+	$d \in \{0, 1\}$	2.9	2.6	2.5	4.6	4.5	4.1	9.3	9.1	8.6
Tgt	$d \in \{-1, 0, 1\}$	2.8	2.1	2.1	3.9	3.4	3.3	8.7	7.7	7.7

information for this purpose.

Classification results. Next, we present the classification results for the orientation model described in Section 4.4.3. In Table 7.18, we present the classification results for three language pairs on the BTEC task. In these experiments, we used two orientation classes, which means that the model has to predict if the next position should be to the left or to the right.

As baseline we always choose the most frequent orientation class; this is done independent of the context. For Arabic-English, the baseline is with 6.3% already very low. For Chinese-English, the baseline is with 12.7% about twice as large. The most differences in word order occur for Japanese-English with a baseline classification error rate of 26.2%. This seems to be reasonable as Japanese has usually a different sentence structure, subject-object-verb compared to subject-verb-object in English.

For each language pair, we present results for several combination of features. The three columns per language pair indicate if the features are based on the words (column label 'W'), on the word classes (column label 'C') or on both (column label 'W+C'). We also distinguish if the features depend on the target sentence ('Tgt'), on the source sentence ('Src') or on both ('Src+Tgt').

For Arabic-English, using features based only on words of the target sentence the classification error rate can be reduced to 4.5%. If the features are based only on the source sentence words, a classification error rate of 2.9% is reached. Combining the features based on source and target sentence words, a classification error rate of 2.8% can be achieved. Adding the features based on word classes, the classification error rate can be further improved to 2.1%. For the other language pairs, the results are similar except that the absolute values of the classification error rates are higher.

We observe the following:

Table 7.19: Effect of the orientation model on the translation performance for the BTEC task.

Language Pair	Reordering	WER[%]	PER[%]	NIST	BLEU[%]
Arabic-English	Distortion	24.1	20.9	10.0	63.8
	+ Phrase orientation	23.6	20.7	10.1	64.8
Chinese-English	Distortion	50.4	43.0	7.67	44.4
	+ Phrase orientation	49.3	42.4	7.36	45.8
Japanese-English	Distortion	32.1	25.2	8.96	56.2
	+ Phrase orientation	31.2	25.2	9.00	56.8

- The features based on the source sentence perform better than features based on the target sentence.

- Combining source and target sentence features performs best.

- Increasing the window always helps, i. e. additional context information is useful.

- Often the word-class based features outperform the word-based features.

- Combining word-based and word-class based features performs best.

- In general, adding features does not hurt the performance.

These are desirable properties of an suitable reordering model. The main point is that these are fulfilled not only on the training data, but on unseen test data. There seems to be no overfitting problem.

For four orientation classes, the classification error rates are a factor 2-4 larger than for two orientation classes. Despite that, we observe the same tendencies as for two orientation classes. The detailed results are reported in Appendix B, Table B.1. Again, using more features does *not* hurt the performance.

Translation results. In Table 7.19, we show the translation results for the BTEC task. In these experiments, the reordering model uses two orientation classes, i. e. it predicts either a left or a right orientation. The features for the phrase orientation model are based on the source and target language words within a window of one. The word-class based features are not used for the translation experiments. The phrase orientation model achieves small but consistent improvement for all the evaluation criteria. The baseline system, i. e. using the distortion-based reordering, was among the best systems in the IWSLT 2005 evaluation campaign [Eck & Hori 05]. Note that only the model scaling factors of Equation 1.8 are optimized using the Downhill Simplex algorithm. The feature weights of the reordering model are trained using the GIS algorithm as described in Section 4.4.3.4.

Some translation examples are presented in Table 7.20. We observe that the system using the phrase orientation model produces more fluent translations.

Table 7.20: Translation examples for the Chinese-English BTEC task.

System	Translation
Distortion	I would like to check out time one day before.
Phrase orientation	I would like to check out one day before the time.
Reference	I would like to check out one day earlier.
Distortion	I hate pepper green.
Phrase orientation	I hate the green pepper.
Reference	I hate green peppers.
Distortion	Is there a subway map where?
Phrase orientation	Where is the subway route map?
Reference	Where do they have a subway map?

Additional translation experiments were carried out on the large data track of the Chinese-English NIST task. The results were already reported in Table 7.9 and in Table 7.10. We observed a significant improvement over the distortion-based reordering model on all test sets. On the 'all' test set, for instance, the Bleu score improved by 1.3 Bleu points. For the translation experiments on the Chinese-English NIST task, we used two orientation classes and the following features: source word, target word, source-target word pair at the current position, i. e. $d = 0$. We also tried additional features, e. g. using a larger window or word classes, but these did not lead to further improvements.

Scaling factor. In Figure 7.3, we show the effect of the model scaling factor for the phrase orientation model on the Chinese-English NIST task.

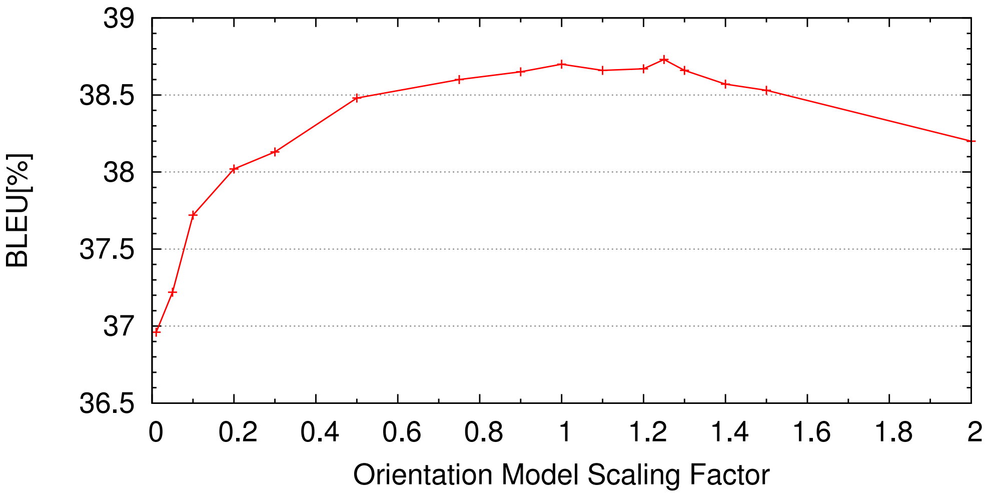


Figure 7.3: Effect of the phrase orientation model scaling factor for the Chinese-English NIST task (NIST 2002 test set, 878 sentences).

7.4.1.3 Distortion model

In Figure 7.4, we show the effect of the distortion model scaling factor on the Chinese-English NIST task. We show two curves, one without the phrase orientation model and one with the phrase orientation model. In Figure 7.5, we show the effect of the distortion limit on the Chinese-English NIST task. We observe a large improvement in translation quality as the distortion limit is increased up to a length of 10. Increasing the distortion limit beyond 10 does not lead to further improvements.

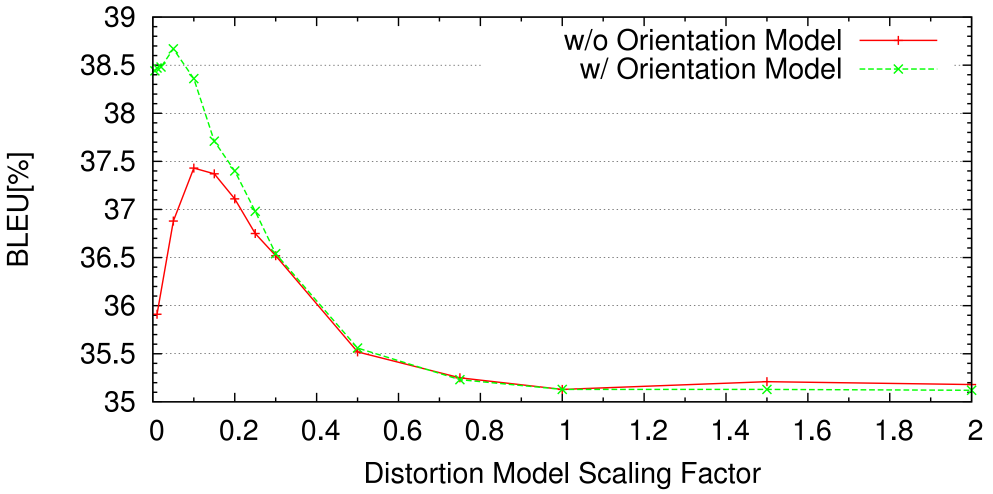


Figure 7.4: Effect of the distortion model scaling factor for the Chinese-English NIST task (NIST 2002 test set, 878 sentences).

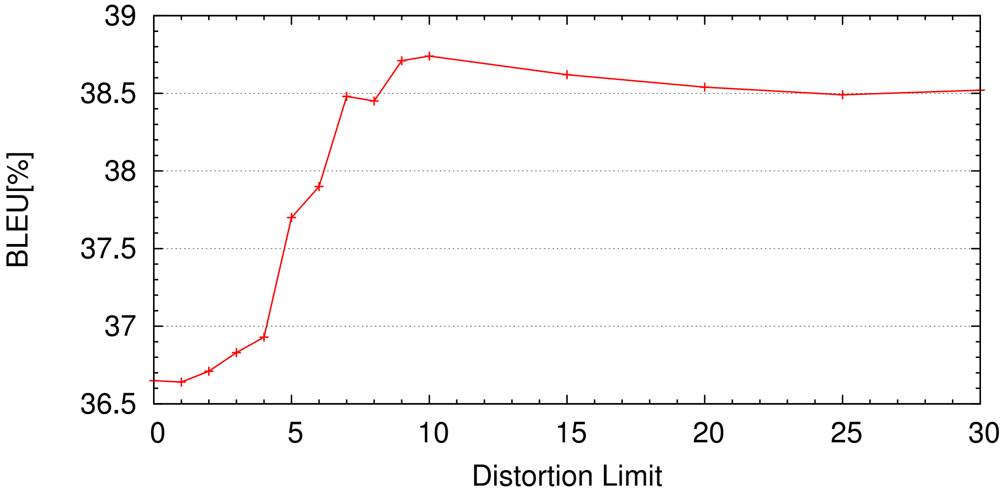


Figure 7.5: Effect of the distortion limit for the Chinese-English NIST task (NIST 2002 test set, 878 sentences).

Table 7.21: Comparison with other groups on the Chinese-English NIST task. Translation performance measured with BLEUr4n4[%].

		NIST test set		
System		2003	2004	2005
[Chiang 07]	ATS	30.8	31.7	30.5
	Hiero	33.7	34.6	31.8
[DeNeefe & Knight$^+$ 07]	ATS	32.8	–	–
	Syntax baseline	37.7	–	–
	improved	41.3	–	–
This work		39.8	39.9	37.8

7.4.2 Comparison with other groups

In Table 7.21, we compare the results obtain in this work with the results of other groups. The first comparison is with the hierarchical approach of [Chiang 07]. There, the alignment template approach of [Och & Ney 04] is used as baseline ('ATS'); the results obtained using hierarchical approach are denoted with 'Hiero'. The second comparison is done with the syntax-based system of [DeNeefe & Knight$^+$ 07], which is a state-of-the-art syntax-based system. Again, 'ATS' denotes the baseline obtained with the alignment template approach of [Och & Ney 04]. To the best of our knowledge, these are currently the best results published for this task.

Note that these results are rather hard to compare and interpret as they measure only the end-to-end performance of the whole MT system, including the effects of preprocessing, word alignment, translation and language models, decoding, tuning etc.. It is difficult to attribute the differences in translation performance to individual system components.

We observe that the phrase-based system described in this work is competitive with the best systems on this task.

7.5 Analysis of the Search

In Figure 7.6, we show the effect of the search errors on the Bleu score for the Chinese-English NIST task. We fixed the model scaling factors and varied the pruning parameters to generate translations of the NIST 2002 test set. We show the Bleu score as a function of the average value of the score function $Q(\cdot)$ in Equation 5.11 per sentence. We observe a strong correlation between the model score and the Bleu score. Thus, it is important to find a translation hypotheses with a good model score, i. e. a good search algorithm is important to achieve good translation quality.

In Figure 7.7, we show the effect of the rest score estimation described in Section 5.2.4 on the translation performance. We compare the following variants of the rest score estimation. We varied the included models: translation model (TM), language model

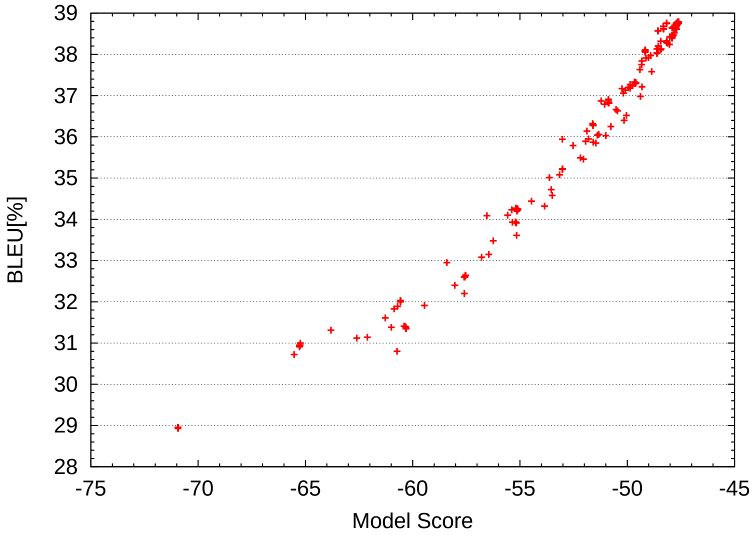


Figure 7.6: Effect of search errors. Bleu score as a function of the average model score per sentence for varied pruning parameters on the Chinese-English NIST task (NIST 2002 test set, 878 sentences). The larger the model score, the better is the solution found by the search algorithm, i. e. there are fewer search errors.

