Word Re-Ordering and Dynamic Programming based Search Algorithm for Statistical Machine Translation

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 Christoph Tillmann

aus

Jülich

Berichter: Universitätsprofessor Dr.-Ing. Hermann Ney
Professor Dr. Francisco Casacuberta

Tag der mündlichen Prüfung: 4.Mai 2001

Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online verfügbar.

Acknowledgements

I would like to thank my scientific advisor Prof. Hermann Ney for his adamant criticism of all my scientific pursuits. I would like to thank the members of my committee: Prof. Francisco Casacuberta, who gave valuable advice on this work as well as Prof. Klaus Indermark and Prof. Gerhard Lakemeyer.

My work has been influenced in a multitude of ways by my group members in Aachen: Sonja Niessen, Franz-Josef Och, Hassan Sawaf, Nicola Ueffing, and Stefan Vogel. The first version of the dynamic programming decoder was joint work with Stefan Vogel, Hassan Sawaf, and Alex Zubiaga. The concept of inverted alignments is due to Sonja Niessen. Franz-Josef Och provided the training software for the IBM-4 model. Nicola Ueffing carried out some of the experiments on the French-to-English translation task. Agniezska Muhr worked on preparing the test data for the German-to-English translation experiments. For all administrative matters concerning my work and my thesis, I would like to thank Christiane Beumers, Ingeborg Jennessen, and Gisela Gillmann.

Thanks a lot to my beloved wife and she might reply to my thankfulness with a cheeky, wry smile of her own. My son is growing up, while I am working. And now, that I might never again want to be a doctor, I could finally explain to him that computers are only the second most interesting things in the world. I would like to thank my mother, my father, and my sister for their support. Still, while being at home I don't feel like touching upon any topic related to my work.

Abstract

In this work, a new search procedure for statistical machine translation (SMT) is proposed that is based on dynamic programming (DP). The starting point is a DP solution to the traveling salesman problem that works by jointly processing tours that visit the same subset of cities. For SMT, the cities correspond to source sentence positions to be translated. Imposing restrictions on the order in which the source positions are translated yields a DP algorithm for carrying out the word re-ordering in SMT efficiently. A simple data-driven search organization allows the algorithm to prune unlikely translation hypotheses. Search restrictions especially useful for the translation directions German-to-English and English-to-German are presented. A generalization of these re-ordering restrictions is given that is applicable to several different translation directions. Translation results are reported with a widely used SMT model.

Zusammenfassung

In dieser Arbeit wird ein neuer Suchalgorithmus für die Statistische Maschinelle Übersetzung vorgestellt. Der Algorithmus beruht auf dem Prinzip der Dynamischen Programmierung (DP). Ausgangspunkt ist eine DP-Lösung für das Traveling-Salesman-Problem, die darauf beruht, daß Untermengen von Städten gemeinsamen bearbeitet werden können, ohne die Reihenfolge, in der die Städte besucht wurden, zu beachten. In der Statistischen Maschinellen Übersetzung entsprechen den Städten Position in dem zu übersetzenden Satz. Unter der Verwendung von geeigneten Einschränkungen der erlaubten Wortumordnungen erhält man einen effizienten DP-basierten Suchalgorithmus. Eine daten-abhängige Suchorganization erlaubt es, unwahrscheinliche Übersetzungshypothesen zu prunen. Sucheinschränkungen, die besonderes nützlich für die Übersetzungsrichtungen Deutsch-Englisch und Englisch-Deutsch sind, werden vorgestellt. Die Umordnungseinschränkungen werden verallgemeinert, wodurch sie auf mehrere unterschiedliche Übersetzungsrichtungen anwendbar sind. Übersetzungsergebnisse werden für ein allgemein verwendetes Übersetzungsmodell und verschiedene Übersetzungsrichtungen vorgestellt.

Contents

1 Introduction **1**
- 1.1 Introduction to Machine Translation . . . 1
- 1.2 Introduction to Statistical Machine Translation . . . 2
- 1.3 Baseline Alignment Models . . . 4
  - 1.3.1 HMM Model Training . . . 6
  - 1.3.2 IBM Model Training . . . 7
- 1.4 State of the Art in Statistical MT . . . 8

2 Objectives **12**

3 Translation Model Extensions **15**
- 3.1 Monotone Alignment Model . . . 15
- 3.2 Extended Language Model . . . 18
  - 3.2.1 Trigger Pairs in Language Modeling . . . 18
  - 3.2.2 Selection of High-Level Triggers . . . 20
  - 3.2.3 Selection of Unigram Level Triggers . . . 22
  - 3.2.4 Selection of Low-Level Triggers . . . 22
  - 3.2.5 Multi-Trigger Model . . . 23
- 3.3 Extended Lexicon Model . . . 25
  - 3.3.1 Word Joining: Viterbi Path Criterion . . . 25
  - 3.3.2 Word Joining: Likelihood Criterion . . . 27
  - 3.3.3 Selection of Target Word Joinings . . . 33
  - 3.3.4 Selection of Multi-Word Extensions . . . 35

4 Search Algorithm for Translation **37**
- 4.1 Introduction into Dynamic Programming . . . 37
- 4.2 Dynamic Programming for Machine Translation . . . 39
- 4.3 DP Algorithm Using Monotone Alignments . . . 40
- 4.4 DP Algorithm Using Inverted Alignments . . . 43
- 4.5 Held & Karp Algorithm for Traveling Salesman Problem . . . 45
- 4.6 DP Algorithm for Statistical Machine Translation . . . 47
  - 4.6.1 Verb Group Re-ordering: German-to-English . . . 48
  - 4.6.2 Word Re-ordering: Generalization . . . 51
  - 4.6.3 Word Re-ordering: IBM Style Restrictions . . . 56
  - 4.6.4 Empirical Complexity Calculations . . . 56

4.7 Beam Search Pruning Techniques . . . 58
4.8 Accelerated Recombination . . . 60
4.9 Details for IBM-4 Model . . . 60
4.10 Beam Search Implementation . . . 62

5 Experimental Results **71**
5.1 Extended Language Model: Trigger Pairs . . . 71
5.1.1 Trigger Pair Examples . . . 72
5.1.2 Trigger Pair Perplexity Results . . . 75
5.1.3 Conclusions on the Use of Word Trigger Pairs . . . 76
5.2 Extended Lexicon Model: Target Word Joining . . . 77
5.3 Details on the IBM Model Training . . . 80
5.4 Details on the Translation Procedure . . . 80
5.5 Performance Measures for Evaluation . . . 82
5.6 Experiments on the German-English Verbmobil Corpus . . . 84
5.7 Experiments on the Italian-English FUB Corpus . . . 100
5.8 Experiments on the French-English Hansard Corpus . . . 103

6 Discussion and Future Work **106**

7 Summary **110**

A Appendix: Quantification of Re-ordering Restrictions **112**

B Complexity of Different Re-ordering Schemata **114**
B.1 Pseudo-Translation Task . . . 114
B.2 Word Re-ordering Complexity . . . 115

C Formal Description for the GE Re-ordering Constraint **121**

D Search Parameter File **123**

E Mathematical Symbols **124**
E.1 Translation Model Symbols . . . 124
E.2 Trigger Language Model Symbols . . . 125
E.3 Extended Lexicon Model Symbols . . . 125
E.4 Translation Search Symbols . . . 126

List of Figures

1.1 Transfer and Interlingua "pyramid" diagram. 1

1.2 Architecture of the statistical translation approach based on Bayes decision rule. 3

1.3 Example of word alignment for two German-to-English sentence pairs taken from the Verbmobil corpus. 4

3.1 Illustration of monotone HMM alignments. 16

3.2 Illustration of two-level monotone HMM alignment. 17

3.3 Example of word trigger pairs. 19

3.4 Example of the word joining carried out for the translation direction German-to-English. The training procedure is carried out subsequently for two different translation directions (Stage II and Stage III). 26

3.5 German-to-English alignment examples, where a single source word is aligned to several target language words. The source word "zum" is mapped to the target words "to the". The German compound nouns "Hauptbahnhof" und "Mittwochnachmittag" are mapped to the English words "main station" and "Wednesday afternoon". 27

3.6 Illustration of the possible target word extensions $(f, \tilde{e})$ for the single-word dependency (f, e_0). The extension shown does not involve additional source language words. 28

3.7 Example for the selection restriction for target language extensions $\tilde{e}$ aligned to a single source word f. The alignment path of the regular translation direction is shown as filled dots. The alignment path of the inverse translation direction is shown as unfilled dots. The following extensions are considered: ("morgens","in the morning") and ("Zahnarzttermin", "dentist's appointment"). 31

3.8 Target language word extension selection for the two German compound nouns "Arbeitskreis" ("work group meeting") and "Personalsitzung" ("staff council meeting"). Here, the single German word in the center should be properly aligned to three English words (a part of the surrounding alignment is shown as well - no aligned words are shown for this part). 33

3.9 Multi-word lexicon extension involving both target and source sentence words. The English phrase of two consecutive words "how about" is mapped to the German phrase "wie sieht es ... aus", where there is a gap of two words between the phrase "wie sieht es" and the German particle "aus". The gap may be filled by different German phrases like "am Freitag", "am Donnerstag" or "etwas später". The unfilled circles indicate additional alignment positions that may be learned as part of a multi-word lexicon extension. . . . 36

4.1 Regular alignment example for the translation direction German-to-English. For each German source word there is exactly one English target word on the alignment path. . . . 43
4.2 Illustration of the transitions in the inverted alignment model. The target sentence is generated from bottom to top. . . . 45
4.3 Illustration of the algorithm by Held & Karp for a traveling salesman problem with $J = 5$ cities. Not all permutations of cities have to be considered. For a given subset of cities the order in which the cities have been visited can be ignored. . . . 47
4.4 Word re-ordering for the translation direction German-to-English: the reordering is restricted to the German verb group. . . . 49
4.5 Order, in which the German source positions are covered for the German-to-English re-ordering example given in Figure 4.4. . . . 50
4.6 Word re-ordering for the translation direction English-to-German: the reordering is restricted to the English verb group. . . . 52
4.7 Order, in which the English source positions are covered for the English-to-German re-ordering example given in Figure 4.6. . . . 53
4.8 Illustration of the IBM-style re-ordering constraint. . . . 56
4.9 Illustration of the traceback procedure for the beam search strategy. . . . 65

6.1 Illustration of the computation of the upper bound calculation for the partial hypothesis extension involving the uncovered positions j, j', and j''. . . . 108

List of Tables

3.1 Algorithm for computing a set $\mathcal{E}$ of candidate extensions $(f, \tilde{e})$. A likelihood threshold τ is used to restrict the extensions selected. 32
3.2 Algorithm for computing a modified target language corpus and a corresponding list of selected extensions. 34

4.1 DP-based search algorithm for the monotone translation model. 42
4.2 DP-based algorithm for solving the traveling salesman problem due to Held and Karp. The outer-most loop is over the cardinality of subsets of already visited cities. 46
4.3 DP-based algorithm for statistical MT which consecutively processes subsets $\mathcal{C}$ of source sentence positions of increasing cardinality. 48
4.4 Procedural description to compute the set $Succ$ of successor hypotheses by which to extend a partial hypothesis $(\mathcal{S}, \mathcal{C}, j)$. 55
4.5 Number or processed arcs for the pseudo-translation task as a function of the input sentence length J (y axis is given in log scale). The complexity for the four different re-ordering constraints MON, GE, EG, and S3 is given. The complexity of the S3 constraint is close to J^4. 57
4.6 Two-list implementation of a DP-based search algorithm. 63
4.7 Two-list implementation of a DP-based search algorithm for statistical MT. . 64
4.8 Implementation of dynamic programing: core of the DP beam search procedure. 67
4.9 Implementation of dynamic programing: program code lines description for Table 4.8. 68
4.10 Implementation of the dynamic programming: the baseline class. 69
4.11 Implementation of dynamic programming: struct representing a partial hypothesis for the IBM-4 model search. 70

5.1 Best word trigger pairs along with perplexity improvement (ΔPP) for the three selection criteria A, B and C (self triggers and same-root triggers excluded). 73
5.2 List of best triggered words b for some triggering words a for the selection criteria A, B and C. 74
5.3 Perplexity results for the extension of a unigram language model (5-million training words) with three different components: low level triggers, unigram triggers and cache. 76
5.4 Perplexity results for the combination of trigger pairs with a trigram/cache language model (1, 5 and 39 million training words). 77

5.5 List of best target word joinings sorted by the log-likelihood improvement $F_{(f,\tilde{e})} - F_{\tilde{e}_0}$ as obtained on the aligned Verbmobil training corpus using the word joining algorithm shown in Table 3.2. 78

5.6 List of best target word joinings where the predicted source word f starts with an upper-case letter. The extensions are sorted by the log-likelihood improvement $F_{(f,\tilde{e})} - F_{\tilde{e}_0}$ as obtained on the aligned Verbmobil training corpus using algorithm 3.2 (The German word "Nachmittag" is abbreviated nach. in compound nouns). 79

5.7 Number of training iterations for the IBM model training as is described in [Brown et al. 1993, Och et al. 2000a]. The IBM-2 training step of the original IBM model training is replaced by an hidden Markov model training step. 81

5.8 Complete list of IBM model parameters that are automatically trained on the training corpora. 81

5.9 Parameters for the search procedure to control the re-ordering restrictions. . 81

5.10 Complete list of pruning parameters to control the beam width within the beam search concept. 82

5.11 Training and test conditions on the German-English Verbmobil corpus (*number of words without punctuation). 84

5.12 Target language vocabulary size, word error rate (WER), and multi-reference word error rate (m-WER) for two cases: 1) Without word joining applied and 2) With word joining applied (using the word joining threshold $\tau = 4.0$) on the TEST-331 test set. 88

5.13 Word error rate (WER) and multi-reference word error rate (m-WER) as a function of the language model scaling factor α on the TEST-331 test set for the GE re-ordering constraint. 88

5.14 Word error rate (WER) and multi-reference word error rate (m-WER) as a function of the distortion model scaling factor α on the TEST-331 test set for the GE re-ordering constraint. 89

5.15 Word error rate (WER) and multi-reference word error rate (m-WER) for four different language models on the TEST-331 test set. For the translation experiments the GE re-ordering constraint is used. 90

5.16 Computing time, multi-reference word error rate (m-WER) and subjective sentence error rate (SSER) for three different search procedures on the TEST-147 test set. No word re-ordering accross punctuation marks is performed. . 90

5.17 Example translations for the translation direction German-to-English using three different re-ordering constraints: MON, GE, and S3. 91

5.18 Effect of the coverage pruning threshold t_C on the number of search errors and the multi-reference word error rate (m-WER) on the TEST-331 test set (no cardinality pruning carried out: $t_c = \infty$). A cardinality histogram pruning of 200 000 is applied to restrict the maximum overall size of the search space. The negative logarithm of t_C is reported. 93

5.19 Effect of the cardinality pruning threshold t_c on the number of search errors and the multi-reference word error rate (m-WER) on the TEST-331 test set (no coverage pruning is carried out: $t_{\mathcal{C}} = \infty$). A coverage histogram pruning of 1 000 is applied to restrict the overall size of the search space. The negative logarithm of t_c is shown. 94
5.20 Effect of the observation pruning on the number of search errors and the m-WER on the TEST-331 test set (parameter setting: $t_c = \infty$, $t_{\mathcal{C}} = 10.0$). No histogram pruning is applied. The results are reported for the GE constraint. 95
5.21 Demonstration of the combination of the two pruning thresholds $t_{\mathcal{C}} = 5.0$ and $t_c = 12.5$ to speed up the search process for the two re-reordering constraints GE and S3 ($n_o = 50$). The translation performance is shown in terms of multi-reference word error rate (m-WER) on the TEST-331 test set. 95
5.22 Translation results for the translation direction English-to-German on the TEST-331 test set. The results are given in terms of computing time, word error rate (WER) and position-independent word error rate (PER) for three different re-ordering constraint: MON, EG, and S3. For the translation direction English-to-German a single reference translation for each test sentence is used for the automatic evaluation. 96
5.23 Example translations for the translation direction English-to-German using three different re-ordering constraints: MON, EG, and S3. 97
5.24 English-to-German translation results on the TEST-251 test set in terms of word error rate (WER) and position-independent word error rate (PER) for the single-word based approach (using three different re-ordering constraints) and for the alignment template approach. 98
5.25 Example translations for the alignment template (AT) and the single-word based (SWB) approach on the TEST-251 test set. The single-word based approach (SWB) uses the GE re-ordering constraint. 99
5.26 Training and test conditions for the FUB task (*number of words without punctuation). 100
5.27 Target language vocabulary size and word error rate (WER) on the test set for the two cases: 1) with word joining and 2) without word joining applied. 101
5.28 Example translations for the translation direction Italian-to-English using the two different re-ordering constraints MON and S2. 101
5.29 Computing time and word error rate (WER) on the 300 test sentences for the translation direction Italian-to-English using four different re-ordering constraints. 102
5.30 Computing time and word error rate (WER) on the 300 test sentences for the translation direction English-to-Italian using four different re-ordering constraints. 102
5.31 Word error rate (WER) on the 300 test set sentences for the translation direction Italian-to-English as a function of the distortion model scaling parameter α_D for the re-ordering constraint S2. 102
5.32 Training and test conditions for the HANSARD task (*number of words without punctuation). 103

5.33 Computing time and word error rate (WER) and position-independent word error rate (PER) for the translation direction French-to-English for the reordering constraints MON and S3 on the 5 432 test sentences. 104
5.34 Computing time and word error rate (WER) and position-independent word error rate (PER) for the translation direction English-to-French for the reordering constraints MON and S3 on the 5 432 test sentences. 104
5.35 Example translations for the translation direction French-to-English using the S3 re-ordering constraint. 105

B.1 Constraints for the MON word re-ordering. 117
B.2 Constraints for the NO word re-ordering. 117
B.3 Constraints for the S3 word re-ordering. 118
B.4 Constraints for the GE word re-ordering. The notation is taken from Appendix A. 120

C.1 Technical Restrictions for the choice of $l, r, r'(r \neq r')$ for the GE constraint partial hypothesis extensions in Table C.2. 121
C.2 Partial hypotheses extensions for the verb group re-ordering using the four verb group states: $\mathcal{I}(Initial), \mathcal{K}(Skip), \mathcal{V}(Move)$, and $\mathcal{U}(Cover)$. 122

Chapter 1

Introduction

1.1 Introduction to Machine Translation

This section briefly reviews basic approaches for **machine translation(MT)**, the use of computers to automate some or all of the process of translating from one language to another. For a more detailed introduction see [Hutchins,Somers 1992, Jurafsky,Martin 2000]. Although the translation of a arbitary text from one language to another might require a deep and rich understanding of the source language and the input text, and a sophisticated and creative command of the target language, simpler translation tasks can be adressed with current computational models. In particular, current machine translation systems often focus on the following:

- Tasks for which **rough translation** is adequate. This can be used for document-finding purposes which then can be submitted to a more detailed translation if necessary.
- Tasks where human **post-editor** can be used to improve MT output. Even a rough draft can sometimes speed-up the overall translation process. Systems used in this way are **computer-added human translation systems** (CAHT or CAT).
- Tasks limited to small **sublanguage domains**. The sublanguage domains include

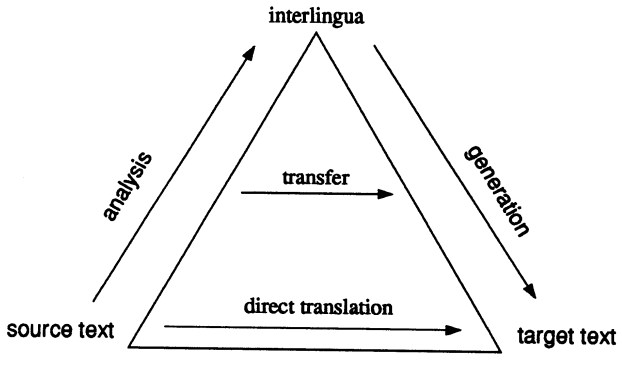


Figure 1.1: Transfer and Interlingua "pyramid" diagram.

weather forecasts, equipement maintenance manuals, air travel queries, appointement scheduling, and restaurant recommendations.

This work will mainly focus on the latter point, namely machine translation in certain sublanguage domains, e.g. appointement scheduling and hotel reservations. There are three basic models for doing MT:

The Transfer Approach involves three phases: The first step is a grammatical **analysis** of the input sentence which produces a parse of some form, followed by a **transfer** step which alters the source sentence structure to get the structure of the target sentence. In the final **generation** step, the actual output sentence is generated.

The Interlingua Approach treats translation as a process to extract the meaning of the input and expressing the meaning in the target language. This scheme assumes the existence of a meaning representation or **interlingua**. The idea is to represent all sentences that mean the "same" thing in the same way. Translation in this model is done by performing a semantic analysis on the input and generating from interlingua to the target language.

The Direct Approach assumes that a MT system should do as little work as possible. No elaborate strutural analyses are carried out at all, just a sequence of simple operations that can be done reliably is performed. Often analysis steps of a procedural character are carried out with only one language pair in mind.

The three approaches are illustrated in Figure 1.1. Analysis and generation steps are carried out to a varying extent. For the interlingua approach the source text is analysized to obtain a meaning representation, which is then further translated to obtain the target text. No transfer of grammatical structure is needed here. The transfer approach uses some analysis and generation to feed its transfer step. For the direct translation approach nearly all the work goes into the transfer step, where a sequence of simple transformation steps are applied.
A statistical model for translation was first described by researchers coming from the field of speech recognition [Brown et al. 1990, Brown et al. 1993] and the approach is very similiar to the Bayesian models used there. This thesis is about a search procedure for the translation model proposed in the two above papers.

1.2 Introduction to Statistical Machine Translation

The goal is the translation of a text given in some source language into a target language. We are given a **source** string $f_1^J = f_1...f_j...f_J$, which is to be translated into a **target** string $e_1^I = e_1...e_i...e_I$. For the mathematical description, we adopt the notation used in [Brown et al. 1993], where the **target language** is English (the symbol e is used for an English word) and the **source language** is French (the symbol f is used for a French word). In this paper, the term *word* always refers to a *full-form* word. Among all possible target strings, we will choose the string with the highest probability which is given by

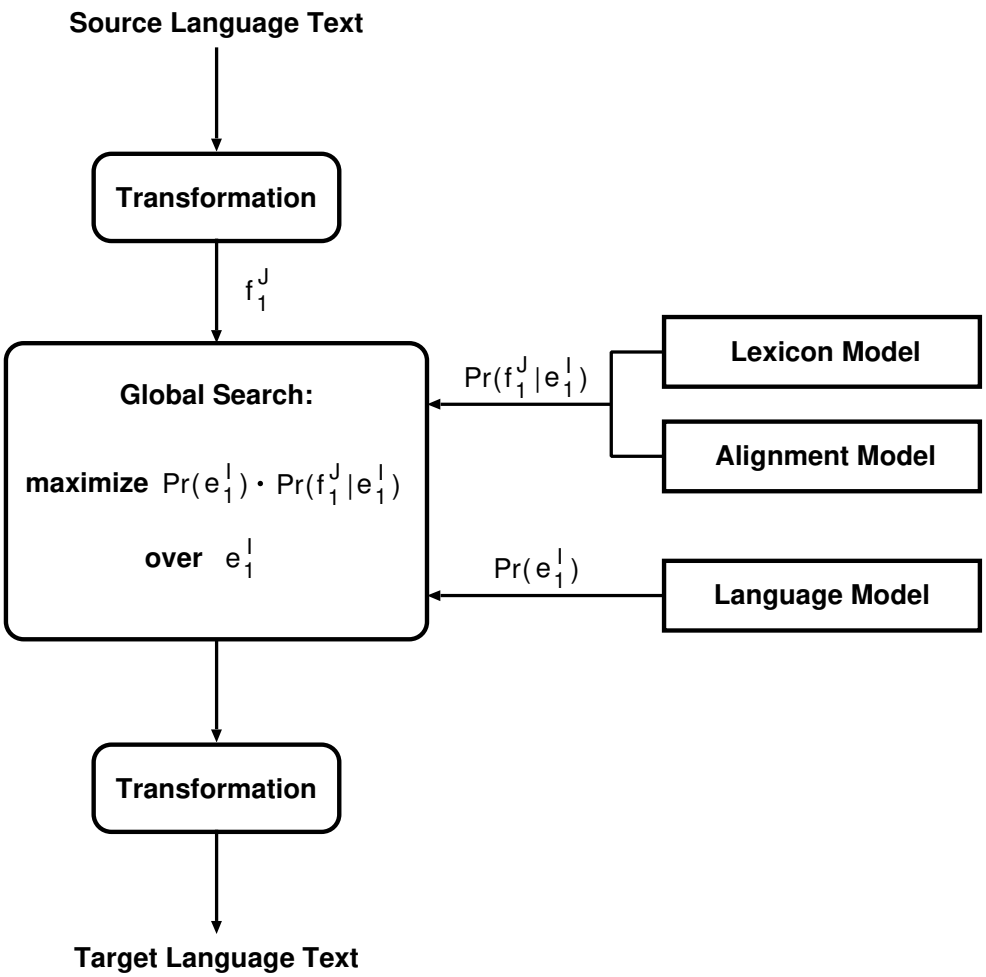


Figure 1.2: Architecture of the statistical translation approach based on Bayes decision rule.

Bayes' decision rule [Brown et al. 1993]:

$$\begin{aligned} \hat{e}_1^I &= \arg\max_{e_1^I} \{Pr(e_1^I|f_1^J)\} \\ &= \arg\max_{e_1^I} \{Pr(e_1^I) \cdot Pr(f_1^J|e_1^I)\} \quad . \end{aligned} \tag{1.1}$$

$Pr(e_1^I)$ is the language model (LM) of the target language, whereas $Pr(f_1^J|e_1^I)$ is the string translation model. The argmax operation denotes the search problem, i.e. the generation of the output sentence in the target language. The overall architecture of the statistical translation approach is summarized in Figure 1.2.
In general, as shown in this figure, there may be additional transformations to make the translation task simpler for the algorithm. The transformations may range from simple word categorization to more complex preprocessing steps that require some parsing of the source string. However, in this work, we will use only word categorization as explicit transformation step. We have to keep in mind that in the search procedure both the language and the translation model are applied *after* the text transformation steps. However, to keep

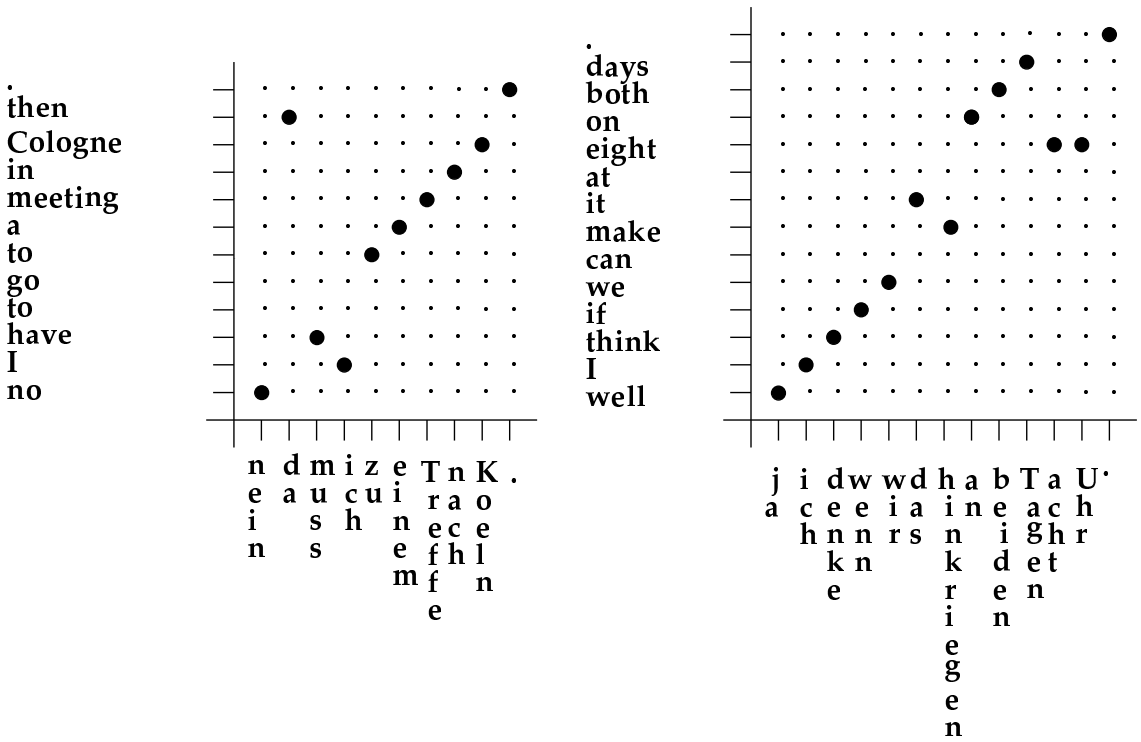


Figure 1.3: Example of word alignment for two German-to-English sentence pairs taken from the Verbmobil corpus.

the notation simple, we will not make this explicit distinction in the subsequent exposition. The notational convention will be as follows. We use the symbol $Pr(.)$ to denote general probability distributions with (nearly) no specific assumptions. In contrast, for model-based probability distributions, we use the generic symbol $p(.)$.

1.3 Baseline Alignment Models

A key issue in modeling the string translation probability $Pr(f_1^J|e_1^I)$ is the question of how we define the correspondence between the words of the target sentence and the words of the source sentence. In typical cases, we can assume a sort of pairwise dependence by considering all word pairs (f_j, e_i) for a given sentence pair $[f_1^J; e_1^I]$. Here, we will further constrain this model by assigning each source word to *exactly one* target word. In Chapter 3 the requirement will be relaxed. Models describing these types of dependencies are referred to as *alignment models* [Brown et al. 1993, Dagan et al. 1993, Röscheisen,Kay 1993, Fung et al. 1994, Vogel et al. 1996]. To arrive at a quantitative specification, we first define the

$$\text{alignment mapping:} \quad j \rightarrow i = a_j \quad ,$$

which assigns a word f_j in position j to a word e_i in position $i = a_j$. An example for such an alignment is given in Figure 1.3. The concept of these alignments is similar to the alignments introduced by [Brown et al. 1993]. By looking at such alignments, it is evident

that the mathematical model should try to capture the strong dependence of a_j on the preceding alignment. Therefore, for our ultimate model, the probability of alignment a_j for position j should have a dependence on the previous alignment position a_{j-1}:

$$p(a_j|a_{j-1}, I, J) \quad .$$

Thus the problem formulation is similar to that of the time alignment problem in speech recognition, where the so-called Hidden Markov models (HMM) have been successfully used for a long time [Jelinek 1976]. Using the same basic principles, we can rewrite the probability by introducing the "hidden" alignments $a_1^J := a_1...a_j...a_J$ for each sentence pair $[f_1^J; e_1^I]$:

$$\begin{aligned} Pr(f_1^J|e_1^I) &= p(J|I) \cdot \sum_{a_1^J} Pr(f_1^J, a_1^J|e_1^I) \qquad (1.2) \\ &= p(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} Pr(f_j, a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) \\ &= p(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} p(f_j, a_j|a_{j-1}, e_1^I) \\ &= p(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} [p(a_j|a_{j-1}, I, J) \cdot p(f_j|e_{a_j})] \quad , \end{aligned}$$

where in the last two equations the dependence has been confined to a first-order dependence. To clarify the meaning of the term "hidden" in comparison with speech recognition, we note that the model states as such (representing words) are *not* hidden but the actual alignments are hidden, i.e. the *sequence* of position index pairs (j, i) with $i = a_j$. To draw the analogy with speech recognition, we have to identify the states (along the vertical axis) with the positions i of the target words e_i and the time (along the horizontal axis) with the positions j of the source words f_j. Putting everything together, we have the following ingredients:

- the sentence length probability: $p(J|I)$,
  which is included here for completeness, but can be omitted without loss of performance;
- the lexicon probability: $p(f|e)$;
- the alignment probability: $p(a_j|a_{j-1}, I, J)$, which here has been chosen as first-order model.

To reduce the number of alignment parameters, we assume that the alignment probabilities $p(a_j|a_{j-1}, I, J)$ depend only on the jump width $(a_j - a_{j-1})$. Using $i = a_j$ and $i' = a_{j-1}$, we have:

$$p(i|i', I) = \frac{q(i - i')}{\sum_{i"=1}^{I} q(i" - i')}$$

with a non-negative table $q(.)$:

$$\Delta i \equiv i - i' \quad \rightarrow \quad q(\Delta i) \quad ,$$

that has to be estimated from the bilingual training corpus. This alignment model will be referred to as *homogeneous* HMM alignment. Rather than a first-order dependence, we can also use a *zero-order* model $p(a_j|j, I, J)$, where there is only a dependence on the *absolute* position index j of the source string. For this zero-order model, it can be shown [Brown et al. 1993] that we have the following identity:

$$\begin{aligned} Pr(f_1^J|e_1^I) &= p(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} [p(a_j|j, I, J) \cdot p(f_j|e_{a_j})] \\ &= p(J|I) \cdot \prod_{j=1}^{J} \sum_{i=1}^{I} [p(i|j, I, J) \cdot p(f_j|e_i)] \end{aligned}$$

The sum in the last equation can be interpreted as a mixture-type distribution with mixture weights $p(i|j, I, J)$ and with component distributions $p(f_j|e_i)$ that model the pairwise dependencies between f_j and e_i. Except for the missing "empty cept" this model is identical to the one denoted by Model 2 in [Brown et al. 1993]. As in the case of HMM alignments, for the general mixture-type alignment model, there may be too many alignment parameters $p(i|j, I, J)$. Therefore, we assume that the position distance relative to the *diagonal* line of the (j, i) plane is the dominating factor (compare Figure 1.3). Thus we obtain:

$$p(i|j, I, J) = \frac{r(i - j \cdot \frac{I}{J})}{\sum_{i'=1}^{I} r(i' - j \cdot \frac{I}{J})}$$

with a non-negative table $r(.)$:

$$\Delta i \equiv i - j \cdot I/J \quad \rightarrow \quad r(\Delta i) \quad ,$$

that has to be estimated from the bilingual training corpus. This alignment model will be referred to as *diagonal* MM alignment (MM = mixture model). Assuming a uniform alignment probability

$$p(i|j, I, J) = \frac{1}{I}$$

we arrive at the so-called IBM-1 model [Brown et al. 1993]. The attractive property of the IBM-1 model is that, for maximum likelihood training [Brown et al. 1993], there is only one optimum and therefore the EM algorithm [Baum 1972] *always* finds the global optimum.

1.3.1 HMM Model Training

To start the training from scratch, the IBM-1 model [Brown et al. 1993] is used to find an initial estimate of the lexicon probabilities. Then, we use a refined model like the diagonal

MM or the homogeneous HMM alignment to train the lexicon model and the alignment model. We use the maximum likelihood criterion in the so-called maximum approximation, i.e. the likelihood criterion covers only the most likely alignment rather than the set of all alignments:

$$\begin{aligned} Pr(f_1^J|e_1^I) &= \sum_{a_1^J} \prod_{j=1}^{J} \Big[p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j})\Big] \\ &\cong \max_{a_1^J} \prod_{j=1}^{J} \Big[p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j})\Big] \quad . \end{aligned}$$

In informal experiments, we found practically no difference between this maximum approximation and the exact sum criterion. To find the optimal alignment, we use dynamic programming (DP). For a general first-order alignment model $p(i|i', I, J)$, we have the following typical recursion formula:

$$Q(i, j) = p(f_j|e_i) \cdot \max_{i'} \{p(i|i', I, J) \cdot Q(i', j-1)\} \quad .$$

Here, $Q(i, j)$ is a sort of partial probability as in time alignment for speech recognition [Jelinek 1976]. As a result, the training procedure amounts to a sequence of iterations, each of which consists of two steps:

- *position alignment:* Given the model parameters, determine the most likely position alignment.
- *parameter estimation:* Given the position alignment, i.e. going along the alignment paths for all sentence pairs, perform maximum likelihood estimation of the model parameters; for model-free distributions, these estimates result in relative frequencies.

1.3.2 IBM Model Training

For the translation experiments presented in Section 5, we use a more complicated translation model ([Brown et al. 1993, Berger et al. 1994]). In this work, the training procedure is based on five models of increasing complexity. Here, the EM-training applied to obtain the experimental results is based on the IBM-4 model training. The IBM-4 model uses the same parameter set as the IBM-5 model which in preliminary experiments did not yield better translation results but took much more time to be carried out. The IBM training procedure is by far more complicated than the hidden Markov model training described above, but in informal experiments the IBM model training resulted in better translation performance and improved alignment accuracy of the resulting alignment as measured on hand-aligned test-data. The actual implementation used during the experiments is described in Section 5.3. The following "types" of parameters are used for the IBM-4 translation model:

Lexicon probabilities: We use the lexicon probability $p(f|e)$ for translating the single target word e as the single source word f. A source word f may be translated by the "null" word e_0, i.e. it does not produce any target word e. A translation probability $p(f|e_0)$ is trained along with the regular translation probabilities.

Fertilities: A single target word e may be aligned to $n = 0, 1$ or more source words. This is explicitly modeled by the fertility parameter $\phi(n|e)$: the probability that the target word e is translated by n source words is $\phi(n|e)$. The fertility for the "null" word is treated specifically (for details see [Brown et al. 1993]). [Berger et al. 1996] describes the extension of a partial hypothesis by a pair of target word (e', e) where e' is not connected to any source word f. In this case, the so-called spontaneous target word e' is accounted for with the fertility probability $\phi(0|e')$ and no translation probability $p(f|e')$.

Class-based distortion probabilities: When covering a source sentence position j, we use distortion probabilities which depend on the previously covered source sentence positions (we say that a source sentence position j is covered for a partial hypothesis when it is taken account for in the translation process by generating a target word or the "null" word e_0). In [Brown et al. 1993], two types of distortion probabilities are distinguished: 1) the leftmost word of a set of source words f aligned to the same target word e is placed (which is called the "head"), and 2) the remaining source words are placed. Two separate distributions are used for the above two cases. For placing the "head" the so-called center function $center(i)$ ([Brown et al. 1993] uses the notation $\odot_i$) is used: the average position of the source words to which the target word e_{i-1} is aligned. The distortion probabilities are class-based: they depend on the word class $\mathcal{F}(f)$ of a covered source word f as well as on the word class $\mathcal{E}(e)$ of the previously generated target word e. The classes are automatically trained [Brown et al. 1992].

Details on the use of IBM-4 model parameters are given in Section 4.9 in the context of the DP-based search procedure.

1.4 State of the Art in Statistical MT

There has been a multitude of papers on statistical machine translation in the recent years starting from IBM's pioneering work in the early nineties. We give a short overview of the different approaches where we will roughly distinguish them according to the type of lexical dependencies they use [Och et al. 1999], namely:

- Single-word based approach, where we use lexical parameters of the type $p(f|e)$, where f is a single source language word and e is a single target language word.
- Multi-word based approach, where the single-word based alignment approach is used as a starting point to learn more complex (grammatical) structures.

This work is about the single-word based approach, where an efficient search algorithm based on dynamic programming is presented.

Single-word based Lexicon Models

A single-word based approach was presented in the pioneering work of the IBM research group on statistical machine translation [Brown et al. 1990, Brown et al. 1993,

Berger et al. 1994, Berger et al. 1996]. It was built on the same statistical principles as used in most speech recognition systems [Jelinek 1976]. [Berger et al. 1994] describes the French-to-English "Candide" translation system. There has been a lot of work that builds upon this original work.
[Wang et al. 1997] presented a search algorithm for the IBM-2 translation model based on the A^* concept. Translation approaches that use the IBM-2 model parameters but are based on dynamic programming are presented in [García et al. 1998a, García et al. 1998b, Niessen et al. 1998]. [García et al. 1998a] presents an iterative search procedure that tries to iteratively improve on a generated target string e_1^I over several iterations using a DP-based search algorithm within each iteration. This approach uses stochastic regular grammars as a language model for the target language. In [García et al. 1998b], the approach is compared to a memory-based translation approach, where both approaches are based on the IBM-2 alignment model. The memory-based approach performs significantly worse on the EuTrans-I data.
An approach based on the HMM alignments as used in speech recognition has been presented in [Tillmann et al. 1997a, Tillmann et al. 1997c]. The work by [Wu 1996] also used the original IBM model parameters but obtained an efficient search algorithm by restricting the possible word re-orderings using the so-called stochastic bracketing transduction grammar (SBTG). In [Wu et al. 1998], this work has been extended to handle a context-free grammar as a target language model.

Multi-word based Lexicon Models

Several approaches build up from the IBM model training or similar methods to learn an alignment mapping. This mapping is used to learn more complex translation models. The alignment template (AT) approach [Och et al. 1998, Ney et al. 2000] takes the original alignment matrix after the final training iteration to learn the so-called alignment templates. Within an alignment template a whole group of source language words is mapped to a whole group of target language words. This approach has the advantage that both word context and local word re-ordering can be taken into account. Similarily, [Wang et al. 1998a] proposes the use of shallow phrase structures that are learned automatically from training data. Both approaches are based on automatically learned word classes.
[Alshawi et al. 1998] presents a statistical translation approach which is based on a collection of so-called *head transducers*. Head transducers are finite-state transducers that try to capture the lexical sensivity of direct statistical translation models by relating the heads of phrases in source and target language. At the same time they take into account the hierarchical phrasal structure of language. The head transducer translation models can be fully automatically learned. The transducer learning uses an alignment matrix that is obtained by a technique presented in [Gale et al. 1991].
Related approaches in a corpus-based or statistical spirit try to learn standard finite-state transducer from bilingual training data. The "Onward Subsequential Transducer Inference Algoritm (OSTIA)" [Vilar et al. 1996, Vidal et al. 1996] is an algorithm for automatically learning subsequential finite-state transducers. The amount of training data needed is drastically reduced using the so-called "OMEGA" algorithm, which makes use of the bilingual alignments provided by statistical techniques.

A new approach that is not based on OSTIA's state-merging paradigm is presented in [Casacuberta 2000]. Stochastic regular grammars are learned by transforming a bilingual training corpus into a string of extended symbols. The extended symbols take into account aligned pairs of source and target language words. Local word re-orderings can be dealt with by including special symbols that allow the re-ordering information to be included into the string of extended symbols. The stochastic regular grammar learned from the strings of extended symbols are then transformed into a finite-state transducer. The approach produced promising results on the Italian-to-English EuTrans-II translation task.

Selection of Baseline Model Extension Using a Likelihood Criterion

The above multi-word based models have in common that they learn more complex structures from underlying single-word based alignment models. In Section 3.3, we will present a method to extract multi-word lexical dependencies using techniques that originally have been developed for the so-called word triggers in language modeling.
The main goal for the use of trigger pairs is to include long-distance dependencies into the language model by means of so-called "trigger pairs", where a trigger pair is defined as a long-distance word pair. We restrict ourselves to trigger pairs where both the triggered and the triggering events are single words (as opposed to word phrases). Thus word trigger pairs can be viewed as long-distance word bigrams. The use of trigger pairs has been presented in [Bahl et al. 1984, Lau et al. 1993, Lau et al. 1993, Rosenfeld 1994]. [Rosenfeld 1994] demonstrates the use of word trigger pairs for his maximum entropy approach to combine different knowledge sources within a single statistical language model. The selection criterion used in this work does not take into account that a trigger pair $a \rightarrow b$ to predict a word b becomes important just in the case where b is not well predicted by a baseline m-gram language model.

Search in Statistical Machine Translation

There are relatively few papers on the search problem in statistical machine translation. A stack decoding algorithm for statistical machine translation for the IBM-2 model has been presented in [Wang et al. 1997]. An extension of this algorithm is demonstrated in [Wang et al. 1998b]. Here, a so-called *reshuffling* step on top of the decoder presented in [Wang et al. 1997] is used to handle more complex translation models, e.g. the IBM-3 model is added.
A detailed description of the decoder used in the IBM translation system is given in [Berger et al. 1996]. Throughout the search process, partial hypotheses are maintained in a set of priority queues. There is a single priority queue for each subset of positions in the source string. In practice, the priority queues are initialized only on demand, many less than the full number of queues possible are actually used. The priority queues are limited in size and only the 1 000 hypothesis with the best score are maintained. Each priority queue is assigned a threshold to select the hypotheses that are going to be extended. The process of assigning these thresholds is rather complicated and among other things involves the product of unigram probabilities of the source sentence words in a given subset. A

restriction to the possible word re-orderings is applied, namely when a new source position is covered, a loop may be made through the first n positions that are not already aligned in a partial hypothesis, where n is set to 4.
Three different decoders for the IBM-4 translation model are compared in [Germann et al. 2001]. The first one is a re-implementation of the stack-based decoder described in [Berger et al. 1996]. The second decoder is a greedy decoder that starts with an approximate solution and then iteratively improves this first rough solution. The third decoder converts the decoding problem into an integer program (IP) and a standard software package for solving IP is used. Although the last approach is guaranteed to find the optimal solution, it is tested only for input sentences of length 8 or shorter.
This thesis will present a DP-based beam search decoder for the IBM-4 translation model. The decoder is designed to carry out an almost full search with a small number of search errors and with little performance degradation as measured by the word error criterion used.

Chapter 2

Objectives

This thesis is about a search procedure for statistical machine translation. It addresses the problem of finding the most probable target language representation of a given source language string. Since the number of possible translations is enormous, we must find the best output without actually generating the set of all possible translations. The approach presented here is based on dynamic programming (DP) which sequentially generates target string words while progressively covering source sentence positions. The translation model used is the so-called IBM translation model as presented in [Brown et al. 1993]. The main contributions of the work are as follows:

- Starting from a dynamic programming solution to the traveling salesman problem, we present a search algorithm for statistical machine translation that is based on dynamic programming (DP). The dynamic programming approach works by jointly processing partial hypotheses that cover the same set of source sentence positions. By imposing constraints on the set of covered source sentence positions the DP approach becomes still more effective. For a certain type of re-ordering constraint (as used in the IBM translation system [Berger et al. 1996]) an upper bound for the complexity of $E^3 \cdot J^4$ is demonstrated (assuming the use of a trigram language model). Here, E is the size of target language vocabulary and J is the length of the source string.

- Several re-ordering constraints useful for different translation directions are shown, e.g. German-to-English, Italian-to-English. The word re-ordering constraints are expressed in terms of the number of uncovered position to the left of the rightmost covered position or in terms of the number of covered positions to the right of the leftmost uncovered position. Furthermore, there are distance constraints for the uncovered or covered positions relatively to the above reference points. A finite-state control may be added to the re-ordering constraints which is especially useful for the translation direction German-to-English where the word-order difference between the two languages is mainly due to the German verb group. The re-ordering constraints allow for word re-ordering to be performed that can not be carried out using the original IBM-style constraint [Berger et al. 1996].

- A simple data-driven search organization for the DP search algorithm is developed. A single list of cardinality-synchronously generated search hypotheses is processed. A

beam search pruning technique is conceived that jointly processes partial hypotheses according to **two** criteria: 1) the partial hypotheses cover the same set of source sentence positions and 2) the partial hypotheses cover sets $\mathcal{C}$ of source sentence positions of equal cardinality. Both the time and and the memory requirements for the search algorithm are strictly linear in the number of search hypotheses. We report and analyse results for several different translation directions: e.g. German-to-English and English-to-German.

- The alignment model introduced in Chapter 1 only allows for single source sentence words to be mapped to single target sentence words. Therefore, lexical correspondences, where a single source sentence word has to be mapped to several target sentence words can not be handled properly. In this work, we propose an approach to join target sentence words on the basis of a likelihood criterion, where an extended lexicon model is defined. The approach applied is similar to the one used in language modeling to learn word trigger pairs. A word trigger pair is a long-distant word pair. An extended model is defined which uses both a baseline m-gram model and a single word-trigger pair. The word-trigger pairs are selected on the basis of the likelihood of the extended model.

The organization of the work is as follows: In **Chapter 3**, we show some extensions that are applied to the baseline alignment model. Namely, in Section 3.1, we introduce the monotone alignment model, which is a simple formalization of the fact that the alignment between many languages is mostly monotone. The concept may be extended to handle alignments where the number of "non-monotonicity" points is restricted. The work on selection criteria for word trigger pairs is presented in Section 3.2: an extended language model, including a single word trigger pair is defined and word trigger pairs are selected on the basis of a likelihood criterion. In Section 3.3, two methods for handling alignments where a single source word is mapped to several target words are presented, e.g. to handle cases like the German word compound "Geschäftsreise" that is mapped to the English compound noun "business trip". A simple solution for joining target words is obtained by looking at the Viterbi path after the final training iteration. A more complex algorithm for word joining presented in Section 3.3 is based on a similar criterion as used for the selection of word trigger pairs in Section 3.2. The approach can be used to learn multi-word lexical dependencies between source and target words.
Chapter 4 constitutes the main contribution of this work. A search algorithm for statistical machine translation based on dynamic programming is presented. Starting from a dynamic programming solution to the traveling salesman problem an effcient search algorithm can be developed. For a certain type of re-ordering constraint (as used in the IBM translation system) the dynamic programming approach becomes even more efficient in reducing the search space. Pruning techniques are introduced to handle the potentially huge search space. A short overview of the implementation of the dynamic programming algorithm is given. **Chapter 5** shows the experimental translation results for several translation tasks, translation directions and re-ordering constraints. The effect of beam search pruning is demonstrated on the German-to-English Verbmobil corpus. Section 5.1 shows examples for selected word trigger pairs in language modeling and perplexity improvements of a combined language model including the selected trigger pairs. **Chapter 6** summerizes

the work carried out in this thesis and gives an outline of future lines to be pursued. In **Chapter 7** , we give a summary of the main results achieved in this work.

Chapter 3

Translation Model Extensions

In this chapter, we show some extensions that are applied to the baseline alignment models described in Section 1.3. In Section 3.1, we introduce the concept of the monotone alignments. A simple DP-based search procedure that uses this concept will be given in Section 4.3. Several criteria for selecting word trigger pairs in language modeling are presented in Section 3.2. In Section 3.3, two different techniques for joining target words are presented. The second technique is based on a likelihood criterion similar to the one used for selecting word trigger pairs.

3.1 Monotone Alignment Model

When aligning the words in parallel texts (for Indo-European language pairs like Spanish-English, French-to-English, Italian-German,...), we typically observe a strong localization effect. Figure 1.3 given in the introduction illustrates this effect for the language pair *German-to-English*. In many cases, although not always, there is an even stronger restriction: over large portions of the source string, the alignment is monotone. To formalize the concept of monotone alignments, we first define the *monotonicity* requirement. To satisfy this requirement, we may introduce suitably restricted permutations of the source string. We introduce the monotonicity property of the alignment model which allows only transitions from a_{j-1} to a_j with the jump width δ:

$$\delta \equiv a_j - a_{j-1} \in \{0, 1, 2\}.$$

Using this restriction the possible alignments between source and target language are shown in Figure 3.1. The problem of finding the most probable alignment path under the monotonicity restriction is very similar to the problem of finding the most probable time alignment path in the acoustic training for speech recognition.
To find the optimal alignment, we use dynamic programming (DP) as introduced for the HMM alignment model in Section 1.3.1 . Now, we restrict the possible jump widths using the monotonicity restriction and obtain the following recursive equation:

$$Q(i,j) = p(f_j|e_i) \cdot \max_{\delta} \{p(\delta) \cdot Q(i-\delta, j-1)\} . \tag{3.1}$$

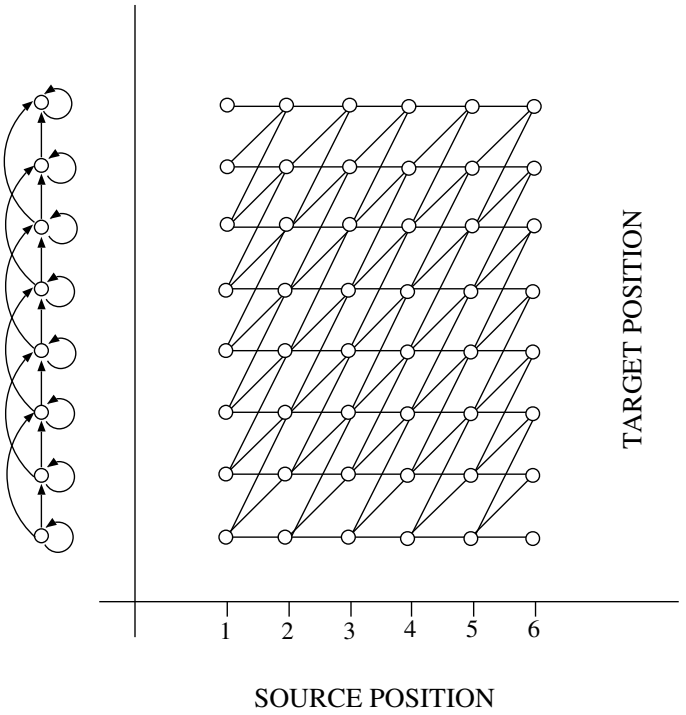


Figure 3.1: Illustration of monotone HMM alignments.

The recursion is carried out only over δ for which $i - \delta \geq 0$. We introduce a special initial state $(0, 0)$ from which all monotone alignments start. The initialization of the dynamic programming equation is as follows:

$$\begin{aligned} Q(0,0) &= 1.0 \\ Q(i,0) &= 0.0 \text{ for } i = 1, \cdots, I \\ Q(0,j) &= 0.0 \text{ for } j = 1, \cdots, J \end{aligned}$$

The best path in the maximum approximation is determined from the following maximization:

$$\max_i Q(i, J)$$

The training procedure assumes that the source word strings or the target word strings have been suitably re-ordered, so that the monotonicity restriction is fullfiled for each pair of training sentences.

Two-Level Monotone Alignment Model

One way to deal with the non-monotonicity of the alignment path encountered in real training data is to allow for a limited number of "big" jumps within the monotone alignment model. One simple way to do so is to allow for one "big" jump in the alignment which is otherwise monotone. This type of alignment is shown in Figure 3.2. Basically, we use

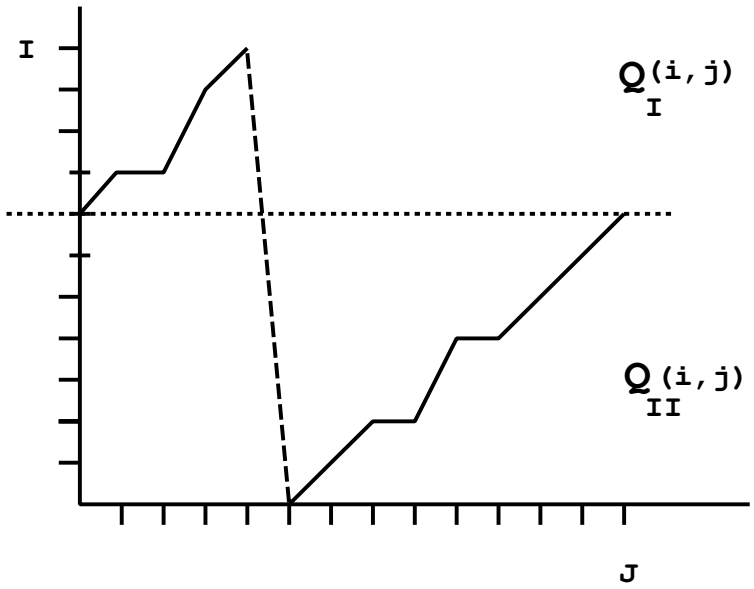


Figure 3.2: Illustration of two-level monotone HMM alignment.

the recursive Eq. 3.1, which we call **one-level** recursive equation. We introduce a **two-level** equation as follows: The first level corresponds to a monotone alignment where no jump has taken place, the second level corresponds to the case where one big jump has occurred. Within the two levels the corresponding recursive equations $Q_I(i,j), Q_{II}(i,j)$ are given in Eq. 3.1. Eq. 3.2 shows the recursive equation for jumping from level I to level II, where only one such jump is allowed. The jump probability may depend on the identity of the preceding source sentence word f_{j-1}, which proved to be useful in the translation experiments. Thus, we use jump probabilities of the type $p(\delta|f)$, where $\delta \in \{-1, 0, 1, 2\}$:

$$\begin{aligned} & Q_{II}(1,j) = \qquad (3.2) \\ & = \; p(f_j|e_1) \max\{p(\delta=-1|f_{j-1}) \cdot Q_I(I, j-1), p(\delta=0|f_{j-1}) \cdot Q_{II}(1, j-1)\} \end{aligned}$$

The initialization for the one-level case is changed accordingly. Here, we do not assume that the monotone alignment for level I starts in the state $(0,0)$:

$$\begin{aligned} Q_I(i,0) &= 1.0 \text{ for } i = 0, \cdots, I \\ Q_I(0,j) &= 0.0 \text{ for } j = 1, \cdots, J \\ Q_{II}(1,j) &= 0.0 \text{ for } j = 0, \cdots, J \end{aligned}$$

The best path is obtained from the following maximization:

$$\max_i \{Q_I(i,J), Q_{II}(i,J)\}$$

Successful experiments are carried out using the monotone two-level alignment training on the Spanish-English EuTrans-I corpus [Vidal et al. 1996]. The word order differences between source and target language in this corpus are very restricted such that allowing for a single big jump already improves translation performance. For a more complex translation task, one would need to allow for an increased number of big jumps.

3.2 Extended Language Model

As has been argued in Section 1.4 more complex multi-word based lexicon models are one way to improve translation accuracy. As a way to improve m-gram language models widely used in speech recognition, we have studied the use of so-called trigger pairs [Tillmann et al. 1996a, Tillmann et al. 1997b]. In Section 3.3, the selection techniques for trigger pairs are carried over to statistical machine translation, in order to obtain more complex lexicon models.

3.2.1 Trigger Pairs in Language Modeling

In this section, we present the use of word-trigger pairs in language modelling. In speech recognition, the most widely used and successful language model is the so-called N-gram model, e.g. a bigram or trigram model, where the dependency of the word under consideration is limited to the immediate predecessor words. We use a reference model $p(w|h)$, i.e. the conditional probability of observing the word w for a given history h. For a trigram model, this history h includes the two predecessor words of the word under consideration, but in general it can be the whole sequence of the last M predecessor words. The criterion for measuring the quality of a language model $p(w|h)$ is the so-called log-likelihood criterion [Ney et al. 1994], which for a corpus $w_1, ..., w_n, ...w_N$ is defined by:

$$F := \sum_{n=1}^{N} \log p(w_n|h_n),$$

According to this definition, the log-likelihood criterion measures for each position n how well the language model can predict the next word given the knowledge about the preceeding words and computes an average over all word positions n. In the context of language modeling, the log-likelihood criterion F is converted to perplexity PP, defined by $PP := -F/N$.

However, it is clear that some sort of long-distance dependencies exist as well. The main goal for the use of trigger pairs is to include long-distance dependencies into the language model by means of so-called "trigger pairs" [Lau et al. 1993, Lau et al. 1993, Rosenfeld 1994], where a trigger pair is defined as a long-distance word pair. Examples for trigger pairs are shown in Figure 3.3. We restrict ourselves to trigger pairs where both the triggered and the triggering events are single words (as opposed to word phrases). Thus word trigger pairs can be viewed as long-distance word bigrams. In this view, we are faced the problem of finding suitable word trigger pairs. This will be achieved by analysing a large text corpus (i.e. several millions of running words) and learning those trigger pairs that are able to improve the baseline language model. A related approach to capturing long-distance dependencies is based on stochastic variants of link grammars [Della Pietra et al. 1994]. For applications where the topic-dependence of the language model is important, e.g. text dictation, the history h may reach back several sentences so that the history length M covers several hundred words, say, $M = 400$ as it is for the cache model. Our main idea works as follows: We extend a baseline language model by a single word trigger pair and use the perplexity of this extended language model as selection criterion, where the baseline language model is typically a trigram model in combination with a cache component. This

$$
\begin{aligned}
airline &\ldots\ldots flights \\
concerto &\ldots\ldots orchestra \\
asks &\ldots\ldots replies \\
neither &\ldots\ldots nor \\
we &\ldots\ldots ourselves \quad .
\end{aligned}
$$

Figure 3.3: Example of word trigger pairs.

extension is applied to all possible trigger pairs, the number of which is the square of the vocabulary size. A similar idea is applied in [Berger et al. 1996] to the so-called feature selection in the maximum entropy framework. We present several methods to select the most significant single word trigger pairs. In a first variant, we directly combine trigger pairs with a given baseline model using a backing-off scheme [Katz 1987]. When using a unigram language model as baseline model, this approach produces the mutual information criterion used in [Lau et al. 1993, Lau et al. 1993, Rosenfeld 1994]. Unlike the approach presented in [Bahl et al. 1984, Lau et al. 1993], where the trigger pairs are selected on the basis of a mutual information criterion, the selection criterion presented in this paper is directly the perplexity improvement obtained by extending the baseline language model by a single trigger pair. A second variant is based on the idea that word trigger pairs should be used only in those cases where the probabilities of the baseline model are not already high. In the context of a bigram/cache language model, this idea amounts to exploiting word trigger dependencies only when neither the cache component nor the bigram component of the language model are "active". The selected trigger pairs are used in a combined model where the interpolation parameters and trigger interaction parameters are trained by the EM algorithm.
What makes the selection criteria for word pair triggers interesting in general, is the following broader view: Given a baseline language model how can we improve this model by including additional types of dependencies? In the next section we will use similar criteria to extend the translation probabilities for the translation models.
In Section 5.1, we present results on the Wall Street Journal corpus. The trigger pairs selected reduce the perplexity of a trigram/cache language model from 138 to 128 for a 5-million word training set and from 92 to 87 for a 38-million word training set. We give examples of selected trigger pairs with and without using the EM algorithm. As might be expected, the experimental results depend heavily on the criterion used to select the word triggers; in particular, the third variant of the trigger selection criteria developed in the next section gives better results than the conventional mutual information criterion used in [Lau et al. 1993, Rosenfeld 1994]. In several papers [Bahl et al. 1984, Lau et al. 1993, Tillmann et al. 1996a], selection criteria for *single* word trigger pairs were studied. In this paper, this work is extended as follows:

- **Single-Trigger Model:** We consider the definition of a single word trigger pair. There are **three** models we consider, namely high level, unigram level and low level triggers.

- **Multi-Trigger Model:** To really use the word triggers for a language model, they must be combined with an existing language model. This is achieved by using linear interpolation between the existing language model and a model for the multi-trigger effects.

3.2.2 Selection of High-Level Triggers

The baseline language model we use in the following is denoted by $p(w|h)$, where w is the word whose probability is to be predicted for the given history h. Here, the word history consists of the last M, say 200, words. To denote a specific word trigger pair, we use the symbol $a \to b$, where a is the *triggering* word and b is the *triggered* word. In order to combine a trigger pair $a \to b$ with the baseline language model $p(w|h)$, we define the extended model $p_{ab}(w|h)$:

$$p_{ab}(w|h) = \begin{cases} q(b|a) & \text{if } a \in h \text{ and } w = b \\ [1 - q(b|a)] \cdot \dfrac{p(w|h)}{\sum\limits_{w' \neq b} p(w'|h)} & \text{if } a \in h \text{ and } w \neq b \\ & \\ q(b|\overline{a}) & \text{if } a \notin h \text{ and } w = b \\ [1 - q(b|\overline{a})] \cdot \dfrac{p(w|h)}{\sum\limits_{w' \neq b} p(w'|h)} & \text{if } a \notin h \text{ and } w \neq b \ , \end{cases} \tag{3.3}$$

where $q(b|a)$ and $q(b|\overline{a})$ are two interaction parameters of the word trigger pair $a \to b$. For symmetry reasons, we introduce a special interaction parameter $q(b|\overline{a})$, when a has not been seen in the history. The unknown parameters $q(b|a)$ and $q(b|\overline{a})$ will be estimated later by maximum likelihood, which is equivalent to the criterion of minimum perplexity. In the above definition, the interaction parameters of the word trigger pair are applied at the highest level, i.e.they have priority over the baseline language model. To distinguish this selection criterion from the criteria to be studied later, we refer to these trigger pairs as *high level triggers*. We consider the difference between the log-perplexity F_{ab} of the extended model $p_{ab}(w|h)$ and the log-perplexity F_0 of the baseline model $p(w|h)$ on a corpus $w_1, ..., w_n, ..., w_N$:

$$\begin{aligned} F_{ab} - F_0 &= \sum_{n=1}^{N} \log \frac{p_{ab}(w_n|h_n)}{p(w_n|h_n)} \\ &= \sum_{h} \Big[N(a;h,b) \log \frac{q(b|a)}{p(b|h)} + N(a;h,\overline{b}) \log \frac{1-q(b|a)}{1-p(b|h)} \\ &\quad + N(\overline{a};h,b) \log \frac{q(b|\overline{a})}{p(b|h)} + N(\overline{a};h,\overline{b}) \log \frac{1-q(b|\overline{a})}{1-p(b|h)} \Big], \end{aligned}$$

where the counts $N(.;.,.)$ are defined in a natural way. E.g. the count $N(a;h,b)$ is the number of occurrences that the word b is observed for the history h and the word a appears at least once in the history. To further rewrite the perplexity improvement $F_{ab} - F_0$, we

introduce another set of counts that is independent of the history h:

$$\begin{aligned} N(a;b) &= \sum_{n:a\in h_n, b=w_n} 1 \\ N(a;\overline{b}) &= \sum_{n:a\in h_n, b\neq w_n} 1 \\ N(\overline{a};b) &= \sum_{n:a\notin h_n, b=w_n} 1 \\ N(\overline{a};\overline{b}) &= \sum_{n:a\notin h_n, b\neq w_n} 1 \quad . \end{aligned} \tag{3.4}$$

Using these count definitions, we obtain after some elementary manipulations:

$$\begin{aligned} F_{ab} - F_0 = \quad & N(a;b)\log q(b|a) + N(a;\overline{b})\log[1-q(b|a)] \\ & + N(\overline{a};b)\log q(b|\overline{a}) + N(\overline{a};\overline{b})\log[1-q(b|\overline{a})] \\ & - S(b) \end{aligned} \tag{3.5}$$

where $S(b)$ depends only on the baseline language model $p(w|h)$ and is defined as:

$$S(b) = \sum_{n=1}^{N} \delta(w_n, b)\log\frac{p(b|h_n)}{1-p(b|h_n)} + \sum_{n=1}^{N}\log[1-p(b|h_n)] \quad , \tag{3.6}$$

where $\delta(x,y)$ is the usual Kronecker delta. Here, for the sake of clarity, we point out the notation: the symbol $p(b|h_n)$ denotes the probability of word b following the history h_n although it has typically never occurred at position n in the training corpus. It is interesting to note that $S(b)$ is independent of the triggering word a. From the representation given by Eq. 3.5, we can derive the conclusions:

- The maximum-likelihood estimates of the interaction parameters $q(b|a)$ and $q(b|\overline{a})$ are obtained by taking the derivates of Eq. 3.5 and equating them to zero:

$$q(b|a) = \frac{N(a,b)}{N(a,b)+N(a,\overline{b})} \tag{3.7}$$

$$q(b|\overline{a}) = \frac{N(\overline{a},b)}{N(\overline{a},b)+N(\overline{a},\overline{b})}, \tag{3.8}$$

- If we fix a triggered word b and consider its triggering words a, the ranking of the a's does not depend in any way on the baseline language model.

The practical problem of computing the perplexity improvement as given by Eq. 3.5 is the second sum in Eq. 3.6. To manage this computational problem, we use a sampling approach, i.e. we take every 20-th position in the training corpus to estimate this sum. We tested this sampling approximation for 200 words (having different unigram probabilities) and found the approximation error to be typically smaller than 5%. To calculate the perplexity improvement for all trigger pairs $a \rightarrow b$, we compute beforehand both the counts in Eq. 3.4 and the quantity $S(b)$ for each triggered word b (by sampling). Then we compute the perplexity improvements on the training corpus for each possible trigger pair $a \rightarrow b$.

3.2.3 Selection of Unigram Level Triggers

Choosing a unigram model $p(w)$ as baseline model $p(w|h)$, we obtain:

$$\begin{aligned} F_{ab} - F_0 &= N(a;b)\log\frac{q(b|a)}{p(b)} + N(a;\overline{b})\log\frac{1-q(b|a)}{1-p(b)} \\ &+ N(\overline{a};b)\log\frac{q(b|\overline{a})}{p(b)} + N(\overline{a};\overline{b})\log\frac{1-q(b|\overline{a})}{1-p(b)} \end{aligned} \tag{3.9}$$

Identifying the probability $p(a;b)$ with the relative frequency $N(a;b)/N$ and similarly for the other joint events $(a;\overline{b})$, $(\overline{a};b)$, $(\overline{a};\overline{b})$, we obtain exactly the mutual information criterion as suggested in [Lau et al. 1993, Rosenfeld 1994]. In other words, this criterion is simply the improvement on the log-perplexity of a unigram model using the above backing-off model for the trigger pair $a \to b$. The trigger pairs selected by this criterion are called *unigram level triggers.*

3.2.4 Selection of Low-Level Triggers

As indicated by the results of several groups [Lau et al. 1993, Rosenfeld 1994], [Tillmann et al. 1996a], the word trigger pairs do not help much to predict the next word if there is already a good model based on specific contexts like trigram, bigram or cache. Therefore, we allow the trigger interaction $a \to b$ only if the probability $p(b|h)$ of the reference model is not sufficiently high, i.e. if $p(b|h) < p_0$ for a certain threshold p_0 (note that, by setting $p_0 := 1.0$, the trigger effect is used in *all* cases). Thus, we use the trigger effect only for the following subset of histories:

$$H_{ab} := \{h : a \in h \wedge p(b|h) < p_0\}$$

In the experiments, we used $p_0 := 1.5/W$, where $W = 20\,000$ is the vocabulary size. We define the model $p_{ab}(w|h)$ as an extension of the reference model $p(w|h)$ by a backing-off technique [Katz 1987]:

$$p_{ab}(w|h) = \begin{cases} q(b|a) & \text{if } h \in H_{ab},\ w = b \\ [1-q(b|a)] \cdot \dfrac{p(w|h)}{\sum\limits_{w' \neq b} p(w'|h)} & \text{if } h \in H_{ab},\ w \neq b \\ p(w|h) & \text{if } h \notin H_{ab} \end{cases} \tag{3.10}$$

For a training corpus $w_1 ... w_N$, we consider the log–likelihood functions of both the extended model and the reference model $p(w_n|h_n)$, where we define the history h_n:

$$h_n := w_{n-M}^{n-1} = w_{n-M} ... w_{n-2} w_{n-1} \quad .$$

For the difference $F_{ab} - F_0$ in the log–likelihoods of the extended language model $p_{ab}(w|h)$ and the reference model $p(w|h)$, we obtain:

$$\begin{aligned}
F_{ab} - F_0 &= \sum_{n=1}^{N} \log \frac{p_{ab}(w_n|h_n)}{p(w_n|h_n)} = \sum_{n:\, h_n \in H_{ab}} \log \frac{p_{ab}(w_n|h_n)}{p(w_n|h_n)} \\
&= \sum_{h:\, h \in H_{ab}} \sum_{w} N(h,w)\ \log \frac{p_{ab}(w|h)}{p(w|h)} \\
&= \sum_{h:\, h \in H_{ab}} \left[N(h,b)\ \log \frac{q(b|a)}{p(b|h)} + \ N(h,\bar{b})\ \log \frac{1-q(b|a)}{1-p(b|h)} \right] \\
&= \tilde{N}(a;b)\ \log q(b|a)\ +\ \tilde{N}(a;\bar{b}) \log[1-q(b|a)] - \sum_{h:\, h \in H_{ab}} [N(h,b)\ \log p(b|h) \\
&\quad +\ N(h,\bar{b})\ \log[1-p(b|h)] \Big]
\end{aligned}$$

where we have used the usual counts $N(h,w)$:

$$N(h,w) \ := \sum_{n: h=h_n, w=w_n} 1$$

and two additional counts $\tilde{N}(a;b)$ and $\tilde{N}(a;\bar{b})$ defined particularly for word trigger modeling:

$$\begin{aligned}
\tilde{N}(a;b) \ &:= \sum_{h: h \in H_{ab}} N(h,b) \ = \sum_{n: h_n \in H_{ab}, w_n = b} 1 \\
\tilde{N}(a;\bar{b}) \ &:= \sum_{h: h \in H_{ab}} N(h,\bar{b}) \ = \sum_{n: h_n \in H_{ab}, w_n \neq b} 1
\end{aligned}$$

Note that, for the counts $\tilde{N}(a;b)$ and $\tilde{N}(a;\bar{b})$, it does *not* matter how often the triggering word a actually occurred in the history $h \in H_{ab}$. The unknown trigger parameter $q(b|a)$ is estimated using maximum likelihood estimation. By taking the derivative and setting it to zero, we obtain the estimate:

$$q(b|a) \ = \ \frac{\tilde{N}(a;b)}{\tilde{N}(a;b) + \tilde{N}(a;\bar{b})}$$

which can be interpreted as the relative frequency of the occurrence of the word trigger $(a \to b)$.

3.2.5 Multi-Trigger Model

The trigger pairs are used in combination with a conventional baseline model $p(w_n|h_n)$ (e.g. m-gram) to define a trigger model $p_T(w_n|h_n)$:

$$p_T(w_n|h_n) \ = \ (1-\lambda) \cdot p(w_n|h_n) + \frac{\lambda}{M_n} \sum_{m} \alpha(w_n|w_{n-m})$$

with the trigger parameters $\alpha(w|v)$ that must be normalized for each v:

$$\sum_w \alpha(w|v) = 1 \quad .$$

To simplify the notation, we have used the convention:

$$\sum_m \alpha(w_n|w_{n-m}) \quad = \quad \sum_{m \in \mathcal{M}_n} \alpha(w_n|w_{n-m})$$

with

- $\mathcal{M}_n$: the set of triggering words for position n
- $M_n = |\mathcal{M}_n|$: the number of triggering words for position n .

A remark about the functional form of the multi-trigger model is in order. The form chosen in this paper is a sort of linear combination of the trigger pairs. A different approach is to combine the various trigger pairs in multiplicative way, which results from a Maximum-Entropy approach [Lau et al. 1993].

3.3 Extended Lexicon Model

In this section, we present two techniques for joining target language words during the translation model training. The second technique presented in Section 3.3.2 is similar to the use of the trigger selection criteria presented in the previous section. Additionally, as explained in Section 3.3.3 the set of considered extensions is further restricted to the *maximally defining* ones.

3.3.1 Word Joining: Viterbi Path Criterion

The baseline alignment model does not permit a source word to be aligned with two or more target words. Therefore, e.g. for the translation direction German to English lexical correspondences like the compound noun *"Zahnarzttermin"* for "dentist's appointment" cause problems because a single source word must be mapped on two or more target words. A simple solution to deal with this problem can be applied. The method joins target sentence words during an extended training procedure and the joined target words are used as new entities for the language and the lexicon model. First, we reverse the translation direction, i. e. for a first training step the role of source and target language are reversed. For this reversed translation direction, we perform the usual training and then check the alignment paths for the most probable alignment a_1^J - the so-called Viterbi alignment. Whenever a source language word of the original translation direction is aligned with a sequence of adjacent target language words, this sequence is added to the target language vocabulary as a new lexical entry, no matter how often the newly generated sequence is seen in the training data. As a result, we have an extended target language vocabulary and an modified target language corpus, where the extensions are used as new target language words . Using this new corpus, we then perform the standard training for the original translation direction. This approach is illustrated in Figure 3.4 for the translation direction German-to-English. For the original translation direction German-to-English, there is no proper word alignment between the German word "Zahnarzttermin" and the corresponding English word sequence "dentist's appointment". To deal with this problem, we first invert the translation direction: English is now the source language and German is the target language. We carry out a usual training and than check the alignment path: whenever a single German word is aligned to several adjacent English words the English words are joined to obtain a new lexical entry (Stage II in Figure 3.4). After the word joining, the translation direction is reversed again and we are able to obtain an adequate alignment for the German-to-English sentence pair, where the new lexical entry "in#the#morning" is aligned to "morgens" and the German word "Zahnarzttermin" is aligned to the new lexical entry "dentist's#appointment" (Stage III in Figure 3.4). A more systematic approach to the word joining problem is presented in following section.