(LM), distortion model (Dist) and we compare if the rest scores are computed per position (Equation 5.24) or per sequence (Equation 5.21). For each of the rest score estimates, we varied the beam size, i. e. the histogram size N_c for the lexical pruning per cardinality. For (very) large beam sizes, the rest score estimation is of course not important. For small and medium beam sizes however, we observe that a good rest score estimate is important to achieve high Bleu scores. Using only the translation model for the rest score estimation does not help much. The curves are very similar to the one without rest score estimation. Both, the language model and the distortion model, help to improve the search results. The rest score estimates based on sequences of source position typically outperform the estimate based on positions, i. e. with the same beam size they achieve a higher Bleu score. The rest score estimate based on translation and language model scores for sequences of source positions in combination with a rest score estimate for the distortion penalty model works best. Using this rest score estimation, we already achieve a very good Bleu score with a beam size of only 64 hypotheses. Without rest score estimation, we would need about 16 K hypotheses to achieve the same Bleu score. In Figure 7.8, we show the effect of different rest score estimates on the model score.

As defined in Section 5.1, the number of lexical hypotheses is the overall number of hypotheses for a given cardinality. In contrast, for the number of coverage hypotheses we count the distinct coverages for a given cardinality. Thus, the coverage hypotheses are independent of the lexical choice. The more coverage hypotheses are retained during the search, the more reorderings are considered. In Figure 7.9, we separated the effect of lexical choice and reordering. For each of the curves, we limited the number of coverage hypotheses and varied the maximum number of lexical hypotheses per coverage hypothesis. Thus, along the x-axis we increase the search space by allowing for more lexical

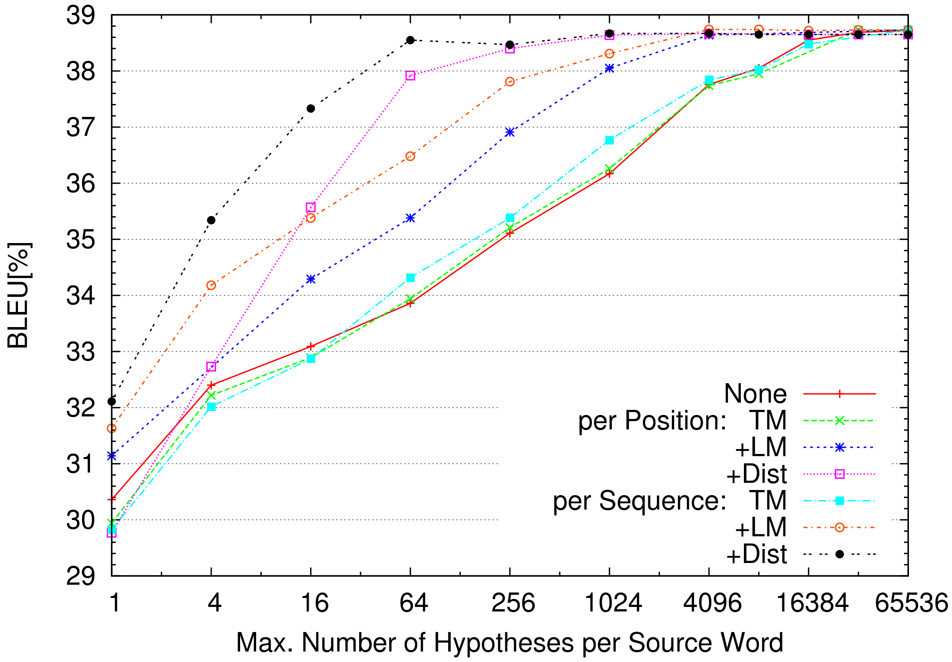


Figure 7.7: Effect of the rest score estimation on the translation performance (BLEUr4n4[%]) for different beam sizes on the Chinese-English NIST task (NIST 2002 test set, 878 sentences).

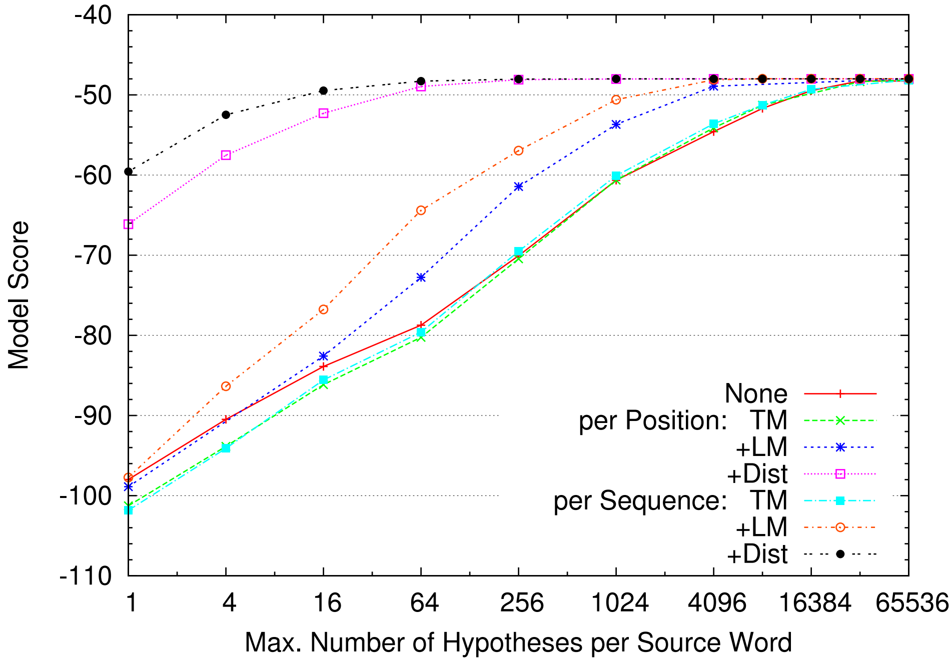


Figure 7.8: Effect of the rest score estimation on the model score for different beam sizes on the Chinese-English NIST task (NIST 2002 test set, 878 sentences).

choice, whereas from curve to curve we allow for more reordering. To be precise: for each curve we fixed the histogram size N_C for the coverage pruning per cardinality and we varied the histogram size N_L for the lexical pruning per coverage. The overall search space is limited by the product of the two numbers; we vary the overall search space between 1 hypothesis and 64 K hypotheses. We observe that increasing the lexical choice beyond 16 hypotheses per coverage does not lead to further improvements; often only 4 lexical hypotheses are sufficient. Furthermore, the improvement that we can achieve by taking lexical alternatives into account is between 1 and 2 Bleu points. If we look at the maximum number of coverage hypotheses, we see a much bigger effect on the Bleu score. There is a considerable improvement by increasing the number of coverage hypotheses up to 64. The curve for a maximum of 256 hypotheses is virtually identical; thus there is no further improvement.

In Figure 7.10, we show the effect of lexical pruning per coverage on the Bleu score. We fixed the histogram size N_L for the lexical pruning per coverage, i. e. the maximum number of lexical hypotheses per coverage and varied the total number of hypotheses, i. e. the histogram size N_c for the lexical pruning per cardinality. We then plotted the average number of hypotheses per source word against the Bleu score. The 'unlimited' means that there is no lexical pruning per coverage. This is equivalent to the pruning in Pharaoh and Moses. Using lexical pruning per coverage, we achieve the best Bleu score already with a small number of hypotheses; in this example about 256 hypotheses are required. Without lexical pruning per coverage, a significantly larger beam is required to achieve this Bleu score.

In Table 7.22, we show the memory usage of different system components. We distinguished the language model, the phrase table, the search, i. e. the hypotheses and trace back information. All other components and data structures are put in the 'Other' category. This includes the word lexica and the phrase orientation model. For these experiments, we used a beam size of 16 K hypotheses per source word. For the LM and the phrase table, we also compared filtering on the test set level and on the sentence level. Using the efficient phrase-table representation described in Section 5.6, there is no time overhead for filtering the phrase table on the sentence level. For filtering the LM per sentence requires about 20-30 seconds, whereas loading the whole LM into memory requires about 15 minutes for the sixgram LM. We measured the memory usage per sentence and report the maximum. If we filter the LM and phrase-table per sentence, the total memory usage is at most 1.6 GB for the fourgram LM and 2.2 GB for the sixgram LM. If the LM and phrase table are filtered for the test set, they clearly dominate the memory usage resulting in a total memory usage of about 5.2 GB for the fourgram LM and about 11.2 GB for the sixgram LM.

In Figure 7.11, we show the effect of the window size for the word-level and block-level skip reordering constraints, i. e. the IBM constraints described in Section 5.5.3. We plotted the curves using a distortion limit of 10 and without distortion limit. The results with and without distortion limit are rather similar for small window size. For larger window sizes, the distortion limit seems to help somewhat.

Table 7.22: Maximum memory usage for different system components. The beam size was limited to 16 K hypotheses per source word.

	LM order	LM	Phrase-table	Search	Other	Total
filtered per test set	4-gram	2.1 GB	1.7 GB	1.0 GB	400 MB	5.2 GB
	6-gram	7.9 GB	1.7 GB	1.2 GB	400 MB	11.2 GB
filtered per sentence	4-gram	205 MB	25 MB	1.0 GB	400 MB	1.6 GB
	6-gram	575 MB	25 MB	1.2 GB	400 MB	2.2 GB

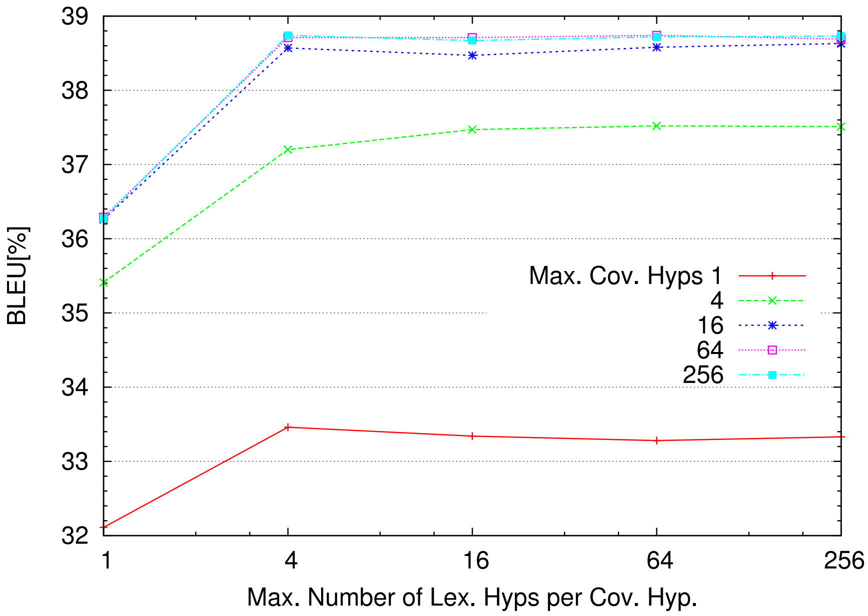


Figure 7.9: Effect of the number of lexical and coverage hypotheses for the Chinese-English NIST task (NIST 2002 test set, 878 sentences).

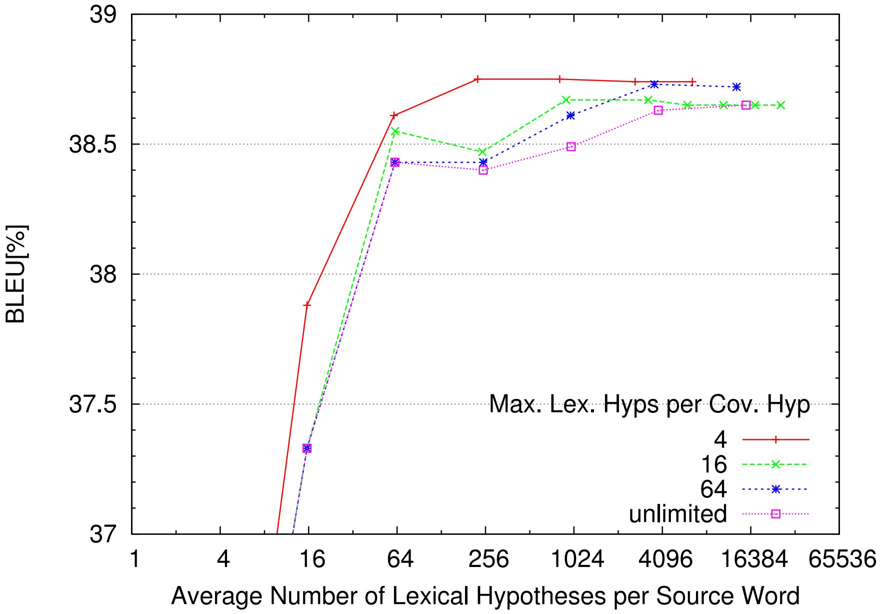


Figure 7.10: Effect of lexical pruning per coverage on the Bleu score for the Chinese-English NIST task (NIST 2002 test set, 878 sentences).