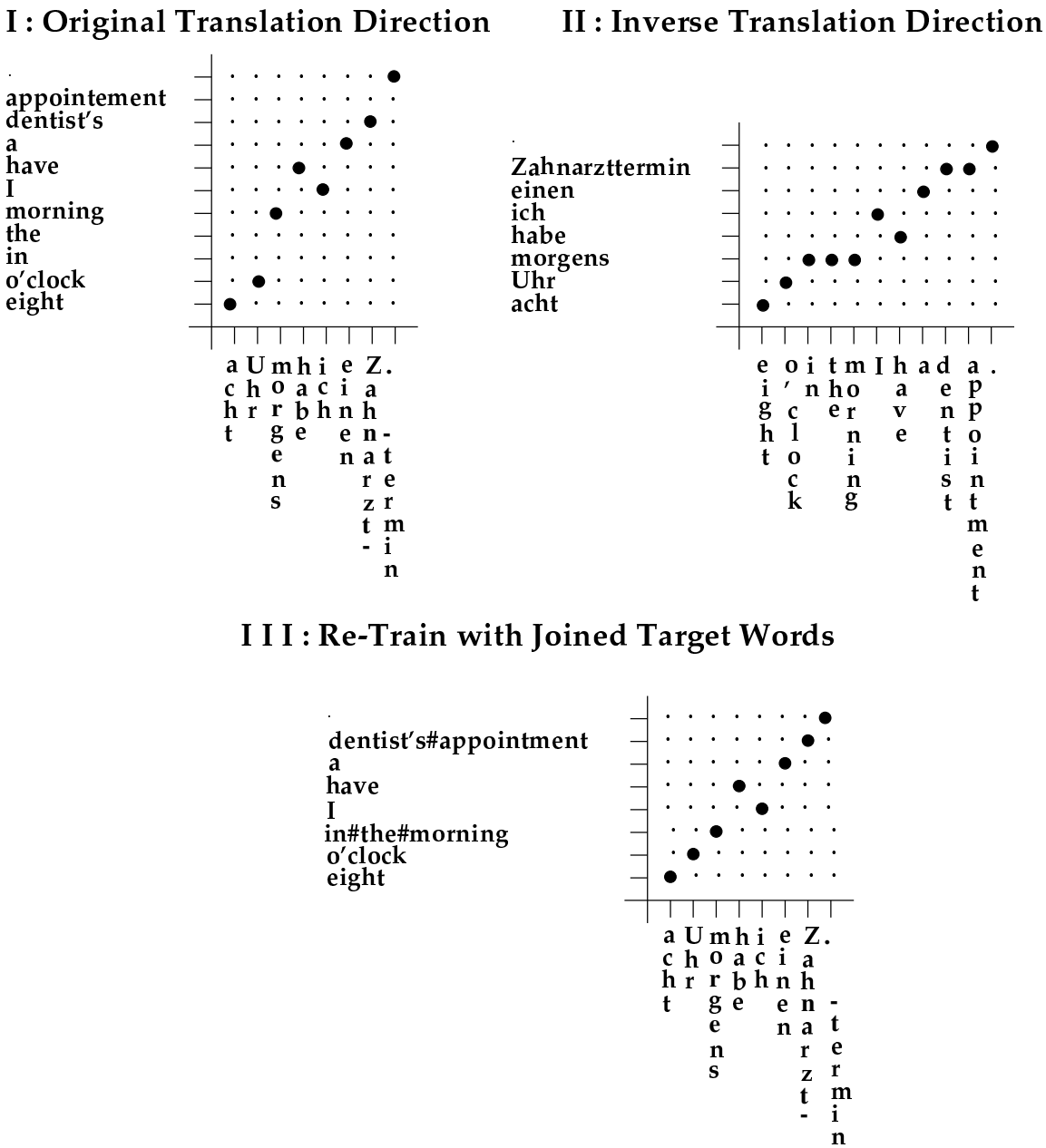


Figure 3.4: Example of the word joining carried out for the translation direction German-to-English. The training procedure is carried out subsequently for two different translation directions (Stage II and Stage III).

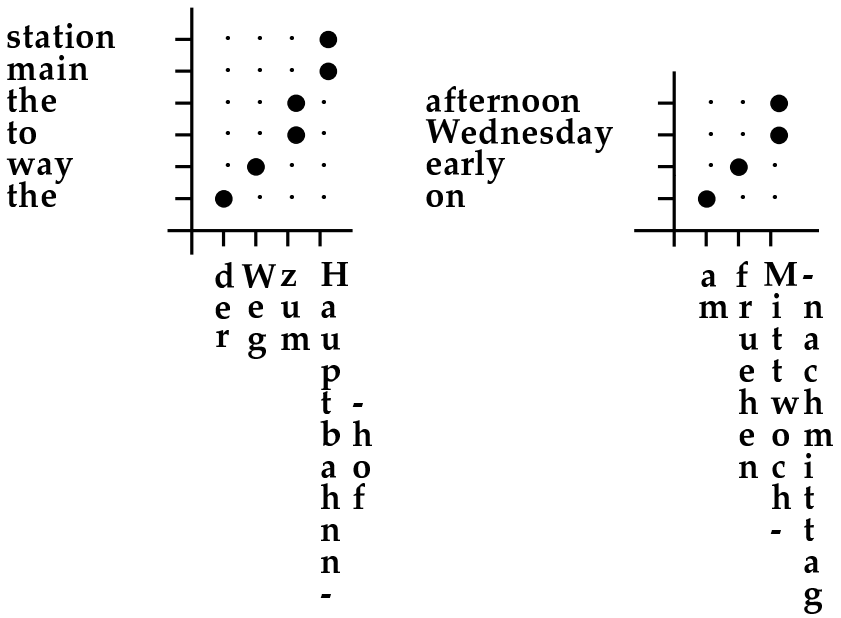


Figure 3.5: German-to-English alignment examples, where a single source word is aligned to several target language words. The source word "zum" is mapped to the target words "to the". The German compound nouns "Hauptbahnhof" und "Mittwochnachmittag" are mapped to the English words "main station" and "Wednesday afternoon".

3.3.2 Word Joining: Likelihood Criterion

So far, only translation dependencies between single words are used for the lexicon model and the lexicon parameters are of the form $p(f|e)$, where f is a single source language word and e is a single target language word. However, we might decompose the joint probability $Pr(f_1^J, a_1^J|e_1^I)$, which occurs in Eq. 1.2 as follows:

$$Pr(f_1^J, a_1^J|e_1^I) = \tag{3.11}$$
$$Pr(J|e_1^I) \cdot \prod_{j=1}^{J} Pr(a_j|a_1^{j-1}, J, e_1^I) \cdot Pr(f_j|f_1^{j-1}, a_1^J, J, e_1^I),$$

where the translation probability for the source word f_j may depend on all predecessor source words f_1^{j-1} as well as on all target words in the target sentence e_1^I. In the previous section, a procedure was presented to join target sentence words that are aligned to the same source word. For the word joining an alignment training in the inverted translation direction is carried out and the joined words are used as new lexicon entries. Figure 3.5 shows possible word joinings, where the German word "zum" is translated by two target words "to the" and the German compound nouns "Hauptbahnhof" and "Mittwochnachmittag" are translated by the English noun phrases "main station" and "Wednesday afternoon", respectively. In this section, we will describe a procedure to carry out the word joining on the basis of a likelihood criterion.

For the single-word based translation approach, the lexicon dependencies are expressed by lexicon probabilities of the form $p(f|e)$. We are looking for dependencies of the form

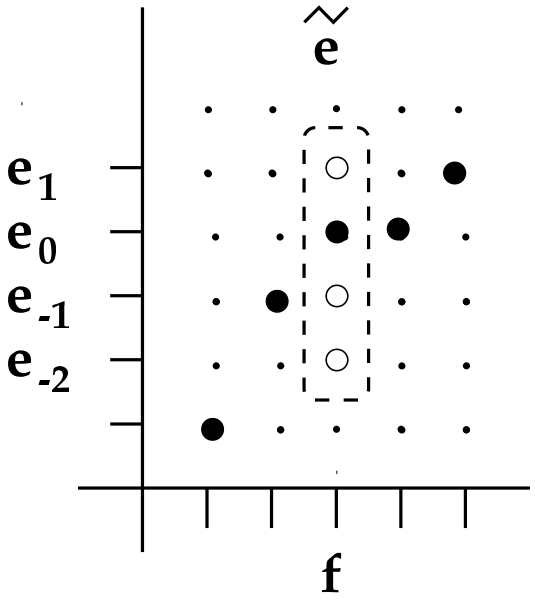


Figure 3.6: Illustration of the possible target word extensions $(f, \tilde{e})$ for the single-word dependency (f, e_0). The extension shown does not involve additional source language words.

$p(f|\tilde{e}, \tilde{f})$ [1]. Here, $\tilde{f}$ and $\tilde{e}$ are subsets of words in the source and target sentence. As can be seen from the last term in Eq. 3.11), for the new lexicon dependencies $p(f|\tilde{f}, \tilde{e}$, $\tilde{f}$ might include all words preceding f_j in the source sentence and $\tilde{e}$ might include all words of the target sentence e_1^I. For the extended lexicon model, we will assign a single new parameter $q(f_0|\tilde{f}, \tilde{e})$ to each of the extensions considered. Here, f_0 is a single source word that forms a fixed lexical unit together with the word groups $\tilde{f}$ and $\tilde{e}$ (The notation suggests that the source word f_0 in an extension is always aligned to the target word $\tilde{e}_0$ defined below.). Starting point for the training of the extended model is the Viterbi alignment path after the final training iteration, e.g. the final iteration of the IBM-4 model training. This alignment path is kept fixed for the training of the lexicon extensions, although in future we might want to allow changing the alignment path within the subsequent lexicon extension training as well.

Each target word extension $\tilde{e}$ must contain a target language word e on the alignment path, where this word is denoted by $\tilde{e}_0$. In Figure 3.6, $\tilde{e}_0$ is marked by a filled dot. The index 0 indicates that all target word extensions will be learned relative to this main translation. Whenever we predict a source language word f on the basis of an extension $\tilde{e}$, we assume that f is aligned to $\tilde{e}_0$. In this section, we consider only extensions that use additional target words, i.d. extensions of the type $p(f|\tilde{e})$. We define an extended model $p_{(f_0,\tilde{e})}(f|e_1^I)$ for a given extension pair $(f_0, \tilde{e})$. We will compute the likelihood improvement of the extended

[1] We use conditional probabilities to denote the new dependencies, since ultimately we will find parameters for these conditional probabilities.

model over the baseline model $p(f|e)$. The extended model is defined as follows:

$$p_{(f_0,\tilde{e})}(f|e_1^I) = \begin{cases} q(f_0|\tilde{e}) & \text{if } f_0 = f \text{ and } match(f,\tilde{e},e_1^I) \\ [1-q(f_0|\tilde{e})]\dfrac{p(f|\tilde{e}_0)}{1-p(f|\tilde{e}_0)} & \text{if } f_0 \neq f \text{ and } match(f,\tilde{e},e_1^I) \\ p(f|\tilde{e}_0) & else \end{cases}$$

where $\tilde{e} \subset e_1^I$ and $\tilde{e}_0$ is the word to which f is aligned. The match functions are defined as follows:

$$match(f,\tilde{e},e_1^I) = \begin{cases} 1 & \text{if } match(\tilde{e},e_1^I) \text{ and } f \text{ is aligned to } \tilde{e}_0 \\ 0 & else \end{cases}$$

$$match(\tilde{e},e_1^I) = \begin{cases} 1 & \text{if } \tilde{e} \text{ is a subset of } e_1^I \\ 0 & else \end{cases}$$

Note that the extended model does not use the alignment probabilities as obtained after the final traning iteration. Only lexical dependencies as expressed by the alignment path are taken into account.
The baseline model that we want to improve on, is the single-word based model trained from the aligned corpus without taking into account the extensions. For each extension $(f_0,\tilde{e})$, the log-likelihood of the extended model $F_{(f_0,\tilde{e})}$ is given by:

$$F_{(f_0,\tilde{e})} = \sum_{(s,j)\in\mathcal{F}_{\mathcal{A}}(\tilde{e}_0)} \log p_{(f_0,\tilde{e})}(f_j|e_1^{I_s}),$$

where $\mathcal{F}_{\mathcal{A}}(e)$ is the set of all 2-tuples (s,j), where j is a source sentence position in the s-th training sentence pair and f_j is aligned to the word e.

$$\mathcal{F}_{\mathcal{A}}(e) = \{(s,j)|e^{(s)}_{a_j^{(s)}} = e\} \tag{3.12}$$

The extended model is compared to the baseline model, which uses only the original single-word based lexicon probabilities $p(f|e)$:

$$F_e = \sum_{(s,j)\in\mathcal{F}_{\mathcal{A}}(e)} \log p(f_j|e). \tag{3.13}$$

The extensions are selected on the basis of the difference $F_{(f_0,\tilde{e})} - F_{\tilde{e}_0}$. When joining target sentence words, we consider extensions, where for a given target word e only its immediate neighbor words are considered. Formally, we look for extensions $(f_0,\tilde{e})$, where $\tilde{e}$ is a sequence of consecutive target words. The single-word translation probabilities $p(f|e)$ in Eq. 3.13 are estimated along the Viterbi alignment path using relative frequencies:

$$p(f|e) = \frac{N(f,e)}{N(\cdot,e)}$$

In the following, for reasons of brevity, we drop the subscript 0 for f_0 when denoting an extension $(f_0,\tilde{e})$. An estimate for the interaction parameter $q(f|\tilde{e})$ can be obtained as follows: The log-likelihood $F_{(f,\tilde{e})}$ of the extended model can be re-written:

$$\begin{aligned} F_{(f,\tilde{e})} &= \sum_{(s,j)\in\mathcal{F}_{\mathcal{A}}(\tilde{e}_0)} \log p_{(f,\tilde{e})}(f_j^{(s)}|e_1^{I_s}) \\ &= N(f,\tilde{e})\cdot\log q(f|\tilde{e}) + \bar{N}(f,\tilde{e})\cdot[1-\log q(f|\tilde{e})] + const(q(f|\tilde{e})), \end{aligned}$$

where we used the following "counts":

$$N(f,\tilde{e}) = \sum_{\substack{(s,j)\in\mathcal{F}_{\mathcal{A}}(\tilde{e}_0) \\ f_j^{(s)}=f \text{ and } match(f,\tilde{e},e_1^I)}} 1$$

$$\bar{N}(f,\tilde{e}) = \sum_{\substack{(s,j)\in\mathcal{F}_{\mathcal{A}}(\tilde{e}_0) \\ f_j^{(s)}\neq f \text{ and } match(f,\tilde{e},e_1^I)}} 1$$

The counts above are obtained from the training corpus consecutively for each target word e. The parameter $q(f|\tilde{e})$ is estimated using maximum likelihood estimation. By taking the derivative of $F_{(f,\tilde{e})}$ according to $q(f|\tilde{e})$ and setting it to zero, we obtain the estimate:

$$q(f|\tilde{e}) = \frac{N(f,\tilde{e})}{N(f,\tilde{e}) + \bar{N}(f,\tilde{e})} \tag{3.14}$$

There is an exponential number of tuples $(f,\tilde{e})$ that may be considered as extension of the single-word based dependency (f,e), since we might consider any subset of target words from e_1^I as an extension candidate. We will present several ways to restrict the number of extensions considered. For the target word joining, we will restrict ourselves to extensions that are obtained from the alignment path after the final training iteration for the *inverse* translation direction. Given a source word f and the target word e to which it is aligned only those extensions $\tilde{e}$ are considered such that all words in $\tilde{e}$ are covered by the inverted alignment path. This approach is illustrated in Figure 3.7. The alignment points on the Viterbi alignment path for both the original and the inverted translation directions are shown in the figure using filled dots. The alignment points resulting from the inverted translation direction only are shown as unfilled dots. When looking for an extension for the lexicon pair "(morgens,morning)" we restrict ourselves to target words that are indicated by the inverted alignment path, in this case the target word sequence "in the". In this example the pairs ("morgens","in the morning") and ("morgens","the morning") are considered as possible extensions.

Table 3.1 shows an algorithm which considers lexicon extensions $(f,\tilde{e}$ for all single-word based dependencies (f,e) where the source word f is aligned to the target word e in the aligned training data. The algorithm uses two alignment mappings obtained from the Viterbi path of a full IBM model training for both translation directions: "source $\rightarrow$ target" and "target $\rightarrow$ source". These alignment paths are kept fixed during the extension training. The extensions are generated in a loop over all target words e. For each pair (f,e) in the aligned training corpus all possible extensions $(f,\tilde{e})$ are considered. The number of extensions $|\mathcal{E}|$ selected is restricted using the threshold τ for the likelihood-improvement $F_{(f,\tilde{e})} - F_{\tilde{e}_0}$. In the next section we will describe how to further restrict the set of candidate selections $\mathcal{E}$ to get the final target word extension which are used for the word joining.

As for the complexity of the algorithm, one has to bear in mind that the maximum number of extensions $(f,\tilde{e})$ considered for a single-word dependency (f,e) is restricted by the length of the target sentence. Since we restrict ourselves to target word sequences of consecutive words, for a target sentence of length I the maximum number of possible extensions is restricted to I^2. In practice, much less than the possible number of extensions are considered, since an extension is restricted by the "target $\rightarrow$ source" alignment matrix. On

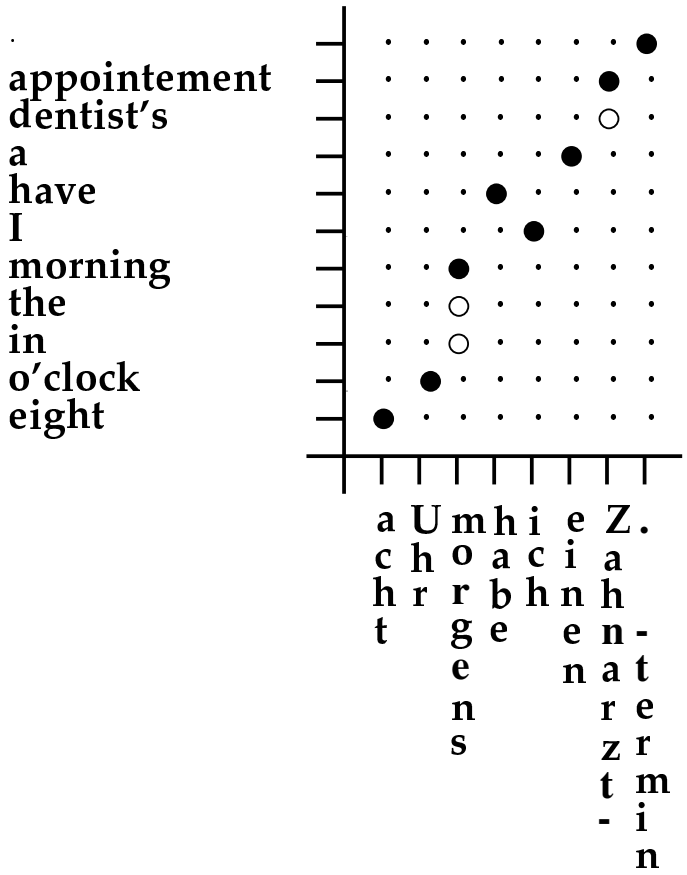


Figure 3.7: Example for the selection restriction for target language extensions $\tilde{e}$ aligned to a single source word f. The alignment path of the regular translation direction is shown as filled dots. The alignment path of the inverse translation direction is shown as unfilled dots. The following extensions are considered: ("morgens","in the morning") and ("Zahnarzttermin", "dentist's appointment").

the Verbmobil task, the word joining procedure (where the algorithms in Table 3.1 and Table 3.2 are executed sequentially) is carried out in a couple of minutes.

Table 3.1: Algorithm for computing a set $\mathcal{E}$ of candidate extensions $(f, \tilde{e})$. A likelihood threshold τ is used to restrict the extensions selected.

input: **two** corpora alignments: "source $\rightarrow$ target" and "target $\rightarrow$ source"
set $\mathcal{E} = \emptyset$
for each target word $e \in E$ do
compute single-word baseline model $p(f\|e)$
for all single-word based pairs (f, e) on "source $\rightarrow$ target"
for all extensions $(f, \tilde{e})$ with $\tilde{e}_0 = e$ and $\tilde{e}$ on "target $\rightarrow$ source"
Get counts $N(f, \tilde{e}), \bar{N}(f, \tilde{e})$
if $F_{(f,\tilde{e})} - F_{\tilde{e}_0} < \tau$ do
add $(f, \tilde{e})$ to $\mathcal{E}$
output: set of candidate extensions $\mathcal{E}$

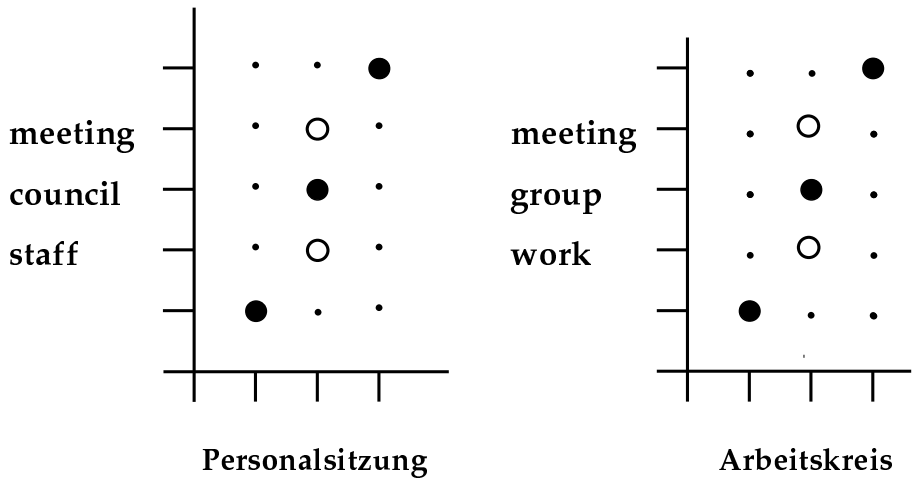


Figure 3.8: Target language word extension selection for the two German compound nouns "Arbeitskreis" ("work group meeting") and "Personalsitzung" ("staff council meeting"). Here, the single German word in the center should be properly aligned to three English words (a part of the surrounding alignment is shown as well - no aligned words are shown for this part).

3.3.3 Selection of Target Word Joinings

If we look at the alignment fragment in Figure 3.8 taken from the aligned German-to-English Verbmobil corpus, we see that the German compound noun "Personalsitzung" is aligned to the three consecutive English target words "staff council meeting". Following the lines described in the previous section, we might consider the following *three* extensions $(f, \tilde{e})$ for the German compound noun "Personalsitzung" which are indicated by the inverted alignment path:

- ("Personalsitzung","staff council")
- ("Personalsitzung","council meeting")
- ("Personalsitzung","staff council meeting")

In all of the three cases "Personalsitzung" is aligned to the English noun "council". The secound example for the German noun "Arbeitskreis" is completely analogous.
If we look at the likelihood difference $F_{(f,\tilde{e})} - F_{\tilde{e}_0}$, we might get a likelihood improvement for all three extensions. We might select the extension to be used for the word joining based on the likelihood improvement: namely, to select the extension which yields the highest likelihood improvement. Actually, we choose another strategy: whenever there are several extensions to select between, we select the extension that is *maximally defining*, i.e. the extension which is based on the largest number of target words, provided it does yield a likelihood improvement. In the case of the above example, we would select the extension ("Personalsitzung","staff council meeting") rather than the extensions ("Personalsitzung","staff council") or ("Personalsitzung","council meeting") . The use of this approach to generate a modified target language corpus based on the maximum extension

Table 3.2: Algorithm for computing a modified target language corpus and a corresponding list of selected extensions.

input: Aligned training corpus and set of extensions $\mathcal{E}$ produced by the algorithm in Table 3.1
set $\mathcal{E}_{max} = \emptyset$
for each aligned training sentence pair $\mathcal{S}$ in the training corpus do
Get extension subset $\mathcal{E}' \subset \mathcal{E}$ contained in the sentence pair $\mathcal{S}$
For each pair of extensions $(f, \tilde{e}), (f', \tilde{e}') \in \mathcal{E}'$
if $(f, \tilde{e}) \subset (f', \tilde{e}')$
$\mathcal{E}' = \mathcal{E}' \setminus (f, \tilde{e})$
Write modified target sentence according to surviving extensions $\mathcal{E}'$
$\mathcal{E}_{max} = \mathcal{E}_{max} \cup \mathcal{E}'$
Output: Modified target language corpus **and** set $\mathcal{E}_{max}$ of maximal extensions

selection criterion is illustrated in Table 3.2.
The algorithm proceeds as follows. First, it generates a list of target extension candidates using the algorithm in Table 3.1. Then a loop over all sentence pairs $\mathcal{S}$ in the aligned training corpus is carried out. A subset of candidate extensions $\mathcal{E}'$ that are contained in the aligned training sentence pair $\mathcal{S}$ is determined. For each pair of extensions $(f, \tilde{e}), (f', \tilde{e}')$ if $(f, \tilde{e})$ is contained in $(f', \tilde{e}')$, symbolically $(f, \tilde{e}) \subset (f', \tilde{e}')$ the contained extension is removed from $\mathcal{E}'$. An extension $(f', \tilde{e}')$ contains another extension $(f, \tilde{e})$, if the same source word f is predicted and the sequence of target words $\tilde{e}'$ contains the sequence of target words $\tilde{e}$, both with respect to the identity of the target words and their position relative to $\tilde{e}_0$. The formal definition of $\subset$ is given below in the context of the multi-word extensions. Finally, the output of the algorithm are the modified target corpus and the final set $\mathcal{E}_{max}$ of maximal extensions.

3.3.4 Selection of Multi-Word Extensions

In the preceeding section, we have restricted ourselves to extensions of the type $(f, \tilde{e})$. But, according to the considerations at the beginning of this chapter, we might choose extensions of the type $p(f|\tilde{e}, \tilde{f})$. Figure 3.9 shows a fragment of the alignment matrix after the final training iteration. The two consecutive English words "how about" are aligned to the German words "wie sieht es aus", where there is a distance of two words between the German words "es" and "aus". According to the alignment matrix, we have the baseline lexical dependency pair (f =aus,e =about) which are marked by filled dots. We can define more complex extensions in exactly the same way as in the preceeding section. Extensions $(f, \tilde{e}, \tilde{f})$ are described as a sequence of triples, where L is the number of triples involved:

$$(f, \tilde{e}, \tilde{f}) \quad = \quad [(tag_i, pos_i, word_i)]_1^L. \tag{3.15}$$

The component triples $(tag, pos, word)$ are defined as follows (we use (t, p, w) as a shorthand description of a triple in the following):

$tag \in \{F, E\}$	determines whether the extension is for a source or a target language word
pos	$tag = E$: the extension position relatively to $\tilde{e}_0$ $pos \in \{-I, \cdots, -1, 0, 1, 2, \cdots, +I\}$ $tag = F$: the extension position relatively to the actual f $pos \in \{-J, \cdots, -1\}$
word	the source, respectively the target word at the corresponding position

Note that each extension $(f, \tilde{e}, \tilde{f})$ must at least contain the two triples "(E,0,"target word")" and "(F,0,"source word")" indicating $\tilde{e}_0$ and f. Sequences of triples describing some possible extensions in Figure 3.3.3 and Figure 3.9 are as follows:

(E,0,council) (F,0,Personalsitzung)	(E,-1,staff) (E,+1,meeting)
(E,0, group) (F,0,Arbeitskreis)	(E,-1,work) (E,+1,meeting)
(E,0, about) (F,0,aus)	(E,-1,how) (F,-3,es) (F,-4,sieht) (F,-5,wie)

Using this notation for an extension $(f, \tilde{e}, \tilde{f})$, we can define counts $N(f, \tilde{e}, \tilde{f})$ and $\bar{N}(f, \tilde{e}, \tilde{f})$ and an extended model $p_{(f,\tilde{e},\tilde{f})}(f|e_1^I)$ exactly in the same way as for the word joining case with the simpler extensions $(f, \tilde{e})$. The relation $\subset$ used in algorithm 3.2 may be defined for the more complex extensions as well:

$$\begin{aligned}&(f, \tilde{e}, \tilde{f}) \subset (f, \tilde{e}', \tilde{f}') := \\ &\quad := \textbf{ for each } (t, p, w) : \textbf{if } (t, p, w) \in (f, \tilde{e}, \tilde{f}) \textbf{then } (t, p, w) \in (f, \tilde{e}', \tilde{f}')\end{aligned}$$

For the concept of the multi-word extensions, it is not clear in which way to use the inverted alignment matrix in order to restrict the possible lexicon extensions since more complex lexical dependencies may not be completely covered by either of the alignment directions. Note that apart from the fact that $\tilde{e}_0$ is aligned to f there is no alignment mapping defined for the remaining source and target words in the extension using the above list of

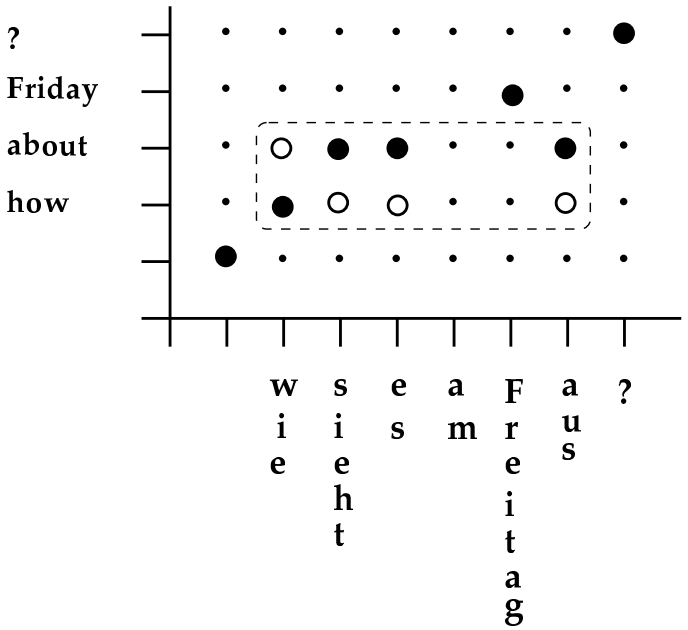


Figure 3.9: Multi-word lexicon extension involving both target and source sentence words. The English phrase of two consecutive words "how about" is mapped to the German phrase "wie sieht es ... aus", where there is a gap of two words between the phrase "wie sieht es" and the German particle "aus". The gap may be filled by different German phrases like "am Freitag", "am Donnerstag" or "etwas später". The unfilled circles indicate additional alignment positions that may be learned as part of a multi-word lexicon extension.

triples. Thus, in a future application, we might think of integrating the multi-word lexicon extension learning into the learning of the unknown alignment mapping itself since each extension marks an alignment mapping for exactly one source sentence word. The multi-word dependencies might be used to replace the fertility parameters in the original IBM-model since different fertilities of a word might actually correspond to certain lexical multi-word dependencies.

Chapter 4

Search Algorithm for Translation

4.1 Introduction into Dynamic Programming

In this section, we give a short introduction into dynamic programming (DP) as a method to solve optimization problems. From the implementation point of view, dynamic programming amounts to creating a table of solutions to all subproblems that might occur. As described in [Rabiner,Juang 1993, Bellmann 1957] dynamic programming is based on the principle of optimality according to Bellmann, where the traditional terminology "policy" is used for a decision rule that determines the next "state" given a predecessor "state". Bellmann's optimality principle is stated as follows:

> An optimal policy has the property that, whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

The expression "dynamic programming" has historical reasons. "Dynamic" in this context means sequential over the time. "Programming" denotes an optimization problem with constraints. The optimality principle says that a sequence of decisions $u(1), u(2), \cdots, u(t), \cdots, u(T-1), u(T)$ can be optimal only if each partial sequences of decisions $u(k+1), \cdots, u(T-1), u(T)$ is optimal, given that the decision $u(1), \cdots, u(k)$ has already been taken. One problem that can be solved using dynamic programming is the optimal path problem: given a set of points and for each pair of points (i, j) the costs for directly moving from point i to point j, find a sequence of moves between two given points with minimum costs. The solution is given in the Dijkstra algorithm. Given that the number of points is n the complexity of the Dijkstra algorithm is $\mathcal{O}(n^2)$. A sequence of moves or decisions in the following is called "path". In more complicated path-finding problems, a point that has an internal structure corresponding to the translation problem is called "state" or "(partial) hypothesis". Similar to "states" in speech recognition "states" in machine translation correspond to gridpoints in a translation lattice where the transitions between states are associated with translation costs. A partial hypothesis is a hypothesis which still does not account for the entire optimization problem.
The search algorithms presented here will try to find shortest paths in translation lattices of some regular structure (as opposed to a search lattice of arbitrary structure as is processed by the plain Dijkstra algorithm): not all possible connections between states exist.

In [Rabiner,Juang 1993], these problems are called sequential decision problems. The algorithm by Held & Karp which will be used for the translation algorithms can be considered as a sequential decision problem, since subsets of visited cities of a given cardinality are processed synchronously. The sequentiality is over the cardinality of the subsets in this case.
For the formalization of the sequential decision problem we follow the representation given in [Ney 1996]. We have to take decisions at $j = 1, \cdots .J$ steps, where we have costs at each step that are denoted by:

$$h(j, u(j-1), u(j))$$

We sum up over all decision steps $u(1), u(2), \cdots, u(j), \cdots, u(J-1), u(J)$:

$$\min_{j \to u(j)} \sum_{j=1}^{J} h(j, u(j-1), u(j))$$

To solve the above optimization problem using dynamic programming, we define an auxiliary quantity $H(j, u)$ as follows:

$$H(j,u) = \begin{array}{l} \text{costs of the best partial decision sequence,} \\ \text{which ends in position } j \text{ and decision } u. \end{array}$$

The auxiliary quantity $H(j, u)$ is recursively evaluated and the decisions are stored at each decision point (j, u) using the so-called backpointer $B(j, u)$:

$$\begin{aligned} H(j,u) &= \min_{u'}\{H(j-1,u') + h(j,u',u)\} \\ B(j,u) &= \arg\min_{u'}\{H(j-1,u') + h(j,u',u)\} \end{aligned}$$

At the end of the computation, the optimal decision sequence is given by:

$$\min_{u} H(J,u).$$

The overall complexity of the DP search is given by suming over the product of the number of decisions $|u(j)|$ at step j and the number of decision $|u(j-1)|$ at step $j-1$:

$$\sum_{j=1}^{J} |u(j)| \cdot |u(j-1)|.$$

Assuming a constant number of decisions at each stage, we get a complexity of:

$$|u|^2 \cdot J \quad .$$

Differing from what is the case in speech recognition, the number of decisions at each stage j must not be a constant in general. E.g. when conceiving the algorithm by Held & Karp as a sequential decision problem, each step corresponds to a subset cardinality j. For a given cardinality j, the number of subsets for that cardinality is not linear in j [1].
In the following, we will presented **three** dynamic programming based algorithm that deal with subsequential decision problems, where the sequentiallity is along different translation "directions":

[1] Although, for the GE and EG re-ordering constraints as presented in Section 4.6.2, the number of subsets for cardinality j will actually be linear in the cardinality j.

- **Monotone DP Search (Section 4.3)**: The source sentence is processed position-synchronously along the positions j of the source sentence.

- **Inverted DP Search (Section 4.4)**: The target sentence is processed position-synchronously along the positions i of the target sentence.

- **Held & Karp - style DP Search (Section 4.5)**: The source sentence is processed cardinality-synchronously along subsets of equal cardinality j.

4.2 Dynamic Programming for Machine Translation

The task of the search algorithm according to Figure 1.2 is to generate the most likely target sentence e_1^I of unknown length I for an observed source sentence f_1^J. To illustrate the specific details of the search problem, we consider a first-order alignment model and obtain the following optimization criterion:

$$\begin{aligned}
&\max_I \left\{ p(J|I) \cdot \max_{e_1^I} \left\{ Pr(e_1^I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} \left[p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j}) \right] \right\} \right\} \qquad (4.1)\\
&\cong \max_I \left\{ p(J|I) \cdot \max_{e_1^I} \left\{ Pr(e_1^I) \cdot \max_{a_1^J} \prod_{j=1}^{J} \left[p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j}) \right] \right\} \right\}\\
&= \max_I \left\{ p(J|I) \cdot \max_{e_1^I} \left\{ \prod_{i=1}^{I} p(e_i|e_{i-1}) \cdot \max_{a_1^J} \prod_{j=1}^{J} \left[p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j}) \right] \right\} \right\},
\end{aligned}$$

where we have replaced the sum over all alignments by the best alignment, which is again the so-called maximum approximation. In addition, we have used a bigram language model, which is given in terms of the conditional probability $p(e_i|e_{i-1})$ of observing word e_i given the predecessor word e_{i-1}:

$$Pr(e_1^I) = \prod_{i=1}^{I} p(e_i|e_{i-1})$$

Considering this criterion, we can see that there are two conflicting requirements for a search strategy: the product over i, i.e. the language model term, tries to cover all positions in the target string, whereas the product over j, i.e. the lexicon and the alignment model term, tries to cover all positions in the source string.
In Section 4.3, we present a simple dynamic programming search algorithm based on the concept of the monotone alignments as introduced in Section 3.1. This approach requires some preprocessing steps to re-order input source strings to deal with word order differences in source and target language. Starting from Section 4.4, a dynamic programming approach is presented that incorporates the word re-ordering into the search procedure. It is based on the concept of the so-called *inverted alignments*.

4.3 DP Algorithm Using Monotone Alignments

In this approach, we first assume that the alignments satisfy the *monotonicity* requirement [Tillmann et al. 1997a] as introduced in Section 3.1. To satisfy this requirement, we will assume suitably restricted permutations of the source string. For the translation direction Spanish-English, the necessary word re-orderings are described in [Tillmann et al. 1997a]. For the translation direction German-to-English the word re-ordering is more complex and mainly linked to the German verb group. The re-ordering carried out for this translation direction is described in [Tillmann et al. 1997c] [2].
Making the monotonicity assumption, the two requirements, namely covering all source positions and all target positions, can be satisfied simultaneously. For the alignment model, the monotonicity property allows only transitions from a_{j-1} to a_j with the *jump width* δ:

$$\delta \equiv a_j - a_{j-1} \in \{0, 1, 2\} \quad .$$

We define a modified probability $p_\delta(e|e')$ for the language model depending on the alignment difference δ. We consider each of the three cases $\delta = 0, 1, 2$ separately:

- $\delta = 0$ (horizontal transition = alignment repetition): This case corresponds to a target word with two or more aligned source words and therefore requires $e = e'$ so that there is no contribution from the language model:

$$p_{\delta=0}(e|e') = \begin{cases} 1 & \text{for} \quad e = e' \\ 0 & \text{for} \quad e \neq e' \end{cases} \quad .$$

- $\delta = 1$ (forward transition = regular alignment): This case is the regular one, and we can use directly the probability of the bigram language model:

$$p_{\delta=1}(e|e') = p(e|e') \quad .$$

- $\delta = 2$ (skip transition = non-aligned word): This case corresponds to skipping a word, i.e. there is a word in the target string with no aligned word in the source string. We have to find the highest probability of placing a non-aligned word $\tilde{e}$ between a predecessor word e' and a successor word e. Thus we optimize the following product over the non-aligned word $\tilde{e}$:

$$p_{\delta=2}(e|e') = \max_{\tilde{e}} \left[p(e|\tilde{e}) \cdot p(\tilde{e}|e') \right] \quad .$$

  If the vocabulary size of the target language is small enough or if the number of candidate words can be somehow limited, this maximization can be done beforehand for all pairs (e, e'), and the result can be stored in a table.

Using this modified LM probability $p_\delta(e|e')$, we can rewrite the search criterion in Eq. 4.1:

$$\max_I \left\{ p(J|I) \cdot \left\{ \max_{e_1^I, a_1^J} \prod_{j=1}^{J} \left[p(a_j|a_{j-1}, I) \cdot p_{[a_j - a_{j-1}]}(e_{a_j}|e_{a_{j-1}}) \cdot p(f_j|e_{a_j}) \right] \right\} \right\} \quad ,$$

[2] Even after several months of rule building, it appeared difficult to cover all cases consistently, e.g. to deal with the German subject phrase.

where the two products over i and j have been merged into a single product over i. Here, apart from the modified bigram language model $p_\delta(e|e')$, we have used the alignment probability $p(a_j|a_{j-1}, I)$ and the lexicon probability $p(f|e)$ of the HMM-based alignment model described in Section 1.3 . For a hypothesized length I, the problem now is to find the unknown mapping:

$$j \rightarrow (a_j, e_{a_j}) \quad ,$$

which defines a path through a network with a uniform trellis structure as shown in Figure 3.1. In each position i along the vertical axis, we have to allow *all* possible words e of the target language. Due to the bigram language model and the monotonicity of the alignment model, we have only first-order type dependencies such that the local probabilities (or costs when using the negative logarithms of the probabilities) depend *only* on the transitions in the lattice. Each possible index triple (i, j, e) defines a grid point in the lattice, and we have the following set of possible transitions leading to grid point (i, j, e):

$$\delta \in \{0, 1, 2\}: \quad (i - \delta, j - 1, e') \rightarrow (i, j, e) \quad .$$

Each of these transitions is assigned a local probability:

$$p(i|i - \delta, I) \cdot p_\delta(e|e') \cdot p(f_j|e) \quad ,$$

where we again have assumed a homogeneous dependence for the HMM alignment model. Apart from the boundary conditions $i \leq I$, there is no dependence on the target string length I so that we will ignore this dependence in the formulation of the search strategy. Using this formulation of the search task, we can now use the method of dynamic programming (DP) to find the best path through the lattice. To this purpose, we introduce the auxiliary quantity:

$$\begin{aligned} Q(i, j, e): \quad & \text{probability of the best partial hypothesis} \\ & (e_1^i, a_1^j) \text{ with } e_i = e \text{ and } a_j = i. \end{aligned}$$

Since we have only first-order dependencies in the model, it is easy to see that the auxiliary quantity must satisfy the following DP recursion equation:

$$Q(i, j, e) \;=\; p(f_j|e) \cdot \max_{\delta, e'} \{p(i|i - \delta, I) \cdot p_\delta(e|e') \cdot Q(i - \delta, j - 1, e')\} \quad .$$

For the Monotone DP search, we omit a special treatment of the start and end conditions like $j = 1$ or $j = J$ in order to simplify the presentation. The DP equation is evaluated recursively to find the best partial path to each grid point (i, j, e). To explicitly construct the unknown word sequence e_1^I of optimal length I, we first incorporate the length model $p(J|I)$ for each possible length I and determine the best end hypothesis:

$$\max_{I, \tilde{e}} \{p(J|I) \cdot Q(I, J, \tilde{e})\} \; ,$$

where $\tilde{e}$ denotes the target words that can occur as final words of the target sentence, i.e. in particular the punctuation marks if they are included in the vocabulary of the target

Table 4.1: DP-based search algorithm for the monotone translation model.

input: source string $f_1 ... f_j ... f_J$
initialization
for each position $j = 1, 2, ...J$ in source sentence do
for each position $i = 1, 2, ..., I_{max}$ in target sentence do
for each target word e do
$Q(i,j,e) = p(f_j\|e) \cdot \max_{\delta,e'}\{p(i\|i-\delta) \cdot p_\delta(e\|e') \cdot Q(i-\delta, j-1, e')\}$
traceback:
- find best end hypothesis: $\max_{I,\tilde{e}}\{p(J\|I) \cdot Q(I,J,\tilde{e})\}$ - recover optimal word sequence

language. For implementation purposes, it is convenient to make use of so-called backpointers which store for each grid point (i, j, e) the best predecessor grid point [Ney et al. 1992]. The complexity of the algorithm is

$$I_{max} \cdot J \cdot E^2 \quad ,$$

where E is the size of the target language vocabulary and I_{max} is the maximum length of the target sentence considered. The resulting search algorithm is presented in Table 4.1.
A speed-up is obtained by changing the model assumptions and dropping the explicit length model $p(J|I)$. The complexity of the search algorithm is reduced to $J \cdot E^2$, since the partial hypotheses (i, j, e) have no longer to be distinguished according to the length i of the target string of the partial hypothesis.
For the rest of this work, the explicit length model $p(J|I)$ will be replaced by the probability to predict the sentence boundary word \$ using the ending target words of the final translation hypotheses. Furthermore, by standard we will assume the use of a trigram language model for the notation used. A bigram language model has been chosen in this section because the three cases $\delta = 0, 1, 2$ are most easily expressed using a bigram language model.

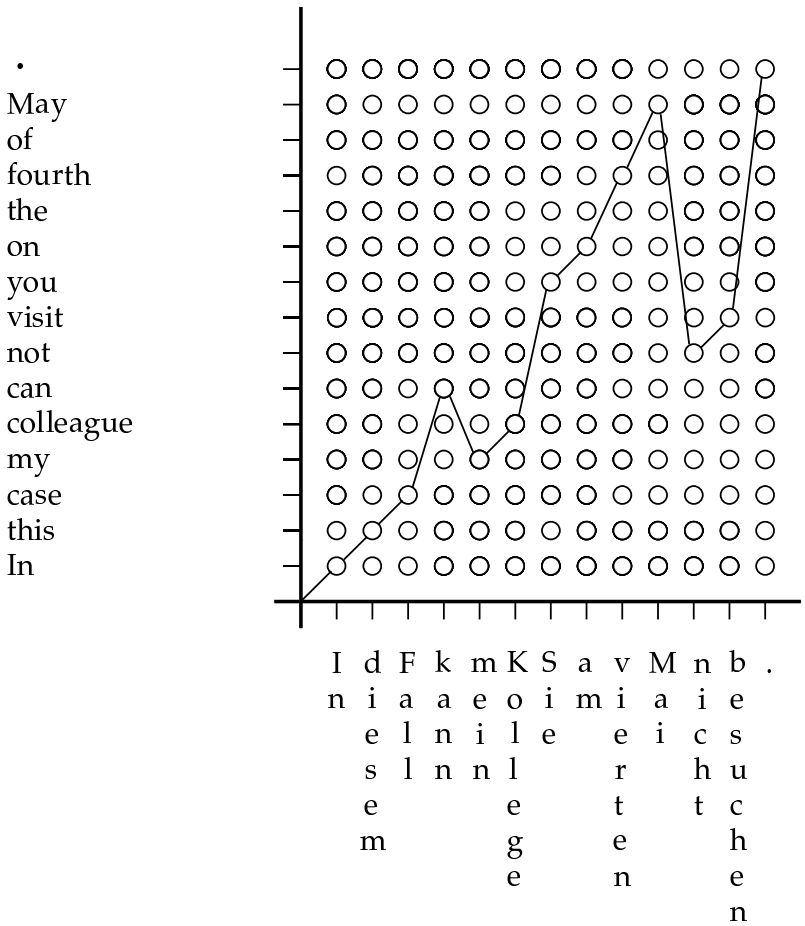


Figure 4.1: Regular alignment example for the translation direction German-to-English. For each German source word there is exactly one English target word on the alignment path.

4.4 DP Algorithm Using Inverted Alignments

So far, we have used the alignment model introduced in Section 1.3. Here, a source position j is mapped to exactly one target position i:

$$j \rightarrow i = a_j \quad .$$

An example for this kind of alignment is given in Figure 4.1, where a German source position j is mapped to an English target position i. This example will be used for the description of the DP-based search procedure in Section 4.6. In [Brown et al. 1993], this "regular" alignment concept is used for the IBM-1, IBM-2, and IBM-3 translation models.

For the IBM-4 model, where the distortion probabilities are defined over input sentence positions only, we use the concept of the so-called inverted alignments as introduced in [Niessen et al. 1998, Ney et al. 2000]. An inverted alignment is defined as follows:

$$\text{inverted alignment:} \quad i \rightarrow j = b_i \quad .$$

Here, a target position i is mapped to a source position j. The so-called coverage constraint for an inverted alignment is not expressed by the notation: each source position j should be "hit" exactly once by the path of the inverted alignment $b_1^I = b_1...b_i...b_I$. The advantage of the inverted alignment concept is that we can construct target sentence hypotheses from bottom to top along the positions of the target sentence.
To keep the mathematical description simple, we will use the following simplified set of translation model parameters: lexicon probabilities $p(f|e)$ and distortion probabilities $p(j|j', J)$. Here, j is the currently covered input sentence position and j' is the previously covered input sentence position. The input sentence length J is included, since we would like to think of the distortion probability as normalized according to J. No fertility probabilities or "null" word probabilities are used, thus each source word f is translated as exactly one target word e and each target word e is translated as exactly one source word f. The simplified notation will help us to focus on the most relevant details of the dynamic programming based search procedure. The simplified set of parameters leads to an unrealistic assumption about the length of the source and target sentence, namely $I = J$. During the translation experiments we will, of course, not make this assumption. The implementation details for using the full set of IBM-4 model parameters are given in Section 4.9.
Using the inverted alignments in the maximum approximation, we re-write Eq. (1.1) to obtain the following search criterion where we are looking for the most likely target sentence e_1^I of unknown length I for an observed source sentence f_1^J of length J:

$$\begin{aligned}
&\max_I \left\{ p(J|I) \cdot \max_{e_1^I} \left\{ p(e_1^I) \cdot p(f_1^J|e_1^I) \right\} \right\} = \qquad (4.2)\\
&\cong \max_I \left\{ p(J|I) \cdot \max_{e_1^I} \left\{ \prod_{i=1}^{I} p(e_i|e_{i-1}, e_{i-2}) \cdot \max_{b_1^I} \prod_{i=1}^{I} [p(b_i|b_{i-1}, J) \cdot p(f_{b_i}|e_i)] \right\} \right\}\\
&= \max_I \left\{ p(J|I) \cdot \max_{e_1^I, b_1^I} \left\{ \prod_{i=1}^{I} [p(e_i|e_{i-1}, e_{i-2}) \cdot p(b_i|b_{i-1}, J) \cdot p(f_{b_i}|e_i)] \right\} \right\} \quad .
\end{aligned}$$

The following notation is used: e_{i-1}, e_{i-2} are the immediate predecessor target words, e_i is the word to be predicted. $p(e_i|e_{i-1}, e_{i-2})$ denotes the trigram language model probability. $p(f_{b_i}|e_i)$ is the lexicon probability for translating the target word e_i as source word f_{b_i}. $p(b_i|b_{i-1}, J)$ is the distortion probability for covering source position b_i after source position b_{i-1}. Note that in Eq. 4.2 two products over i are merged into a single product over i. The translation probability $p(f_1^J|e_1^I)$ is computed in the so-called maximum approximation using the distortion and the lexicon probabilities. $p(J|I)$ is the sentence length model that will be dropped in the following (it is not used in the IBM-4 translation model). For each source sentence f_1^J to be translated, we are searching for the unknown mapping that optimizes Eq. 4.2:

$$i \rightarrow (b_i, e_i) \quad .$$

In Section 4.6, we will introduce an auxiliary quantity that can be evaluated recursively using dynamic programming to find this unknown mapping. We will explicitly take care of the coverage constraint by introducing a coverage set $\mathcal{C}$ of source sentence positions that

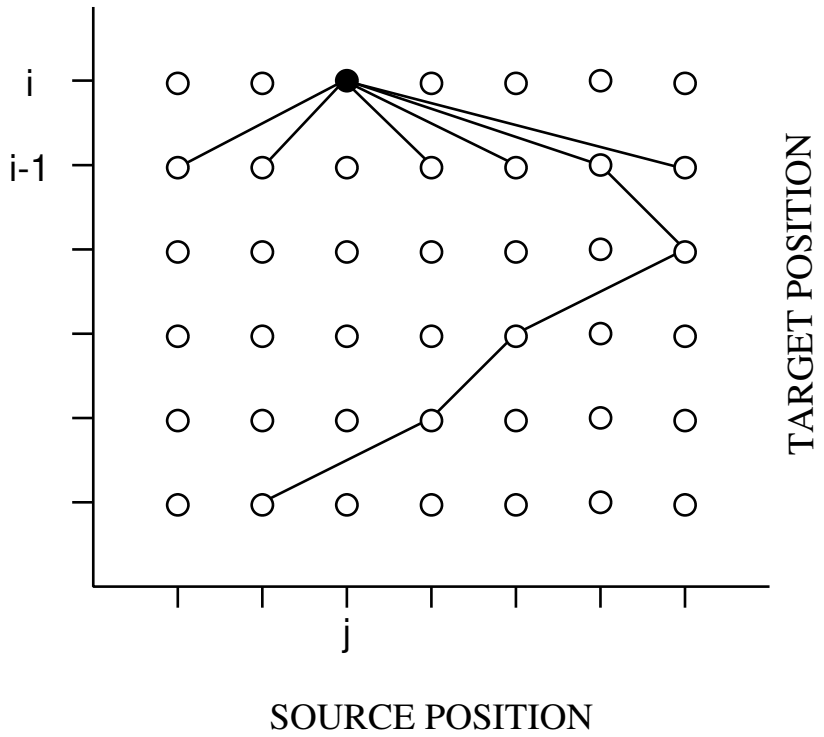


Figure 4.2: Illustration of the transitions in the inverted alignment model. The target sentence is generated from bottom to top.

have already been processed. Figure 4.2 illustrates the concept of the search algorithm using inverted alignments: partial hypotheses are constructed from bottom to top along the positions of the target sentence. Partial hypotheses of length $i-1$ are extended to obtain partial hypotheses of the length i. Extending a partial hypothesis means covering a source sentence position j that has not been covered yet. For a given gridpoint in the translation lattice, the unknown target word sequence can be obtained by tracing back the translation decisions to the partial hypothesis at stage $i = 1$.

4.5 Held & Karp Algorithm for Traveling Salesman Problem

[Held,Karp 1962] presents a dynamic programming approach to solve the Traveling Salesman Problem (TSP). The TSP is an optimization problem which is defined as follows: given is a set of cities $\{1, \cdots, J\}$ and for each pair of cities j, j' the cost $d_{jj'} > 0$ for traveling from city j to city j'. We are looking for the shortest tour visiting all cities exactly once while starting and ending in city 1. We are using the notation $\mathcal{C}$ for the set of cities, since it corresponds to a coverage set of processed source positions in machine translation. A straightforward way to find the shortest tour is by trying all possible permutations of the J cities. The resulting algorithm has a complexity of $O(J!)$. However, dynamic programming

Table 4.2: DP-based algorithm for solving the traveling salesman problem due to Held and Karp. The outer-most loop is over the cardinality of subsets of already visited cities.

input: cities $j = 1, \cdots, J$ with distance matrix $d_{jj'}$
initialization: $D(\{k\}, k) := d_{1k}$
for each path length $c = 2, \cdots, J$ do
for each pair $(\mathcal{C}, j)$, where $\mathcal{C} \subseteq \{2, \cdots, J\}$ and $j \in \mathcal{C}$ and $\lvert\mathcal{C}\rvert = c$ do
$D(\mathcal{C}, j) = \min_{j' \in \mathcal{C} \setminus \{j\}} \{d_{jj'} + D(\mathcal{C} \setminus \{j\}, j')\}$
traceback:
- find shortest tour: $D^* = \min_{k \in \{2, \cdots, J\}} [D(\{2, \cdots, J\}, k) + d_{k1}]$
- recover optimal sequence of cities

can be used to find the shortest tour in $O(J^2 \cdot 2^J)$, which is a much smaller complexity for larger values of J. The approach recursively evaluates the quantity $D(\mathcal{C}, j)$:

$$D(\mathcal{C}, j) := \begin{array}{l}\text{costs of the partial tour starting in city 1, ending} \\ \text{in city } j \text{ and visiting all cities in } \mathcal{C}.\end{array}$$

Subsets of cities $\mathcal{C}$ of increasing cardinality c are processed. The algorithm works due to the fact that not all permutations of cities have to be considered explicitly. During the computation, for a pair $(\mathcal{C}, j)$, the order in which the cities in $\mathcal{C}$ have been visited can be ignored (except j), only the costs for the best path reaching j has to be stored. The algorithm is shown in Table 4.2. For the initialization the costs for starting from city 1 are set: $D(\{k\}, k) = d_{1k}$ for each $k \in \{2, \cdots, |\mathcal{C}|\}$. Then, subsets $\mathcal{C}$ of increasing cardinality are processed. Finally, the cost for the optimal tour is obtained in the second to the last line. The optimal tour itself can be found using a so-called backpointer-array, where the optimal decision for each gridpoint $(\mathcal{C}, j)$ is stored.
Figure 4.3 illustrates the use of the algorithm by showing the "supergraph" that is searched in the Held & Karp algorithm for a TSP with $J = 5$ cities. When traversing the lattice from left to right following the different possibilities, a partial path to a node j corresponds to the subset $\mathcal{C}$ of all cities on that path together with the last visited city j. Of all the different paths merging into the node j only the best partial path has to be retained for further computation.

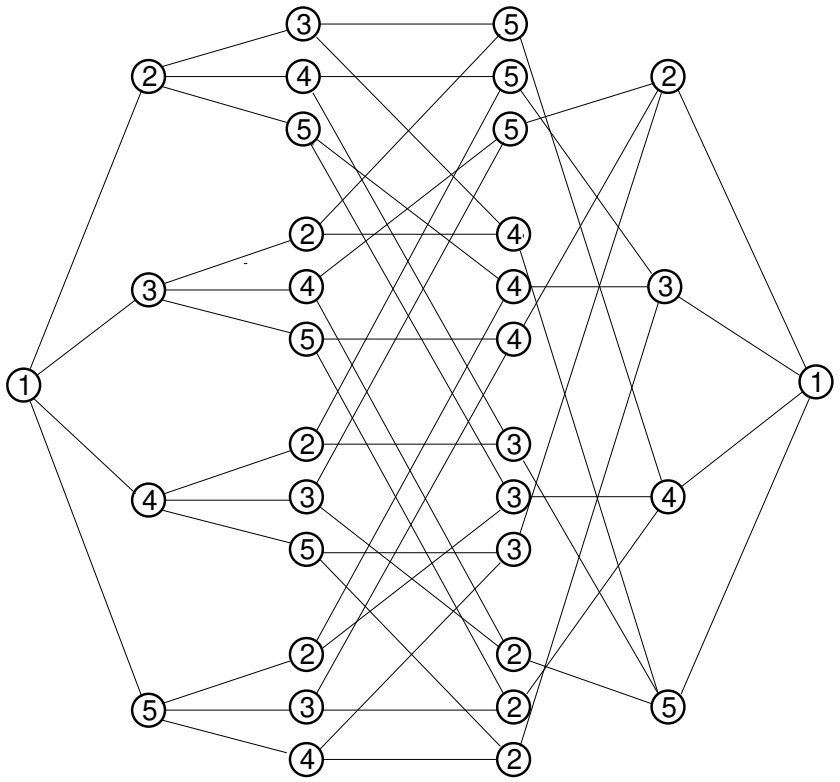


Figure 4.3: Illustration of the algorithm by Held & Karp for a traveling salesman problem with $J = 5$ cities. Not all permutations of cities have to be considered. For a given subset of cities the order in which the cities have been visited can be ignored.

4.6 DP Algorithm for Statistical Machine Translation

In this section, the Held & Karp algorithm is applied to statistical machine translation. Using the concept of inverted alignments as introduced in Section 4.4, we explicitly take care of the coverage constraint by introducing a coverage set $\mathcal{C}$ of source sentence positions that have already been processed. Here, the correspondence is according to the fact that each source sentence position has to be covered exactly once fullfilling the coverage constraint. The cities of the more complex translation TSP correspond roughly to triples (e', e, j) where the notation is given below. The final path output by the translation algorithm will contain exactly one triple (e', e, j) for each source position j.
The algorithm processes subsets of partial hypotheses with coverage sets $\mathcal{C}$ of increasing cardinality c. For a trigram language model, the partial hypotheses are of the form $(e', e, \mathcal{C}, j)$. e', e are the last two target words, $\mathcal{C}$ is a coverage set for the already covered source positions and j is the last covered position. The target word sequence that ends in e', e is stored as a backpointer to the predecessor partial hypothesis (and recursively to its predecessor hypotheses) and is not shown in the notation. Each distance in the traveling salesman problem now corresponds to the negative logarithm of the product of the translation, distortion, and language model probabilities. The following auxiliary quantity is defined:

Table 4.3: DP-based algorithm for statistical MT which consecutively processes subsets $\mathcal{C}$ of source sentence positions of increasing cardinality.

input: source language string $f_1 ... f_j ... f_J$
initialization
for each cardinality $c = 1, 2, \cdots, J$ do
for each pair $(\mathcal{C}, j)$, where $\mathcal{C} \subseteq \{1, \cdots, J\}$ and $j \in \mathcal{C}$ and $\|\mathcal{C}\| = c$ do
for each pair of target words $e', e \in E$
$Q_{e'}(e, \mathcal{C}, j) = p(f_j\|e) \max_{\substack{e'' \\ j' \in \mathcal{C} \setminus \{j\}}} \{p(j\|j', J) \cdot p(e\|e', e'') \cdot Q_{e''}(e', \mathcal{C} \setminus \{j\}, j')\}$
traceback:
- find best end hypothesis: $\max_{e, e', j} \{p(\$\|e, e') \cdot Q_{e'}(e, \{1, \cdots, J\}, j)\}$
- recover optimal word sequence

$$Q_{e'}(e, \mathcal{C}, j) := \quad \text{probability of the best partial hypothesis } (e_1^i, b_1^i), \text{ where } \mathcal{C} = \{b_k | k = 1, \cdots, i\},\ b_i = j,\ e_i = e, \text{ and } e_{i-1} = e'.$$

The above auxiliary quantity satisfies the following recursive DP equation:

$$Q_{e'}(e, \mathcal{C}, j) \quad = \quad p(f_j|e) \cdot \max_{\substack{e'' \\ j' \in \mathcal{C} \setminus \{j\}}} \Big\{ p(j|j', J) \cdot \ p(e|e', e'') \cdot Q_{e''}(e', \mathcal{C} \setminus \{j\}, j') \Big\}.$$

Here, j' is the previously covered source sentence position and e', e'' are the immediate predecessor words. The DP equation is evaluated recursively for each hypothesis $(e', e, \mathcal{C}, j)$. The resulting algorithm is depicted in Table 4.3. Some details concerning the initialization and the finding of the best target language string are presented in Section 4.6.1. $p(\$|e, e')$ is the trigram language probability to predict the sentence boundary symbol \$. The complexity of the algorithm is $O(E^3 \cdot J^2 \cdot 2^J)$, where E is the size of the target language vocabulary.

4.6.1 Verb Group Re-ordering: German-to-English

The above search space is still too large to translate even a medium length input sentence. On the other hand, only very restricted re-orderings are necessary, e.g. for the translation direction German-to-English the word order difference is mostly restricted to the German verb group. The approach presented here assumes a mostly monotonic traversal of the

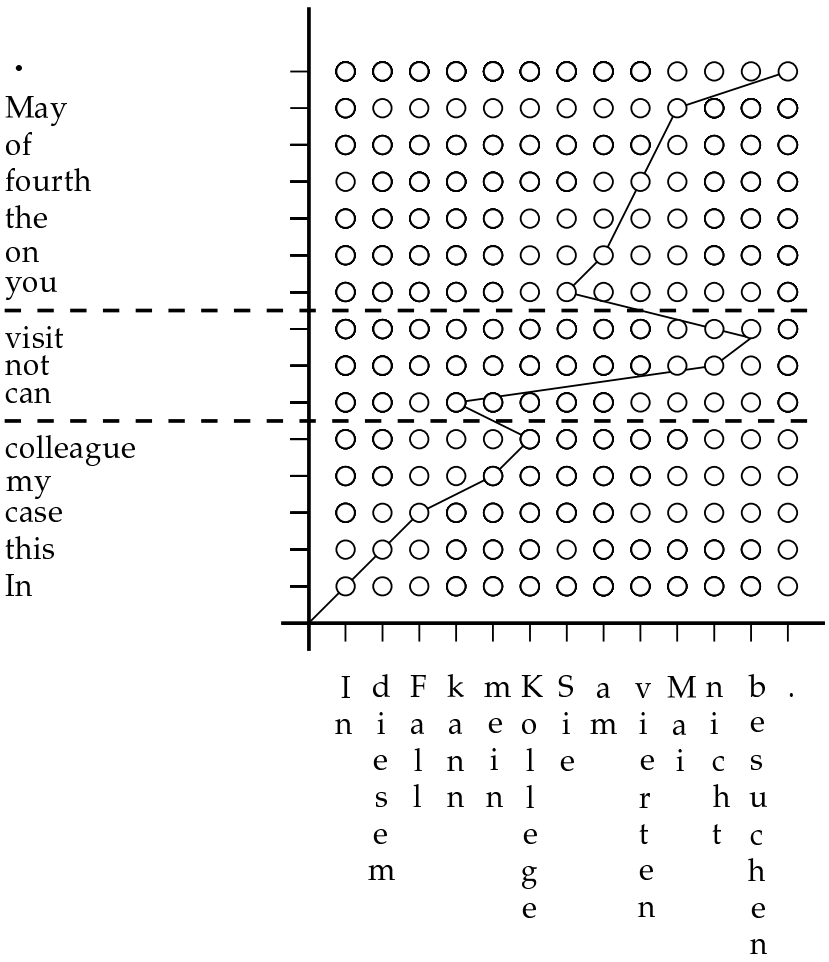


Figure 4.4: Word re-ordering for the translation direction German-to-English: the re-ordering is restricted to the German verb group.

source sentence positions from left to right [3]. A small number of positions may be processed sooner than they would be in that monotonic traversal. Each source position then generates a certain number of target words. The restrictions are fully formalized in Section 4.6.2.

A typical situation is shown in Figure 4.4. When translating the sentence monotonically from left to right, the translation of the German finite verb "kann", which is the left verbal brace in this case, is skipped until the German noun phrase "mein Kollege" is translated, which is the subject of the sentence. Then, the right verbal brace is translated: the infinitive "besuchen" and the negation particle "nicht". The following restrictions are used: one position in the source sentence may be skipped for a distance of up to $L = 4$ source positions, and up to two source positions may be moved for a distance of at most $R = 10$ source positions (The notation L and R shows the relation to the handling of the left and right verbal brace). To formalize the approach, we introduce four verb group states $\mathcal{S}$:

[3]The beam-search pruning techniques in Section 4.7 require that the partial hypotheses cover mostly identical parts of the input sentence.

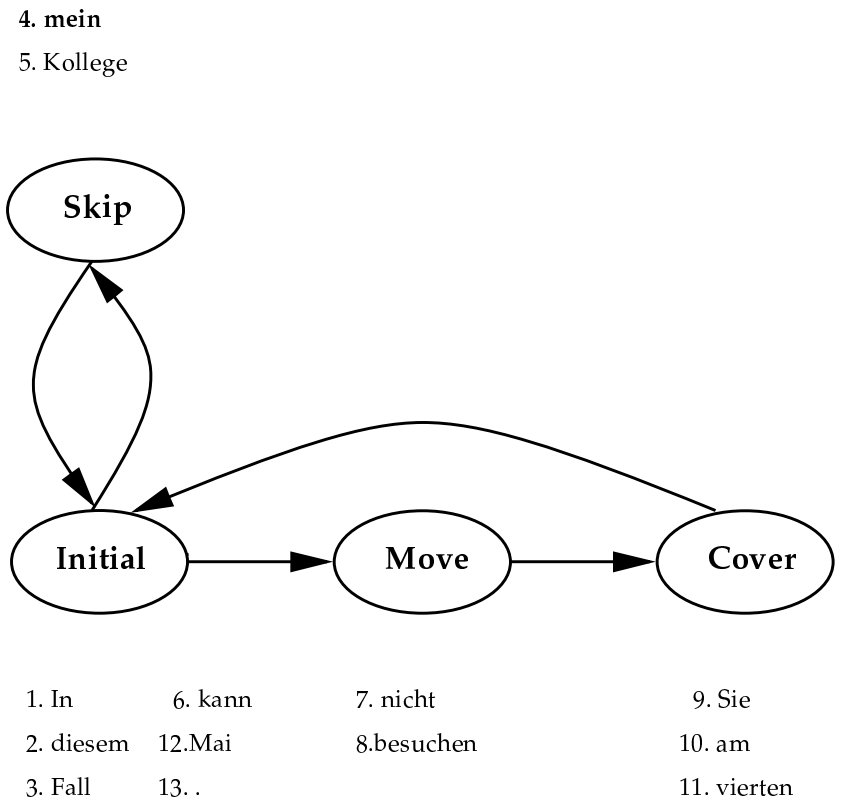


Figure 4.5: Order, in which the German source positions are covered for the German-to-English re-ordering example given in Figure 4.4.

- *Initial* : A contiguous, initial block of source positions is covered.
- *Skip* : One word may be skipped leaving a "hole" in the monotonic traversal.
- *Move* : Up to two words may be "moved" from later in the sentence.
- *Cover* : The sentence is traversed monotonically until the state *Initial* is reached.

The states "Move" and "Skip" both allow to process a set of upcoming words sooner then would be the case in the monotonic traversal. The state "Initial" is entered whenever there are no uncovered positions to the left of the rightmost covered position. The sequence of states needed to carry out the word re-ordering example in Figure 4.4 is given in Figure 4.5. The 13 source sentence positions are processed in the order shown. A position is presented by the word at that position. A formal specification of the state transitions is given in Section 4.6.2. Any number of consecutive German verb phrases in a sentences can be processed by the algorithm. The finite-state control presented here is obtained from a simple analysis of the German-English word re-ordering problem and is not estimated from the training data. It can be viewed as an addition to the IBM-4 model distortion probabilities.
Using the above states, we define partial hypothesis extensions of the following type:

$$(\mathcal{S}', \mathcal{C} \setminus \{j\}, j') \rightarrow (\mathcal{S}, \mathcal{C}, j).$$

Not only the coverage set $\mathcal{C}$ and the positions j, j', but also the verb group states $\mathcal{S}, \mathcal{S}'$ are taken into account. To be short, we have omitted the target language words e, e' in the notation of the partial hypothesis extension. For each extension an uncovered position is added to the coverage set $\mathcal{C}$ of the partial hypothesis and the verb group state $\mathcal{S}$ may change. A more detailed description of the partial hypothesis extension for a certain state $\mathcal{S}$ is given in the next section in a more general context. Covering the first uncovered position in the source sentence, we use the language model probability $p(e|\$, \$)$. Here, \$ is the sentence boundary symbol, which is thought to be at position 0 in the target sentence. The search starts in the hypothesis $(Initial, \{\emptyset\}, 0)$. $\{\emptyset\}$ denotes the empty set, where no source sentence position is covered. The following recursive equation is evaluated:

$$\begin{aligned} & Q_{e'}(e, \mathcal{S}, \mathcal{C}, j) = \\ & = p(f_j|e) \max_{\substack{e'', \mathcal{S}', j' \\ (\mathcal{S}', \mathcal{C}\setminus\{j\}, j') \to (\mathcal{S}, \mathcal{C}, j) \\ j' \in \mathcal{C}\setminus\{j\}}} \Big\{ p(j|j', J) \cdot p(e|e', e'') \cdot Q_{e''}(e', \mathcal{S}', \mathcal{C} \setminus \{j\}, j') \Big\}. \end{aligned} \tag{4.3}$$

The search ends in the hypotheses $(Initial, \{1, \cdots, J\}, j)$, the last covered position may be in the range $j \in \{J - L, \cdots, J\}$, because some source positions may have been skipped at the end of the input sentence. $\{1, \cdots, J\}$ denotes a coverage set including all positions from position 1 to position J. The final translation probability Q_F is:

$$Q_F = \max_{\substack{e, e' \\ j \in \{J-L, \cdots, J\}}} p(\$|e, e') \cdot Q_{e'}(e, Initial, \{1, \cdots, J\}, j), \tag{4.4}$$

where $p(\$|e, e')$ denotes the trigram language model, which predicts the sentence boundary \$ at the end of the target sentence. Q_F can be obtained using an algorithm very similar to the one given in Table 4.3. The complexity of the verb group re-ordering for the translation direction German-to-English is $\mathcal{O}(J \cdot (R^2 \cdot L \cdot R))$ as shown in Appendix B.

4.6.2 Word Re-ordering: Generalization

For the translation direction English-to-German the word re-ordering can be restricted in a similar way as for the translation direction German-to-English. Again, the word-order difference is mainly due to the German verb group. During the translation process, the English verb group is decomposed as shown in Fig 4.6. When translating the sentence monotonically from left to right, the translation of the English finite verb "can" is moved and it is translated as the German left verbal brace before the English noun phrase "my colleague", which is the subject of the sentence. The translation of the infinitive "visit" and of the negation particle "not" are skipped until later in the translation process. For this translation direction, the translation of one source sentence position may be moved for a distance of up to $L = 4$ source positions, and the translation of up to two source positions may be skipped for a distance of up to $R = 10$ source positions (we take over the L and R notation from the previous section). Thus, the role of the skipping and the moving are just reversed. For the example translation in Figure 4.6, the order in which the source sentence positions are covered is given in Figure 4.7.

We generalize the two approaches for the different translation directions as follows: Still, we assume that the source sentence is mainly processed monotonically. A small number

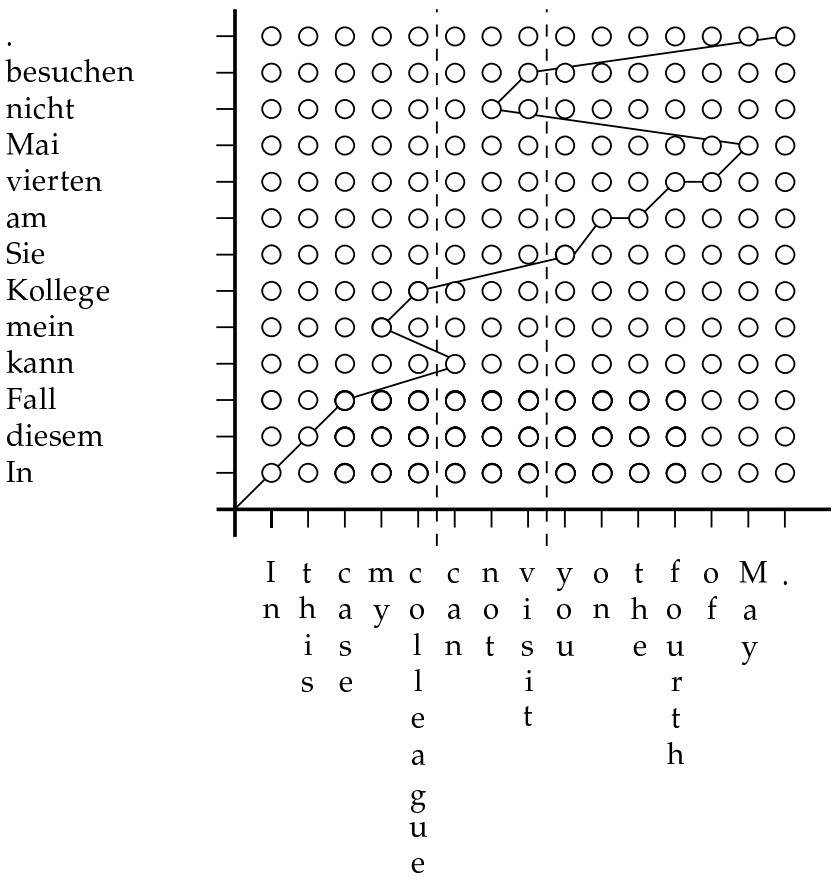


Figure 4.6: Word re-ordering for the translation direction English-to-German: the re-ordering is restricted to the English verb group.

of upcoming source sentence positions may be processed earlier than they would be in the monotonic traversal: the states "Skip" and "Move" are used as explained in the preceeding section. The positions to be processed outside the monotonic traversal are restricted:

- The number of positions dealt with in the states $Move$ and $Skip$ is restricted.
- There are distance restrictions for the source positions processed in those states.