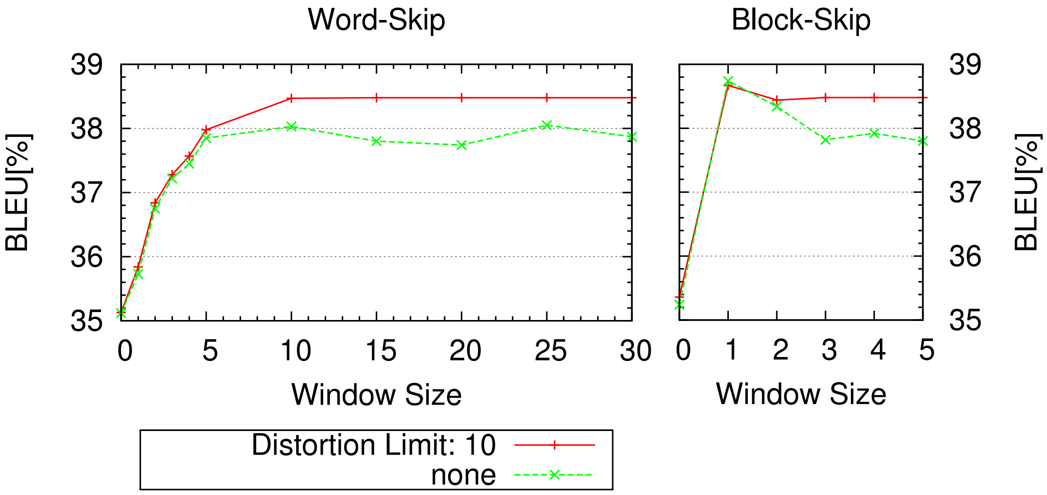


Figure 7.11: Effect of the window size for the word and block-level skip reordering constraints (IBM constraints) for the Chinese-English NIST task (NIST 2002 test set, 878 sentences).

Table 7.23: Comparison with Moses on the Chinese-English NIST task (NIST 2002 test set, 878 sentences). Bleu in percent; speed in words per second.

	Moses		RWTH		RWTH (skip)	
Beam Size	Bleu	Speed	Bleu	Speed	Bleu	Speed
1	31.7	415.7	32.0	886.5	32.4	790.7
4	34.6	128.3	35.3	544.6	35.7	620.0
16	35.9	34.9	36.7	221.8	36.9	225.3
64	36.9	9.8	37.0	58.9	37.5	89.2
256	36.9	2.0	37.2	14.4	37.4	29.5
1 024	37.2	0.5	37.2	3.4	37.5	7.9
4 096	37.2	0.1	37.2	0.8	37.5	3.3

7.5.1 Comparison with Moses

In this section, we will compare the decoder described in this work with Moses [Koehn & Hoang$^+$ 07], a publicly available decoder for phrase-based SMT. The experiments were carried out using the Moses version from May 25, 2007 and compiled with the optimization flags enabled. We use the SRI library for the language model (this is identical in both decoders). More details on Moses can be found at `http://www.statmt.org/moses`.

We use the same phrase table, i. e. the same models, for the two decoders: a fourgram LM, the phrase models $p(\tilde{f}|\tilde{e})$ and $p(\tilde{e}|\tilde{f})$, the noisy-or lexicon model $p(e|\tilde{f})$, word and phrase penalty and the distortion penalty model. The phrase orientation model is not used. To compare the performance, we translated the NIST 2002 test set with varying beam sizes and measured the Bleu score and the translation speed (in words per second) on machines with AMD Opteron CPUs and 8 GB of memory. As Moses does not support multi-threading, we run both decoders single-threaded. The results are presented in Table 7.23. We used two variant of our decoder: the first version uses the same reordering constraints as Moses, the second version uses the IBM-style phrase-level skip reordering with window 1.

Comparing Moses and the first version of the RWTH decoder, we observe that they achieve very similar Bleu scores with a small advantage for the RWTH decoder in case of a small beam size. Both decoders reach a Bleu score of about 37.2%. In terms of translation speed, the RWTH decoder clearly outperforms Moses. It is faster by a factor of 7-8 for beam sizes ≥ 64. The second variant of the RWTH decoder, using the phrase-level skip reordering constraints with window 1, we achieve a higher Bleu score of about 37.5%.

In Figure 7.12, we plotted the Bleu score versus the translation speed. We observe that we can achieve the same Bleu score as Moses with much less computation time. If we look, for instance, at the Bleu score level of about 36.9%, the RWTH decoder is about 6 times faster than Moses (58.9 words per second for the RWTH decoder versus 9.8 words per second for Moses), using the phrase-level skip constraints, we can achieve the same Bleu score even 23 times faster than Moses (with a speed of 225.3 words per second). If

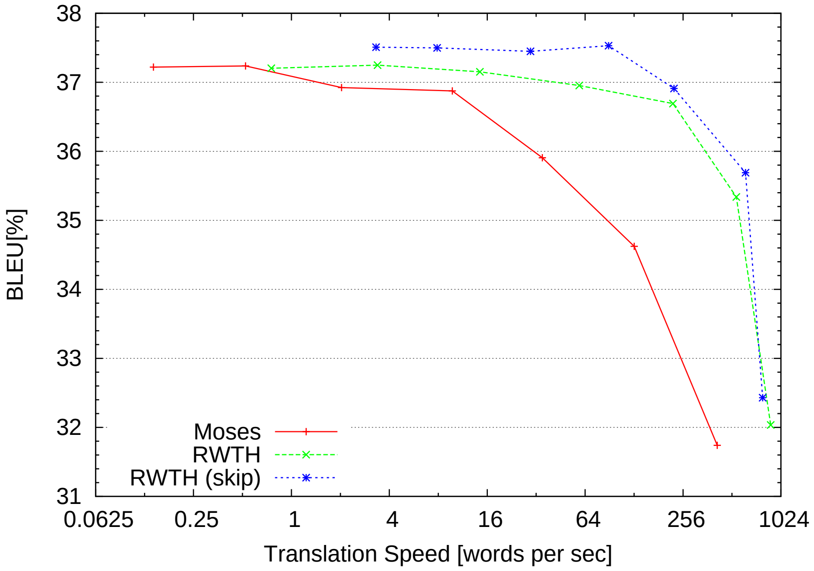


Figure 7.12: Comparison with Moses on the Chinese-English NIST task (NIST 2002 test set, 878 sentences).

Table 7.24: Translation results (BLEUr2n4c [%]) for various training criteria (MERT: minimum error rate training; ML: maximum likelihood estimation; $\mathbb{E}$[Bleu]: expected Bleu score) and the maximum a-posteriori (MAP) and the minimum Bleu risk (MBR) decision rule for the Chinese-English TC-Star task.

			Development		Eval'06		Eval'07	
		Decision Rule	MAP	MBR	MAP	MBR	MAP	MBR
Training Criterion	MERT-Bleu (baseline)		**19.5**	**19.4**	16.7	17.2	22.2	23.0
	ML	sentence-level	17.8	18.9	14.8	17.1	18.9	22.7
		n-gram level	18.6	18.8	17.0	17.8	22.8	23.5
	$\mathbb{E}$[Bleu]	sentence-level	19.1	18.9	17.5	**18.1**	23.5	**24.1**
		n-gram level	18.6	18.8	**17.7**	17.6	**24.0**	24.0

we look at a Bleu score of 37.2%, which is the best we achieved using Moses, the difference is even larger. Moses processes about 0.5 words per second; using the RWTH decoder (skip), we achieve an even better Bleu score with a speed of about 89.2 words per second, i. e. about 178 times faster than Moses.

7.6 Effect of Different Training Criteria

In Table 7.24, we present the translation results for different training criteria for the development set and the two blind test sets on the Chinese-English TC-Star task. The reported case-sensitive Bleu scores are computed using two reference translations, i. e. BLEUr2n4c. Note that already the baseline system (MERT-Bleu) would have achieved the first rank in the official TC-Star evaluation 2006; the best Bleu score in that evaluation was 16.1% (cf. Table 7.25).

We also compare the effect of two loss function in the Bayes decision rule:

- **MAP**: the maximum a-posteriori decision rule (cf. Equation 1.4), i. e. the Bayes decision rule for the 0-1 loss function.
- **MBR**: the minimum Bleu risk decision rule. Here, we use a Bleu induced loss function as described in [Kumar & Byrne 04]. We use the algorithm described in [Ehling & Zens$^+$ 07] on a 10 000-best list.

On the development data, the MERT-Bleu achieves the highest Bleu score. This seems reasonable as it is the objective of this training criterion.

The maximum likelihood (ML) criteria perform somewhat worse under MAP decoding. Interestingly, the MBR decoding can compensate this to a large extent: all criteria achieve a Bleu score of about 18.9% on the development set. The benefits of MBR decoding become even more evident on the two test sets. Here, the MAP results for the sentence-level ML criterion are rather poor compared to the MERT-Bleu. Nevertheless, using MBR decoding results in very similar Bleu scores for most of the criteria on these two test sets. We can therefore support the claim of [Smith & Eisner 06] that MBR tends to have better generalization capabilities.

The n-gram level ML criterion seems to perform better than the sentence-level ML criterion, especially on the test sets. The reasons might be that there is no need for the use of pseudo references as described in Section 6.4 and that partial correctness is taken into account.

The best results are achieved using the expected Bleu score criteria described in Section 6.5. Here, the sentence level and n-gram level variants achieve more or less the same results. The overall improvement on the Eval'06 set is about one Bleu point for MAP decoding and 0.9 Bleu points for MBR decoding. On the Eval'07 set, the improvements are even larger, about 1.8 Bleu points for MAP and 1.1 Bleu points for MBR. All these improvements are statistically significant at the 99% level using a pairwise significance test.

Given that currently the most popular approach is to use MERT-Bleu MAP decoding, the overall improvement is 1.4 Bleu points for the Eval'06 set and 1.9 Bleu points on the Eval'07 set. Note that the MBR decision rule almost always outperforms the MAP decision rule. In the rare cases where the MAP decision rule yields better results, the difference in terms of Bleu score are small and *not* statistically significant.

We also investigated the effect of the maximum n-gram order for the n-gram level maximum likelihood estimation (ML). The results are shown in Figure 7.13. We observe an

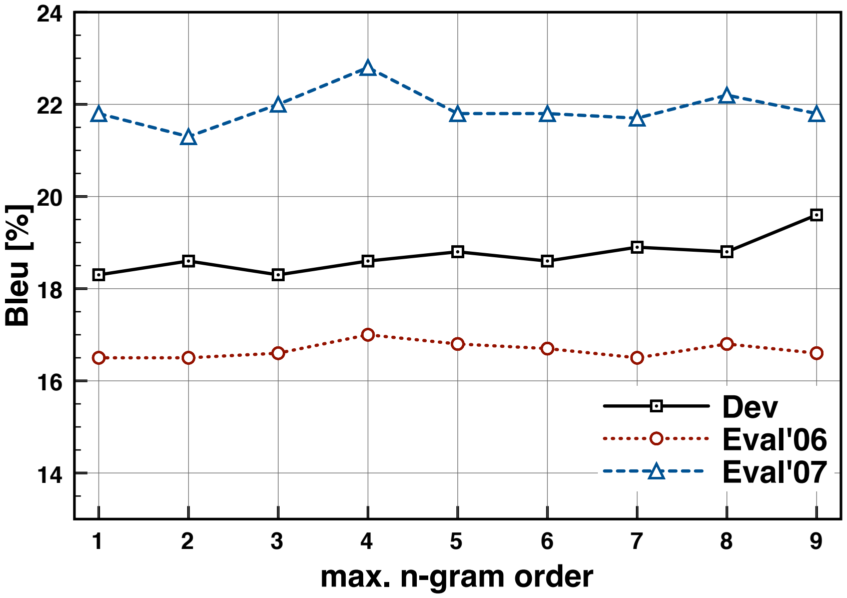


Figure 7.13: Effect of the maximum n-gram order on the Bleu score for the n-gram level maximum likelihood estimation under the maximum a-posteriori decision rule for the Chinese-English TC-Star task.

increase of the Bleu score with increasing maximum n-gram order for the development corpus. On the evaluation sets, however, the maximum is achieved if the maximum n-gram order is limited to four. This seems intuitive as the Bleu score uses n-grams up to length four. However, one should be careful here: the differences are rather small, so it might be just statistical noise.

We can conclude that the expected Bleu score is not only a theoretically sound training criterion, but also achieves the best results in terms of Bleu score on this task. The improvement over a state-of-the-art MERT baseline is 1.3 Bleu points for the MAP decision rule and 1.1 Bleu points for the MBR decision rule for the large Chinese-English TC-Star speech translation task.

We presented two methods for computing the expected Bleu score: a sentence-level and an n-gram level approach. Both yield similar results. We think that the n-gram level computation has certain advantages: The n-gram posterior probabilities could be computed from a word graph which would result in more reliable estimates. Whether this pays off in terms of translation quality is left open for future work.

Another interesting result of our experiments is that the MBR decision rule seems to be less affected by sub-optimal parameter settings. Although it is well-known that the MBR decision rule is more appropriate than the MAP decision rule, the latter is more popular in the SMT community (and many other areas of natural language processing). The presented results show that it can be beneficial to use the MBR decision rule. On the other hand, the computation of the MBR hypotheses is more time consuming.

Table 7.25: TC-Star: Official results of the public evaluations in 2005, 2006 and 2007 for the Chinese-English task (top five systems). We report case-sensitive BLEU and NIST scores.

		Verbatim			ASR		
Year	Rank	Group	BLEU[%]	NIST	Group	BLEU[%]	NIST
2005	1	**RWTH**	**15.7**	**5.80**	**RWTH**	**15.0**	**5.61**
	2	IBM	13.8	5.78	UKA	13.3	5.39
	3	UKA	13.7	5.64	JHU	13.2	5.48
	4	JHU	13.5	5.64	IRST	11.6	5.23
	5	IRST	12.2	5.42	IBM	5.3[c]	2.68
2006	1	**RWTH**	**16.1**	**6.45**	**RWTH**	**12.4**	**5.17**
	2	IRST	14.0	6.01	IRST	11.1	4.95
	3	ICT	13.7	6.03	ICT	10.9	4.90
	4	NRC	12.8	5.80	Systran	8.6	4.38
	5	UKA	10.8	5.51	UKA	8.5	4.59
2007	1	**RWTH**	**24.5**	**7.35**	**RWTH**	**22.5**	**6.80**
	2	IRST	21.8	7.09	IRST	19.7	6.45
	3	ICT	20.1	6.67	ICT	18.3	6.01
	4	UKA	18.6	6.47	UKA	16.5	5.82
	5	NICT-ATR	18.4	6.51	XMU	11.6	5.27

7.7 Official TC-Star Evaluations

In the European Union project TC-Star, annual public evaluation were carried out. The official results of the evaluation in 2005, 2006 and 2007 for the Chinese-English task are summarized in Table 7.25. We show only the top five systems for the verbatim and ASR condition. There were two data conditions: in the primary condition, the training data was limited (very similar to the NIST large data track), in the secondary condition, the training was unlimited. If a group participated in both tasks, we have chosen the better of the two results. The effect on the Bleu and NIST score is small and does not affect the ranking of the systems. RWTH participated in the primary, i. e. limited, data condition. More details can be found in the public evaluation reports [Mostefa & Arranz$^+$ 05, Mostefa & Garcia$^+$ 06, Mostefa & Hamon$^+$ 07]. In all three years and in both conditions, verbatim and ASR, the RWTH system achieved the first rank.