The restrictions will be fully formalized later in this section. In the state $Move$ some source sentence positions are "moved" from later in the sentence. After source sentence positions are moved, they are marked and the translation of the sentence is continued monotonically, keeping track of the already covered positions. To formalize the approach, we introduce four **re-ordering** states $\mathcal{S}$:

- $Initial$: A contiguous, initial block of source positions is covered.
- $Skip$: A restricted number of source positions may be skipped leaving "holes" in the monotonic traversal.
- $Move$: A restricted number of words may be "moved" from later in the sentence.

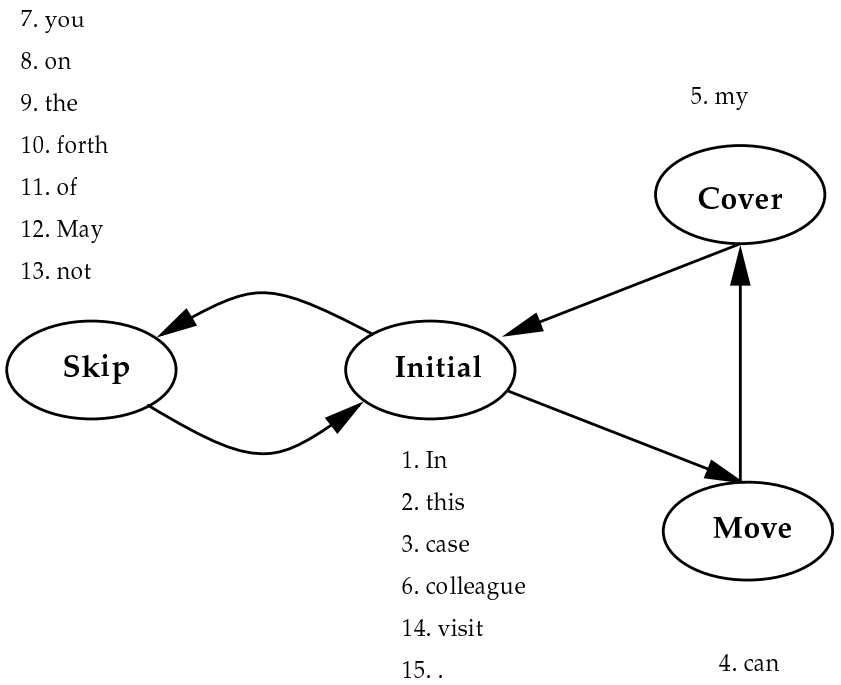


Figure 4.7: Order, in which the English source positions are covered for the English-to-German re-ordering example given in Figure 4.6.

- *Cover* : The sentence is traversed monotonically until the state *Initial* is reached.

To formalize the approach, the following notation is introduced:

$$\begin{aligned} r_{max}(\mathcal{C}) &= \max_{c \in \mathcal{C}} c \\ l_{min}(\mathcal{C}) &= \min_{c \notin \mathcal{C}} c \\ u(\mathcal{C}) &= card(\{c | c \notin \mathcal{C} \text{ and } c < r_{max}(\mathcal{C})\}) \\ m(\mathcal{C}) &= card(\{c | c \in \mathcal{C} \text{ and } c > l_{min}(\mathcal{C})\}) \\ w(\mathcal{C}) &= r_{max}(\mathcal{C}) - l_{min}(\mathcal{C}) \end{aligned}$$

$r_{max}(\mathcal{C})$ is the rightmost covered and $l_{min}(\mathcal{C})$ is the leftmost uncovered source position. $u(\mathcal{C})$ is the number of "skipped" positions and $m(\mathcal{C})$ is the number of "moved" positions. The function $card(\cdot)$ returns the cardinality of a set of source positions. The function $w(\mathcal{C})$ describes the "window" size in which the word re-ordering takes place.
A procedural description for the computation of the set of successor hypotheses for a given partial hypothesis $(\mathcal{S}, \mathcal{C}, j)$ is given in Table 4.4. There are restrictions on the possible successor states: a partial hypothesis in state *Skip* can **not** be expanded into a partial hypothesis in state *Move* and vice versa. If the coverage set for the newly generated hypothesis covers a contiguous initial block of source positions, the state *Initial* is entered. No other state $\mathcal{S}$ is considered as a successor state in this case (hence the use of the continue statement in the procedural description). The set of successor hypotheses *Succ*, by which to extend the partial hypothesis $(\mathcal{S}, \mathcal{C}, j)$ is computed using the constraints defined by the values for *numskip*, *widthskip*, *nummove*, and *widthmove* as explained in the Appendix A.

In particular, a source position k is discarded for extension if the "window" restrictions are violated. Within the restrictions all possible successors are computed. It can be observed that the set of successors as computed in Table 4.4 is never empty.
There is an asymmetry between the two re-ordering states $Move$ and $Skip$: while being in state $Move$, the algorithm is not allowed to cover the position $l_{min}(\mathcal{C})$. It must first enter the state $Cover$ to do so. In contrast, for the state $Skip$, the newly generated hypothesis always remains in the state $Skip$ (until the state $Initial$ is entered.) This is motivated by the word re-ordering for the German verb group. After the right verbal brace has been processed, no source words may be moved into the verbal brace from later in the sentence.
There is a redundancy in the re-orderings: the same re-ordering might be carried out using either the state $Skip$ or $Move$, especially if $widthskip$ and $widthmove$ are about the same. The additional computational burden is somewhat alleviated by the fact that the pruning as introduced in Section 4.7 does not distinguish hypotheses according to the states. A complexity analysis for different re-ordering constraints is given in Appendix B.

Table 4.4: Procedural description to compute the set $Succ$ of successor hypotheses by which to extend a partial hypothesis $(\mathcal{S}, \mathcal{C}, j)$.

input: partial hypothesis $(\mathcal{S}, \mathcal{C}, j)$

$Succ := \{\emptyset\}$

for each $k \notin \mathcal{C}$ do

- Set $\mathcal{C}' = \mathcal{C} \cup \{k\}$
- if $u(\mathcal{C}') = 0$
  - $Succ := Succ \cup (Initial, \mathcal{C}', k)$
  - continue
- if $(\mathcal{S} = Initial)$ or $(\mathcal{S} = Skip)$
  - if $w(\mathcal{C}') \leq widthskip$ **and** $u(\mathcal{C}') \leq numskip$
    - $Succ := Succ \cup (Skip, \mathcal{C}', k)$
- if $(\mathcal{S} = Initial)$ or $(\mathcal{S} = Move)$
  - if $k \neq l_{min}(\mathcal{C}')$ **and** $w(\mathcal{C}') \leq widthmove$ **and** $m(\mathcal{C}') \leq nummove$
    - $Succ := Succ \cup (Move, \mathcal{C}', k)$
- if $(\mathcal{S} = Move)$ or $(\mathcal{S} = Cover)$
  - if $(l_{min}(\mathcal{C}') = k)$
    - $Succ := Succ \cup (Cover, \mathcal{C}', k)$

output: set $Succ$ of successor hypotheses

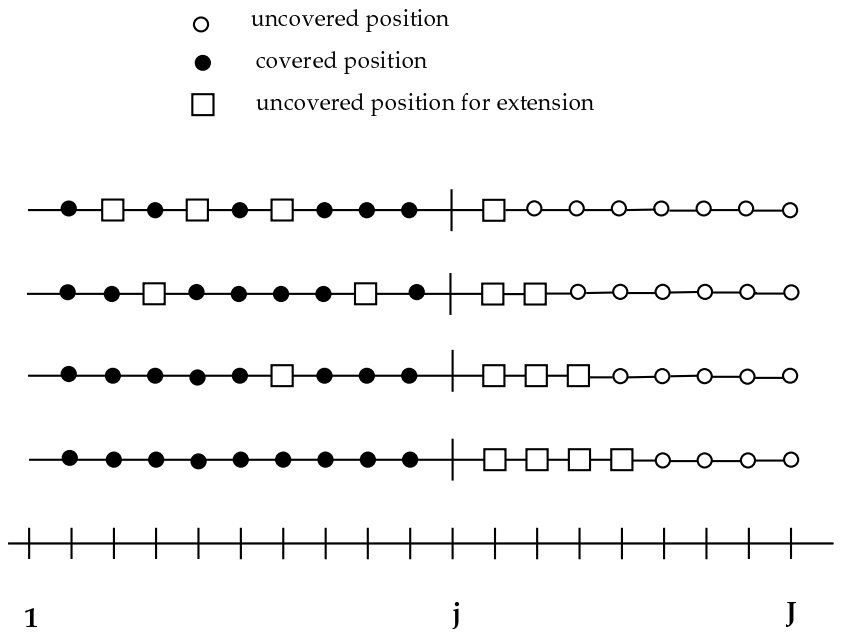


Figure 4.8: Illustration of the IBM-style re-ordering constraint.

4.6.3 Word Re-ordering: IBM Style Restrictions

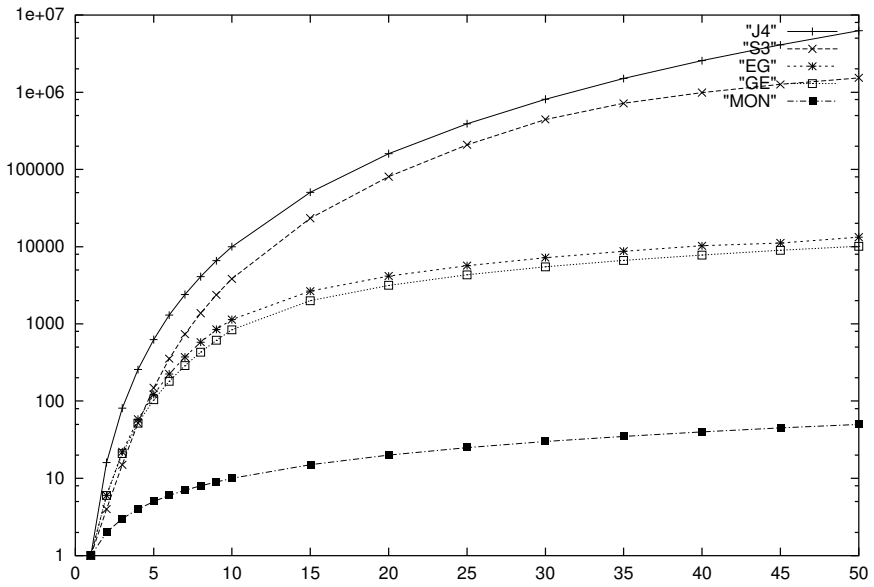
We compare the new word re-ordering approach with the approach used in [Berger et al. 1996]. In this approach, source sentence words are aligned with hypothesized target sentence words [4]. When a source sentence word is aligned, we say its position is covered. During the search process, a partial hypothesis is extended by choosing an uncovered source sentence position, where the choice is restricted. Only one of the first n uncovered positions in a coverage set may be chosen, where n is set to 4. This choice is illustrated in Figure 4.8. Covered positions are marked by a filled circle and uncovered positions are marked by an unfilled circle. Positions that may be covered next are marked by an unfilled square. The restrictions for a coverage set C can be expressed in terms of the expression $u(\mathcal{C})$ defined in the previous section: the number of uncovered source sentence positions to the left of the rightmost covered position. Demanding $u(\mathcal{C}) \leq 3$ we obtain the S3 restriction introduced in the preceding section. An upper bound of $\mathcal{O}(E^3 \cdot J^4)$ for the word re-ordering complexity is given in Appendix B.

4.6.4 Empirical Complexity Calculations

In order to demonstrate the complexity of the proposed re-ordering constraints we have modified our translation algorithm to show the overall number of successor states generated by the algorithm given in Table 4.4 for the different re-ordering constraints. The number of successors shown in Table 4.5 is counted for a pseudo-translation task where a pseudo-source word "x" is translated into the identically pseudo-target word "x". No actual optimization is carried out, just the total number of successors is counted as the algorithm proceeds through

[4] In [Berger et al. 1996], a morphological analysis is carried out and word morphemes are processed during the search. Here, we process only full-form words.

Table 4.5: Number or processed arcs for the pseudo-translation task as a function of the input sentence length J (y axis is given in log scale). The complexity for the four different re-ordering constraints MON, GE, EG, and S3 is given. The complexity of the S3 constraint is close to J^4.



subsets of increasing cardinality. The complexity differences for the different re-ordering constraints result from the different number of coverage subsets $\mathcal{C}$ and corresponding re-ordering states $\mathcal{S}$ allowed. For the different re-ordering constraints we obtain the following results (The abbreviations MON, GE, EG, and S3 are taken from Appendix A):

- MON: For this re-ordering restriction, a partial hypothesis is always extended by the position $l_{min}(\mathcal{C})$, hence the number of processed arcs is J.
- GE,EG: The two re-ordering constraints GE and EG are very similar in terms of complexity - the number of word re-orderings is heavily restricted. Actually, since the distance restrictions (expressed by the variables $widthskip$ and $widthmove$) apply the complexity is linear in the length of the input sentence J.
- S3: The S3 re-ordering constraint has a complexity close to J^4. Since no distance restrictions for the skipped positions apply the overall search space is significantly larger than for the GE or EG restriction.

4.7 Beam Search Pruning Techniques

To speed up the search the so-called beam search strategy is used. There is a direct analogy to the data-driven search organization used in continuous speech recognition (Ney et al. 1992). The full DP search algorithm proceeds cardinality-synchronously over subsets of source sentence positions of increasing cardinality. Using the beam search concept, the search can be focused on the most likely hypotheses. The hypotheses $Q_{e'}(e, \mathcal{C}, j)$ are distinguished according to the coverage set $\mathcal{C}$ with two kinds of pruning based on this coverage set:

1. The **coverage** pruning is carried out separately for each coverage set $\mathcal{C}$.
2. The **cardinality** pruning is carried out jointly for all coverage sets $\mathcal{C}$ with the same cardinality $c = c(\mathcal{C})$.

After the pruning is carried out, we retain for further consideration only hypotheses with a probability close to the maximum probability. The number of surviving hypotheses is controlled by **four** kinds of thresholds:

- The coverage pruning threshold $t_{\mathcal{C}}$
- The coverage histogram threshold $n_{\mathcal{C}}$
- The cardinality pruning threshold t_c
- The cardinality histogram threshold n_c

For the coverage and the cardinality pruning, the probability $Q_{e'}(e, \mathcal{C}, j)$ is adjusted to take into account the uncovered source sentence positions $\bar{\mathcal{C}} = \{1, \cdots, J\} \setminus \mathcal{C}$. To do so, for a source word f at an uncovered source position, we pre-compute an upper bound $\bar{p}(f)$ for the product of language model and lexicon probability:

$$\bar{p}(f) \quad = \quad \max_{e'', e', e} \{p(e|e', e'') \cdot p(f|e)\}.$$

The above optimization is carried out only over the word trigrams (e, e', e'') which have actually been seen in the training data. Aditionally, the observation pruning described below is applied to the possible translations e of a source word f. The upper bound is used in the beam-search concept to increase the comparability between hypotheses covering different coverage sets. Even more benefit from the upper bound $\bar{p}(f)$ can be expected if the distortion and the fertility probabilities are taken into account as shown in Chapter 6. Using the definition of $\bar{p}(f)$, the following modified probability $\bar{Q}_{e'}(e, \mathcal{C}, j)$ is used to replace the original probability $Q_{e'}(e, \mathcal{C}, j)$, where all pruning is applied to the new probability:

$$\bar{Q}_{e'}(e, \mathcal{C}, j) \quad = \quad Q_{e'}(e, \mathcal{C}, j) \cdot \prod_{j \in \bar{\mathcal{C}}} \bar{p}(f_j).$$

For the translation experiments, Eq. 4.3 is recursively evaluated over subsets of source positions of equal cardinality. For reasons of brevity, we omit the state description $\mathcal{S}$ in Eq. 4.3, since no separate pruning according to the states $\mathcal{S}$ is carried out. The set of

surviving hypotheses for each cardinality c is referred to as the so-called beam. The size of the beam for cardinality c depends on the ambiguity of the translation task for that cardinality. To fully exploit the speed-up of the DP beam search, the search space is dynamically constructed as described in [Tillmann et al. 1997c] rather than using a static search space. To carry out the pruning, the maximum probabilities with respect to each coverage set $\mathcal{C}$ and cardinality c are computed:

- Coverage Pruning: Hypotheses are distinguished according to the subset of covered positions $\mathcal{C}$. The probability $\hat{Q}(\mathcal{C})$ is defined:

$$\hat{Q}(\mathcal{C}) = \max_{e,e',j} \bar{Q}_{e'}(e, \mathcal{C}, j)$$

- Cardinality Pruning: Hypotheses are distinguished according to the cardinality $c(\mathcal{C})$ of subsets $\mathcal{C}$ of covered positions. The probability $\hat{Q}(c)$ is defined for all hypotheses with c(C)=c:

$$\hat{Q}(c) = \max_{\substack{\mathcal{C} \\ c(\mathcal{C})=c}} \hat{Q}(\mathcal{C})$$

The coverage pruning threshold $t_{\mathcal{C}}$ and the cardinality pruning threshold t_c are used to prune active hypotheses. We call this pruning **translation** pruning, hypotheses are pruned according their translation probability:

$$\begin{aligned} \bar{Q}_{e'}(e, \mathcal{C}, j) &< t_{\mathcal{C}} \cdot \hat{Q}(\mathcal{C}) \\ \bar{Q}_{e'}(e, \mathcal{C}, j) &< t_c \cdot \hat{Q}(c) \end{aligned}$$

An hypothesis $(e', e, \mathcal{C}, j)$ is discarded if its probability is below the corresponding threshold. For the current experiments, the coverage and the cardinality threshold are constant for different coverage sets $\mathcal{C}$ and cardinalities c. Together with the translation pruning, the **histogram** pruning is carried out: The overall number $N(\mathcal{C})$ of active hypotheses for the coverage set $\mathcal{C}$ and the overall number $N(c)$ of active hypotheses for all sub-sets of a given cardinality may not exceed a given number, where again different numbers are used for coverage and cardinality pruning. The coverage histogram pruning is denoted by $n_{\mathcal{C}}$, and the cardinality histogram pruning is denoted by n_c:

$$\begin{aligned} N(\mathcal{C}) &> n_{\mathcal{C}} \\ N(c) &> n_c \end{aligned}$$

If the numbers of active hypotheses for each coverage set $\mathcal{C}$ and cardinality c, $N(\mathcal{C})$ and $N(c)$, exceed the above threshold, only the partial hypotheses with the highest translation probability are retained , e.g. we may use $n_{\mathcal{C}} = 1\,000$ for the coverage histogram pruning. The third type of pruning used is the so-called **observation** pruning: the number of words that may be produced by a source word f is limited. For each source language word f the list of its possible translations e is sorted according to:

$$p(f|e) \cdot p_{uni}(e),$$

where $p_{uni}(e)$ is the unigram probability of the target language word e. Only the best n_o target words e are hypothesized during the search process, e.g. during the experiments to hypothesize the best $n_o = 50$ words was sufficient. Additionally, a threshold t_o may be used to control the number of hypothesized target words e relatively to $o(f)$:

$$o(f) \quad = \quad \max_e \{p(f|e) \cdot p_{uni}(e)\}$$

Only those target language words e for which $p(f|e) \cdot p_{uni}(e) \geq t_o \cdot o(f)$ are hypothesized during the search process. This type of observation pruning threshold is not applied during the translation experiments since the use of the t_o threshold turned out to be sufficient.

4.8 Accelerated Recombination

Here, we briefly present a method to further speed up the search in the translation process, namely accelerated recombination. For accelerated recombination, the idea is to reduce the number of predecessor words e' to be evaluated in the DP recursion. This can be achieved as follows. Assuming the use of a bigram language model, a partial hypothesis is determined by the tuple $(e, \mathcal{C}, j)$. The edges leading to the tuple $(e, \mathcal{C}, j)$ are processed in a specific order so that the evaluation can be terminated whenever it is sure that no better partial path to $(e, \mathcal{C}, j)$ can be found. By carrying out this procedure for each hypothesis $(e, \mathcal{C}, j)$, the search algorithm is still guaranteed to find the optimal path through the lattice. Although the complexity of the algorithm is not guaranteed to be reduced in the worst case, we obtain significant computational savings in the average case. In the tests reported in [Tillmann et al. 1997c], a speed-up factor of 40 is obtained on the Verbmobil task for the search algorithm presented in Section 4.3.

4.9 Details for IBM-4 Model

In this section, we show how the DP-based beam search approach can be carried over to the full set of IBM-4 parameters. First, the full set of IBM-4 parameters does not make the simplifying assumption that source and target sentences are of equal length made in Section 4.4. Depending on how many target words e are generated by a covered source word f and whether the word is aligned to the "null" word, different subsets out of the full set of parameters for the IBM-4 model are used to compute the probability of an extended partial hypothesis. A single target word may be aligned to several source words (this target word then has fertility greater than 1). Conversely, a single source word may generate $0, 1$ or 2 target words as desribed in [Berger et al. 1996]. 0 target words are generated if f is aligned to the "null" word e_0. Two target words (e', e'') may be generated where we do not use a lexicon probability to generate the first target word e'. The costs for generating the target word e' are given by its fertility $\phi(0|e')$ and the language model probability. When predicting two target words, we will restrict ourselves to triples of target words (e, e', e'') actually seen in the training data. This approach is used for the French-English translation experiments. During the German-English translation experiments the word joining procedure described in Section 3.3 is used instead.

In order to deal with the IBM-4 fertility parameters within the DP-based concept, we adopt the distinction between **open** and **closed** hypotheses given in [Berger et al. 1996]. An hypothesis is said to be open if it is to be aligned with more source positions than it currently is, i.e. at least two. Otherwise it is called closed. Partial hypotheses are distinguished according to the following entries:

- The last two target words u, v of the target string.
- The coverage set $\mathcal{C}$ of already processed ("covered") source sentence positions.
- The last covered source sentence position j.
- Hypotheses are distinguished according to whether they are **open** or **closed**.
- Hypotheses are distinguished according to the current fertility of the "null" word.

The above partial hypothesis struct description for the IBM-4 model is a generalization of the description used in Section 4.6, namely partial hypotheses were charactarized by the tuple $(e', e, \mathcal{C}, j)$ (Here, the notation u, v is used for the target words e', e). To distinguish hypotheses by just two additional components is enough to handle partial hypotheses for the full IBM-4 translation model. Furthermore, the word re-ordering restrictions and the beam search pruning techniques are directly carried over to the full set of IBM-4 parameters since they are based on restrictions on the partial hypothesis coverage vector $\mathcal{C}$ only.
To ensure its correctness, the implementation was tested by carrying out a forced alignments on 500 German-English training sentence pairs. In a forced alignment, the source sentence f_1^J and the target sentence e_1^I are kept fixed and a full search without re-ordering restrictions is carried out only over the unknown alignment a_1^J. The language model probability is divided out and the resulting probability is compared to the Viterbi probability as obtained by the training procedure. For 499 training sentences the Viterbi alignment probability as obtained by the forced alignment search was exactly the same as the one produced by the training procedure. In one case the forced-alignment search did obtain a better Viterbi probability than the training procedure.

4.10 Beam Search Implementation

In this section, we sketch an implementation of the dynamic programming algorithm presented in the previous section. The implementation of the dynamic programming search procedure is given in C++, where we use the so-called STL (Standard Template Library). The STL is a C++ library of container classes, algorithms, and iterators; it provides many of the basic algorithms and data structures of computer science. The STL is build upon the template concept inherent to C++.
The implementation described here is similiar to the one used in speech recognition systems based on dynamic programming as presented in [Ney et al. 1992, Ney 1996]. The similarities are given mainly in the following:

- The implementation is data-driven. Both its time and memory requirements are strictly linear in the number of path hypotheses.
- The search procedure is developed to work most efficiently, when the input sentences are processed mainly monotonically from left to right. The algorithm works cardinality-synchronously, meaning that all the hypotheses that are processed cover subsets of source sentence positions of equal cardinality.
- Since full search is prohibitive, we use a beam search concept as in speech recognition. We use appropriate pruning techniques in connection with the cardinality-synchronous search procedure.

The baseline DP algorithm is given in Table 4.6. It uses the notation introduced in section 4.1. Table 4.7 shows how the DP-concept is applied to a search algorithm for statistical MT. The corresponding C++ implementation is given in Figure 4.8.
In Table 4.6, the DP-search starts in the state $(0, u_0)$. J is the number of the decision steps of the optimization problem. The main loop is over that number J of decision steps. The implementation uses two lists of hyptheses: S_{new} and S [5]. The set S contains the currently active hypotheses. The set S_{new} contains the set of currently generated hypotheses. Sets of hypotheses S that survive the previous decision step are expanded simultaneously at search stage j. The newly generated hypotheses after recombination are collected in S_{new}. The resulting set of hypotheses S_{new} is pruned and the surviving hypotheses are then written into the bookkeeping array.
Table 4.7 shows how the two-list implementation concept is applied to a search algorithm for statistical machine translation. The search procedure processes subsets of covered source sentence positions of increasing cardinality. The search starts in the state $(\$, \$, \{\emptyset\}, 0)$, where \$ denotes the sentence start symbol for the immediate two predecessor words and $\{\emptyset\}$ denotes the empty coverage set, where no source position is covered yet. For the initial search state, the position last covered is set to 0. A set $\mathcal{S}$ of active hypotheses is expanded using lexicon model, language model and alignment model costs. During the partial hypothesis extensions, an anticipated pruning is carried out: hypotheses are discarded before they are considered for recombination. The anticipated pruning is not shown in Table 4.6. After

[5] In principal, the two lists can be replaced by a single list, where the active and newly generated hypotheses are seperated accordingly.

Table 4.6: Two-list implementation of a DP-based search algorithm.

initial set of hypotheses: $\mathcal{S}_{old} = \{(0, u_0)\}$
for each cardinality $j = 1, 2, \cdots, J$ do
$S_{new} = \{\emptyset\}$
for each hypothesis $(j-1, u') \in \mathcal{S}_{old}$ do
Expand $(j-1, u')$ using costs $h(j, u', u)$
Look-up and add or update expanded hypothesis (j, u) in S_{new}
Pruning of hypotheses in S_{new}
Bookkeeping of surviving hypotheses in S_{new}
$S = S_{new}$
traceback:
-get best hypothesis from S -decision sequence from bookkeeping array

the extension of all partial hypotheses in $\mathcal{S}$, a pruning step is carried out for the hypotheses in the newly generated set $\mathcal{S}_{new}$. The partial hypotheses are first sorted according to their translation score and the cardinality pruning is carried out. Then, the partial hypotheses are sorted a second time according to their coverage set $\mathcal{C}$ and their translation score [6]. After this sorting step, all partial hypotheses that cover the same subset of source sentence positions are located in consecutive fragments in the overall list of partial hypotheses. The coverage pruning is carried out in a single run over the overall list of partial hypotheses: for each fragment corresponding to the same coverage set $\mathcal{C}$ the coverage pruning thresholds are applied. The partial hypotheses that survive the two pruning stages are then written into the bookkeeping array. For the next expansion step, the set $\mathcal{S}$ is set to the newly generated list of hypotheses. Finally, the target translation is constructed from the bookkeeping array.

[6]The re-ordering complexity given for the different re-ordering constraints does not take into account the sorting steps used in the implementation.

Table 4.7: Two-list implementation of a DP-based search algorithm for statistical MT.

input: source string $f_1 ... f_j ... f_J$
initial hypothesis lists: $\mathcal{S} = \{(\$, \$, \{\emptyset\}, 0)\}$
for each cardinality $k = 0, 1, 2, \cdots, J$ do
$\mathcal{S}_{new} = \{\emptyset\}$
for each hypothesis $(e', e, \mathcal{C}, j) \in \mathcal{S}$, where $j \in \mathcal{C}$ and $\lvert\mathcal{C}\rvert = k$ do
Expand $(e', e, \mathcal{C}, j)$ using probabilities $p(f_j \lvert e) \cdot p(j \lvert j', J) \cdot p(e \lvert e', e'')$
Look-up and add or update expanded hypothesis in $\mathcal{S}_{new}$
Sort hypotheses in $\mathcal{S}_{new}$ according to translation score and carry out cardinality pruning
Sort hypotheses in $\mathcal{S}_{new}$ according to coverage set $\mathcal{C}$ **and** translation score and carry out coverage pruning
Bookkeeping of surviving hypotheses in $\mathcal{S}_{new}$
$\mathcal{S} := \mathcal{S}_{new}$
traceback:
- find best end hypothesis in $\mathcal{S}$ - recover optimal word sequence

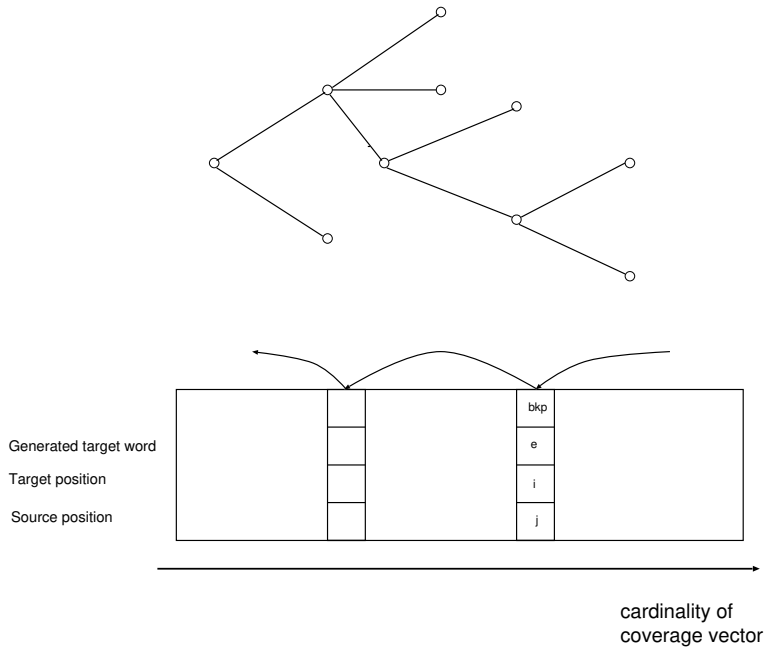


Figure 4.9: Illustration of the traceback procedure for the beam search strategy.

Bookkeeping Procedure

The bookkeeping procedure is sketched in Figure 4.9. For each surviving partial hypothesis in $\mathcal{S}_{new}$, a bookkeeping entry is written into the bookkeeping array. A bookkeeping entry consists out of the following four components:

- source position j
- target position i
- target language word e
- backpointer entry bkp

Here, j is the source sentence position that has been covered by the partial hypotheses. e is the newly generated target language word. i is the the target sentence position to which j is aligned. bkp is the backpointer into the backpointer-array that points to the entry which has been written by the predecessor hypothesis. The backpointers constitute a tree as is shown in Figure 4.9 where each leaf of the tree corresponds to a target sentence hypothesis.

C++ Implementation

The C++ implementation of the two-list implementation concept is given in Table 4.8. Table 4.9 comments on each line of the programming code in Table 4.8. It tries to follow as closely as possible the above description of the two-list algorithm. The baseline class for the dynamic programming procedure is given in Table 4.10. The basic data structures and functions are as follows:

- There are two sets of hypotheses: `set, set_new`. `set` contains the hypotheses that are currently being expanded. Before the pruning `set_new` contains the hypotheses that have survived the recombination. After the pruning `set_new` contains the hypotheses that are used for the expansion the following stage $j + 1$.
- The search object `search` is used to compute costs for transitions between states. A state is defined by a C-struct containing all its components. The C-struct used for the IBM model search is given in Table 4.11. The use of its compenents is explained below. The search object also defines a linear order on the states, which is used to implement the recombination using the STL-container `map`.
- The result object `r` is used to store the hypothesis with the maximal score .
- The two member functions `DP_init` and `DP_search` may be called from outside the DP_SEARCH-class to perform the translation search. Some pre-processing, e.g. dealing with unknown words is done in the procedure `pre_processing`.
- The variable `num_arcs` counts the number of hypothesis expansions within the search process, no matter whether the resulting new hypothesis is recombined or not. However, the number of hypothesis expansions is affected by the anticipated pruning.
- The core of the search procedure is the function `dp_search`. Its details are given in Table 4.8. The function `bookkeeping` is called to perform the bookkeeping. The function `add_sentence_end_score` is called to add the language model score to predict the sentence end symbol $ at the end of the sentence. For the best translation hypothesis the function `traceback` is called to get the target word sequences with the highest translation score.
- The four variables `coverage_thr`, `coverage_histogram , cardinality_thr` and `cardinality_histogram` are used to control the beam search as described in Section 4.7. The corresponding pruning is carried out by the functions `coverage_pruning` and `cardinality_pruning`.

Table 4.11 shows the struct which charactarized a partial hypothesis for the IBM model search. The first six components are explained in Section 4.9. They are used to distinguish partial hypotheses during the recombination process. The partial hypothesis score is stored in `sco`. The pointer into the bookkeeping array is given in `bkp`. The following five components are not used for recombination and could in principle be computed for each partial hypothesis on the fly using the backpointer array when needed during the search process. Instead, they are stored in a partial hypothesis to avoid the computational overhead of generating them from the bookkeeping array. The following five values are stored:

- **i:** The length i of the target sequence generated so far.
- **mon:** The right-most covered source position such that all source position to the left of it are covered as well.
- **expand_closed:** For closed hypotheses: the center value decribed in Section 4.9.
- **fert:** For open hypotheses: the current fertility of the target word v.
- **sum:** For open hypotheses: the sum of the source positions aligned to the target word v. This is used to compute the center value.

Table 4.8: Implementation of dynamic programing: core of the DP beam search procedure.

```
1 : template<class SEARCH, class HYP>
2 : void DP_SEARCH<SEARCH,HYP>::dp_search() {
3 : book.reset(); set.Clear(); set_new.Clear();
4 : set=search.initbeam();
5 : bookkeeping(set);
6 : num_arcs=0;
7 : for (uint j=1;j<=search.len(); j++) {
8 :       for(uint i=0; i<set.getsize(); i++ ) {
9 :            vector<HYP> v = search.expand(set[i]);
10:            for (uint k=0; k<v.size(); k++) {
11:                    num_arcs++;
12:                    if (set_new.lookup(v[k])) {
13:                       if (set_new.sco(v[k]) > v[k].sco) {
14:                          set_new.update(v[k]);
15:                       }
16:                    } else {
17:                            set_new.insert(v[k]);
18:                    }
19:             }
20:       }
21:       set.free();
22:       coverage_pruning(set_new);
23:       bookkeeping(set_new);
24:       set.swap(set_new);
25:    }
26: }
```

Table 4.9: Implementation of dynamic programing: program code lines description for Table 4.8.

Line	Description
3	Initializaton of bookkeeping and hypothesis sets.
4-5	Initialize beam with starting hypothesis.
7	Loop over all `search.len()` stages of the sequential decision problem.
8	Loop over all current partial hypotheses.
9	Expand each partial hypotheses.
12-19	Look-up and add or recombine each new hypothesis.
21	The memory for the expanded hypotheses is freed.
22	The new set of hypotheses is pruned.
23	The bookkeeping for the surviving hypotheses is carried out.
24	set_new becomes set.

Table 4.10: Implementation of the dynamic programming: the baseline class.

```
template <class SEARCH, class HYP>
class DP_SEARCH {
Set<HYP,typename SEARCH::order>  set, set_new;
SEARCH          search;
Book<HYP>       book;
result          r;

vocClass       *ve, *vf;
satz            f_input;    // sentence to translate
uint            num_arcs;   // number of arcs processed

const satz &pre_processing( const string &input );
void dp_search  ();
void bookkeeping(Set<HYP,typename SEARCH::order>& s);
void add_sentence_end_score(Set<HYP,typename SEARCH::order>& s);
void traceback();
// PRUNING
double  coverage_thr;
uint    coverage_histogram;
double  cardinality_thr;
uint    cardinality_histogram;

void coverage_pruning   ( Set<HYP,typename SEARCH::order> &s );
void cardinality_pruning( Set<HYP,typename SEARCH::order> &s );
public:
void DP_init ( const param &p, vocClass &ve, vocClass &vf );
const   result DP_search( const char input[] );
};
```

Table 4.11: Implementation of dynamic programming: struct representing a partial hypothesis for the IBM-4 model search.

```
struct hypothesis {
 WIdx               v,u;        // predecessor words v,u
 bitset<MAXBITSET>  set;        // coverage set
 ushort             last;       // last covered position
 char               open;       // open / closed hypothesis
 int                emptywords;// number of words aligned to
                                // the null word

 double             sco;        // hypothesis score
 int                bkp;        // backpointer to bookkeeping entry

 int                expand_closed; // used to compute the center value
 ushort             i;             // length of the target string
                                   // in partial hypothesis
 ushort             mon;           // initial block of covered source
                                   // positions up to position mon
 char               fert;          // current fertility of v
 int                sum;           // sum of source sentence
                                   // positions aligned to target
                                   // word v
};
```

Chapter 5

Experimental Results

In this section, we present experimental results for the techniques presented in this thesis. Examples of selected trigger pairs are given in Section 5.1. Examples for joined target language word sequences are given in Section 5.2. Details on the training procedure and the translation search procedure are given in Section 5.3 and in Section 5.4. The performance measures used are introduced in Section 5.5. Translation experiments are carried out for several different translation directions as reported from Section 5.6 to Section 5.8: for the translation directions German-to-English and English-to-German (Verbmobil task), for the translation directions Italian-to-English and English-to-Italian (FUB task), and for the translation directions French-to-English and English-to-French (Canadian Hansards task). The difficulty of the translation task varies for the different translation directions in terms of vocabulary sizes and amount of training data. The most detailed experiments are carried out on the German-English Verbmobil corpus.

5.1 Extended Language Model: Trigger Pairs

In this section, we present experimental results for the different trigger pair selection criteria presented in Chapter 3.2. Results for the following selection criteria are presented:

A:	unigram level selection criterion	in Eq. 3.9
B:	high level selection criterion	in Eq. 3.3
C:	low level selection criterion	in Eq. 3.10.