[c] The official IBM submission for the ASR condition was erroneous. The corrected version, which was submitted after the evaluation, obtained a Bleu score of 12.7%.

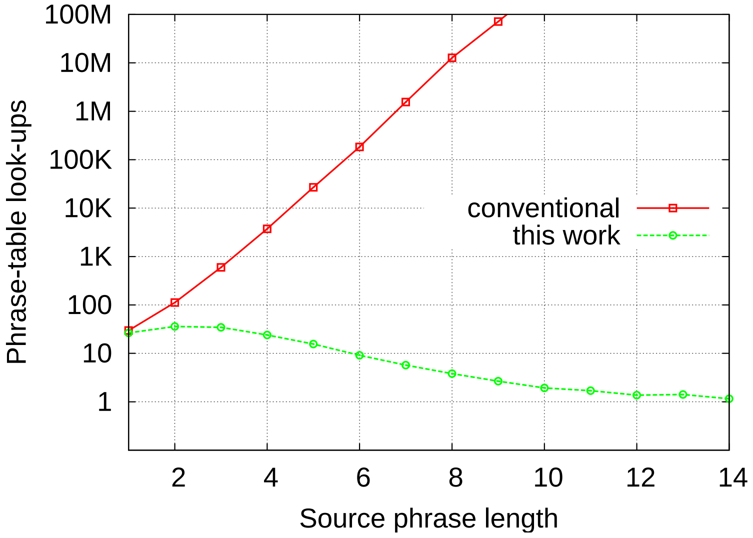


Figure 7.14: Average number of phrase-table look-ups per sentence as a function of the source phrase length for the Spanish-English EPPS task.

7.8 Phrase Matching

In this section, we will present an empirical complexity analysis of the phrase matching algorithm described in Section 5.7.2. In Figure 7.14, we present the average number of phrase-table look-ups for the full EPPS confusion networks (CNs) as a function of the source phrase length. The maximum phrase length for these experiments is seven. The curve 'conventional' represents the total number of source phrases in the CNs for a given length. This is the number of phrase-table look-ups using the naive algorithm. Note the exponential growth with increasing phrase length. Therefore, the naive algorithm is only applicable for very short phrases and heavily pruned CNs, as e. g. in [Bertoldi & Federico 05].

The curve 'this work' is the number of phrase-table look-ups using the `phrase-match` algorithm described in Figure 5.16. We do *not* observe the exponential explosion as for the naive algorithm. Thus, the presented algorithm effectively solves the combinatorial problem of matching phrases of the input CNs and the phrase-table.

In Table 7.26, we present the translation results and the translation times for different input conditions. We observe a significant improvement in translation quality as more ASR alternatives are taken into account. The best results are achieved for the full CNs. On the other hand, the decoding time increases only moderately. Using the new algorithm, the ratio of the time for decoding the CNs and the time for decoding the single-best input is 3.4 for the full CNs and 1.8 for the pruned CNs. In previous work [Bertoldi & Federico 05], the ratio for the pruned CNs was about 25 and the full CNs could not be handled.

To summarize, the presented algorithm has two main advantages for speech translation: first, it enables us to utilize large CNs, which was prohibitively expensive beforehand and second, the efficiency is improved significantly.

Table 7.26: Translation quality and time for different input conditions for the Spanish-English EPPS task (time in seconds per sentence [sec] and as factor of the single-best decoding time [× SB]).

			Time	
Input type	oracle ASR-WER[%]	BLEU[%]	[sec]	[× SB]
Single-best	21.4	37.6	2.7	1.0
Confusion Network pruned	16.7	38.5	4.8	1.8
full	8.4	38.9	9.2	3.4

Whereas the previous approaches required careful pruning of the CNs, we are able to utilize the unpruned CNs. Experiments on other tasks have shown that even larger CNs are unproblematic.

The confusion network decoding experiments in this section were carried out using the Moses decoder [Koehn & Hoang$^+$ 07], which is a open-source decoder for phrase-based machine translation. More details on Moses can be found at `http://www.statmt.org/moses`.

8 Conclusions

In this chapter, we will summarize the achievements of this work and point out directions for future work.

8.1 Summary

- We have extended the standard IBM word alignment models with a symmetric lexicon and we have described the corresponding training procedure. Furthermore, we have reduced the word alignment problem to the minimum-weight edge cover problem and presented an efficient algorithm to solve this problem. Both approaches resulted in significantly reduced word alignment error rates on the Verbmobil and Canadian Hansards tasks.

- We investigate the contribution of several phrase-based translation models to the overall translation quality. The resulting system achieves state-of-the-art performance on the large scale Chinese-English NIST task. Furthermore, the system was used in the official TC-Star evaluations in 2005, 2006 and 2007 for the Chinese-English broadcast news speech translation task. We achieved the first rank in all three years and for all input conditions.

- We have described the search problem in detail and presented efficient algorithms for solving this problem. The analysis of the search showed that it is important to focus the search on alternative coverage hypotheses, i. e. alternative reorderings. On the other hand, already a small number of lexical hypotheses per coverage hypothesis are sufficient to achieve good translation quality. A comparison with the public tool Moses showed, that the presented decoder is more than 23 times faster at the same level of translation quality.

- We have presented an efficient phrase-table representation that is loaded on-demand from disk. This enables online phrase-based SMT, i. e. the translation of arbitrary text without the time-consuming pre-filtering of the phrase table. The memory consumption is reduced to less than 25 MB for the large Chinese-English NIST task where previous approaches required about 1.5-2 GB for test set specific phrase tables and about 15 GB for the unfiltered phrase table. The implementation was made available as part of the public open-source toolkit Moses.

- In addition to the search for text input, we also described the search for lattice input. Here, the input to the MT system is a lattice of alternative source sentences, e. g. a lattice of transcriptions from a speech recognizer. The phrase matching of input lattice and phrase table can be done efficiently by exploiting the prefix tree structure of the phrase table. Using the presented algorithm on the Spanish-English TC-Star task, we achieved a significant speed-up compared to previous work and generated translations of better quality.

- The reordering problem in MT is difficult for two reasons: first, it is computational expensive to explore all possible permutations; second, it is hard to select a good permutation. Thus, reordering is a difficult search problem and a difficult modeling problem. We have described and compared different reordering constraints to address the search problem. We showed that the phrase orientation model can help to address the modeling problem. Both approaches resulted in improved translation quality.

- We presented different training criteria for optimizing the free parameters of a phrase-based SMT system. The experimental results on the large-scale Chinese-English TC-Star task have shown that significant improvements over the standard minimum error rate training can be achieved. In this context, we introduced n-gram and sentence length posterior probabilities. Using these posterior probabilities in a rescoring/reranking framework resulted in improved translation quality.

8.2 Future Directions

- **More Data.** The most obvious way to improve a data-driven approach like the one presented here is of course to utilize more data for training the system. [Pantel & Ravichandran$^+$ 04] has shown that simple (or better efficient) methods applied to larger volumes of data often outperform more sophisticated methods that do not scale to large volumes of data and therefore can utilize only a limited subset of the available data. The automatic collection of new bilingual training data would significantly increase the amount of available data for training SMT systems. The work of [Fry 05] seems to indicate that we can collect bilingual data of high quality in a rather simple (but clever) way using RSS news feeds. Furthermore, utilizing comparable corpora, as e. g. in [Fung & Cheung 04, Munteanu & Fraser$^+$ 04], can result in additional training data. But as only a fraction of the comparable corpora can be added to the bilingual training corpus, huge amounts of comparable corpora are required to achieve significant improvements.

- **Better Data.** Despite the sheer amount of data, the quality of the data is of course important. Currently, the preprocessing of the training data, e. g. cleaning, tokenization, text normalization, is done using a set of hand-written rules or regular expressions. As a proper preprocessing can affect the translation quality quite significantly, it would be desirable to automate this procedure. As the purpose is to improve translation quality, this should be done in a bilingual way, i. e. both the source and target language sentence should be taken into account.

- **Better Models.** Despite using more data, improved models can lead to better translation quality. Iterative training of the phrase translation probabilities using maximum likelihood or even discriminative training could result in better estimates.

  A weakness of the phrase-based approach is the long-range reordering. Local reordering is captured within the phrases, but long-range reordering is not. Translation quality could benefit from a stronger reordering model. Wether this is based on syntactic parse trees as e. g. in [Galley & Hopkins$^+$ 04, Galley & Graehl$^+$ 06, DeNeefe & Knight$^+$ 07] or hierarchical phrases without explicit syntax as in e. g. [Chiang 05, Chiang 07] or some other concept has still to be investigated.

  Most SMT systems look just at the surface-level of the words. Factored translation models [Koehn & Hoang$^+$ 07] try to utilize models at the stem or part-of-speech level. This seems to be promising for morphologically rich languages and/or tasks with scarce resources.

- **Public Corpora, Tools & Evaluations**. MT systems are getting more and more complicated. Thus, the entry barrier of a research group to start in the area of MT is rather high. Freely available tools such as GIZA++ [Och & Ney 03], Pharaoh [Koehn 04a] and Moses [Koehn & Hoang$^+$ 07] lower the entry barrier significantly. They foster collaboration among research groups and speed up progress in the area of MT. Public corpora permit a comparison of different approaches trained on the same data and public evaluations, such as the NIST evaluations from 2002 to 2006[a] or the shared task of the Workshop on Machine Translation[b], allow for a comparison on unseen test data.

- **Better Evaluation Metrics**. Automatic evaluation metrics are important for a rapid development cycle. During the development and tuning phase, the quality of the MT system is evaluated several (hundred) times. The parameters of the MT system are adjusted to achieve a high score of a given automatic evaluation metric. Nevertheless, the ultimate goal is to improve the translation quality using this parameter tuning. Therefore, automatic evaluation metrics should have a high correlation with human judgment of translation quality. Furthermore, it should not be possible to cheat the metric, i. e. to improve the score without improving translation quality. Current metrics have their limitations as pointed out in [Callison-Burch & Osborne$^+$ 06] for the Bleu score. As MT systems are tuned toward a specific metric, improved MT evaluation metrics will lead to better machine translation quality.

[a] http://www.nist.gov/speech/tests/mt/index.htm

[b] http://www.statmt.org

A Symbols & Acronyms

A.1 Mathematical Symbols

$\mathbf{f} = f_1^J = f_1, ..., f_j, ..., f_J$	source language sentence
$\mathbf{e} = e_1^I = e_1, ..., e_i, ..., e_I$	target language sentence
$A \subseteq J \times I$	word alignment (general)
$\mathbf{a} = a_1^J = a_1, ..., a_j, ..., a_J$	word alignment (mapping)
$Pr(\cdot)$	general probability distribution with no specific assumptions
$p(\cdot)$	model-based probability distribution
$L(\cdot)$	loss function
λ	model scaling factor
$h(\cdot)$	component of log-linear model
s_1^K	segmentation into K phrase-pairs
i_k	end position of k^{th} target phrase
j_k	end position of k^{th} source phrase
b_k	start position of k^{th} source phrase
$\tilde{e}$	target phrase
$\tilde{f}$	source phrase
$\lvert\tilde{e}\rvert$	length of the phrase $\tilde{e}$
$\tilde{e}^i$	the i^{th} word of phrase $\tilde{e}$
$q_{\text{TM}}(\cdot)$	weighted translation model score
$q_{\text{LM}}(\cdot)$	weighted language model score
$q_{\text{DM}}(\cdot)$	weighted distortion model score
N_o	histogram size for observation pruning
τ_o	threshold for observation pruning
N_c	histogram size for lexical pruning per cardinality
τ_c	threshold for lexical pruning per cardinality
N_L	histogram size for lexical pruning per coverage
τ_L	threshold for lexical pruning per coverage
N_C	histogram size for coverage pruning per cardinality
τ_C	threshold for coverage pruning per cardinality
\$	sentence start or sentence end symbol
$E(j, j')$	translation candidates for source phrase $f_j, \ldots, f_{j'}$

A.2 Acronyms

AER	alignment error rate
ASR	automatic speech recognition
BLEU	bilingual evaluation understudy
BTEC	basic travel expression corpus
BTG	bracketing transduction grammar
CN	confusion network
EM	expectation maximization
EPPS	European Parliament Plenary Speeches
FST	finite state transducer
GB	gigabyte
GIS	generalized iterative scaling
HMM	hidden Markov model
ITG	inversion transduction grammar
LDC	Linguistic Data Consortium
LM	language model
MAP	maximum a-postiori
MB	megabyte
MBR	minimum Bleu risk
MERT	minimum error rate training
ML	maximum likelihood
MT	machine translation
PER	position-independent word error rate
POS	part-of-speech
PP	phrase penalty
SLDB	spoken language database
SMT	statistical machine translation
TER	translation edit rate
TM	translation model
WER	word error rate
WP	word penalty

B Additional Results

B.1 Reordering Model

In Table B.1, we present the classification results of the lexicalized reordering model for four orientation classes. The final error rates are a factor 2-4 larger than for two orientation classes. Despite of that, we observe the same tendencies as for two orientation classes. Again, using more features does *not* hurt the classification performance.