The experimental tests are performed on the the Wall Street Journal (WSJ) task [Paul et al. 1992] for a vocabulary size of $V = 20\,000$. In Section 5.1.1, we present samples of the selected trigger pairs for the three selection criteria. These samples are chosen because we deem them typical for the three criteria. In Section 5.1.2, we present perplexity results for the combination of trigger pairs with a baseline language model. For the baseline language model, there are three training corpora with sizes of $1, 5$ and 39 million running words. Unlike the baseline language models, the word trigger pairs are *always* selected from the 39-million word training corpus. Finally, in Section 5.1.3 we draw conclusions on the use of trigger pairs in language modeling. The results are taken from [Tillmann et al. 1996a, Tillmann et al. 1997b].

5.1.1 Trigger Pair Examples

For a vocabulary size of V words, there are V^2 possible word trigger pairs, therefore only trigger pairs that are observed at least 3 times in the training corpus are used to calculate the perplexity improvement using the different criteria. For the unigram level triggers and the low level triggers, the calculation of all possible trigger pairs took a maximum of 6 hours (on a R4000 SGI workstation). For the high level triggers, the computation time is dominated by the time for calculating $S(b)$ in Eq. 3.6 using the sampling approximation; typically it took 48 hours when every 20th position is sampled. Table 5.1 shows those trigger pairs that yield the maximum likelihood improvement for each of the three selection criteria. In each case, it is found that the word trigger pairs are dominated by the self-triggers and the so-called same-root triggers. A trigger pair $a \rightarrow b$ is a self-trigger pair if $a = b$, i.e. it is equivalent to the cache effect. Same-root triggers are trigger pairs $a \rightarrow b$, where the two words a and b differ only by an ending s-character, e.g. the possessive and plural forms for nouns. To single out the really interesting word trigger pairs, the self-triggers and same-root triggers are removed. To illustrate the effect of these self-triggers and same-root triggers, the first column of Table 5.1 shows the rank of each trigger pair within the original list. The second column presents the perplexity improvement ΔPP of the extended model over the baseline model. In addition, Table 5.1 gives the four counts defined in Eq. 3.4, where, of course, for the low level selection criterion, only the reduced corpus (resulting in 1.8 million positions) is used.
For selecting the trigger pairs, the baseline model used is a unigram or a bigram language model with cache. For the trigger pair selecting procedure, a bigram language model is used as an approximation to the trigram language model due to computational restrictions. For the perplexity results using the trigger pairs a trigram language model with cache is used instead [Ney et al. 1995]. Table 5.2 gives another representation of the best trigger pairs. For a chosen set of triggering words a, Table 5.2 shows the best triggered words b. The words b are ordered by decreasing perplexity improvement. For each of the three selection criteria, the trigger pairs are taken from a list of the best 500 000 trigger pairs.
We now discuss the selection criteria in more detail. All in all, the results for the high level triggers are less satisfactory than for the low level triggers, but there are some interesting facts to note. There is un unsymmetry as for the trigger pair "Fe $\rightarrow$ Santa", which has a high ranking position. The inverse trigger pair "Santa $\rightarrow$ Fe" did not get a high ranking, because "Fe" is already well predicted by the bigram model in the case of predecessor word "Santa". In Table 5.1, the high level triggers shown consist of proper names only; all of these trigger pairs seem reasonable considering the financial domain of the WSJ corpus. However, the situation is different when looking at Table 5.2. Here, some of the high level triggers (i.e. criterion B) are more difficult to understand. An interesting fact to note for high level triggers is that only 3000 out of V^2 possible trigger pairs are able to improve a given bigram/cache model. In both Tables 5.1 and 5.2, the low level trigger pairs yield the best overall result of the three selection criteria. Some words produce very interesting trigger pairs, e.g. the triggering words "asked" and "says" in Table 5.2. These verbs mostly trigger verbs again, which even have the same tense. Additional interesting examples are the triggering words "airlines" and "Ford" in Table 5.2, where the triggered words are airlines or car models produced by the "Ford" company. The corresponding unigram level

Table 5.1: Best word trigger pairs along with perplexity improvement (ΔPP) for the three selection criteria A, B and C (self triggers and same-root triggers excluded).

	Rank	ΔPP	a	b	$N(a,b)$	$N(a,\overline{b})$	$N(\overline{a},b)$	$N(\overline{a},\overline{b})$
A	3	-2.22	the	a	839783	31263065	6175	3901023
	4	-2.21	a	share	15107	33430899	39833	2524207
	5	-1.75	in	nineteen	72010	33119066	54615	2764355
	11	-1.45	point	dollars	174009	14577007	66658	21192372
	12	-1.44	of	the	1793280	31921904	246783	2048079
	13	-1.41	the	company	75945	32026903	58876	3848322
	14	-1.29	the	U.	49630	32053218	47096	3860102
	16	-1.22	a	the	1985329	31460677	54734	2509306
	17	-1.17	the	of	767430	31335418	197787	3709411
	18	-1.10	percent	point	149707	12327082	92944	23440313
	19	-1.06	to	be	112112	33569246	44121	2284567
	20	-1.03	the	S.	80343	32022505	50732	3856466
	26	-0.96	the	company's	4693	32098155	19640	3887558
	27	-0.95	rose	point	65694	3275140	176957	32492255
B	18	-0.0103	Texaco	Pennzoil	1423	294204	433	35713986
	19	-0.0102	Pennzoil	Texaco	1911	152412	2312	35853411
	30	-0.0074	Fe	Santa	1111	95276	1379	35912280
	34	-0.0071	distillers	Guinness	835	79004	802	35929405
	38	-0.0064	Am	Pan	1241	346056	975	35661774
	41	-0.0062	Campeau	Federated	844	134468	542	35874192
	45	-0.0061	Cola	Coca	807	144817	634	35863788
	64	-0.0051	oil	Opec	2274	2138246	221	33869305
	72	-0.0048	Federated	Campeau	941	129385	856	35878864
	107	-0.0039	multiples	negotiable	367	54612	86	35954981
	130	-0.0035	Geller	Lord	494	26838	652	35982062
	131	-0.0035	Beazer	Koppers	262	25132	131	35984521
	137	-0.0034	soviet	Moscow	1712	1173777	663	34833894
	163	-0.0031	rales	Interco	243	22795	147	35986861
	165	-0.0031	Eddie	crazy	478	67269	565	35941734
C	1	-0.00371	neither	nor	411	28775	567	1853529
	14	-0.00109	tip	iceberg	55	4944	4	1878279
	15	-0.00107	soviet	Moscow's	119	80652	26	1802485
	26	-0.00101	named	succeeds	147	63692	164	1819279
	27	-0.00100	Iraq	Baghdad	74	13766	45	1869397
	33	-0.00093	Eastman	Kodak's	49	3919	16	1879298
	40	-0.00090	Eastman	photographic	55	3913	61	1879253
	43	-0.00089	Carbide	Danbury	51	3350	46	1879835
	50	-0.00088	Eurodollar	syndication	60	3758	139	1879325
	55	-0.00086	filed	alleges	103	52441	80	1830658
	57	-0.00085	asked	replied	120	67419	110	1815633
	60	-0.00085	Kodak	photographic	57	6367	59	1876799
	68	-0.00083	motor	Ford's	74	25221	47	1857940
	71	-0.00083	South	Pretoria	87	71047	18	1812130

Table 5.2: List of best triggered words b for some triggering words a for the selection criteria A, B and C.

a	b	
asked	A:	point replied Mr. I he percent asked one seven eight
	B:	Deltona Prism Benequity Taiyo Ropak Genesis Quintessential Envirodyne target's
	C:	replied answered responded refused replies responses reply yes request requesting
airlines	A:	airlines airline air passenger fares carriers traffic flights miles continental
	B:	Delta's Northwest's Maxsaver Transtar Swissair Primark United's Motown
	C:	American's passengers Airlines' Eastern's United's hubs fares Northwest's carriers
buy	A:	buy shares stock dollars company price offer million share stake
	B:	Sheller Deltona Motown Northview Barren Philipp Selkirk Oshkosh Radnor
	C:	repurchased Landover purchases repurchases Kohlberg repurchase Southland's
concerto	A:	orchestra concerto music symphony piano violin philharmonic ballet composer
	B:	Mozart violin Bach poignant
	C:	strings orchestra violin score Mozart pianist recordings keyboard listen variations
Ford	A:	Ford Ford's cars auto Chrysler car G. Jaguar models M.
	B:	Ford Ford's Edsel ambulances Dearborn Jaguar Bronco Mustang Jaguar's Sheller
	C:	Ford's Dearborn Bronco Taurus Escort Chrysler's Tempo Mustang Thunderbird
love	A:	her love she point his I said dollars percent You
	B:	Genex polly soothing boyish pathetic authenticity quaint Horace chalk Domino's
	C:	beautifully passion sweet sexy romantic hero pop lovers pale wit
Microsoft	A:	Microsoft software Lotus computer Microsoft's Apple computers personal O. one
	B:	Microsoft Microsoft's Borland Ashton Lotus's Adobe Oracle Redmond Novell
	C:	Microsoft's Redmond Apple's Borland spreadsheets Ashton Lotus's database
says	A:	says said point million dollars adds seven he five one
	B:	Benham Barren accredited Philipp Panasonic Radnor Deltona kids' Battelle
	C:	concedes explains adds agrees recalls asks insists acknowledges asserts predicts

triggers look worse for the same verbs, but for some nouns as triggering words, the triggered words seem to be more general. In addition, we emphasize the following results for the low level triggers:

- Among the best low level triggers are nouns that trigger their possessives, while self triggers do not occur at all.
- In another experiment, we found that the low-level triggers are basically the same as those obtained by the following criterion: only those corpus positions n are used where the count of the bigram (w_{n-1}, w_n) is exactly 1 and where w_n is not in the cache, i.e. in the history h_n.
- If we confine the history to the current sentence, we get trigger pairs related to grammatical structure, e.g. "I $\rightarrow$ myself" "We $\rightarrow$ ourselves" These results can be compared to the link grammar results in [Della Pietra et al. 1994], where long-distant word pairs (and triples) are used to find the lexicalized grammar rules.

As the experimental tests reported in the following will show, the low level triggers are also able to improve the perplexity of a trigger/cache baseline model.

5.1.2 Trigger Pair Perplexity Results

In this section, we present perplexity results for the trigger pairs. The trigger pairs are selected as described in the preceding section and are used to extend a baseline language model, which is a trigram/cache model. We have used the following model to incorporate the selected trigger pairs into a full language model and to define an extended language model $p_E(w_n|h_n)$:

$$p_E(w_n|h_n) \quad = \quad (1-\lambda_1-\lambda_2)\cdot p_S(w_n|h_n) + \lambda_1 \cdot p_C(w_n|h_n) + \lambda_2 \cdot p_T(w_n|h_n) \quad ,$$

where the history $h_n = w_{n-M}^{n-1}$ consists of the M predecessor words of w_n. The cache model $p_C(w_n|w_{n-M}^{n-1})$ is defined as:

$$p_C(w_n|w_{n-M}^{n-1}) \quad = \quad \frac{1}{M}\sum_{m=1}^{M}\delta(w_n, w_{n-m}) \quad ,$$

with $\delta(w,v)=1$ if and only if $w=v$. The trigger model $p_T(w_n|h_n)$ is defined as:

$$p_T(w_n|w_{n-M}^{n-1}) \quad = \quad \frac{1}{M}\sum_{m=1}^{M}\alpha(w_n|w_{n-m}) \quad .$$

The $\alpha(b|a)$ are obtained by renormalization:

$$\alpha(b|a) \quad = \quad \frac{q(b|a)}{\sum\limits_{b'} q(b'|a)},$$

where the interaction parameters $q(b|a)$ are the maximum likelihood estimates given by Eq. 3.8. In the experiments, the history h is defined to start with the most recent article delimiter. The baseline trigram model is a backing-off model described in [Ney et al. 1995]. Although the interpolation parameters λ_i in Eq. 5.1 can be trained by the EM procedure [Dempster et al. 1977, Jelinek 1991], they are adjusted by trial and error in informal experiments. For the different extended language models, the perplexity is computed on a test corpus of 325 000 words from the WSJ task. In [Tillmann et al. 1997b], the use of the EM algorithm is demonstrated to obtain slightly improved perplexity results.

In a first experiment, we try to improve a unigram model with the unigram level triggers and with the low level triggers. The unigram model is trained on the 5-million corpus. We use the 500 000 best trigger pairs for the low level and unigram level selection criterion. The resulting perplexities are presented in Table 5.3. The unigram level triggers improve the perplexity to a much higher extent than the low level triggers as can be expected from the corresponding selection criteria. In a second experiment, the baseline language model is a trigram/cache model which is extended by different sets of trigger pairs. The perplexity results are given in Table 5.4. For comparison purposes, the effect of the cache component in the baseline model is studied, too. Table 5.4 shows the perplexities for the following conditions: trigram with no cache, trigram/cache and its extensions using a set of 1.5 million

unigram level trigger pairs, a set of 0.5 million low level trigger pairs and a set of 1.5 million low level trigger pairs. By far the best results are obtained for the low level trigger pairs. Increasing the number of low level trigger pairs from 0.5 million to 1.5 million improves the perplexity only slightly.
As expected, the combination of the low level trigger pairs produces the best perplexity improvement using the model defined in Eq. 5.1. The problem with all the selection criteria presented is that the combination of the selected trigger pairs into one global language model is not captured by any of the criteria. However, the low level criterion provides a better approximation to the use of the trigger pairs in Eq. 5.1. As opposed to the low level triggers, the high level triggers are never found to result in perplexity improvements because the selection criterion as defined by Eq.(3.3) is not consistent with the extended model defined in Eq. 5.1.

5.1.3 Conclusions on the Use of Word Trigger Pairs

In this section, we have described the use of word trigger pairs for language modeling. We have shown new methods for finding word trigger pairs: given a baseline language model to start with, we extend it by including one word trigger pair at a time and compute the perplexity improvement of this extended model over the baseline model. This perplexity improvement is used to select the most important word trigger pairs. For the special case that the baseline language model is a unigram model, this new method results in the well-known mutual information criterion. The more interesting case is that the trigger pair under consideration is integrated at a lower level of the baseline language model, which leads to the so-called low level selection criterion. Instead of using some more or less arbitrary ad hoc selection criterion, we have presented a new method for finding word trigger pairs: In the experimental tests, we found the following results:

1. The low level selection criterion produces intuitively better word trigger pairs than the usual mutual information criterion.

2. When combined with a full baseline language model, namely a trigram/cache model, the low level triggers reduce the perplexity from 138 to 128 for the 5-million training set and from 92 to 87 for the 39-million training set. In comparison, when using the conventional mutual information criterion, the perplexity improvements are significantly smaller.

Table 5.3: Perplexity results for the extension of a unigram language model (5-million training words) with three different components: low level triggers, unigram triggers and cache.

model	5 Mio
unigram	1027
+ low level triggers	960
+ unigram level triggers	860
+ cache	750

Table 5.4: Perplexity results for the combination of trigger pairs with a trigram/cache language model (1, 5 and 39 million training words).

model	Number of Triggers	1 Mio	5 Mio	39 Mio
trigram with no cache		252	168	105
trigram/cache		197	138	92
+ unigram level triggers	1500000	191	135	91
+ low level triggers	500000	182	130	88
+ low level triggers	1500000	180	128	87

5.2 Extended Lexicon Model: Target Word Joining

In this section, we present examples for target word extensions that are learned from the bilingual aligned training data using the algorithm shown in Table 3.2. The word joinings are carried out in the original target training corpus to obtain a modified target corpus. In Table 5.5 we list the overall best extensions sorted by the likelihood improvement $F_{\tilde{e}_0} - F_{(f,\tilde{e})}$. The details of the selection criterion are presented in Section 3.3. In Table 5.5 and Table 5.6 we use the following notation:

- The first column shows the log-likelihood improvement $F_{(f,\tilde{e})} - F_{\tilde{e}_0}$ of the extended model over the baseline model.
- The source word f is aligned to its main translation $\tilde{e}_0$ on the Viterbi training path.
- The target word e_1 follows the target word $\tilde{e}_0$ in the target sentence.
- The target word e_{-1} preceedes the target word $\tilde{e}_0$ in the target sentence.
- The target word e_{-2} preceedes the target word $\tilde{e}_0$ in the target sentence by two positions.
- An empty position in a column means that no target word is chosen within the extension at this position in the target sentence.

The top extensions shown in Table 5.5, namely ("zum","to the" and ("müssen","have to") are grammatically meaningful, since the German "zum" is a short form of the two German words "zu dem" and "müssen" followed by an infinitive is translated as "have to" followed by an infinitive in English. The lexicon extension further down in the list in most cases seem to correspond to fragements of larger phrases. The second Table 5.6 shows the lexicon extensions where the predicted source word f starts with an upper-case letter. The selected German words are often compound nouns that correspond to an English noun phrase that consists of several English nouns. A single German word may be mapped to up to three target words, e.g. the extension ("Maiwoche","week of may"). Again, some extensions seem to be fragements from larger mappings between source and target words, e.g. the extension ("Ihnen","suit you") which is part of the mapping between the German phrase "wie würde es Ihnen passen" and the English phrase "how would it suit you".

Table 5.5: List of best target word joinings sorted by the log-likelihood improvement $F_{(f,\tilde{e})} - F_{\tilde{e}_0}$ as obtained on the aligned Verbmobil training corpus using the word joining algorithm shown in Table 3.2.

$F_{(f,\tilde{e})} - F_{\tilde{e}_0}$	f	e_{-1}	$\tilde{e}_0$	e_1
-1291.02	zum	to	the	
-1107.28	müssen		have	to
-865.66	muß		have	to
-776.448	müßten		have	to
-769.864	da	I	have	
-748.993	ich	I	have	
-744.827	bin	I	have	
-741.848	mir	I	have	
-736.209	wäre		would	be
-701.24	hätte	I	would	
-683.904	bin	I	will	
-682.832	werde	I	will	
-666.873	müßte		have	to
-657.21	brauchen		have	to
-657.21	muß		have	to
-613.087	vom		from	the
-595.902	gibt	there	is	
-594.563	wir	I	will	
-594.486	dann	I	will	
-591.956	mir	I	will	
-591.913	da	I	will	
-549.744	gerne		like	to
-549.731	kann	I	can	
-542.464	möchte	I	would	
-525.141	zwar	I	would	
-524.746	mir	I	would	
-494.826	bis		you	see
-463.731	einundzwanzigsten	twenty	first	
-429.353	gern		like	to

Table 5.6: List of best target word joinings where the predicted source word f starts with an upper-case letter. The extensions are sorted by the log-likelihood improvement $F_{(f,\tilde{e})} - F_{\tilde{e}_0}$ as obtained on the aligned Verbmobil training corpus using algorithm 3.2 (The German word "Nachmittag" is abbreviated nach. in compound nouns).

$F_{(f,\tilde{e})} - F_{\tilde{e}_0}$	f	e_{-2}	e_{-1}	$\tilde{e}_0$	e_1
-387.059	Wiederhören			see	you
-386.419	Wiederschauen			see	you
-264.725	Vormittagstermin	in	the	morning	
-173.002	Geschäftsreise			business	trip
-162.83	Wiedersehen		good	bye	
-153.956	Dienstreise			business	trip
-148.958	Nachmittagsbereich		the	afternoon	
-148.72	Geschäftsfahrt			business	trip
-126.826	Wiederschauen			good	bye
-115.658	Wiederschauen		good	bye	
-105.997	Wiederhören		good	bye	
-104.429	Konferenzraum		conference	room	
-102.71	Dank	you	very	much	
-97.6821	Dank		very	much	
-93.444	Abendtreffen		the	evening	
-92.896	Ende			end	of
-91.1279	Ihnen			suit	you
-83.5556	Wochendseminar		weekend	seminar	
-83.5223	Karfreitag		good	Friday	
-82.8975	Hauptbahnhof			central	station
-75.6793	Geschäftsessen		business	diner	
-74.3292	Donnerstagnach.		Thursday	afternoon	
-71.2007	Januarwoche	week	of	January	
-71.1551	Dienstagnach.		Tuesday	afternoon	
-69.153	Ende		the	end	
-68.4362	Ihnen		would	suggest	
-68.4063	Vorschlag		would	suggest	
-67.1598	Novemberwoche	week	of	November	
-64.9799	Freitagmorgen			Friday	morning
-63.5067	Montagnach.		Monday	afternoon	
-63.152	Mittwochnach.		Wednesday	afternoon	
-62.6842	Maiwoche	week	of	May	
-59.627	Märzwoche	week	of	March	
-59.3157	Juliwoche	week	of	July	
-59.0109	Vorschlag		my	suggestion	

5.3 Details on the IBM Model Training

The details of the training procedure used in the experiments are given in [Al-Onaizan et al. 1999, Och et al. 2000b]. The training procedure is essentially the training procedure given in [Brown et al. 1993]. A sequence of four models of increasing complexity is trained (The model 5 training is not used during the translation experiments). The following main changes are carried out to the training procedure given in [Brown et al. 1993]:

- Instead of the IBM-2 iteration step of the original IBM model training an hidden Markov model training step is carried out. Hidden Markov models for statistical translation were introduced in [Vogel et al. 1996]. Deviating from that work, the HMM training in [Och et al. 2000a] is not carried out in the maximum approximation and the "empty" target word is introduced into the translation model as well. Additionally, the HMM alignment probabilities are made class-dependent.

- Some smoothing for the lexicon probabilities and the distortion probabilities is done which was not mentioned in [Brown et al. 1993].

- The auxiliary parameter p_0 to compute the fertility of the empty word is held fixed during the IBM model training: $p_0 = 0.98$. In the original IBM training, it was re-estimated.

- Some simple heuristics are applied, e.g. for the translation direction English-to-German a coupling between the character string length and the fertility of a German word is introduced. German words that are longer in terms of character string length turn out to have a higher fertility.

In order to estimate the usefulness of the above methods, a measure for the alignment quality as produced by a certain training procedure after the final training iteration is introduced. The Viterbi alignment as obtained by a certain training procedure is compared to a hand-aligned training corpus. The above methods have been proven to lower alignment error rates as shown in [Och et al. 2000a].
Table 5.7 shows the number of training iterations that are carried out during the IBM model training. No model 5 training step is carried out during the translation experiments since the the training procedure performance as measured by the alignment error rate is not improved. A typical training on the Verbmobil task takes about 1.5 hours.

5.4 Details on the Translation Procedure

Before presenting the experimental results on the different training corpora, we give a complete list of all parameters used for the statistical single-word based approach used in this work. Details on the use of the IBM-4 model parameters are given in Section 1.3.2 and in Section 4.9.
In Table 5.8 the list of parameters that are trained automatically from the bilingual training data is given. Table 5.8, Table 5.9, Table 5.10, and Table 5.10 show parameters and

Table 5.7: Number of training iterations for the IBM model training as is described in [Brown et al. 1993, Och et al. 2000a]. The IBM-2 training step of the original IBM model training is replaced by an hidden Markov model training step.

Translation Model	#iterations
IBM-1	5
HMM	5
IBM-3	4
IBM-4	3

Table 5.8: Complete list of IBM model parameters that are automatically trained on the training corpora.

Type of parameter	Symbol
Lexicon probabilities	$p(f\|e)$
Fertility probabilities	$\phi(e)$
Distortion probabilities	$p(j - \bar{\pi}(i-1)\|\mathcal{F}(f_j), \mathcal{E}(e))$
Trigram language model probabilities	$p(e\|e', e'')$
Parameter to compute the fertility of empty word	p_0
Class mapping	$\mathcal{E}$,$\mathcal{F}$

parameter values that are used to control the speed and the re-ordering performance of the search algorithm. These parameter values are either kept fixed to certain values according to informal experiments not reported in this work or are optimized on held-out test data. In the case of the Verbmobil corpus the parameters are optimized on held-out test data and additional evaluation is carried out on a second set of test data. The notation is as follows: scaling factors are denoted by the greek letter α, linear interpolation factors are denoted by λ's, and thresholds by the letter τ. Subscripts are used to distinguish the factors according to their use. The subscripts may be omitted in order to shorten the description.

Appendix D shows a parameter file used to control the search procedure. There are three types of re-ordering schemata that can be selected as shown in Table 5.9. Actually, the type MR_SKIP4 yields identical results as the type MR_QMS4 with QMS_STRING S_3_99_M_0_xx but its implementation is kept due to comparison reasons. The parameter *ScanType* allows typing in sentences from standard input. The parameter *SplitPunctuation* determines whether word re-ordering is restricted to occur within source sentence fragments

Table 5.9: Parameters for the search procedure to control the re-ordering restrictions.

MR_TRI4	Search procedure with monotonicity constraint
MR_SKIP4	Search procedure with IBM-style re-ordering restrictions
MR_QMS4	Search procedure with quasi-monotone re-ordering restrictions as given in QMS_STRING (cf. Appendix A)

Table 5.10: Complete list of pruning parameters to control the beam width within the beam search concept.

Type of parameter	Symbol
Coverage translation pruning	t_C
Coverage historgram pruning	n_C
Cardinality translation pruning	t_c
Cardinality translation pruning	n_c
Observation histogram pruning	n_o
Observation pruning	t_o

that are determined by punctuation marks. This turned out to improve translation performance in some cases. The parameter *ScanFile* is the name of the input file. A list of data files containing the distributions listed up in Table 5.8 is given in the following lines. If the parameter *ManualLexicon* is given, a manually generated lexicon is linearly interpolated with the trained lexicon using a fixed interpolation parameter $\lambda_M = 0.5$ (see next section). The lexicon probabilities (after the linear interpolation is carried out) may be smoothed using the linear interpolation factor λ_T. This kind of smoothing is not used for the translation experiments reported here. When carrying out a free search, target words e whose lexicon probability $p(f|e)$ is zero for all source words f cannot be generated. A *Labeling* for the target and source language may be applied as described in the following section. A language model scaling factor α_{LM} and a distortion model scaling factor α_D may be set using the parameters *LMScale* and *InvScale*.
Six parameters are used to control the pruning as described in Section 4.7. Four parameters are used for coverage and cardinality histogram pruning, both translation and histogram pruning. Two ways of observation pruning are implemented: The observation histogram pruning is controled by the parameter *ObservationHistogram*. The observation threshold pruning which is actually not used during the experiments is controled by the parameter *ObservationFactor*.

5.5 Performance Measures for Evaluation

To measure the performance of the translation methods, we need an automatic and easy-to-use measure of the translation errors. Although such an automatic measure is necessarily very crude, it seems to be sufficient in the early phases of the system development when the translation task is very limited and/or the performance is poor anyway. In the experiments, we will always use a set of bilingual test sentences so that, for each source sentence, an associated target sentence is given. We use the following three error criteria:

- 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 string into the target string. This performance criterion is widely used in speech recognition. This minimum is computed using a dynamic programming algorithm and is typically referred to as

edit or *Levenshtein* distance.

- PER: position-independent word error rate:
  A shortcoming of the WER is the fact that a perfect word ordering is required. This is particularly a problem for the Verbmobil task, where the word order of the German-to-English sentence pair can be quite different. As a result, the word order of the automatically generated target sentence can be different from that of the target sentence, but nevertheless acceptable so that the WER measure alone could be misleading. In order to overcome this problem, we introduce as additional measure the position-independent word error rate (PER). This measure compares the words in the two sentences *without* taking the word order into account. Words that have no matching counterparts are counted as substitution errors. Depending on whether the translated sentence is longer or shorter than the target translation, the remaining words result in either insertion or deletion errors. The PER is guaranteed to be less than or equal to the WER.

- m-WER: multi-reference WER:
  We use the Levenshtein distance between the automatic translation and several reference translations as a measure of the translation errors. E.g. on the Verbmobil TEST-331 test set on average 6 reference translations per automatic translation are available. The Levenshtein distance between the automatic translation and each of the reference translations is computed, and the minimum Levenshtein distance is taken. This measure is more robust than the word error rate which takes into account only a single reference translation.

- SSER: subjective sentence error rate:
  For a more detailed analysis, subjective judgments by test persons are more appropriate [Niessen et al. 2000b]. The following scale for the error count per sentence is used:

0.0	:	semantically correct and syntactically correct
...	:	...
0.5	:	semantically correct and syntactically wrong
...	:	...
1.0	:	semantically wrong (independent of syntax)

  Each translated sentence is judged by a human examiner according to the above error scale, where several human judges might have been involved. The subjective evaluation is carried out only for the Verbmobil TEST-147 test set.

Table 5.11: Training and test conditions on the German-English Verbmobil corpus (*number of words without punctuation).

		German	English
Training:	Sentences	58 073	
	Words	519 523	549 921
	Words*	418 979	453 632
Vocabulary	Size	7 911	4 648
	Singletons	3 453	1 699
TEST-331:	Sentences	331	
	Words	5 591	6 279
	Bi./Tri. Perplexity	84.0/68.2	49.3/38.3
TEST-147:	Sentences	147	
	Words	1 968	2 173
	Bi./Tri. Perplexity	–	34.6/28.1
TEST-251:	Sentences	251	
	Words	2 627	2 866
	Bi./Tri. Perp.	-	39.3/30.7

5.6 Experiments on the German-English Verbmobil Corpus

The Task and the Corpus

The translation system is tested on the "Verbmobil Task" [Wahlster 2000]. The Verbmobil task is an appointment scheduling task, where two subjects are each given a calendar and asked to schedule a meeting. We carry out experiments for both translation directions: German-to-English and English-to-German. Although the Verbmobil task is still a limited-domain task, it is rather difficult in terms of vocabulary size namely about 5 000 words or more for each of the two languages; second, the syntactic structures of the sentences are rather unrestricted. Although the ultimate goal of the Verbmobil project is the translation of *spoken* language, the input used for the translation experiments reported on in this work is mainly the (more or less) correct orthographic transcription of the spoken sentences. Thus, the effects of spontaneous speech are present in the corpus, however the effect of speech recognition errors is not covered. The corpus consists of 58 073 training pairs. The corpus charactaristics are given in Table 5.11, where no word categorization or word joining is applied. For the translation experiments, a trigram language model with a perplexity of 28.1 is used.

The Test Data

For the translation experiments, we use three different test corpora:

TEST-331 This test set consists of 331 test sentences. Only automatic evaluation is

carried out on this test corpus: the word error rate (WER) and the multi-reference word error rate (m-WER) are computed. For each test source sentence there is a range of acceptable reference translations, namely 6 reference translations on the average. They are provided by a human translator, who is asked to produce word-to-word translations wherever it is possible. Partly, the reference sentences are obtained by correcting automatic translations which have been obtained by using different re-ordering constraints on the same test data. The TEST-331 test set is used as held-out data for parameter optimization (for the language mode scaling factor and for the distortion model scaling factor). Furthermore, the beam search experiments where the effect of the different pruning thresholds is demonstrated are carried out on the TEST-331 test set.

TEST-147 The second, separate test set consists of 147 test sentences. Translation results are given in terms of multi-reference word error rate (m-WER) and subjective sentence error rate (SSER). No parameter optimization is carried out on the TEST-147 test set - the parameter values as obtained from the experiments on the TEST-331 test set are used.

TEST-251 Another separate test set consists of 251 test sentences. Translation results are given in terms of word error rate (WER) and position-independent word error rate (PER). It is used for direct comparison of different translation approaches: namely, for comparison with the alignment template approach [Ney et al. 2000].

Preprocessing Steps

To improve the translation performance the following pre-processing steps are carried out:

Categorization We use some categorization which consists of replacing a single word by a category. The only words that are replaced by a category label are proper nouns denoting German cities. Using the new labeled corpus, all probability models are trained anew. To produce the translation in the "normal" language, the categories are translated by rule and are inserted into the target sentence.

Word Joining Words are joined on the basis of the likelihood- criterion presented in Section 3.3. The word joining is done to handle cases like the German compound noun "Zahnarzttermin" for the English "dentist's appointement", because a single word has to be mapped to two or more target words. Examples for word joinings that have been selected are given in the following section 5.2. The effect of the word joining on the translation performance on the TEST-331 test set is reported in Section 5.6. Note that the word joining is applied only to the target language words, the source language sentences remain unchanged. During the search process several joined target language words may be generated by a single source language word. For the translation experiments, where no word joining is carried out, we retain the possibility to generate two target words by a single source word in the way as has been described in [Berger et al. 1996]: when two target words (v, w) are generated we do not use a lexicon probability for generating the intermediate target word v.

The costs for generating the target word v are given by its fertility $\phi(0|v)$ and the language model probability.

Manual Lexicon To account for unseen words in the test sentences and to obtain more focused lexicon probabilities $p(f|e)$, we use a bilingual German-to-English dictionary. For each word e in the target vocabulary, we create a list of source translations f according to this dictionary. The probability of the manual lexicon $p_{dic}(f|e)$ for the dictionary entry (f, e) is defined as:

$$p_{dic}(f|e) = \begin{cases} \frac{1}{N_e} & \text{if } (f,e) \text{ is in dictionary} \\ 0 & \text{otherwise} \end{cases}$$

where N_e is the number of source words listed as translations of the target word e. The dictionary probability $p_{dic}(f|e)$ is linearly combined with the automatically trained lexicon probabilities $p_{aut}(f|e)$ to obtain smoothed probabilities $p(f|e)$:

$$p(f|e) = (1-\lambda) \cdot p_{dic}(f|e) + \lambda \cdot p_{aut}(f|e).$$

For the translation experiments, the value of the interpolation parameter is fixed to $\lambda = 0.5$.

Translation Results

In this section, we carry out a series of translation experiments for the German-to-English "Verbmobil" task. The translation experiments reported here are carried out with conservatively large pruning thresholds. Only minor translation errors due to search errors occur. The following main results are obtained:

- The word joining technique presented in Section 3.3 is successfully applied and reduces the multi-reference word error rate (m-WER) on the TEST-331 test set from 30.7% to 24.7% .
- We show the effect of two scaling factors α for the language model (α_{LM}) and the distortion model (α_D) probabilities which significantly improve translation performance in terms of multi-reference word error rate (m-WER).
- A significant loss in translation accuracy is occurred for a zerogram or a unigram language model whereas there is nearly no difference in the use of a bigram or a trigram language model.
- For the translation direction German-to-English the re-ordering restriction GE performs at least as good as/ or better than the re-ordering restriction S3 in both terms of multi-reference word error rate (m-WER) and subjective sentence error rate (SSER). Table 5.17 shows example translation where the necessary word re-ordering can only be handled by the GE re-ordering constraint. For the translation direction English-to-German, the re-ordering restriction EG is successfully tested.

- The translation speed can be decreased to a few seconds per sentence without significant loss in translation performance using **two** pruning parameters: 1) coverage pruning and 2) cardinality pruning. Here, observation pruning and histogram pruning thresholds are set to conservatively large thresholds.

For the experiments, the following re-ordering constraints are tested (cf. Appendix A):

- The MON constraint where no word re-ordering is allowed.
- The GE constraint for the German-to-English verb group re-ordering.
- The EG constraint for the English-to-German verb group re-ordering.
- The S3 constraint which is the restriction described in the IBM patent [Berger et al. 1996].

Effect of the Word Joining

In this subsection the effect of the word joining as described in Section 3.3 is demonstrated. The actual word joinings that are found using the likelihood criterion are presented in Section 5.2. These word joinings are applied to the target language training data sentences. Two types of translation experiments are carried out: 1) with and 2) without word joining applied. The results are shown in Table 5.12. For a first experiment, where no word joining has been applied, a full IBM model training obtaining the translation model parameters described in Section 4.9 is carried out. The word error rate (WER) and the multi-reference word error rate (m-WER) obtained on the TEST-331 test set are reported in first line of Table 5.12. In a second experiment, we first carry out the word joining algorithm given in Table 3.2. The resulting modified target language corpus is used to rerun the full IBM model training procedure. The translation model parameters may now include single-word entries that may consist of several joined target language words. Hence, the target language vocabulary goes up from 4 648 to 7 908. For the word joining a threshold of $\tau = 4.0$ is used. The translation results on the TEST-331 test set are reported in the second line of Table 5.12. As the experimental results show, the word joining improves the translation performance significantly, the multi-reference word error rate (m-WER) is reduced from 30.7 to 24.9. The word joining is applied in all subsequent translation experiments on the Verbmobil corpus.

Effect of the Language Model Scaling Factor

In speech recognition, where Bayes' decision rule is applied, a language model scaling factor α is used, where a typical value is $\alpha \approx 15$. The reason for this scaling factor is that the language model probabilities are more reliably estimated than the acoustic probabilities. Following the use of a language model scaling factor in speech recognition, it is introduced into statistical machine translation, too. The optimization criterion in Eq. 1.1 is modified as follows:

$$\hat{e}_1^I = \arg\max_{e_1^I} \{p(e_1^I)^{\alpha_{LM}} \cdot p(f_1^J|e_1^I)\},$$

Table 5.12: Target language vocabulary size, word error rate (WER), and multi-reference word error rate (m-WER) for two cases: 1) Without word joining applied and 2) With word joining applied (using the word joining threshold $\tau = 4.0$) on the TEST-331 test set.

Word Joining	Target vocabulary size	WER [%]	m-WER [%]
No	4 648	61.3	30.7
Yes	7 908	60.1	24.7

Table 5.13: Word error rate (WER) and multi-reference word error rate (m-WER) as a function of the language model scaling factor α on the TEST-331 test set for the GE re-ordering constraint.

Language model scaling factor α_{LM}	WER [%]	m-WER [%]
0.5	61.6	25.8
0.6	61.2	25.0
0.7	60.7	25.0
0.8	60.1	24.7
0.9	60.1	25.2
1.0	60.0	25.9
1.1	59.9	27.3
1.2	60.1	28.5
1.3	60.1	29.4
1.4	60.7	30.8
1.5	61.2	32.4

where $p(e_1^I)$ is the language model probability of the target language sentence. In the experiments presented here, a trigram language model is used to compute $p(e_1^I)$. The effect of the language model scaling factor α on the multi-reference word error rate (m-WER) on the TEST-331 test set is presented in Table 5.13. The effect of the language model scaling factor is rather small in terms of word error rate (WER) and multi-reference word error rate (m-WER) within the range $\alpha \in [0.5, \cdots, 1.0]$. A language model scaling factor $\alpha > 1.0$ entails a significant loss in translation performance. Deviating from what is the case in speech recognition, the translation model probabilities seem to be estimated as reliably as the language model probablities in statistical machine translation. A minimum multi-reference word error rate (m-WER) of 24.7 is obtained on the TEST-331 test set using a language model scaling factor of $\alpha = 0.8$.