B.2 Reordering Constraints Evaluation in Training

In this section, we will investigate for the IBM and ITG reordering constraints the coverage of the training corpus alignment. For this purpose, we compute the Viterbi alignment of IBM model 5 with GIZA++ [Och & Ney 00, Och & Ney 03]. This alignment is produced without any restrictions on word-reorderings and has good quality. Then, we check for every sentence if the alignment satisfies each of the constraints. The ratio of the number of satisfied alignments and the total number of sentences is referred to as coverage.

The first task we will present results on is the Verbmobil task [Wahlster 00]. The domain of this corpus is appointment scheduling, travel planning, and hotel reservation. It consists

Table B.1: Classification error rates [%] using four orientation classes for the BTEC task (W: words, C: classes).

		Arabic-English			Chinese-English			Japanese-English		
Baseline		31.4			44.9			59.0		
Lang.	Window	W	C	W+C	W	C	W+C	W	C	W+C
Tgt	$d = 0$	24.5	27.7	24.2	30.0	34.4	29.7	28.9	31.4	28.7
	$d \in \{0, 1\}$	23.9	27.2	23.7	29.2	32.9	28.9	28.7	30.6	28.3
	$d \in \{-1, 0, 1\}$	22.1	25.3	21.9	27.6	31.4	27.4	28.3	30.1	28.2
Src	$d = 0$	22.1	23.2	20.4	25.9	27.7	20.4	24.1	24.9	22.3
	$d \in \{0, 1\}$	11.9	12.0	10.8	14.0	14.9	13.2	18.6	19.5	17.7
	$d \in \{-1, 0, 1\}$	10.1	8.7	8.0	11.4	11.1	10.5	15.6	15.6	14.5
Src	$d = 0$	20.9	21.8	19.6	24.1	26.8	19.6	22.3	23.4	21.1
+	$d \in \{0, 1\}$	11.8	11.5	10.6	13.5	14.5	12.8	18.6	18.8	17.1
Tgt	$d \in \{-1, 0, 1\}$	9.6	7.7	7.6	11.3	10.1	10.1	15.6	15.2	14.2

Table B.2: Corpus statistics of the Verbmobil task.

		German	English
Train	Sentences	58 073	
	Words	519 523	549 921
	Vocabulary	7 939	4 672
	Singletons	3 453	1 698

Table B.3: Corpus statistics of the Canadian Hansards task.

		French	English
Train	Sentences	1.5 M	
	Words	24 M	22 M
	Vocabulary	100 269	78 332
	Singletons	40 199	31 319

of transcriptions of spontaneous speech. Table B.2 shows the corpus statistics of this corpus. The training corpus (Train) was used to train the IBM model parameters.

Additionally, we carried out experiments on the Canadian Hansards task. This task consists of the proceedings of the Canadian parliament, which are kept by law in both French and English. About 3 million parallel sentences of this bilingual data have been made available by the Linguistic Data Consortium (LDC). Here, we use a subset of the data containing only sentences with a maximum length of 30 words. Table B.3 shows the training and test corpus statistics.

Table B.4 shows the results for the Verbmobil task and for the Canadian Hansards task. It contains the results for both translation directions German-English (S→T) and English-German (T→S) for the Verbmobil task and French-English (S→T) and English-French (T→S) for the Canadian Hansards task, respectively.

Table B.4: Coverage on the training corpus for alignment constraints for the Verbmobil task and for the Canadian Hansards task (S→T: source-to-target direction, T→S: target-to-source direction).

			Coverage [%]	
Task	Constraint		S→T	T→S
Verbmobil	IBM		91.0	88.1
	ITG	baseline	91.6	87.0
		extended	96.5	96.9
Canadian Hansards	IBM		87.1	86.7
	ITG	baseline	81.3	73.6
		extended	96.1	95.6

For the Verbmobil task, the baseline ITG constraints and the IBM constraints result in a similar coverage. It is about 91% for the German-English translation direction and about 88% for the English-German translation direction. A significantly higher coverage of about 96% is obtained with the extended ITG constraints. Thus, with the extended ITG constraints, the coverage increases by about 8% absolute.

For the Canadian Hansards task, the baseline ITG constraints yield a worse coverage than the IBM constraints. Especially for the English-French translation direction, the ITG coverage of 73.6% is very low. One reason might be that the negation "ne ... pas" ↔ "not" cannot be handled with the standard ITG constraints. Again, the extended ITG constraints obtained the best results. Here, the coverage increases from about 87% for the IBM constraints to about 96% for the extended ITG constraints.

We have described the ITG constraints in detail and compared them to the IBM constraints. We draw the following conclusions: especially for long sentences the ITG constraints allow for higher flexibility in word-reordering than the IBM constraints. Regarding the Viterbi alignment in training, the baseline ITG constraints yield a similar coverage as the IBM constraints on the Verbmobil task. On the Canadian Hansards task the baseline ITG constraints were not sufficient. With the extended ITG constraints the coverage improves significantly on both tasks. On the Canadian Hansards task the coverage increases from about 87% to about 96%.

B.3 Results for Different Phrase-level Reordering Constraints

B.3.1 Corpus statistics

To investigate the effect of reordering constraints, we have chosen two Japanese–English tasks, because the word order in Japanese and English is rather different. The first task is the *Basic Travel Expression Corpus* (BTEC) task [Takezawa & Sumita$^+$ 02]. The corpus statistics are shown in Table B.5. This corpus consists of phrasebook entries. Note that this version of the BTEC corpus is not identical to the one used in Section 7.2.1. Here, we use a significantly larger portion of the Japanese-English corpus. This corpus was kindly provided by the Advanced Telecommunication Research Institute International (ATR), Kyoto, Japan.

The second task is the *Spoken Language DataBase* (SLDB) task [Morimoto & Uratani$^+$ 94]. This task consists of transcription of spoken dialogs in the domain of hotel reservation. Here, we use domain-specific training data in addition to the BTEC corpus. The corpus statistics of this additional corpus are shown in Table B.6. The development corpus is the same for both tasks.

Table B.5: Statistics of the BTEC corpus.

		Japanese	English
Train	Sentences	152 K	
	Words	1 044 K	893 K
	Vocabulary	17 047	12 020
Dev	sentences	500	
	words	3 361	2 858
Test	sentences	510	
	words	3 498	–

Table B.6: Statistics of the SLDB corpus. This corpus is used in addition to the BTEC corpus.

		Japanese	English
Train	Sentences	15 K	
	Words	201 K	190 K
	Vocabulary	4 757	3 663
Test	sentences	330	
	words	3 940	–

B.3.2 System comparison

The experiments in this section were carried out using the alignment template approach [Och & Ney 04] which is closely related to the phrase-based approach. The main differences is that in the alignment template approach word classes are used to generalize the phrases. In Table B.7 and Table B.8, we show the translation results for the BTEC task. First, we observe that the overall quality is rather high on this task. The average length of the used alignment templates is about five source words in all systems. The monotonic search (**mon**) shows already good performance on short sentences with less than 10 words. We conclude that for short sentences the reordering is captured within the alignment templates. On the other hand, the monotonic search degrades for long sentences with at least 10 words resulting in a WER of 16.6% for these sentences.

We present the results for various non-monotonic search variants: the first one is with the IBM constraints (**skip**) as described in Section 5.5.3. We allow for one or two gaps, i. e. skip 1 or skip 2.The experiments showed that if we set the maximum number of gaps to three or more the translation results are equivalent to the search without any reordering constraints (**free**). The results for the ITG constraints as described in Section 5.5.2 are also presented.

The unconstrained reorderings improve the total translation quality down to a WER of 11.5%. We see that especially the long sentences benefit from the reorderings resulting in

Table B.7: Effect of phrase-level reordering constraint for the BTEC task (510 sentences). Sentence lengths: short: < 10 words, long: ≥ 10 words; times in milliseconds per sentence.

		WER[%]			
		sentence length			
reorder		short	long	all	time[ms]
mon		11.4	16.6	12.7	73
skip	1	10.8	13.5	11.4	134
	2	10.8	13.4	11.4	169
free		10.8	13.8	11.5	194
ITG		10.6	12.2	11.0	164

Table B.8: Translation performance for the BTEC task (510 sentences).

		error rates[%]		accuracy measures	
reorder		WER	PER	BLEU[%]	NIST
mon		12.7	10.6	86.8	14.14
skip	1	11.4	10.1	88.0	14.19
	2	11.4	10.1	88.1	14.20
free		11.5	10.0	88.0	14.19
ITG		11.0	9.9	88.2	14.25

an improvement from 16.6% to 13.8%. Comparing the results for the free reorderings and the ITG reorderings, we see that the ITG system always outperforms the unconstrained system. The improvement on the whole test set is statistically significant at the 95% level.[a]

In Table B.9 and Table B.10, we show the results for the SLDB task. First, we observe that the overall quality is lower than for the BTEC task. The SLDB task is a spoken language translation task and the training corpus for spoken language is rather small. This is also reflected in the average length of the used alignment templates that is about three source words compared to about five words for the BTEC task.

The results on this task are similar to the results on the BTEC task. Again, the ITG constraints perform best. Here, the improvement compared to the unconstrained search is statistically significant at the 99% level. Compared to the monotonic search, the BLEU score for the ITG constraints improves from 54.4% to 57.1%.

[a] The statistical significance test were done for the WER using boostrap resampling.

Table B.9: Effect of phrase-level reordering constraint for the SLDB task (330 sentences). Sentence lengths: short: < 10 words, long: ≥ 10 words; times in milliseconds per sentence.

		WER[%]			
		sentence length			
reorder		short	long	all	time[ms]
mon		32.0	52.6	48.1	911
skip	1	31.9	51.1	46.9	3 175
	2	32.0	51.4	47.2	4 549
free		32.0	51.4	47.2	4 993
ITG		31.8	50.9	46.7	4 472

Table B.10: Translation performance for the SLDB task (330 sentences).

		error rates[%]		accuracy measures	
reorder		WER	PER	BLEU[%]	NIST
mon		48.1	35.5	54.4	9.45
skip	1	46.9	35.0	56.8	9.71
	2	47.2	35.1	57.1	9.74
free		47.2	34.9	57.1	9.75
ITG		46.7	34.6	57.1	9.76

Bibliography

[Al-Onaizan & Curin$^+$ 99] Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J.D. Lafferty, I.D. Melamed, D. Purdy, F.J. Och, N.A. Smith, D. Yarowsky: Statistical Machine Translation, Final Report, JHU Workshop, 1999. `http://www.clsp.jhu.edu/ws99/projects/mt/final_report/mt-final-report.ps`.

[Ayan & Dorr 06] N.F. Ayan, B.J. Dorr: Going Beyond AER: An Extensive Analysis of Word Alignments and Their Impact on MT. Proc. *21st Int. Conf. on Computational Linguistics and 44th Annual Meeting of the Assoc. for Computational Linguistics (COLING/ACL)*, pp. 9–16, Sydney, Australia, July 2006.

[Banerjee & Lavie 05] S. Banerjee, A. Lavie: METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. Proc. *43rd Annual Meeting of the Assoc. for Computational Linguistics (ACL): Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, pp. 65–72, Ann Arbor, MI, June 2005.

[Baum 72] L.E. Baum: An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes. *Inequalities*, Vol. 3, pp. 1–8, 1972.

[Baxter 64] G. Baxter: On Fixed Points of the Composite of Commuting Functions. *American Mathematical Society*, Vol. 15, No. 6, pp. 851–855, December 1964.

[Bellman 57] R. Bellman: *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[Bender & Hasan$^+$ 05] O. Bender, S. Hasan, D. Vilar, R. Zens, H. Ney: Comparison of Generation Strategies for Interactive Machine Translation. Proc. *10th Annual Conf. of the European Assoc. for Machine Translation (EAMT)*, pp. 33–40, Budapest, Hungary, May 2005.

[Bender & Zens$^+$ 04] O. Bender, R. Zens, E. Matusov, H. Ney: Alignment Templates: the RWTH SMT System. Proc. *Int. Workshop on Spoken Language Translation (IWSLT)*, pp. 79–84, Kyoto, Japan, September 2004.

[Berger & Brown$^+$ 96] A.L. Berger, P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, J.R. Gillett, A.S. Kehler, R.L. Mercer: Language Translation Apparatus and Method of Using Context-based Translation Models, United States Patent 5510981, April 1996.

[Berger & Della Pietra$^+$ 96] A.L. Berger, S.A. Della Pietra, V.J. Della Pietra: A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, Vol. 22, No. 1, pp. 39–72, March 1996.

[Bertoldi 05] N. Bertoldi: *Statistical Models and Search Algorithms for Machine Translation.* Ph.D. thesis, University of Trento, Trento, Italy, March 2005.

[Bertoldi & Federico 05] N. Bertoldi, M. Federico: A New Decoder for Spoken Language Translation Based on Confusion Networks. Proc. *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 86–91, San Juan, Puerto Rico, November/December 2005.

[Bertoldi & Zens$^+$ 07] N. Bertoldi, R. Zens, M. Federico: Speech Translation by Confusion Networks Decoding. Proc. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. IV, pp. 1297–1300, Honolulu, Hawaii, April 2007.

[Birch & Callison-Burch$^+$ 06] A. Birch, C. Callison-Burch, M. Osborne, P. Koehn: Constraining the Phrase-Based, Joint Probability Statistical Translation Model. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL): Workshop on Statistical Machine Translation*, pp. 154–157, New York City, NY, June 2006.

[Bisani & Ney 04] M. Bisani, H. Ney: Bootstrap Estimates for Confidence Intervals in ASR Performance Evaluationx. Proc. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 409–412, Montreal, Canada, May 2004.

[Blatz & Fitzgerald$^+$ 03] J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, N. Ueffing: Confidence Estimation for Machine Translation. Final report, JHU/CLSP Summer Workshop, 113 pages, 2003. `http://www.clsp.jhu.edu/ws2003/groups/estimate/`.

[Blatz & Fitzgerald$^+$ 04] J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, N. Ueffing: Confidence Estimation for Machine Translation. Proc. *20th Int. Conf. on Computational Linguistics (COLING)*, pp. 315–321, Geneva, Switzerland, August 2004.

[Brown & Cocke$^+$ 88] P.F. Brown, J. Cocke, S. Della Pietra, V.J. Della Pietra, F. Jelinek, R.L. Mercer, P.S. Roossin: A Statistical Approach to Language Translation. Proc. *12th Int. Conf. on Computational Linguistics (COLING)*, pp. 71–76, Budapest, Hungary, August 1988.

[Brown & Cocke$^+$ 90] P.F. Brown, J. Cocke, S.A. Della Pietra, V.J. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, P.S. Roossin: A Statistical Approach to Machine Translation. *Computational Linguistics*, Vol. 16, No. 2, pp. 79–85, June 1990.

[Brown & Della Pietra$^+$ 93] P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, R.L. Mercer: The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, Vol. 19, No. 2, pp. 263–311, June 1993.

[Callison-Burch & Osborne$^+$ 06] C. Callison-Burch, M. Osborne, P. Koehn: Re-evaluating the Role of BLEU in Machine Translation Research. Proc. *11th Conf. of the Europ. Chapter of the Assoc. for Computational Linguistics (EACL)*, pp. 249–256, Trento, Italy, April 2006.

[Cettolo & Federico 04] M. Cettolo, M. Federico: Minimum Error Training of Log-linear Translation Models. Proc. *Int. Workshop on Spoken Language Translation (IWSLT)*, pp. 103–106, Kyoto, Japan, September 2004.

[Chen & Goodman 98] S.F. Chen, J. Goodman: An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, Cambridge, MA, 63 pages, August 1998.

[Chen & Rosenfeld 99] S.F. Chen, R. Rosenfeld: A Gaussian Prior for Smoothing Maximum Entropy Models. Technical Report CMUCS-99-108, Carnegie Mellon University, Pittsburgh, PA, 25 pages, February 1999.

[Cherry & Lin 03] C. Cherry, D. Lin: A Probability Model to Improve Word Alignment. Proc. *41st Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 88–95, Sapporo, Japan, July 2003.

[Chiang 05] D. Chiang: A Hierarchical Phrase-Based Model for Statistical Machine Translation. Proc. *43rd Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 263–270, Ann Arbor, Michigan, June 2005.

[Chiang 07] D. Chiang: Hierarchical Phrase-Based Translation. *Computational Linguistics*, Vol. 33, No. 2, pp. 201–228, June 2007.

[Chung & Graham$^+$ 78] F. Chung, R. Graham, V. Hoggatt, M. Kleimann: The Number of Baxter Permutations. *Journal of Combinatorial Theory Series A*, Vol. 24, No. 3, pp. 382–394, 1978.

[Darroch & Ratcliff 72] J.N. Darroch, D. Ratcliff: Generalized Iterative Scaling for Log-Linear Models. *Annals of Mathematical Statistics*, Vol. 43, pp. 1470–1480, 1972.

[Dempster & Laird$^+$ 77] A.P. Dempster, N.M. Laird, D.B. Rubin: Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistics Society Series B*, Vol. 39, No. 1, pp. 1–22, 1977.

[DeNeefe & Knight$^+$ 07] S. DeNeefe, K. Knight, W. Wang, D. Marcu: What Can Syntax-based MT Learn from Phrase-based MT? Proc. *Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pp. 755–763, Prague, Czech Republic, June 2007.

[Doddington 02] G. Doddington: Automatic Evaluation of Machine Translation Quality Using N-gram Co-occurrence Statistics. Proc. *ARPA Workshop on Human Language Technology*, 2002.

[Dreyer & Hall$^+$ 07] M. Dreyer, K. Hall, S. Khudanpur: Comparing Reordering Constraints for SMT Using Efficient BLEU Oracle Computation. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL): Workshop on Syntax and Structure in Statistical Translation (SSST)*, pp. 103–110, Rochester, NY, April 2007.

[Duda & Hart$^+$ 00] R.O. Duda, P.E. Hart, D.G. Stork: *Pattern Classification.* John Wiley and Sons, New York, NY, 2nd edition, 2000.

[Dulucq & Guibert 96] S. Dulucq, O. Guibert: Stack Words, Standard Tableaux and Baxter Permutations. *Discrete Mathematics*, Vol. 157, No. 4, pp. 91–106, 1996.

[Eck & Hori 05] M. Eck, C. Hori: Overview of the IWSLT 2005 Evaluation Campaign. Proc. *Int. Workshop on Spoken Language Translation (IWSLT)*, pp. 11–32, Pittsburgh, PA, October 2005.

[Ehling & Zens$^+$ 07] N. Ehling, R. Zens, H. Ney: Minimum Bayes Risk Decoding for BLEU. Proc. *45th Annual Meeting of the Assoc. for Computational Linguistics (ACL): Poster Session*, pp. 101–104, Prague, Czech Republic, June 2007.

[Eppstein 01] D. Eppstein: Bibliography on k shortest paths and other "k best solutions" problems, `http://www.ics.uci.edu/∼eppstein/bibs/kpath.bib`. URL, March 2001.

[EU 07] European Commission - Directorate-General for Translation - FAQ, 2007. `http://ec.europa.eu/dgs/translation/navigation/faq/faq_facts_en.htm`.

[Foster & Kuhn$^+$ 06] G. Foster, R. Kuhn, H. Johnson: Phrasetable Smoothing for Statistical Machine Translation. Proc. *Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pp. 53–61, Sydney, Australia, July 2006.

[Fraser & Marcu 07] A. Fraser, D. Marcu: Measuring Word Alignment Quality for Statistical Machine Translation. *Computational Linguistics*, Vol. 33, No. 3, pp. 293–303, September 2007.

[Fry 05] J. Fry: Assembling a Parallel Corpus from RSS News Feeds. Proc. *MT Summit X: Workshop on Example-based Machine Translation*, pp. 59–62, Phuket, Thailand, September 2005.

[Fung & Cheung 04] P. Fung, P. Cheung: Mining Very-Non-Parallel Corpora: Parallel Sentence and Lexicon Extraction via Bootstrapping and EM. Proc. *Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pp. 57–63, Barcelona, Spain, July 2004.

[Galley & Graehl$^+$ 06] M. Galley, J. Graehl, K. Knight, D. Marcu, S. DeNeefe, W. Wang, I. Thayer: Scalable Inference and Training of Context-Rich Syntactic Translation Models. Proc. *21st Int. Conf. on Computational Linguistics and 44th Annual Meeting of the Assoc. for Computational Linguistics (COLING/ACL)*, pp. 961–968, Sydney, Australia, July 2006.

[Galley & Hopkins$^+$ 04] M. Galley, M. Hopkins, K. Knight, D. Marcu: What's in a translation rule? Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL)*, pp. 273–280, Boston, MA, May 2004.

[Germann & Jahr$^+$ 01] U. Germann, M. Jahr, K. Knight, D. Marcu, K. Yamada: Fast Decoding and Optimal Decoding for Machine Translation. Proc. *39th Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 228–235, Toulouse, France, July 2001.

[Germann & Jahr$^+$ 04] U. Germann, M. Jahr, K. Knight, D. Marcu, K. Yamada: Fast Decoding and Optimal Decoding for Machine Translation. *Artificial Intelligence*, Vol. 154, pp. 127–143, 2004.

[Gildea 03] D. Gildea: Loosely Tree-Based Alignment for Machine Translation. Proc. *41st Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 80–87, Sapporo, Japan, July 2003.

[Graham & Knuth$^+$ 94] R.L. Graham, D.E. Knuth, O. Patashnik: *Concrete Mathematics.* Addison-Wesley Publishing Company, Reading, MA, 2nd edition, 1994.

[Hasan & Khadivi$^+$ 06] S. Hasan, S. Khadivi, R. Zens, H. Ney: A Flexible Architecture for CAT Applications. Proc. *11th Annual Conf. of the European Assoc. for Machine Translation (EAMT)*, pp. 81–88, Oslo, Norway, June 2006.

[Hasan & Zens$^+$ 07] S. Hasan, R. Zens, H. Ney: Are Very Large N-best Lists Useful for SMT? Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL), Companion Volume: Short Papers*, pp. 57–60, Rochester, NY, April 2007.

[Held & Karp 62] M. Held, R.M. Karp: A Dynamic Programming Approach to Sequencing Problems. *Journal of the Society of Industrial and Applied Mathematics (SIAM)*, Vol. 10, No. 1, pp. 196–210, 1962.

[Huang & Zhang$^+$ 05] L. Huang, H. Zhang, D. Gildea: Machine Translation as Lexicalized Parsing with Hooks. Proc. *9th Int. Workshop on Parsing Technology (IWPT)*, pp. 65–73, Vancouver, Canada, October 2005.

[Jelinek 98] F. Jelinek: *Statistical Methods for Speech Recognition.* MIT Press, Cambridge, MA, 1998.

[Kanthak & Ney 04] S. Kanthak, H. Ney: FSA: An Efficient and Flexible C++ Toolkit for Finite State Automata Using On-Demand Computation. Proc. *42nd Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 510–517, Barcelona, Spain, July 2004.

[Kanthak & Vilar$^+$ 05] S. Kanthak, D. Vilar, E. Matusov, R. Zens, H. Ney: Novel Reordering Approaches in Phrase-Based Statistical Machine Translation. Proc. *43rd Annual Meeting of the Assoc. for Computational Linguistics (ACL): Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, pp. 167–174, Ann Arbor, MI, June 2005.

[Kasami 65] T. Kasami: An Efficient Recognition and Syntax Analysis Algorithm for Context-free Languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA, August 1965.

[Keijsper & Pendavingh 98] J. Keijsper, R. Pendavingh: An Efficient Algorithm for Minimum-weight Bibranching. *Journal of Combinatorial Theory Series B*, Vol. 73, No. 2, pp. 130–145, July 1998.

[Khadivi & Zens$^+$ 06] S. Khadivi, R. Zens, H. Ney: Integration of Speech to Computer-Assisted Translation Using Finite State Automata. Proc. *21st Int. Conf. on Computational Linguistics and 44th Annual Meeting of the Assoc. for Computational Linguistics (COLING/ACL)*, pp. 467–474, Sydney, Australia, July 2006.

[Kneser & Ney 95] R. Kneser, H. Ney: Improved Backing-Off for M-gram Language Modelling. Proc. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 181–184, Detroit, MI, May 1995.

[Knight 99] K. Knight: Decoding Complexity in Word-Replacement Translation Models. *Computational Linguistics*, Vol. 25, No. 4, pp. 607–615, December 1999.

[Knight & Al-Onaizan 98] K. Knight, Y. Al-Onaizan: Translation with Finite-State Devices. Proc. D. Farwell, L. Gerber, E.H. Hovy, editors, *Third Conference of the Association for Machine Translation in the Americas*, Vol. 1529 of *Lecture Notes in Computer Science*, pp. 421–437. Springer Verlag, October 1998.

[Knuth 73] D.E. Knuth: *The Art of Computer Programming*, Vol. 1 - Fundamental Algorithms. Addison-Wesley, Reading, MA, 2nd edition, 1973.

[Koehn 03] P. Koehn: *Noun Phrase Translation.* Ph.D. thesis, University of Southern California, 2003.

[Koehn 04a] P. Koehn: Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. Proc. *6th Conf. of the Assoc. for Machine Translation in the Americas (AMTA)*, pp. 115–124, Washington DC, September/October 2004.

[Koehn 04b] P. Koehn: Statistical Significance Tests for Machine Translation Evaluation. Proc. *Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pp. 388–395, Barcelona, Spain, July 2004.

[Koehn & Axelrod$^+$ 05] P. Koehn, A. Axelrod, A.B. Mayne, C. Callison-Burch, M. Osborne, D. Talbot: Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. Proc. *Int. Workshop on Spoken Language Translation (IWSLT)*, pp. 78–85, Pittsburgh, PA, October 2005.

[Koehn & Hoang$^+$ 07] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantine, E. Herbst: Moses: Open Source Toolkit for Statistical Machine Translation. Proc. *45th Annual Meeting of the Assoc. for Computational Linguistics (ACL): Poster Session*, pp. 177–180, Prague, Czech Republic, June 2007.

[Koehn & Och$^+$ 03] P. Koehn, F.J. Och, D. Marcu: Statistical Phrase-Based Translation. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL)*, pp. 127–133, Edmonton, Canada, May/June 2003.

[Kumar & Byrne 03] S. Kumar, W. Byrne: A weighted finite state transducer implementation of the alignment template model for statistical machine translation. Proc. *Human*

Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL), pp. 142–149, Edmonton, Canada, May/June 2003.

[Kumar & Byrne 04] S. Kumar, W. Byrne: Minimum Bayes-Risk Decoding for Statistical Machine Translation. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL)*, pp. 169–176, Boston, MA, May 2004.

[Kumar & Byrne 05] S. Kumar, W. Byrne: Local Phrase Reordering Models for Statistical Machine Translation. Proc. *Human Language Technology Conf. / Conf. on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pp. 161–168, Vancouver, Canada, October 2005.

[Lavie & Agarwal 07] A. Lavie, A. Agarwal: METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. Proc. *45th Annual Meeting of the Assoc. for Computational Linguistics (ACL): Workshop on Statistical Machine Translation*, pp. 228–231, Prague, Czech Republic, June 2007.

[Lehmer 70] D.H. Lehmer: Permutations with strongly restricted displacements. In *Combinatorial Theory and its Applications*, Vol. 2, pp. 755–770. North-Holland, Amsterdam, The Netherlands, 1970.

[Lin & Och 04] C.Y. Lin, F.J. Och: ORANGE: a Method for Evaluating Automatic Evaluation Metrics for Machine Translation. Proc. *20th Int. Conf. on Computational Linguistics (COLING)*, pp. 501–507, Geneva, Switzerland, August 2004.