Effect of the Distortion Model Scaling Factor

A second scaling factor is introduced for the distortion model probabilities $p(j|j', J)$. For reasons of brevity, we discard the fact that the distortion probabilities in the full IBM model are class-based and show only the dependence on the two source positions j and j'.

Table 5.14: Word error rate (WER) and multi-reference word error rate (m-WER) as a function of the distortion model scaling factor α on the TEST-331 test set for the GE re-ordering constraint.

Scaling parameter α_D	WER [%]	m-WER [%]
0.0	64.0	32.8
0.1	61.9	29.2
0.2	60.9	26.9
0.3	60.4	25.4
0.4	60.1	24.7
0.5	60.2	24.6
0.6	60.4	24.6
0.7	60.5	24.8
0.8	60.4	24.9
0.9	60.4	25.3
1.0	60.7	25.8

The distortion probability $p(j|j', J)$ is replaced by $p(j|j', J) = p(j|j', J)^{\alpha_D}$ [1]. The effect of distortion model scaling factor α is shown in Table 5.14 on the TEST-331 test set. The minimum word error rate (WER) is obtained for $\alpha = 0.4$ with nearly no difference in the multi-reference word error rate (m-WER) for the cases $\alpha = 0.4, 0.5, 0.6$. The word error rate (WER) and the multi-reference word error rate (m-WER) on the TEST-331 test set increase significantly, if no distortion probability is used for the case $\alpha = 0.0$. The benefit of using a distortion probability scaling factor of $\alpha = 0.4$ instead of a scaling factor of $\alpha = 1.0$ comes from the fact that high costs due to the distortion probability might supress long-distant word re-orderings that are important for the German verb group re-ordering.

Effect of the Language Model

Table 5.15) shows the word error rate (WER) and multi-reference word error rate (m-WER as the function of the language model used. The translation performance decreases significantly when a zerogram or an unigram language model is used. On the other hand, the difference between the use of a bigram or a trigram language model is not significant.

Effect of the Word Re-Ordering Constraints

Table 5.16) shows the computing time, the multi-reference word error rate (m-WER, and the subjective sentence error rate (SSER) on the TEST-147 test set as a function of three re-ordering constraints MON, GE and S3. The computing time is given in terms of CPU time per sentence (on a 450-MHz Pentium-III-PC). For the subjective sentence error rate

[1] We have studied a slightly different version of the distortion probability scaling, where a renormalization is carried out: $p_\alpha(j|j') = \frac{p^{\alpha_D}(j|j')}{\sum_j p^{\alpha_D}(j|j')}$. This approach has resulted in slightly higher word error rates.

Table 5.15: Word error rate (WER) and multi-reference word error rate (m-WER) for four different language models on the TEST-331 test set. For the translation experiments the GE re-ordering constraint is used.

Language modell	WER [%]	m-WER [%]
Zerogram	85.8	77.1
Unigram	64.2	38.8
Bigram	59.1	25.6
Trigram	60.1	24.7

Table 5.16: Computing time, multi-reference word error rate (m-WER) and subjective sentence error rate (SSER) for three different search procedures on the TEST-147 test set. No word re-ordering accross punctuation marks is performed.

Re-ordering Constraint	CPU time [sec]	m-WER [%]	SSER [%]
MON constr.	0.2	40.6	28.6
GE constr.	5.2	33.3	21.0
S3 constr.	13.7	34.4	19.9

(SSER), it turns out that restricting the word re-ordering not to cross punctuation marks improves translation performance significantly. The average length of the sentence fragments that are seperated by punctuation marks is rather small - 4.5 words per fragments. A coverage pruning threshold of $t_C = 5.0$ and an observation pruning of $n_o = 50$ are applied during the experiments [2]. No other type of pruning is used [3].
The MON constraint performs worst in terms of both multi-reference word error rate (m-WER) and subjective sentence error rate (SSER). The computing time is low, since no re-ordering is carried out. The restrictions GE and S3 perform nearly identically in terms of both error rates m-WER and SSER. However, the GE restriction works about 3 times as fast as the S3 restriction.
Table 5.17 shows example translations obtained by the three different approaches. Again, the MON re-ordering constraint performs worst. In the second and third translation examples, the S3 word re-ordering constraint performs worse than the GE re-ordering constraint, since it can not take properly into account the word re-ordering due to the German verb group. The German finite verbs "bin" (second example) and "könnten" (third example) are too far away from the personal pronouns "ich" and "Sie" (6 respectively 5 source sentence positions). In the last example, the less restrictive S3 re-ordering constraint leads to a better translation, the GE translation is still acceptable, though.

[2]For the translation experiments, the negative logarithm of the actual pruning thresholds t_c and t_C is reported, where for simplicity reasons we do not change the notation.

[3]In a speech-to-speech demo system, we would use the GE re-ordering restriction and a slightly sharper pruning in order to achieve translation times of about one second per sentence.

Table 5.17: Example translations for the translation direction German-to-English using three different re-ordering constraints: MON, GE, and S3.

Input:	Ja , wunderbar . Können wir machen .
MON :	Yes, wonderful. Can we do .
GE :	Yes, wonderful. We can do that .
S3 :	Yes, wonderful. We can do that .

Input:	Das ist zu knapp , weil ich ab dem dritten in Kaiserslautern bin . Genaugenommen nur am dritten . Wie wäre es denn am ähm Samstag , dem zehnten Februar ?
MON :	That is too tight , because I from the third in Kaiserslautern . In fact only on the third . How about ähm Saturday , the tenth of February ?
GE :	That is too tight , because I am from the third in Kaiserslautern . In fact only on the third . Ähm how about Saturday , February the tenth ?
S3 :	That is too tight , from the third because I will be in Kaiserslautern. In fact only on the third . Ähm how about Saturday , February the tenth ?

Input:	Wenn Sie dann noch den siebzehnten könnten , wäre das toll , ja .
MON :	If you then also the seventeenth could , would be the great , yes .
GE :	If you could then also the seventeenth , that would be great , yes .
S3 :	Then if you could even take seventeenth , that would be great , yes .

Input:	Ja , das kommt mir sehr gelegen . Machen wir es dann am besten so .
MON :	Yes , that suits me perfectly . Do we should best like that .
GE :	Yes , that suits me fine . We do it like that then best .
S3 :	Yes , that suits me fine . We should best do it like that .

Beam Search Pruning Results

In this section, the effect of the beam search pruning is demonstrated. Translation results on the TEST-331 test set are presented in order to evaluate the effectiveness of the pruning techniques [4]. The quality of the search algorithm with respect to the GE and S3 re-ordering constraints is demonstrated using two criteria:

1. The number of search errors for a certain combination of pruning thresholds is counted. A search error occurs for a test sentence if the final translation probability Q_F for a candidate translation e_1^I as given in Eq. 4.4 is smaller than a reference probability for that test sentence. We will use two ways to compute reference probabilities as explained below.

2. The multi-reference word error rate (m-WER) performance measure is computed as a function of the used pruning thresholds. Generally, decreasing the pruning threshold leads to a higher word error rate since the optimal path through the translation lattice is missed resulting in translation errors.

Two automatically generated reference probabilities are used. These probabilities are computed **separately** for the re-ordering constraints GE and S3. The difference is not shown in the notation, but will be clear from the context:

Q_{ref} : A forced alignment is carried out between each of the test sentences and its corresponding reference translation, where only a single reference translation for each test sentence is used. The probability obtained for the reference translation is denoted by Q_{ref}.

Q_{F^*} : A translation is carried out with conservatively large pruning thresholds yielding a translation close to the one with the maximum translation probability. The translation probability for that translation is denoted by Q_{F^*}.

First, in a series of experiments we study the effect of the coverage and the cardinality pruning for the re-ordering constraints GE and S3. When reporting on the different pruning thresholds, we will show the negative logarithm of those pruning thresholds. The experiments are carried out into two different pruning "dimensions":

1. In Table 5.18 only the coverage pruning using threshold $t_{\mathcal{C}}$ is carried out while no cardinality pruning is applied: $t_c = \infty$.

2. In Table 5.19 only the cardinality pruning using threshold t_c is carried out while no coverage pruning is applied: $t_{\mathcal{C}} = \infty$.

Both tables use an observation pruning of $n_o = 50$. The effect of the coverage pruning threshold $t_{\mathcal{C}}$ is demonstrated in Table 5.18. For these translation experiments, the cardinality pruning threshold is set to $t_c = \infty$, thus no comparison between partial hypotheses that do not cover the same set $\mathcal{C}$ of source sentence positions is carried out. To restrict

[4] The CPU times on the TEST-331 set are higher, since the average fragment length is greater than for the TEST-147 set.

Table 5.18: Effect of the coverage pruning threshold t_C on the number of search errors and the multi-reference word error rate (m-WER) on the TEST-331 test set (no cardinality pruning carried out: $t_c = \infty$). A cardinality histogram pruning of 200 000 is applied to restrict the maximum overall size of the search space. The negative logarithm of t_C is reported.

Re-ordering constraint	t_C	CPU time [sec]	search errors		m-WER [%]
			$Q_{ref} > Q_F$	$Q_{F^*} > Q_F$	
GE constr.	0.01	0.21	318	323	73.5
	0.1	0.43	231	301	53.1
	1.0	1.43	10	226	30.3
	2.5	4.75	5	142	25.8
	5.0	29.6	-	35	24.6
	7.5	156	-	2	24.9
	10.0	630	-	-	24.9
	12.5	1300	-	-	24.9
S3 constr.	0.01	5.48	314	324	70.0
	0.1	9.21	225	303	50.9
	1.0	46.2	4	223	31.6
	2.5	190	-	129	28.4
	5.0	830	-	-	28.3

the overall size of the search space in terms of computing time and memory restrictions, a cardinality pruning of $n_c = 200\,000$ is applied. As can be seen from Table 5.18, the multi-reference word error rate (m-WER and the number of search errors decrease significantly as the coverage pruning threshold t_C goes up. For the GE re-ordering constraint the m-WER goes down from 73.5% to 24.9%. For a coverage pruning threshold $t_C \geq 5.0$ the m-WER remains nearly constant at 25.0% although search errors still occur. For the S3 re-ordering constraint the m-WER goes down from 70.0% to 28.3%. Here, the largest coverage threshold tested is $t_C = 5.0$, since for larger threshold values t_C, the search procedure can not be carried out due to memory and time restrictions. The number of search errors is reduced as the coverage pruning threshold is increased. It turns out to be difficult to verify search errors by looking at the reference translation probabilities Q_{ref} alone. The translation with the maximum translation probability seems to be quite narrowly defined. The coverage pruning is more effective for the GE constraint than for the S3 constraint since the overall search space for the GE re-ordering is smaller.

Table 5.19 shows the effect of the cardinality pruning threshold t_c on the m-WER when no coverage pruning is carried out (a histogram coverage pruning of 1 000 is applied to restrict the overall size of the search space). There is a strong effect of the cardinality threshold t_c on the m-WER. The word error rate decreases significantly as the cardinality threshold t_c goes up. For the GE re-ordering constraint the m-WER goes down from 48.5% to 24.9%, for the S3 re-ordering constraint the m-WER rate goes down from 51.4% to 28.2%. For the coverage threshold $t = 15.0$, the GE constraint works about 4 times as fast as the S3 re-ordering constraint, since the overall search space for the S3 re-ordering constraint is much

Table 5.19: Effect of the cardinality pruning threshold t_c on the number of search errors and the multi-reference word error rate (m-WER) on the TEST-331 test set (no coverage pruning is carried out: $t_\mathcal{C} = \infty$). A coverage histogram pruning of 1 000 is applied to restrict the overall size of the search space. The negative logarithm of t_c is shown.

Re-ordering Constraint	t_c	CPU time [sec]	search errors		m-WER [%]
			$Q_{ref} > Q_F$	$Q_{F^*} > Q_F$	
GE constr.	1.0	0.03	45	287	48.5
	2.0	0.06	20	277	41.9
	3.0	0.13	16	266	37.7
	4.0	0.30	6	239	34.1
	5.0	0.55	2	212	30.5
	7.5	3.2	-	106	26.6
	10.0	14.2	-	32	25.1
	12.5	42.2	-	5	24.9
	15.0	93.9	-	-	24.9
	17.5	176.7	-	-	24.9
S3 constr.	1.0	0.02	10	331	51.4
	2.0	0.05	1	283	46.2
	3.0	0.10	1	274	43.3
	4.0	0.22	-	251	40.2
	5.0	0.50	-	227	37.5
	7.5	4.3	-	171	32.9
	10.0	26.8	-	99	30.8
	12.5	123.3	-	49	28.9
	15.0	430	-	-	28.2

larger. Although the overall search space is much larger for the S3 re-ordering restriction, for smaller values of the coverage threshold $t_\mathcal{C} \leq 5.0$ the S3 constraint works as fast as the GE constraint or even faster. The reason is that only a very small portion of the overall search space is searched for small values of the cardinality pruning threshold t_c. There is some computational overhead in expanding a partial hypothesis for the GE re-ordering restriction because the finite-state control has to be handled. No results are obtained for the S3 constraint and the coverage threshold $t_c = 17.5$ due to memory restrictions. The number of search errors is reduced as the cardinality pruning threshold is increased. Again, it is difficult to verify search errors by looking at the reference translation probabilities alone.

Both coverage and cardinality pruning are more efficient for the GE re-ordering restriction than for the S3 re-ordering restriction. For the S3 restriction the situation is even worse: no translation results are obtained for a coverage threshold $t_c > 5.0$ without cardinality pruning applied, due to memory and computing time restrictions. For the GE restriction a virtually full search can be carried out where only the observation pruning is applied: the target translations and translation probabilities for the hypothesis files produced for the two cases: 1) $t_\mathcal{C} = 10.0$,$t_c = \infty$ and 2) $t_\mathcal{C} = \infty$, $t_c = 15.0$ are identical. (Actually, for one

Table 5.20: Effect of the observation pruning on the number of search errors and the m-WER on the TEST-331 test set (parameter setting: $t_c = \infty$, $t_{\mathcal{C}} = 10.0$). No histogram pruning is applied. The results are reported for the GE constraint.

observation pruning $n_{\mathcal{O}}$	CPU time [sec]	search errors		m-WER [%]
		$Q_{ref} > Q_F$	$Q_{F^*} > Q_F$	
1	2.0	13	284	29.3
2	5.9	6	239	26.9
3	10.8	2	196	25.7
5	23.6	2	140	25.3
10	62.9	-	99	24.8
25	238	-	44	24.5
50	630	-	-	24.9

Table 5.21: Demonstration of the combination of the two pruning thresholds $t_{\mathcal{C}} = 5.0$ and $t_c = 12.5$ to speed up the search process for the two re-reordering constraints GE and S3 ($n_o = 50$). The translation performance is shown in terms of multi-reference word error rate (m-WER) on the TEST-331 test set.

Re-ordering Constraint	$t_{\mathcal{C}}$	t_c	CPU time [sec]	search errors		m-WER [%]
				$Q_{ref} > Q_F$	$Q_{F^*} > Q_F$	
GE constr.	5.0	12.5	6.9	0	38	24.7
S3 constr.	5.0	12.5	26.9	0	65	29.2

test sentence in the TEST-331 test set the translations are different although the translation probabilities are exactly the same). Since the pruning is carried out independently into two different" pruning dimensions, no search errors will occur if the thresholds are further increased.

Table 5.20 shows the effect of the observation pruning parameter n_o on the m-WER for the re-ordering constraint GE. The multi-reference word error rate (m-WER) is significantly reduced by hypothesizing up to the best 50 target words e for a source language word f. The m-WER goes up from 24.9% to 29.3% when decreasing the number of hypothesized words to only a single word.

Table 5.21 demonstrates the effect of the combination of the coverage pruning $t_{\mathcal{C}} = 5.0$ and the cardinality pruning thresholds $t_c = 12.5$, where the actual values are found in informal experiments: a good rule of thumb is that t_c should be at least twice as big as $t_{\mathcal{C}}$. For the GE re-ordering constraint the average computing time is about 7 seconds per sentence without any loss in translation performance as measured in terms of multi-reference word error rate (m-WER). For the S3 re-ordering constraint the average computing time per sentence is 27 seconds. Again, the combination of coverage and cardinality pruning works more efficiently for the GE re-ordering constraint. The memory requirement for the algorithm is about 100 MB.

Experimental Results: English-to-German

A series of translation experiments for the translation direction English-to-German are carried out, too. The results are shown in Table 5.22. The translation results are given in terms of word error rate (WER) and position-independent word error rate (PER). For the English-to-German translation direction, a single reference translation for each test sentence is used to carry out the automatic evaluation. The translation task for the translation direction English-to-German is somewhat more difficult, the trigram language model perplexity goes up from 38.3 to 68.2 as can be seen in Table 5.11. The following main results have been obtained:

- There is only a very small difference for the word error rates obtained for the re-ordering constraints EG and S3, namely the word error rate (WER) is 70.1% for both re-ordering constraints.
- The re-ordering constraint MON performs slightly worse: the word error rate (WER) is increased to 70.6% and the position-independent word error rate (PER) is increased to 57.0%
- The word error rate (WER) obtained for the translation direction English-to-German is about 10.0% higher than the word error rates obtained for the translation direction German-to-English. This is due to the fact that the trigram language model for the translation direction English-to-German has a significantly higher perplexity.

Table 5.23 shows translation examples for the translation direction English-to-German. There is no significant difference between the re-ordering

Comparison of Different Approaches

On the TEST-251 test set translation experiments are carried in order to compare two different translation approaches:

- **Single-word based approch (SWB approach):** This is the approach presented in this work using the IBM-model parameters. The approach is called single-word

Table 5.22: Translation results for the translation direction English-to-German on the TEST-331 test set. The results are given in terms of computing time, word error rate (WER) and position-independent word error rate (PER) for three different re-ordering constraint: MON, EG, and S3. For the translation direction English-to-German a single reference translation for each test sentence is used for the automatic evaluation.

Re-ordering Constraint	CPU time [sec]	WER [%]	PER [%]
MON constr.	0.5	70.6	57.0
EG constr.	10.1	70.1	55.9
S3 constr.	53.2	70.1	55.8

Table 5.23: Example translations for the translation direction English-to-German using three different re-ordering constraints: MON, EG, and S3.

Input:	Yeah , that wouldn't be bad . do you have any ideas where I could stay ?
MON :	Ja , das wäre schade . haben Sie irgendwelche Ideen wo ich könnten übernachten ?
EG :	Ja , das wäre nicht schlecht . haben Sie irgendwelche Ideen wo wir wohnen könnten ?
S3 :	Ja , das wäre nicht schlecht . haben Sie irgendwelche Ideen wo wir wohnen könnten ?

Input:	Oh , that sounds great . could you arrange a suite for me ?
MON :	Oh , das klingt gut . könnten Sie unbedingt ein Suite bei mir ?
EG :	Oh , das klingt gut . könnten Sie einen Suite ausmachen für mich ?
S3 :	Oh , das klingt gut . könnten Sie mir einen Suite ausmachen ?

Input:	Well , I still need your signature here and then I will check with your company .
MON :	Also , ich konnte Arbeitskraft Unterschrift hier und ich werde nachsehen mit Ihrer Firma .
EG :	Also , ich bräuchte noch Ihre Unterschrift und dann gucke ich hier mit Ihrer Firma .
S3 :	Also , ich brauche hier noch Ihre Unterschrift und dann werde ich veranlassen mit Ihrer Firma .

based, since the lexicon parameters map single target words to single source words only.

- **Alignment Template Approach (AT approach):** This is the approach presented in [Och et al. 1998, Och et al. 1999] that maps multiple target words to multiple source words.

The translation results are given in Table 5.24. Example translations for the different translation approaches are given in Table 5.25. For the experiments, the test sentences are splitted at punctuation marks. The average length of the corresponding fragments is rather small: 6.1 words per fragment. From the translation results, we can make the following observations:

- For most of the produced translations the difference between the two approaches is rather small: often they differ only in a few words.
- The alignment template approach performs better in both terms of word error rate (WER) and position-independent word error rate (PER) although the difference is rather smaller. The reason for the better translation performance can be seen in the example translations for the two approaches given in Table 5.25. They are selected

Table 5.24: English-to-German translation results on the TEST-251 test set in terms of word error rate (WER) and position-independent word error rate (PER) for the single-word based approach (using three different re-ordering constraints) and for the alignment template approach.

Re-Ordering Constraint	CPU [sec]	WER [%]	PER [%]
MON const.	0.2	43.9	32.0
EG const.	8.4	41.8	30.5
S3 const.	31.8	42.2	30.6
AT approach	2.5	41.3	27.3

to underlign the differences between the two approaches. The single-word based approach produces rather word-to-word translation whereas the the alignment template approach tends to translate whole phrases, e.g. in the second translation example the translation of the German phrase "Ich habe nicht verstanden", which is translated as "I haven't understand" by the alignment template approach and as "I did not hear" by the single-word based approach.
Another example is the translation of "über das Hotel sagen" which is correctly translated by the alignment template approach as "say about" wheras the SWB translation "say at" is not correct since the SWB approach fails to take into account the local context to properly translated the proposition "über".

Table 5.25: Example translations for the alignment template (AT) and the single-word based (SWB) approach on the TEST-251 test set. The single-word based approach (SWB) uses the GE re-ordering constraint.

Input	ja , wann möchten Sie zurückfliegen ?
AT approach	yes , when do you want to fly back ?
SWB approach	well , when you want to fly back ?

Input	ich habe nicht verstanden , was Sie über das Hotel gesagt haben .
AT approach	I haven't understand , what you said about the hotel .
SWB approach	I did not hear what you said at the hotel .

Input	lassen Sie uns um zwanzig Uhr losfliegen . da fliegt ein Flug nach Hamburg über Mailand . ist das nicht toll ?
AT approach	let's leave at eight . there is a flight out about to Hamburg Mailand. that isn't great ?
SWB approach	let us leave at eight o'clock . that is a flight from Hamburg over Mainland is not that great ?

5.7 Experiments on the Italian-English FUB Corpus

The Task and the Corpus

The single-word based translation approach is tested on a speech input corpus obtained within the EuTrans-II project [Vidal et al. 2000]. The translation direction is from Italian to English. The corpus consisted of queries, requests and complains made through the telephone to the front desk in a hotel. The details of the corpus are given in Table 5.26. The translation results are given in terms of word error rate (WER) on the 300 test sentence. The following main results for the FUB corpus are presented:

- We present the effect of the word joining: the word error rate is reduced from 32.1 to 26.0 when the word joining is applied as can be seen in Table 5.27. Hence, the word joining is important for the translation direction Italian-to-English as well.
- We present example translation for the translation direction Italian-to-English. Example translations are given in Table 5.28.
- We study the effect of different re-ordering restrictions in Table 5.29. The re-ordering restriction S2 produces the minimum word error rate of 26.0.

Effect of Word Re-ordering

The effect of different word re-ordering schemes is shown in Table 5.29, where we test four different re-ordering schemes. For the re-ordering restriction MON no re-ordering is allowed. For the re-ordering constraint "Sk", it is allowed to skip at most k source sentence positions. The case $k = 3$ corresponds to the IBM-style re-ordering constraint. The computing time varies significantly for the different re-ordering restrictions. The best translation results in terms of word error rate (WER) are obtained for the S2 re-ordering constraint where the computing time is about 4 seconds per sentence. Table 5.28 shows example translations for the translation direction Italian-to-English. For the first translation example, the re-ordering constraint S2 leads to a better translation for the Italian phrase "un servizio di

Table 5.26: Training and test conditions for the FUB task (*number of words without punctuation).

		Italian	English
Train:	Sentences	3 038	
	Words	55 302	65 446
	Words*	47 606	57 588
Vocabulary	Size	2 459	1 701
	Singletons	1 118	662
Test:	Sentences	300	
	Words	6 121	7 243
	Bigr./Tri. Perplexity	25.2/19.2	14.7/10.1

Table 5.27: Target language vocabulary size and word error rate (WER) on the test set for the two cases: 1) with word joining and 2) without word joining applied.

Word Joining	Target vocabulary size	WER [%]
Yes	2 459	32.1
No	3 193	26.0

Table 5.28: Example translations for the translation direction Italian-to-English using the two different re-ordering constraints MON and S2.

Input:	Volevo sapere se era possibile effettuare un servizio di lavanderia .
MON :	I would like to know if it is possible to make a service of laundry .
S2 :	I would like to know if it is possible to make a laundry service .

Input:	Buongiorno l' albergo Excelsior di Parigi ? volevo sapere se per la settimana di Natale c' e' disponibilita' di una camera matrimoniale con doccia , con frigobar e con aria condizionata .
MON :	Good morning the Excelsior of Paris ? I would like to know if for the week of Christmas there is availability of a double bed room with shower , with minibar and with air conditioning .
S2 :	Good morning the Excelsior hotel in Paris ? I would like to know if for the week of Christmas there is availability of a double bed room with shower , with minibar and with air conditioning .

lavanderia" which is translated as "a laundry service" after the corresponding word reordering is carried out. In general, there are only minor difference in the translation results for the re-ordering constraints MON and S2. Rather long sentences of more than 20 words can be translated in a few seconds.
Table 5.30) contains translation results in terms of word error rate (WER for the translation direction English-to-Italian. Again, the four re-ordering constraints MON, S1, S2, and S3 are used to carry out the translation experiments.

Effect of the Distortion Model Scaling Factor

In this section, the effect of a distortion model scaling factor for the inverted alignment probabilities $p(j|j', J)$ on the FUB corpus is shown in Table 5.31. Very much the same results as for the Verbmobil task are obtained: the optimum scaling factor is $\alpha = 0.3$. The word error rate increases significantly when no distortion probabilities are used for the case $\alpha = 0.0$.

Table 5.29: Computing time and word error rate (WER) on the 300 test sentences for the translation direction Italian-to-English using four different re-ordering constraints.

Re-ordering Constraint	CPU time [sec]	WER [%]	PER [%]
MON constr.	0.2	27.9	20.8
S1 constr.	1.3	26.3	20.1
S2 constr.	4.1	26.0	20.0
S3 constr.	49.0	26.3	20.1

Table 5.30: Computing time and word error rate (WER) on the 300 test sentences for the translation direction English-to-Italian using four different re-ordering constraints.

Re-ordering Constraint	CPU time [sec]	WER [%]	PER [%]
MON constr.	0.4	37.1	30.7
S1 constr.	3.7	34.8	29.0
S2 constr.	15.2	34.7	28.7
S3 constr.	39.0	35.2	29.0

Table 5.31: Word error rate (WER) on the 300 test set sentences for the translation direction Italian-to-English as a function of the distortion model scaling parameter α_D for the re-ordering constraint S2.

Scaling parameter α_D	WER [%]
0.0	30.2
0.1	26.5
0.2	26.0
0.3	26.0
0.4	26.2
0.5	26.3
0.6	26.4
0.7	26.6
0.8	26.7
0.9	27.0
1.0	27.2

Table 5.32: Training and test conditions for the HANSARD task (*number of words without punctuation).

		French	English
Train:	Sentences	1 470 473	
	Words	24 338 195	22 163 092
	Words*	22 175 069	20 063 378
Vocabulary	Size	100 269	78 332
	Singletons	40 199	31 319
Test:	Sentences	5 432	
	Words	97 446	88 773
	Bigr./Tri. Perplexity	196.9/121.8	269.9/179.8

5.8 Experiments on the French-English Hansard Corpus

The third corpus on which we perform translation experiments, is the so-called Hansard corpus [5]. The proceedings of the Canadian parliament are kept in both French and English. The remarks of the parliament members are written down in whichever of the two languages they use. They are translated to produce a complete set of the proceedings in both French and English. The proceedings for historical reasons are called "Hansards". The resulting bilingual data has been sentence-aligned using statistical methods [Brown et al. 1990]. Originally, about 3 million sentences were selected. Here, we use a subset of the original training data, where the details are given in Table 5.32. The Hansards corpus is by far the most difficult task in terms of vocabulary size and number of training sentences. The training and test sentences are by far less restricitive than for the Verbmobil and the FUB task.

The experimental results are given in Table 5.33 and Table 5.34. For the translation experiments on the Hansard corpus no word joining is carried out. Two target words can be produced by a single source word in the way as was described in [Berger et al. 1996]: when two target words (e', e) are generated we do not use a lexicon probability for generating the intermediate target word e'. The costs for generating the target word e' are given by its fertility $\phi(0|e')$ and the language model probability. As can be seen in Table 5.33 and Table 5.34, the word error rate (WER) and the position-independent word error rate (PER) are rather high. The reason is that the Hansard task is by far less restrictive and the vocabulary size is much larger. There is no significant difference in performance between the MON and the S3 re-ordering constraints. The computation time is rather high: for the S3 re-ordering constraint the average translation time is about 3 minutes per sentence, where the following settings is used: $t_{\mathcal{C}} = 5.0$, $t_c = 10.0$, $n_{\mathcal{C}} = 250$, and $t_o = 12$. No cardinality histogram pruning is carried out. In an additional experiment for the translation direction

[5] Part of the results presented in this subsection are due to the help of Nicola Ueffing who carried out the experiments on this corpus [Och et al. 2001].

Table 5.33: Computing time and word error rate (WER) and position-independent word error rate (PER) for the translation direction French-to-English for the re-ordering constraints MON and S3 on the 5 432 test sentences.

Re-ordering constraint	CPU time [sec]	WER [%]	PER [%]
MON constr.	2.5	65.5	53.0
S3 constr.	580.0	64.9	51.4

Table 5.34: Computing time and word error rate (WER) and position-independent word error rate (PER) for the translation direction English-to-French for the re-ordering constraints MON and S3 on the 5 432 test sentences.

Re-ordering constraint	CPU time [sec]	WER [%]	PER [%]
MON constr.	2.2	66.6	56.3
S3 constr.	189.1	66.0	54.4

French-to-English and the re-ordering constraint S3, we speeded up the translation time to about 18 seconds per sentence by using the following pruning parameter setting: $t_{\mathcal{C}} = 3.0$, $t_c = 7.5$, $n_{\mathcal{C}} = 20$, $n_c = 400$, and $t_o = 5$. For the resulting hypotheses file the position independent word error rate (PER) went up only slightly from 51.4 % to 51.6 %.

Translation examples are given in Table 5.35. The French input sentences show some pre-processing that is carried out beforehand to simplify the translation task, e.g. "des" $\rightarrow$ "de les" and "l'est" $\rightarrow$ "le est". The translations produced are rather rough in some cases, although the general meaning is often preserved.

Table 5.35: Example translations for the translation direction French-to-English using the S3 re-ordering constraint.

Input	Je crois que cela donne une bonne idée de les principes à retenir et de ce que devraient être nos responsabilités .
S3	I think it is a good idea of the principles and to what should be our responsibility .

Input	Je pense que , indépendamment de notre parti , nous trouvons tous cela inacceptable .
S3	I think , regardless of our party , we find that unacceptable .

Input	Je ai le intention de parler surtout aujourd' hui de les nombreuses améliorations apportées à les programmes de pensions de tous les Canadiens .
S3	I have the intention of speaking today about the many improvements in pensions for all Canadians especially those programs .

Input	Chacun en lui - même est très complexe et le lien entre les deux le est encore davantage de sorte que pour beaucoup la situation présente est confuse .
S3	Each in itself is very complex and the relationship between the two is more so much for the present situation is confused .

Chapter 6

Discussion and Future Work

Statistical techniques are now widely used in natural language problems. Statistical machine translation is a relatively new approach to the problem of translating human languages. Given statistical models for the translation probability $p(f_1^J|e_1^I)$ and for the language model probability $p(e_1^I)$, we are trying in Eq. 1.1 to find the most probable target sentence e_1^I given a source sentence f_1^J. Since the number of possible translations e_1^I considered is E^I, where E is the size of the target vocabulary, we must find a way to generate the most likely target sequence e_1^I without trying all possible word sequences. In what follows, we will disscuss the main goals of this work as described in Chapter 2 and we will outline main directions for further developments.

Dynamic Programming Search Algorithm. In this work, we have presented a dynamic programming-based approach to solve the optimization criterion in Eq. 1.1. The starting point is a dynamic programming solution to the traveling salesman problem. The approach works by synchronously processing subsets of covered cities of equal cardinality while ignoring the order in which the cities are covered. In statistical machine translation, the cities correspond to source positions processed. The dynamic programming approach becomes more efficient in reducing the search space by introducing certain types of re-ordering constraints. These constraints are expressed in terms of the order the source positions may be covered. The following are lines that can be readily followed from the dynamic programming approach:

- A translation word-graph can be generated by extending the bookkeeping method presented in Section 4.10. At each gridpoint point in the dynamic programming search lattice, not only the best hypothesis at this grid point is stored, but also a set of partial hypotheses within a range of the optimum score. The approach may be carried out along the lines in [Ortmanns et al. 1997] as is done in speech recognition. The word graph generated may be used to apply more sophisticated language models, e.g. stochastic context free language models within the translation process.
- Finite-state language models can be directly included into the search process. The dynamic programming recursive equations are easily changed accordingly. Finite-state language models can be used to carry out a robust translation since

the generated target language sentences are syntactically very restricted, e.g. in the case of speech input translation, where the input source sentence is corrupted by speech-recognition errors. On the other hand, finite-state language models are able to cover long-distant dependencies that go behond the range of m-gram language models.

Language-Specific Re-Ordering Restrictions. A re-ordering constraint especially useful for the translation direction German-to-English is presented. The re-ordering constraints are presented in terms of covered or uncovered positions to the left and right of the rightmost covered position or leftmost uncovered position. The re-ordering constraint might be easily adopted to a new language pair: a single character string is used as a parameter for the search procedure to control the re-ordering restriction (cf. Appendix A). Parts-of-speech for the source sentence words might be used to further restrict the word re-ordering, e.g. by charactarizing the German verb group. Finally, the finite-state control for the re-ordering might be attached to certain words in the target or source language to handle re-ordering phenomena perculiar to certain grammatical structures, e.g. the target words produced by the source sentence positions which are processed non-monotonically could be constrained to belong to the English verb group (using English parts-of-speech information).

Beam Search Techniques. In Section 4.7, we have presented beam search techniques that are developed in a way consistent with the search space construction. The function $u(f)$ is used to take into account uncovered source sentence positions. This idea can be extended to take into account the distortion probabilities. Making the observation that the distortion probabilities are focused on small-distant jumps, we define two modified distortion probability $l(j|k)$ and $r(k|j)$. Here, $l(j|k)$ is an upper bound used for the distortion probabilities when jumping from at least position k on the right to position j on the left. $r(k|j)$ is an upper bound for the distortion probabilities when jumping from position j on the left to at least position k on the right. The modified distortion probability is defined as follows:

$$\begin{aligned} l(j|k) &= \max_{k' \geq k} p(j|k') \\ r(k|j) &= \max_{k' \geq k} p(k'|j) \end{aligned}$$

The use of the modified distortion probabilities is illustrated using an example involving the S3 re-ordering constraint (cf. Appendix A), where at most three source sentence positions may be left uncovered to the left of the rightmost covered position. The upper bound is computed using the fact that to reach the left-most uncovered position we cannot avoid jumps of a certain width. Infact, the jumps with the minimal possible width involve the two immediate uncovered positions to the right (if not a position still further to the right). We compute an upper bound for the involved distortion probabilities as illustrated in Figure 6.1. This specific upper bound works only in the given special case, different re-ordering constraints would lead to different upper bound computations. The three uncovered positions are denoted j, j', j''. Covering position j envolves probabilities that depend on the positions j', j'' and on

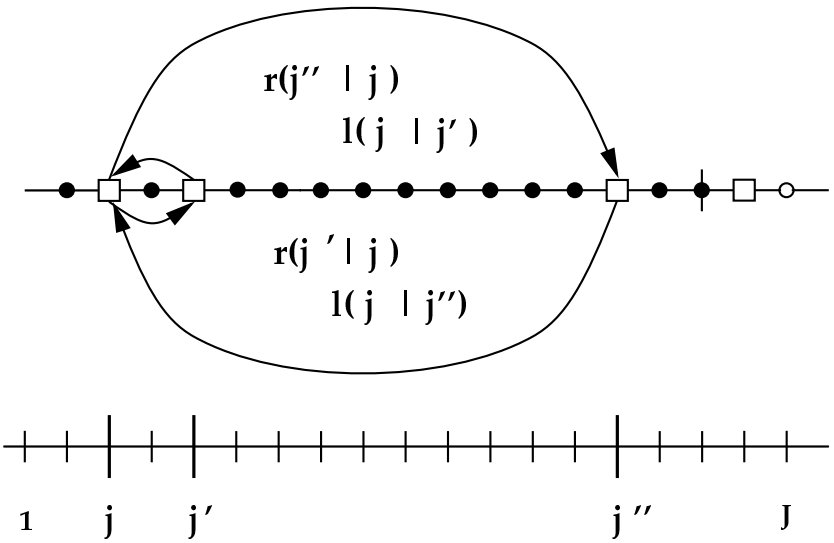


Figure 6.1: Illustration of the computation of the upper bound calculation for the partial hypothesis extension involving the uncovered positions j, j', and j''.

the order in which the positions are covered. The upper bound is given by:

$$\min\left\{r(j''|j)\cdot l(j|j'), r(j'|j)\cdot l(j|j'')\right\}$$

For the modified distortion probabilities $l(j|k)$ and $r(k|j)$, we have dropped not only the dependence on the word classes $\mathcal{F}, \mathcal{E}$, but also the dependence on the length of the source sentence J.

Extended Lexicon Models. Section 3.3 has demonstrated the use of an extended lexicon model for the target word joining. The concept is readily extended to cover multi-word lexicon dependencies. The lexicon extensions are similar to the so-called alignment templates [Och et al. 1998] or the head-transducers [Alshawi et al. 1997]. Unlike the alignment templates they allow for non-consecutive source and target word sequences to be processed. They do not yield the hierarchical structure of the head-transducers but allow for a tight integration of the translation and the search process using the DP-based search procedure. Finally, lexicon extensions might be used to simplify the rather complicated IBM model training. The fertility parameters might be replaced by direct lexical dependencies of the type presented in Section 3.3. A certain fertility value for a target word e with a high probability may be due to a specific combination of source language words to which e is typically aligned.