[Malouf 02] R. Malouf: A Comparison of Algorithms for Maximum Entropy Parameter Estimation. Proc. *Conf. on Natural Language Learning (CoNLL)*, pp. 49–55, Taipei, Taiwan, August/September 2002.

[Mangu & Brill$^+$ 00] L. Mangu, E. Brill, A. Stolcke: Finding Consensus in Speech Recognition: Word Error Minimization and Other Applications of Confusion Networks. *Computer, Speech and Language*, Vol. 14, No. 4, pp. 373–400, October 2000.

[Marcu & Wong 02] D. Marcu, W. Wong: A Phrase-Based, Joint Probability Model for Statistical Machine Translation. Proc. *Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pp. 133–139, Philadelphia, PA, July 2002.

[Mathias & Byrne 06] L. Mathias, W. Byrne: Statistical Phrase-based Speech Translation. Proc. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 561–564, Toulouse, France, May 2006.

[Matusov & Ney 05] E. Matusov, H. Ney: Phrase-based Translation of Speech Recognizer Word Lattices using Loglinear Model Combination. Proc. *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 110–115, San Juan, Puerto Rico, November/December 2005.

[Matusov & Popović$^+$ 04] E. Matusov, M. Popović, R. Zens, H. Ney: Statistical Machine Translation of Spontaneous Speech with Scarce Resources. Proc. *Int. Workshop on Spoken Language Translation (IWSLT)*, pp. 139–146, Kyoto, Japan, September 2004.

[Matusov & Zens$^+$ 04] E. Matusov, R. Zens, H. Ney: Symmetric Word Alignments for Statistical Machine Translation. Proc. *20th Int. Conf. on Computational Linguistics (COLING)*, pp. 219–225, Geneva, Switzerland, August 2004.

[Matusov & Zens$^+$ 06] E. Matusov, R. Zens, D. Vilar, A. Mauser, M. Popović, S. Hasan, H. Ney: The RWTH Machine Translation System. Proc. *TC-Star Workshop on Speech-to-Speech Translation*, pp. 31–36, Barcelona, Spain, June 2006.

[Melamed 00] I.D. Melamed: Models of Translational Equivalence among Words. *Computational Linguistics*, Vol. 26, No. 2, pp. 221–249, June 2000.

[Melamed 04] I.D. Melamed: Statistical Machine Translation by Parsing. Proc. *42nd Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 653–660, Barcelona, Spain, July 2004.

[Mohri & Pereira$^+$] M. Mohri, F.C. Pereira, M.D. Riley: AT&T FSM Library - Finite State Machine Library. `http://www.research.att.com/~fsmtools/fsm`.

[Morimoto & Uratani$^+$ 94] T. Morimoto, N. Uratani, T. Takezawa, O. Furuse, Y. Sobashima, H. Iida, A. Nakamura, Y. Sagisaka, N. Higuchi, Y. Yamazaki: A Speech and Language Database for Speech Translation Research. Proc. *3rd Int. Conf. on Spoken Language Processing (ICSLP)*, pp. 1791–1794, Yokohama, Japan, September 1994.

[Mostefa & Arranz$^+$ 05] D. Mostefa, V. Arranz, K. Choukri, O. Hamon, S. Surcin: TC-STAR Deliverable no. D 30: Evaluation Report. Technical report, Integrated project TC-STAR (IST-2002-FP6-506738) funded by the European Commission, 42 pages, August 2005. http://www.tc-star.org/.

[Mostefa & Garcia$^+$ 06] D. Mostefa, M.N. Garcia, O. Hamon, N. Moreau: TC-STAR Deliverable no. D 16: Evaluation Report. Technical report, Integrated project TC-STAR (IST-2002-FP6-506738) funded by the European Commission, 102 pages, September 2006. http://www.tc-star.org/.

[Mostefa & Hamon$^+$ 07] D. Mostefa, O. Hamon, N. Moreau, K. Choukri: TC-STAR Deliverable no. D 30: Evaluation Report. Technical report, Integrated project TC-STAR (IST-2002-FP6-506738) funded by the European Commission, 91 pages, May 2007. http://www.tc-star.org/.

[Munteanu & Fraser$^+$ 04] D.S. Munteanu, A. Fraser, D. Marcu: Improved Machine Translation Performance via Parallel Sentence Extraction from Comparable Corpora. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL)*, pp. 265–272, Boston, MA, May 2004.

[Nagata & Yamamoto$^+$ 06] M. Nagata, K. Yamamoto, K. Ohashi: A Clustered Global Phrase Reordering Model for Statistical Machine Translation. Proc. *21st Int. Conf. on Computational Linguistics and 44th Annual Meeting of the Assoc. for Computational Linguistics (COLING/ACL)*, pp. 713–720, Sydney, Australia, July 2006.

[Nelder & Mead 65] J.A. Nelder, R. Mead: A Simplex Method for Function Minimization. *The Computer Journal*, Vol. 7, pp. 308–313, 1965.

[Ney 99] H. Ney: Speech Translation: Coupling of Recognition and Translation. Proc. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 517–520, Phoenix, AR, March 1999.

[Ney 01] H. Ney: Stochastic Modelling: from Pattern Classification to Language Translation. Proc. *39th Annual Meeting of the Assoc. for Computational Linguistics (ACL): Workshop on Data-Driven Machine Translation*, pp. 1–5, Morristown, NJ, July 2001.

[Ney & Haeb-Umbach$^+$ 92] H. Ney, R. Haeb-Umbach, B.H. Tran, M. Oerder: Improvements in Beam Search for 10000-Word Continuous Speech Recognition. Proc. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 9–12, San Francisco, CA, March 1992.

[Ney & Martin$^+$ 97] H. Ney, S. Martin, F. Wessel: Statistical Language Modeling Using Leaving-One-Out. In S. Young, G. Bloothooft, editors, *Corpus-Based Methods in Language and Speech Processing*, pp. 174–207. Kluwer, 1997.

[Och 99] F.J. Och: An Efficient Method for Determining Bilingual Word Classes. Proc. *9th Conf. of the Europ. Chapter of the Assoc. for Computational Linguistics (EACL)*, pp. 71–76, Bergen, Norway, June 1999.

[Och 01] F.J. Och: YASMET: Toolkit for Conditional Maximum Entropy Models, 2001. http://www-i6.informatik.rwth-aachen.de/web/Software/YASMET.html.

[Och 02] F.J. Och: *Statistical Machine Translation: From Single-Word Models to Alignment Templates.* Ph.D. thesis, Lehrstuhl für Informatik 6, Computer Science Department, RWTH Aachen University, Aachen, Germany, October 2002.

[Och 03] F.J. Och: Minimum Error Rate Training in Statistical Machine Translation. Proc. *41st Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 160–167, Sapporo, Japan, July 2003.

[Och & Gildea$^+$ 03] F.J. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, D. Radev: Syntax for Statistical Machine Translation. Technical report, Johns Hopkins University 2003 Summer Workshop on Language Engineering, Center for Language and Speech Processing, Baltimore, MD, 120 pages, August 2003.

[Och & Gildea$^+$ 04] F.J. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, D. Radev: A Smorgasbord of Features for Statistical Machine Translation. Proc. *Human Language Technology Conf.*

/ North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL), pp. 161–168, Boston, MA, May 2004.

[Och & Ney 00] F.J. Och, H. Ney: Improved Statistical Alignment Models. Proc. *38th Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 440–447, Hong Kong, China, October 2000.

[Och & Ney 02] F.J. Och, H. Ney: Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. Proc. *40th Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 295–302, Philadelphia, PA, July 2002.

[Och & Ney 03] F.J. Och, H. Ney: A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, Vol. 29, No. 1, pp. 19–51, March 2003.

[Och & Ney 04] F.J. Och, H. Ney: The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, Vol. 30, No. 4, pp. 417–449, December 2004.

[Och & Tillmann$^+$ 99] F.J. Och, C. Tillmann, H. Ney: Improved Alignment Models for Statistical Machine Translation. Proc. *Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP)*, pp. 20–28, College Park, MD, June 1999.

[Och & Zens$^+$ 03] F.J. Och, R. Zens, H. Ney: Efficient Search for Interactive Statistical Machine Translation. Proc. *10th Conf. of the Europ. Chapter of the Assoc. for Computational Linguistics (EACL)*, pp. 387–393, Budapest, Hungary, April 2003.

[Ohashi & Yamamoto$^+$ 05] K. Ohashi, K. Yamamoto, K. Saito, M. Nagata: NUT-NTT Statistical Machine Translation System for IWSLT 2005. Proc. *Int. Workshop on Spoken Language Translation (IWSLT)*, pp. 128–133, Pittsburgh, PA, October 2005.

[Pantel & Ravichandran$^+$ 04] P. Pantel, D. Ravichandran, E. Hovy: Towards Terascale Semantic Acquisition. Proc. *20th Int. Conf. on Computational Linguistics (COLING)*, pp. 771–777, Geneva, Switzerland, August 2004.

[Papineni & Roukos$^+$ 98] K.A. Papineni, S. Roukos, R.T. Ward: Maximum Likelihood and Discriminative Training of Direct Translation Models. Proc. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 189–192, Seattle, WA, May 1998.

[Papineni & Roukos$^+$ 02] K. Papineni, S. Roukos, T. Ward, W.J. Zhu: Bleu: a Method for Automatic Evaluation of Machine Translation. Proc. *40th Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 311–318, Philadelphia, PA, July 2002.

[Pearl 88] J. Pearl: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988. Revised second printing.

[Press & Teukolsky$^+$ 02] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery: *Numerical Recipes in C++.* Cambridge University Press, Cambridge, UK, 2002.

[Schröder 70] E. Schröder: Vier combinatorische Probleme. *Zeitschrift für Mathematik und Physik*, Vol. 15, pp. 361–376, 1870.

[Shapiro & Stephens 91] L. Shapiro, A.B. Stephens: Boostrap Percolation, the Schröder Numbers, and the N-Kings Problem. *SIAM Journal on Discrete Mathematics*, Vol. 4, No. 2, pp. 275–280, May 1991.

[Shen & Sarkar$^+$ 04] L. Shen, A. Sarkar, F.J. Och: Discriminative Reranking for Machine Translation. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL)*, pp. 177–184, Boston, MA, May 2004.

[Shen & Zens$^+$ 06] W. Shen, R. Zens, N. Bertoldi, M. Federico: The JHU Workshop 2006 IWSLT System. Proc. *Int. Workshop on Spoken Language Translation (IWSLT)*, pp. 59–63, Kyoto, Japan, November 2006.

[Smith & Eisner 06] D.A. Smith, J. Eisner: Minimum Risk Annealing for Training Log-Linear Models. Proc. *21st Int. Conf. on Computational Linguistics and 44th Annual Meeting of the Assoc. for Computational Linguistics (COLING/ACL)*, pp. 787–794, Sydney, Australia, July 2006.

[Snover & Dorr$^+$ 06] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, J. Makhoul: A Study of Translation Edit Rate with Targeted Human Annotation. Proc. *Conf. of the Assoc. for Machine Translation in the Americas (AMTA)*, pp. 223–231, Cambridge, MA, August 2006.

[Steinbiss & Tran$^+$ 94] V. Steinbiss, B.H. Tran, H. Ney: Improvements in Beam Search. Proc. *Int. Conf. on Spoken Language Processing (ICSLP)*, pp. 2143–2146, September 1994.

[Stolcke 02] A. Stolcke: SRILM – An Extensible Language Modeling Toolkit. Proc. *Int. Conf. on Spoken Language Processing (ICSLP)*, Vol. 2, pp. 901–904, Denver, CO, September 2002.

[Takezawa & Sumita$^+$ 02] T. Takezawa, E. Sumita, F. Sugaya, H. Yamamoto, S. Yamamoto: Toward a Broad-coverage Bilingual Corpus for Speech Translation of Travel Conversations in the Real World. Proc. *3rd Int. Conf. on Language Resources and Evaluation (LREC)*, pp. 147–152, Las Palmas, Spain, May 2002.

[Tillmann 01] C. Tillmann: *Word Re-Ordering and Dynamic Programming based Search Algorithms for Statistical Machine Translation.* Ph.D. thesis, Lehrstuhl für Informatik 6, Computer Science Department, RWTH Aachen University, Aachen, Germany, May 2001.

[Tillmann 03] C. Tillmann: A Projection Extension Algorithm for Statistical Machine Translation. Proc. *Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pp. 1–8, Sapporo, Japan, July 2003.

[Tillmann 06] C. Tillmann: Efficient Dynamic Programming Search Algorithms for Phrase-Based SMT. Proc. *Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, pp. 9–16, New York City, NY, June 2006.

[Tillmann & Ney 00] C. Tillmann, H. Ney: Word Re-ordering and DP-based Search in Statistical Machine Translation. Proc. *18th Int. Conf. on Computational Linguistics (COLING)*, pp. 850–856, Saarbrücken, Germany, July 2000.

[Tillmann & Ney 03] C. Tillmann, H. Ney: Word Reordering and a Dynamic Programming Beam Search Algorithm for Statistical Machine Translation. *Computational Linguistics*, Vol. 29, No. 1, pp. 97–133, March 2003.

[Tillmann & Vogel$^+$ 97] C. Tillmann, S. Vogel, H. Ney, A. Zubiaga: A DP-based Search Using Monotone Alignments in Statistical Translation. Proc. *35th Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 289–296, Madrid, Spain, July 1997.

[Tillmann & Xia 03] C. Tillmann, F. Xia: A Phrase-based Unigram Model for Statistical Machine Translation. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL), Companion Volume: Short Papers*, pp. 106–108, Edmonton, Canada, May/June 2003.

[Tillmann & Zhang 05] C. Tillmann, T. Zhang: A Localized Prediction Model for Statistical Machine Translation. Proc. *43rd Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 557–564, Ann Arbor, MI, June 2005.

[Tillmann & Zhang 06] C. Tillmann, T. Zhang: A Discriminative Global Training Algorithm for Statistical MT. Proc. *21st Int. Conf. on Computational Linguistics and 44th Annual Meeting of the Assoc. for Computational Linguistics (COLING/ACL)*, pp. 721–728, Sydney, Australia, July 2006.

[Tomás & Casacuberta 01] J. Tomás, F. Casacuberta: Monotone Statistical Translation using Word Groups. Proc. *Machine Translation Summit VIII*, pp. 357–361, Santiago de Compostela, September 2001.

[Tomás & Casacuberta 04] J. Tomás, F. Casacuberta: Statistical Machine Translation Decoding Using Target Word Reordering. In *Structural, Syntactic, and Statistical Pattern Recognition*, Vol. 3138 of *Lecture Notes in Computer Science*, pp. 734–743. Springer-Verlag, Lisbon, Portugal, August 2004.

[Toutanova & Ilhan$^+$ 02] K. Toutanova, H.T. Ilhan, C.D. Manning: Extensions to HMM-based Statistical Word Alignment Models. Proc. *Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pp. 87–94, Philadelphia, PA, July 2002.

[Turian & Shen$^+$ 03] J.P. Turian, L. Shen, I.D. Melamed: Evaluation of Machine Translation and its Evaluation. Technical Report Proteus technical report 03-005, Computer Science Department, New York University, 8 pages, 2003.

[Udupa & Maji 06] R. Udupa, H.K. Maji: Computational Complexity of Statistical Machine Translation. Proc. *11th Conf. of the Europ. Chapter of the Assoc. for Computational Linguistics (EACL)*, pp. 25–32, Trento, Italy, April 2006.

[Ueffing 05] N. Ueffing: *Confidence Measures for Statistical Machine Translation.* Ph.D. thesis, Lehrstuhl für Informatik 6, Computer Science Department, RWTH Aachen University, Aachen, Germany, 2005.

[Ueffing & Macherey$^+$ 03] N. Ueffing, K. Macherey, H. Ney: Confidence Measures for Statistical Machine Translation. Proc. *MT Summit IX*, pp. 394–401, New Orleans, LA, September 2003.

[Ueffing & Och$^+$ 02] N. Ueffing, F.J. Och, H. Ney: Generation of Word Graphs in Statistical Machine Translation. Proc. *Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pp. 156–163, Philadelphia, PA, July 2002.

[Venugopal & Zollmann$^+$ 05] A. Venugopal, A. Zollmann, A. Waibel: Training and Evaluating Error Minimization Rules for Statistical Machine Translation. Proc. *43rd Annual Meeting of the Assoc. for Computational Linguistics (ACL): Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, pp. 208–215, Ann Arbor, MI, June 2005.

[Vidal 97] E. Vidal: Finite-State Speech-to-Speech Translation. Proc. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 111–114, Munich, Germany, April 1997.

[Vilar 98] J.M. Vilar: *Aprendizaje de Transductores Subsecuenciales para su empleo en tareas de Dominio Restringido.* Ph.D. thesis, Universidad Politecnica de Valencia, 1998.

[Vilar & Matusov$^+$ 05] D. Vilar, E. Matusov, S. Hasan, R. Zens, H. Ney: Statistical Machine Translation of European Parliamentary Speeches. Proc. *MT Summit X*, pp. 259–266, Phuket, Thailand, September 2005.

[Vilar & Popović$^+$ 06] D. Vilar, M. Popović, H. Ney: AER: Do we need to "improve" our alignments? Proc. *Int. Workshop on Spoken Language Translation (IWSLT)*, pp. 205–212, Kyoto, Japan, November 2006.

[Vilar & Vidal 05] J.M. Vilar, E. Vidal: A Recursive Statistical Translation Model. Proc. *43rd Annual Meeting of the Assoc. for Computational Linguistics (ACL): Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, pp. 199–207, Ann Arbor, MI, June 2005.

[Vogel 03] S. Vogel: SMT Decoder Dissected: Word Reordering. Proc. *Int. Conf. on Natural Language Processing and Knowledge Engineering (NLP-KE)*, pp. 561–566, Beijing, China, October 2003.

[Vogel & Ney$^+$ 96] S. Vogel, H. Ney, C. Tillmann: HMM-Based Word Alignment in Statistical Translation. Proc. *16th Int. Conf. on Computational Linguistics (COLING)*, pp. 836–841, Copenhagen, Denmark, August 1996.

[Wahlster 00] W. Wahlster, editor: *Verbmobil: Foundations of speech-to-speech translations*. Springer Verlag, Berlin, Germany, July 2000.

[Wang & Waibel 97] Y.Y. Wang, A. Waibel: Decoding Algorithm in Statistical Machine Translation. Proc. *35th Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 366–372, Madrid, Spain, July 1997.

[Wasserman 05] L. Wasserman: *All of Statistics: A Concise Course in Statistical Inference*. Springer Science+Business Media, New York City, NY, 2005. 2nd printing.

[Weaver 55] W. Weaver: Translation. In W.N. Locke, A.D. Booth, editors, *Machine Translation of Languages: fourteen essays*, pp. 15–23. MIT Press, Cambridge, MA, 1955.

[Wessel 02] F. Wessel: *Word Posterior Probabilities for Large Vocabulary Continuous Speech Recognition*. Ph.D. thesis, Lehrstuhl für Informatik 6, Computer Science Department, RWTH Aachen University, Aachen, Germany, January 2002.

[West 95] J. West: Generating Trees and the Catalan and Schröder Numbers. *Discrete Mathematics*, Vol. 146, pp. 247–262, November 1995.

[Wu 95] D. Wu: Stochastic Inversion Transduction Grammars, with Application to Segmentation, Bracketing, and Alignment of Parallel Corpora. Proc. *14th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 1328–1334, Montreal, Canada, August 1995.

[Wu 96] D. Wu: A Polynomial-Time Algorithm for Statistical Machine Translation. Proc. *34th Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 152–158, Santa Cruz, CA, June 1996.

[Wu 97] D. Wu: Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, Vol. 23, No. 3, pp. 377–403, September 1997.

[Wu & Wong 98] D. Wu, H. Wong: Machine Translation with a Stochastic Grammatical Channel. Proc. *36th Annual Meeting of the Assoc. for Computational Linguistics and 17th Int. Conf. on Computational Linguistics (COLING-ACL)*, pp. 1408–1414, Montréal, Québec, Canada, August 1998.

[Xiong & Liu$^+$ 06] D. Xiong, Q. Liu, S. Lin: Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. Proc. *21st Int. Conf. on Computational Linguistics and 44th Annual Meeting of the Assoc. for Computational Linguistics (COLING/ACL)*, pp. 521–528, Sydney, Australia, July 2006.

[Xu & Matusov$^+$ 05] J. Xu, E. Matusov, R. Zens, H. Ney: Integrated Chinese Word Segmentation in Statistical Machine Translation. Proc. *Int. Workshop on Spoken Language Translation (IWSLT)*, pp. 141–147, Pittsburgh, PA, October 2005.

[Xu & Zens$^+$ 04] J. Xu, R. Zens, H. Ney: Do We Need Chinese Word Segmentation for Statistical Machine Translation? Proc. *Third SIGHAN Workshop on Chinese Language Learning*, pp. 122–128, Barcelona, Spain, July 2004.

[Xu & Zens$^+$ 05] J. Xu, R. Zens, H. Ney: Sentence Segmentation Using IBM Word Alignment Model 1. Proc. *10th Annual Conf. of the European Assoc. for Machine Translation (EAMT)*, pp. 280–287, Budapest, Hungary, May 2005.

[Xu & Zens$^+$ 06] J. Xu, R. Zens, H. Ney: Partitioning Parallel Documents Using Binary Segmentation. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL): Workshop on Statistical Machine Translation*, pp. 78–85, New York City, NY, June 2006.

[Yamada & Knight 01] K. Yamada, K. Knight: A Syntax-Based Statistical Translation Model. Proc. *39th Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 523–530, Toulouse, France, July 2001.

[Yamada & Knight 02] K. Yamada, K. Knight: A Decoder for Syntax-based Statistical MT. Proc. *40th Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 303–310, Philadelphia, PA, July 2002.

[Younger 67] D. Younger: Recognition and Parsing of Context-free Languages in Time n^3. *Information and Control*, Vol. 10, No. 2, pp. 189–208, 1967.

[Zens & Bender$^+$ 05] R. Zens, O. Bender, S. Hasan, S. Khadivi, E. Matusov, J. Xu, Y. Zhang, H. Ney: The RWTH Phrase-based Statistical Machine Translation System. Proc. *Int. Workshop on Spoken Language Translation (IWSLT)*, pp. 155–162, Pittsburgh, PA, October 2005.

[Zens & Hasan$^+$ 07] R. Zens, S. Hasan, H. Ney: A Systematic Comparison of Training Criteria for Statistical Machine Translation. Proc. *Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pp. 524–532, Prague, Czech Republic, June 2007.

[Zens & Matusov$^+$ 04] R. Zens, E. Matusov, H. Ney: Improved Word Alignment Using a Symmetric Lexicon Model. Proc. *20th Int. Conf. on Computational Linguistics (COLING)*, pp. 36–42, Geneva, Switzerland, August 2004.

[Zens & Ney 03] R. Zens, H. Ney: A Comparative Study on Reordering Constraints in Statistical Machine Translation. Proc. *41st Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pp. 144–151, Sapporo, Japan, July 2003.

[Zens & Ney 04a] R. Zens, H. Ney: Improvements in Phrase-Based Statistical Machine Translation. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL)*, pp. 257–264, Boston, MA, May 2004.

[Zens & Ney$^+$ 04b] R. Zens, H. Ney, T. Watanabe, E. Sumita: Reordering Constraints for Phrase-Based Statistical Machine Translation. Proc. *20th Int. Conf. on Computational Linguistics (COLING)*, pp. 205–211, Geneva, Switzerland, August 2004.

[Zens & Ney 05] R. Zens, H. Ney: Word Graphs for Statistical Machine Translation. Proc. *43rd Annual Meeting of the Assoc. for Computational Linguistics (ACL): Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, pp. 191–198, Ann Arbor, MI, June 2005.

[Zens & Ney 06a] R. Zens, H. Ney: Discriminative Reordering Models for Statistical Machine Translation. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL): Workshop on Statistical Machine Translation*, pp. 55–63, New York City, NY, June 2006.

[Zens & Ney 06b] R. Zens, H. Ney: N-Gram Posterior Probabilities for Statistical Machine Translation. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL): Workshop on Statistical Machine Translation*, pp. 72–77, New York City, NY, June 2006.

[Zens & Ney 07] R. Zens, H. Ney: Efficient Phrase-table Representation for Machine Translation with Applications to Online MT and Speech Translation. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL)*, pp. 492–499, Rochester, NY, April 2007.

[Zens & Och$^+$ 02] R. Zens, F.J. Och, H. Ney: Phrase-Based Statistical Machine Translation. Proc. M. Jarke, J. Koehler, G. Lakemeyer, editors, *25th German Conf. on Artificial Intelligence (KI2002)*, Vol. 2479 of *Lecture Notes in Artificial Intelligence (LNAI)*, pp. 18–32, Aachen, Germany, September 2002. Springer Verlag.

[Zhang & Huang$^+$ 06] H. Zhang, L. Huang, D. Gildea, K. Knight: Synchronous Binarization for Machine Translation. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL)*, pp. 256–263, New York City, NY, June 2006.

[Zhang & Vogel 05] Y. Zhang, S. Vogel: Measuring Confidence Intervals for the Machine Translation Evaluation Metrics. Proc. *Int. Conf. on Theoretical and Methodological Issues in Machine Translation (TMI)*, pp. 85–94, Baltimore, MD, October 2005.

[Zhang & Zens$^+$ 07a] Y. Zhang, R. Zens, H. Ney: Chunk-Level Reordering of Source Language Sentences with Automatically Learned Rules for Statistical Machine Translation. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL): Workshop on Syntax and Structure in Statistical Translation (SSST)*, pp. 1–8, Rochester, NY, April 2007.

[Zhang & Zens$^+$ 07b] Y. Zhang, R. Zens, H. Ney: Improved Chunk-level Reordering for Statistical Machine Translation. Proc. *Int. Workshop on Spoken Language Translation (IWSLT)*, pp. 21–28, Trento, Italy, October 2007.

[Zollmann & Venugopal 06] A. Zollmann, A. Venugopal: Syntax Augmented Machine Translation via Chart Parsing. Proc. *Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics Annual Meeting (HLT-NAACL): Workshop on Statistical Machine Translation*, pp. 138–141, New York City, NY, June 2006.

Lebenslauf

Angaben zur Person

Richard Zens
geboren am 29. März 1977
Geburtsort: Düren

Schulbildung

1984 - 1988 Grundschule Müddersheim

1988 - 1996 Frankengymnasium Zülpich

Wehrdienst

1996 - 1997 Stabs- und Fernmeldebatallion, Coesfeld

Studium

1997 - 2002 Informatikstudium an der RWTH Aachen
Vertiefungsgebiet: Mustererkennung und Sprachverarbeitung
Abschluss als Diplom-Informatiker

2002 - 2008 Promotionsstudium an der RWTH Aachen

2006 Teilnahme am 6-Wochen Summer Research Workshop des Center for Language and Speech Processing an der Johns Hopkins University, Baltimore, Maryland, USA

Arbeitstätigkeiten

1998 - 2001 Studentische Hilfskraft am Lehrstuhl für Mustererkennung und Sprachverarbeitung (Informatik 6) an der RWTH Aachen

2002 - 2008 Wissenschaftlicher Mitarbeiter am Lehrstuhl für Mustererkennung und Sprachverarbeitung (Informatik 6) an der RWTH Aachen

2003 3-monatiger Aufenthalt beim Advanced Telecommunication Research Institute International, Kyoto, Japan

2008 - Research Scientist bei Google Inc., Mountain View, Kalifornien, USA