Coupling of Recognition and Translation. In speech translation, we are faced with the problem of how to couple the speech recognition process and the translation process, i.e. we have to cope with the additional complication of speech recognition errors. Considering the problem of speech input as opposed to text input for translation, we can distinguish three levels, i.e. the acoustic vectors $x_1^T = x_1, \cdots, x_T$ over time $t = 1, \cdots, T$, the source words f_1^J , and the target words e_1^I:

$$x_1^T \rightarrow f_1^J \rightarrow e_1^I$$

When doing speech input translation, the source sequence f_1^J is not of direct interest. The source word string may be introduced into the Bayes decision rule as hidden variables as shown in [Ney 1999, Tillmann et al. 2001]. This approch becomes particularly interesting in cases where the target language perplexity is much lower than the source language perplexity.

Chapter 7

Summary

In this chapter, we will sum up the most important results of the work presented. This work is about the search for statistical machine translation. Given a source sentence string f_1^J, to search for the most probable target language string e_1^I using the so-called IBM model parameters described in [Brown et al. 1993] is equivalent to finding the shortest path in a finite translation graph. The vertices of the graph correspond to subsets of covered source positions together with hypothesized target language words. The edges between the vertices correspond to hypothesis expansions. Both the number of vertices and the number of edges are restricted as is described in the preceding chapters. Costs are assigned to the vertices using the IBM model parameters. The following main results are presented:

Dynamic Programming Search Algorithm A dynamic programming solution to the general traveling salesman problem is modified to search for the shortest path in the above finite translation graph. As the original algorithm presented by Held & Karp, the DP-based search algorithm for statistical machine translation works by jointly processing the same set of covered source sentence positions. Using restrictions on the order in which the source positions are processed the DP approach becomes still more effective: for a certain type of re-ordering constraint (as used in the IBM translation system [Berger et al. 1996]) an upper bound for the complexity of $E^3 \cdot J^4$ is demonstrated (assuming the use of a trigram language model). Here, E is the size of target language vocabulary and J is the length of the source string.

Language-specific Re-ordering restrictions A parametrization of the re-ordering constraint is given that allows to quickly adopt to several different translation directions, e.g. German-to-English, Italian-to-English. The re-ordering restrictions are expressed in terms of **four** numbers: 1) the number of **skipped** positions to the left of the rightmost covered position, 2) the size of the window in which the skipped positions are placed 3) the number of **moved** positions to the right of the leftmost uncovered position, and 4) the size of the window in which the moved positions are placed. A finite-state control may be added to the re-ordering constraints which is especially useful for the translation direction German-to-English. These new types of re-ordering constraints allow for word re-ordering to be performed that can not be carried out using the original IBM-style constraint [Berger et al. 1996].

Data-driven Search Organization A simple data-driven search organization for the DP search algorithm is developed. The cardinality-synchronuously generated partial hypotheses are pruned in two steps for each processed cardinality: 1) partial hypotheses that cover the same set of source sentence positions are pruned and 2) the partial hypotheses that cover sets $\mathcal{C}$ of source sentence positions of equal cardinality are pruned. These types of pruning become particularly efficient with the above re-ordering restrictions.

Extended Alignment Model The alignment model introduced in Chapter 1 accounts for single source sentence words to be mapped to single target sentence words only. Chapter 3 describes a method for mapping several target language words to a single source language word, using the Viterbi alignment path after the final training iteration of the IBM-model training. A likelihood criterion is used to learn extensions of the baseline lexicon model. The approach presented rests upon similar ideas to learn word trigger pairs in language modeling.

Appendix A

Appendix: Quantification of Re-ordering Restrictions

To quantify the re-ordering restrictions in Section 4.6.2, the **four** non-negative numbers *numskip*, *widthskip*, *nummove*, and *widthmove* are used (*widthskip* corresponds to L, *widthmove* corresponds to R in the Section 4.6.1 - here, we use a more intuitive notation). Within the implementation of the DP search, the restrictions are provided to the algorithm as an input parameter of the following type:

$$S_numskip_widthskip_M_nummove_widthmove.$$

The meaning of the re-ordering string is as follows: the two numbers following S that are separated by a single character "_" describe the way words may be skipped, the two numbers following M describe the way words may be moved during the word re-ordering. The first number after S and M denote the number of positions that may be moved or skipped, e.g. for the translation direction German-to-English 1 position may be skipped and 2 positions may be moved. The second number after S and M restricts the distance a word may be skipped or moved. These "width" parameters restrict the word re-ordering to take place within a "window" of a certain size. The window is defined by the distance between the positions $l_{min}(\mathcal{C})$ and $r_{max}(\mathcal{C})$ as defined in Section 4.6.2. For the notation, a substring headed by S or M may be omitted altogether to indicate that the corresponding re-ordering is not allowed. Any numerical value in the string may be set to INF, denoting that an arbitrary number of positions may be skipped/moved or the moving or skipping distance may be arbitrarily large. The following re-ordering strings are used in this paper:

Word re-ordering string	Description of specified word re-ordering
ϵ (short: **MON**)	The empty string denotes the re-ordering restriction where no re-ordering is allowed.
S_01_04_M_02_10 (short: **GE**)	This string describes the German-to-English word re-ordering. Up to one word may be skipped for at most 4 positions and up to 2 words may be moved up to 10 positions.
S_02_10_M_01_04 (short: **EG**)	This string describes the English-to-German word re-ordering. Up to two words may be skipped for at most 10 positions and up to 1 word may be moved for up to 4 positions.
S_03_INF (short: **S3**)	This string describes the IBM-style word re-ordering given in Section 4.6.3. Up to three words may be skipped for an unrestricted number of positions.
S_INF_INF or M_INF_INF (short: **NO**)	These strings denote the word re-ordering without restrictions.

The word re-ordering strings can be directly used as input parameters to the dynamic programming-based search procedure in order to test different re-ordering restrictions within a single implementation.

Appendix B

Complexity of Different Re-ordering Schemata

This appendix shows the complexity for four different word re-ordering constraints (cf. Appendix A) . The presentation of the different re-ordering constraints is sorted by the increasing complexness of the calculation involved:

- The **MON** re-ordering constraint where no word re-ordering is carried out
- The **NO** re-ordering constraint where no word re-ordering constraints are applied
- The **S3** re-ordering constraint for the IBM-style re-ordering
- The **GE** re-ordering constraint for the translation direction German-to-English

The complexity for the re-ordering scheme **EG** for the translation direction English-to-German can be shown in a similar way.

B.1 Pseudo-Translation Task

In order to calculate the complexity of the search algorithm with respect to different word re-order constraints, we define a pseudo-translation task. In this task, an artificial target and source word "x" is translated into itself. The lexicon probability $p(\text{"}x\text{"}|\text{"}x\text{"})$ and the language model probability $p(\text{"}x\text{"}|\text{"}x\text{"},\text{"}x\text{"})$ are set to be constant values:

$$\begin{aligned} p(\text{"}x\text{"}|\text{"}x\text{"}) &= 1.0 \\ p(\text{"}x\text{"}|\text{"}x\text{"},\text{"}x\text{"}) &= 1.0 \end{aligned}$$

With the above parameter values, an input sentence $\text{"}x\text{"}_1^J$ of length J in the pseudo source language is translated into a target sentence $\text{"}x\text{"}_1^J$ of the same length. We assume that the distortion probabilities $p(j|j', J)$ are non-constant. The problem is then to find the optimal inverted alignment b_1^I, where its choice depends merely on the distortion probabilities $p(j|j', J)$. The above problem is equivalent to searching a shortest TSP tour for a problem

where the cities are linearly ordered and constraints on the order in which the cities may be visited do exist. The complexity to evaluate the following recursive equation is calculated:

$$Q(\mathcal{S}, \mathcal{C}, j) = \max_{\substack{(\mathcal{S}', j') \\ (\mathcal{S}', \mathcal{C} \setminus \{j\}, j') \rightarrow (\mathcal{S}, \mathcal{C}, j) \\ j' \in S \setminus \{j\}}} p(j|j', J) \cdot Q(\mathcal{S}', \mathcal{C} \setminus \{j\}, j')\}, \tag{B.1}$$

where we omit the language model and the translation model since they are set to constant values. The notation is taken from Section 4.5. For the re-ordering, we consider partial hypothesis extensions of the following type (cf. Section 4.6.1):

$$(\mathcal{S}', \mathcal{C} \setminus \{j\}, j') \rightarrow (\mathcal{S}, \mathcal{C}, j). \tag{B.2}$$

For each extension the uncovered position j is added to the coverage set $\mathcal{C}$ and and the verb group state may change from $\mathcal{S}'$ to $\mathcal{S}$. The starting state in the translation lattice is the 3-tuple

$$(Initial, \{\emptyset\}, 0),$$

and the ending state is:

$$(Initial, \{1, \cdots, J\}, j).$$

The search starts at position 0: we assume a special distortion probability $p(\cdot|0)$ for jumping from that position. There are several final states, each of which covers the whole set of source positions $\{1, \cdots, J\}$ - they differ in the last covered position j. The verb group state $\mathcal{S}$ is only need for the GE constraint and is ignored for all other re-ordering constraints. The transitions in Eq. B.2 define a search lattice and the complexity will be computed in terms of the number of arcs in that search lattice. Each arc processed is associated with a probability lookup for the distortion probability $p(j|j', J)$. The search lattice does have a regular structure, similar to the one shown in Fig (4.3). The gridpoints $(\mathcal{C}, j)$ in the lattice can be naturally ordered according to the cardinality of the coverage set $\mathcal{C}$ and arcs do go only between gridpoints for which the difference of the cardinality of the coverage set is exactly 1.
From the complexity for evaluating Eq. B.1, we can obtain the complexity of evaluating Eq. 4.3 which additionally takes into account the translation model probabilities and the trigram language model probabilities. Assuming that the lexicon probability $p(f|e) > 0.0$ for all e and f and the language model probability $p(e|e', e'') > 0.0$ for all e, e', e'', the complexity is increased by a factor E^3, where E is the size of the target vocabulary. Additionally, the use of the full set of IBM parameters involving the fertility parameters, the fertility for the null word and the computation of the center function $center(i)$ may increase the overall complexity slightly.

B.2 Word Re-ordering Complexity

In this section, we show the complexity of the pseudo-translation task for four re-ordering constraints: 1) no word re-ordering (**MON**), 2) word re-ordering without restrictions applied (**NO**), 3) IBM-style word re-ordering (**S3**), and 4) word re-ordering with verb group constraints (**GE**). We compute the complexity for the different re-ordering constraints in

terms of the number of arcs in the translation lattice corresponding to that re-ordering constraint. The number of arcs is given in terms of the allowed pairs $(\mathcal{C}, j)$ - coverage set $\mathcal{C}$ and last covered position j (the finite state component $\mathcal{S}$ is only needed for the GE constraint). The overall number of arcs $\mathcal{A}_{CON}$ for a re-ordering constraint CON is computed as the double sum over all cardinalities c and all corresponding pairs $(\mathcal{C}, j)$:

$$\begin{aligned} \mathcal{A}_{CON} &= \sum_{c=0}^{J} \sum_{\substack{(\mathcal{C},j) \\ j \in \mathcal{C} \wedge c(\mathcal{C})=c}} \delta_{CON}(\mathcal{C}, j) \cdot \sigma_{CON}(\mathcal{C}, j) \\ &= \sum_{c=0}^{J} \sigma_{CON}(c) \sum_{\substack{(\mathcal{C},j) \\ j \in \mathcal{C} \wedge c(\mathcal{C})=c}} \delta_{CON}(\mathcal{C}, j) \end{aligned} \tag{B.3}$$

Here, $\delta_{CON}(\mathcal{C}, j)$ is an indicator function that for each constraint CON equals to 1 if and only if the 2-tuple $(\mathcal{C}, j)$ is allowed for that restriction (we assume $j \in \mathcal{C}$ as the only restriction on j unless otherwise stated). Furthermore, for the initial state we allow to write "$0 \in \{\emptyset\}$" for shortness reasons, meaning that the last visited position 0 is assumed to be contained in the empty coverage set. The indicator function is always 1 for the initial state. For each re-ordering constraint it can be checked that the indicator function is 1 only for states that can be reached from the initial state $(\{\emptyset\}, 0)$.
$\sigma_{CON}(\mathcal{C}, j)$ gives the number of successor states for each 2-tuple $(\mathcal{C}, j)$. Rather than computing the number of successors, we might have computed the number of predecessor states within our complexity calculations - this turned out to be more difficult for some re-ordering restrictions. The number of successor states for a final state is 0: $\sigma(\{1, \cdots, J\}, k) = 0$. The complexity calculation is done in a way, that the number of successors depends only on the cardinality c, hence we write $\sigma_{CON}(c)$ instead of $\sigma_{CON}(\mathcal{C}, j)$ (This simplification is not valid in connection with the verb group state $\mathcal{S}$). When using the σ_{CON} function, we ignore any restrictions due to the sentence length J. For the re-ordering involving the verb group state the number of successors is equal to the cardinality of the set $Succ$ in Table 4.4.
The constraint functions for the MON constraint and the NO constraint are given in Table B.1 and Table B.2. For these constraints, the sum $\mathcal{A}_{CON}$ is carried out exactly. For the S3 constraint and the GE constraint, where the constraint functions are given in Table B.3 and Table B.4, we give upper bounds for the number of arcs in the corresponding translation lattices. For the GE constraint, we have additionally to sum over the verb group states $\mathcal{S}$:

$$\mathcal{A}_{GE} = \sum_{\mathcal{S}} \sum_{c=0}^{J} \sum_{\substack{(\mathcal{C},j) \\ j \in \mathcal{C} \wedge c(\mathcal{C})=c}} \delta_{CON}(\mathcal{S}, \mathcal{C}, j) \cdot \sigma_{CON}(\mathcal{S}, \mathcal{C}, j) \tag{B.4}$$

MON Re-ordering Constraint

For the monotone search, the sum $\mathcal{A}_{MON}$ is calculated as follows: the only allowed pairs $(\mathcal{C}, j)$ are of the form $(\{1, \cdots, j\}, j)$ where a sequence of j consecutive words is covered. There is only a single successor for each allowed 2-tuple. The complexity of the pseudo-

Table B.1: Constraints for the MON word re-ordering.

$\delta_{MON}(\mathcal{C}, j) = 1$	**iff**	$(\mathcal{C}, j) = (\{1, \cdots, j\}, j)$
$\sigma_{MON}(j)$	$=$	1

task is J (since $\sigma(J) = 0$). The constraints are given in Table B.1.

$$\begin{aligned}
\mathcal{A}_{MON} &= \sum_{c=0}^{J} \sigma_{MON}(c) \cdot \sum_{\substack{(\mathcal{C},j) \\ j \in \mathcal{C} \wedge c(\mathcal{C})=c}} \delta_{MON}(\mathcal{C}, j) \\
&= \sum_{c=0}^{J} \sigma_{MON}(c) \cdot 1 \\
&= J
\end{aligned}$$

NO Re-ordering Constraint

For this word re-ordering, the constraints are given in Table B.2. There are neither restrictions on the coverage set $\mathcal{C}$ nor on the last covered position j. We compute the number of arcs $\mathcal{A}_{NO}$ as follows:

$$\begin{aligned}
\mathcal{A}_{NO} &= \sum_{c=0}^{J} \sigma_{NO}(c) \sum_{\substack{(\mathcal{C},j) \\ j \in \mathcal{C} \wedge c(\mathcal{C})=c}} \delta_{NO}(\mathcal{C}, j) \\
&= \sum_{c=0}^{J} (J - c) \cdot \sum_{\substack{(\mathcal{C},j) \\ j \in \mathcal{C} \wedge c(\mathcal{C})=c}} \delta_{NO}(\mathcal{C}, j) \\
&= \sum_{c=0}^{J} (J - c) \cdot c \sum_{\substack{c \\ c=c(\mathcal{C})}} 1 \\
&= \sum_{c=0}^{J} (J - c) \cdot c \cdot \binom{J}{c} \\
&= \sum_{c=0}^{J} \binom{J-2}{c-1} \cdot J \cdot (J-1) \\
&= J \cdot (J-1) \cdot 2^{J-2} \\
&= \mathcal{O}(J^2 \cdot 2^J)
\end{aligned}$$

where the equation: $\sum_{m=0}^{J} \binom{J}{m} = 2^J$ is used. The complexity of the algorithm is still exponential, but much smaller than the complexity for testing all possible permutations of

Table B.2: Constraints for the NO word re-ordering.

$\delta_{NO}(\mathcal{C}, j)$	$=$	1
$\sigma_{NO}(c)$	$=$	$J - c$

Table B.3: Constraints for the S3 word re-ordering.

$\delta_{S3}(\mathcal{C}, j) = 1$	**iff**	$(u(\mathcal{C}) < 3) \parallel (u(\mathcal{C}) = 3 \;\&\; j = r_{max}(\mathcal{C}))$
$\sigma_{S3}(c)$	$=$	4

source positions which is $J!$.

S3 Re-ordering Constraint

The partial hypothesis extensions carried out for the S3 re-ordering constraint are described in in Section 4.6.3. The restrictions on the type of the grid points $(\mathcal{C}, j)$ are given in Table B.3: for a coverage set $\mathcal{C}$ the number of uncovered positions $u(\mathcal{C})$ to the left of the rightmost covered position is restricted. Note, that for the complexity computation there are two cases for a 2-tuple $(\mathcal{C}, j)$: 1) $u(\mathcal{C}) < 3$ and 2) $u(\mathcal{C}) = 3$. For the second case, the last position covered has to be the rightmost covered position. Otherwise , there would have been four uncovered positions violating the S3 constraint for some predecessor partial hypothesis.

$$
\begin{aligned}
\mathcal{A}_{S3} &= \sum_{c=0}^{J} \sigma_{S3}(c) \cdot \sum_{\substack{(\mathcal{C},j) \\ j \in \mathcal{C} \wedge c(\mathcal{C})=c}} \delta_{S3}(\mathcal{C}, j) \\
&< 4 \cdot \sum_{c=0}^{J} \sum_{\substack{(\mathcal{C},j) \\ j \in \mathcal{C} \wedge c(\mathcal{C})=c}} \delta_{S3}(\mathcal{C}, j) \\
&= 4 \cdot \sum_{c=0}^{J} \left\{ \sum_{\substack{(\mathcal{C},j) \\ j \in \mathcal{C} \wedge c(\mathcal{C})=c \\ u(\mathcal{C})<3}} \delta_{S3}(\mathcal{C}, j) + \sum_{\substack{(\mathcal{C},j) \\ j \in \mathcal{C} \wedge c(\mathcal{C})=c \\ u(\mathcal{C})=3}} \delta_{S3}(\mathcal{C}, j) \right\} \\
&= 4 \cdot \sum_{c=0}^{J} \left\{ c \cdot \sum_{\substack{\mathcal{C} \\ c(\mathcal{C})=c \\ u(\mathcal{C})<3}} 1 + \sum_{\substack{\mathcal{C} \\ c(\mathcal{C})=c \\ u(\mathcal{C})=3}} 1 \right\} \\
&< 4 \left\{ J \cdot \left[\sum_{c=0}^{J} \binom{c}{0} + \sum_{c=0}^{J} \binom{c}{1} + \sum_{c=0}^{J} \binom{c}{2} \right] + \sum_{c=0}^{J} \binom{c}{3} \right\} \\
&< 4 \cdot \left\{ J \cdot \left[\binom{J+1}{1} + \binom{J+1}{2} + \binom{J+1}{3} \right] + \binom{J+1}{4} \right\} \\
&< 4 \cdot \left\{ J \cdot \left[(J+1) + (J+1)^2 + (J+1)^3 \right] \;+\; (J+1)^4 \right\} \\
&= \mathcal{O}(J^4)
\end{aligned}
$$

where we used the following approximation and equation:

$$
\binom{n}{k} \;<\; n^k
$$

$$\sum_{m=0}^{n} \binom{m}{k} = \binom{n+1}{k+1}$$

The above result does not mean that the overall number of S3 word re-orderings is restricted to J^4. However, the dynamic programming approach allows to jointly process word re-orderings in order to achieve the above complexity.

GE Re-ordering Constraint

The complexity of the GE re-ordering constraint is computed in a way similar to the S3 constraint. In Table B.4, we have more severe restrictions on the 3-tuples $(\mathcal{S}, \mathcal{C}, j)$ and on the number of successors which results in more complex calculation. For example, the *Initial* state has three successor states (cf. Figure 4.5), namely the states *Initial*, *Skip* and *Move* (the transitions going from one state to itself are not shown in that figure). Each of these states comes with a number of source positions to be covered next. If we stay in the state *Initial* or switch to the state *Skip* there is only 1 possible source position to be covered next. If we switch to the state *Move*, we have a choice of $R-1$ source position to chose from (The L and R notation for the skip width and the move width is taken from the Section 4.6.1). The number of successor for the other states is computed in a similar way. Furthermore, there are strong restrictions on the last covered position: for example, for the state *Initial*, j might be one of the last L positions in the coverage set $\mathcal{C}$. For the states *Skip* and *Cover* there is only one possible last covered source position. For the state *Move*, there are two possible last covered positions: the rightmost covered or the second rightmost covered position. $\mathcal{A}_{GE}$ is calculated as a sum over the four possible verb group state $\mathcal{S}$. Each line after the first line corresponds to one of the four states: *Initial*, *Skip*, *Move*, and *Cover*.

$$\begin{aligned}
\mathcal{A}_{GE} &= \sum_{\mathcal{S}} \sum_{c=0}^{J} \sum_{\substack{(\mathcal{C},j) \\ j \in \mathcal{C} \wedge c(\mathcal{C})=c}} \delta_{GE}(\mathcal{S},\mathcal{C},j) \cdot \sigma_{GE}(\mathcal{S},\mathcal{C},j) \\
&< \sum_{c=0}^{J} 1 \cdot L \cdot (1+1+(R-1)) \\
&\quad + \sum_{c=0}^{J} \binom{L-1}{1} \cdot 1 \cdot (1+1) \\
&\quad + \sum_{c=0}^{J} \left\{ \binom{R-1}{1} \cdot 1 \cdot (R-2) + \binom{R-1}{2} \cdot 2 \cdot 1 \right\} \\
&\quad + \sum_{c=0}^{J} \left\{ \binom{R-1}{1} + \binom{R-1}{2} \right\} \cdot 1 \cdot 1 \\
&= \mathcal{O}(J \cdot \left(R \cdot L + R^2 \right))
\end{aligned}$$

Omitting the details, the complexity of the verb group re-ordering procedure is $\mathcal{O}(J \cdot (R^2 + L \cdot R))$.

Table B.4: Constraints for the GE word re-ordering. The notation is taken from Appendix A.

$$
\begin{aligned}
\delta_{GE}(Initial, \mathcal{C}, j) &= \begin{cases} 1 & \text{if } (\mathcal{C}, j) = (\{1, \cdots, k\}, j) \ \& \ j \in \{k - L, \cdots, k\} \\ 0 & \text{else} \end{cases} \\
\delta_{GE}(Skip, \mathcal{C}, j) &= \begin{cases} 1 & \text{if } (w(\mathcal{C}) \leq L) \ \& \ (u(\mathcal{C}) = 1) \ \& \ (j = r_{max}(\mathcal{C})) \\ 0 & \text{else} \end{cases} \\
\delta_{GE}(Move, \mathcal{C}, j) &= \begin{cases} 1 & \text{if } (w(\mathcal{C}) \leq R) \ \& \ (m(\mathcal{C}) = 1 \vee m(\mathcal{C}) = 2) \ \& \ (j = r_{max}(\mathcal{C})) \\ 1 & \text{if } (w(\mathcal{C}) \leq R) \ \& \ (m(\mathcal{C}) = 2) \ \& \ (j = r_{max}(\mathcal{C} \setminus r_{max}(\mathcal{C})) \\ 0 & \text{else} \end{cases} \\
\delta_{GE}(Cover, \mathcal{C}, j) &= \begin{cases} 1 & \text{if } (w(\mathcal{C}) \leq R) \ \& \ (m(\mathcal{C}) = 1 \vee m(\mathcal{C}) = 2) \ \& \ (j = l_{min}(\mathcal{C}) - 1) \\ 0 & \text{else} \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
\sigma_{GE}(Initial, \mathcal{C}, j) &= 1 + 1 + (R - 1) \\
\sigma_{GE}(Skip, \mathcal{C}, j) &\leq 1 + 1 \\
\sigma_{GE}(Move, \mathcal{C}, j) &= \begin{cases} R - 2 & \text{if } (m(\mathcal{C}) = 1) \\ 1 & else \end{cases} \\
\sigma_{GE}(Cover, \mathcal{C}, j) &= 1
\end{aligned}
$$

Appendix C

Formal Description for the GE Re-ordering Constraint

In this appendix, we give a formal definition of the re-ordering lattice extensions needed for the GE constraint. There are 13 different types of extensions that are of the form given in Eq. (B.2). The extensions are listed in Table C.2: for each type of partial hypothesis $(\mathcal{S}', \mathcal{C} \setminus \{j\}, j')$ the successor partial hypothesis $(\mathcal{S}, \mathcal{C}, j)$ is given. For each extension an uncovered position is added to the coverage set $\mathcal{C}$ of the partial hypothesis and the verb group state $\mathcal{S}$ may change. The extensions are sorted according to the verb group state $\mathcal{S}$ of the target partial hypothesis $(\mathcal{S}, \mathcal{C}, j)$. The last column of Table C.2 comments on the use of the transition in the verb group re-ordering. l, r, r' denote word positions within the left and right verbal brace, where restrictions for the choice of l, r, r' are given in Table C.1. The L and R notation for the skip width and the move width is taken from the Section 4.6.2. The second column of Table C.1 refers to the line number in Table C.2 where the restriction applies.

Table C.1: Technical Restrictions for the choice of $l, r, r'(r \neq r')$ for the GE constraint partial hypothesis extensions in Table C.2.

Restriction	Line to which it applies
$1 \leq l \leq L$	1), 4) , 5) , 7)
$2 \leq l \leq L$	6)
$m+2 \leq r \leq m+R$	7) , 8)
$m+2 \leq r' \leq m+R$	8)
$m+2 \leq r \leq m+R-1$	9) - 12)
$m+2 \leq r' \leq m+R-1$	10), 12)
$r+1 < r' < r+R-1$	13)

Table C.2: Partial hypotheses extensions for the verb group re-ordering using the four verb group states: $\mathcal{I}(Initial), \mathcal{K}(Skip), \mathcal{V}(Move)$, and $\mathcal{U}(Cover)$.

No.	*Transition*	*Comment*
1	$(\mathcal{I}, \{1,\cdots,m-1\}, m-1-l)$ $\rightarrow (\mathcal{I}, \{1,\cdots,m\}, m)$	process sentence monotonically.
2	$(\mathcal{U}, \{1,\cdots,m-2\} \cup \{m\}, m-2)$ $\rightarrow (\mathcal{I}, \{1,\cdots,m\}, m-1)$	skip already processed words of verbal brace.
3	$(\mathcal{U}, \{1,\cdots,m-3\} \cup \{m-1,m\}, m-3)$ $\rightarrow (\mathcal{I}, \{1,\cdots,m\}, m-2)$	
4	$(\mathcal{K}, \{1,\cdots,m\} \setminus \{m-l\}, m)$ $\rightarrow (\mathcal{I}, \{1,\cdots,m\}, m-l)$	process word of left verbal brace that has been skipped.
5	$(\mathcal{I}, \{1,\cdots,m-2\}, m-2-l)$ $\rightarrow (\mathcal{K}, \{1,\cdots,m-2\} \cup \{m\}, m)$	skip word of left verbal brace.
6	$(\mathcal{K}, \{1,\cdots,m-1\} \setminus \{m-l\}, m-1)$ $\rightarrow (\mathcal{K}, \{1,\cdots,m\} \setminus \{m-l\}, m)$	process sentence monotonically remember skipped word.
7	$(\mathcal{I}, \{1,\cdots,m\}, m-l)$ $\rightarrow (\mathcal{V}, \{1,\cdots,m\} \cup \{r\}, r)$	process up to two words of right verbal brace.
8	$(\mathcal{V}, \{1,\cdots,m\} \cup \{r\}, r)$ $\rightarrow (\mathcal{V}, \{1,\cdots,m\} \cup \{r,r'\}, r')$	
9	$(\mathcal{V}, \{1,\cdots,m-1\} \cup \{r\}, r)$ $\rightarrow (\mathcal{U}, \{1,\cdots,m\} \cup \{r\}, m)$	continue to process sentence monotonically while keeping track of already covered positions.
10	$(\mathcal{V}, \{1,\cdots,m-1\} \cup \{r,r'\}, r \mid r')$ $\rightarrow (\mathcal{U}, \{1,\cdots,m\} \cup \{r,r'\}, m)$	
11	$(\mathcal{U}, \{1,\cdots,m-1\} \cup \{r\}, m-1)$ $\rightarrow (\mathcal{U}, \{1,\cdots,m\} \cup \{r\}, m)$	
12	$(\mathcal{U}, \{1,\cdots,m-1\} \cup \{r,r'\}, m-1)$ $\rightarrow (\mathcal{U}, \{1,\cdots,m\} \cup \{r,r'\}, m)$	
13	$(\mathcal{U}, \{1,\cdots,m-2\} \cup \{m,r'\}, r-2)$ $\rightarrow (\mathcal{U}, \{1,\cdots,m\} \cup \{r'\}, m-1)$	

Appendix D

Search Parameter File

```
!! Name              MR_QMS4
!! Parameters
ScanType             FromFile
SplitPunctuation     SPLIT SENTENCES AT PUNCTUATION
# FILES
ScanFile             TEST SENTENCES
French               SOURCE LANGUAGE VOCABULARY WITH UNIGRAM COUNTS
English              TARGET LANGUAGE VOCABULARY WITH UNIGRAM COUNTS
LMFile               PARAMETER FILE FOR LANGUAGE MODEL
LMList               LIST OF BIGRAMS  SEEN IN TRAINING DATA SORTED BY PROBABILITY
TrigramData          LIST OF TRIGRAMS SEEN IN TRAINING DATA
LexFile              LEXICON FILE
ManualFile           MANUAL LEXICON
Fertility            FERTILITY PROBABILITIES
ClassesF             CLASS-MAPPING WORD -> CLASS
ClassesE             CLASS-MAPPING WORD -> CLASS
Distortion           DISTORTION PROBABILITIES
Labeling             LABELING FOR TEST DATA
QMS_STRING           S_01_04_M_02_10
HypFile              GENERATED TRANSLATIONS
# PARAMETER
p_0                  0.98
LexiconSmooth        0.0
LMScale              0.8
InvScale             0.4
# PRUNING
CoverageThr          5.0
CoverageHistogram    1000
CardinalityThr       12.5
CardinalityHistogram 200000
ObservationHistogram 50
ObservationFactor    0.001
```

Appendix E

Mathematical Symbols

The following list of mathematical symbols is not complete. It reports only those symbols that are used throughout the thesis or symbols that are especially imported for the key concepts introduced.

E.1 Translation Model Symbols

f	:	source language word
e	:	target language word
e_1^I	:	target word sequence of length I
f_1^J	:	source word sequence of length J
$p(e_1^I)$	:	target language model probability
$p(f_1^J \mid e_1^I)$	:	translation model probability
$p(J \mid I)$	:	target sentence length probability given source sentence length
$a_1^J = a_1, \cdots, a_j, \cdots, a_J$	:	"hidden" alignments
$j \rightarrow i = a_j$	:	target position a_j is aligned to the j-th source position
$p(f \mid e)$	:	lexicon probability for translating the target word e as the source word f

E.2 Trigger Language Model Symbols

$a \rightarrow b$	:	word trigger pair: word a triggers word b
$p_{ab}(w\|h)$	:	probability of an extended model using $a \rightarrow b$
$q(b\|a)$	:	single trigger interaction parameter using the trigger pair $a \rightarrow b$
$F_{ab} - F_0$	:	likelihood improvement of the extended model

E.3 Extended Lexicon Model Symbols

(f, e)	:	single word lexicon dependence between f and e
$(f_0, \tilde{e})$	:	extended lexicon dependence between a single source word f_0 and a sequence of target words $\tilde{e}$
$(f_0, \tilde{e}, \tilde{f})$	:	extended lexicon dependence between a single source word f_0 and a sequence of target words $\tilde{e}$ as well as a sequence of source words $\tilde{f}$ to the left of f
$\tilde{e}_0$	:	main translation on the alignment path for each extension $(f_0, \tilde{e}, \tilde{f})$
$p_{(f_0,\tilde{e},\tilde{f})}(f\|e_1^I)$	:	probability of an extended lexicon model using the extension $(f_0, \tilde{e}, \tilde{f})$
$q(f_0\|\tilde{e}, \tilde{f})$	:	single extension parameter for the extension $(f_0, \tilde{e}, \tilde{f})$
$F_{(f_0,\tilde{e},\tilde{f})} - F_{e_0}$	:	likelihood improvement of the extended model

E.4 Translation Search Symbols

$i \rightarrow b_i = j$	:	inverted alignment for search
$\mathcal{C}$	:	coverage vector $\mathcal{C}$ of already processed source sentence positions
$p(j\|j', J)$	:	distortion probability
$p(e\|e', e'')$	:	trigram language model probability
$D(\mathcal{C}, j)$	:	auxiliary quantity for the Held & Karp algorithm
$Q_{e'}(e, \mathcal{C}, j)$	:	auxiliary quantity for DP-based search algorithm
$S_numskip_widthskip_$ $M_nummove_widthmove$	:	string describing the word re-ordering constraints
$\mathcal{S}$	:	re-ordering state to restrict word re-ordering
$(\mathcal{S}, \mathcal{C}, j)$	:	search state 3-tuple
MON	:	re-ordering constraint: no word re-ordering is allowed
GE	:	re-ordering constraint for the translation direction German-to-English: S_01_04_M_02_10
EG	:	re-ordering constraint for the translation direction English-to-German: S_02_10_M_01_04
S3	:	re-ordering constraint for the IBM-style re-ordering restriction
NO	:	word re-ordering without restrictions

Pruning Parameters

$\hat{Q}(\mathcal{C})$	:	best hypothesis score over all hypotheses of given coverage vector $\mathcal{C}$
$\hat{Q}(c)$	:	best hypothesis score over all hypotheses with a coverage vector $\mathcal{C}$, such that $\|\mathcal{C}\| = c$
$t_{\mathcal{C}}, n_{\mathcal{C}}$	:	translation and histogram coverage pruning
t_c, n_c	:	translation and histogram cardinality pruning
t_o, n_o	:	observation pruning

Bibliography

[Al-Onaizan et al. 1999] Al-Onaizan, Yaser, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan I. Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. 1999. Statistical Machine Translation: Final Report,Johns Hopkins University 1999 Summer Workshop (WS 99) on Language Engineering, Center for Language and Speech Processing, Baltimore, MD, USA.

[Alshawi et al. 1997] H. Alshawi and F. Xiang. 1997. English-to-Mandarin Speech Translation with Head Transducers. In *Proceedings of the Workshop on Spoken Language Translation (in conjunction with ACL/EACL'97)*, pp. 54-60, Madrid, Spain, July.

[Alshawi et al. 1998] H. Alshawi,S. Bangalore, and S. Douglas. 1998. Automatic Acquisition of Hierarchical Transduction Models for Machine Translation. In *Proceedings of the 36th Annual Conf. of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING/ACL'98)*, pp. 41–47, Montreal, Canada, August.

[Amengual et al. 1997] J. C. Amengual, J. M. Benedi, F. Casacuberta, A. Castano, A. Castellanos, D. Llorens, A. Marzal, F. Prat, E. Vidal, and J. M. Vilar. 1997. Error Correcting Parsing for Text-to-Text Machine Translation using Finite State Models. In *Proceedings of the Conf. on Theoretical and Methodological Issues in Machine Translation*, pp. 135–142, Santa Fe, New Mexico, July.

[Bahl et al. 1984] L.R. Bahl, F. Jelinek, R.L. Mercer and A. Nadas. 1984. Next Word Statistical Predictor. *IBM Techn. Disclosure Bulletin*, 27(7A), pp. 3941–3942.

[Bellmann 1957] R. Bellmann. 1957. *Dynamic Programming*. Princeton University Press, Princton, NJ.

[Berger et al. 1996] A.L. Berger, S.A. Della Pietra, and V.J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, Vol. 22, No. 1, pp. 39–71, March.

[Baum 1972] L. E. Baum. 1972. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities*, Vol. 3, pp. 1–8.

[Berger et al. 1994] A.L. Berger, P.F. Brown, J. Cocke, S.A. Della Pietra, V.J. Della Pietra, J.R. Gillett, J.D. Lafferty, R.L. Mercer, H. Printz, and L. Ures. 1994. The Candide

System for Machine Translation. In *Proceedings of the ARPA Human Language Technology Workshop*, Plainsboro, NJ, Morgan Kaufmann Publishers, pp. 152-157, San Mateo, CA, March.

[Berger et al. 1996] A. L. Berger, P. F. Brown, S.A. Della Pietra, V.J. Della Pietra, J.R. Gillett, A.S. Kehler, and R.L. Mercer. 1996. Language Translation apparatus and method of using context-based translation models. United States Patent, Patent Number 5 510 981, April.

[Brown et al. 1990] P.F. Brown, J. Cocke, V.J. Della Pietra, S.A. Della Pietra, F. Jelinek, J. Lafferty, R.L. Mercer, and P.S. Roosin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics*, Vol. 16, No. 2, pp. 79–85.

[Brown et al. 1992] P.F. Brown, P.V. deSouza, V.J. Della Pietra, and R.L. Mercer. 1992. Class-based n-gram Models of Natural Language. *Computational Linguistics*, Vol. 18, No. 4, pp. 467–479.

[Brown et al. 1993] P.F. Brown, V.J. Della Pietra, S.A. Della Pietra, and R.L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, Vol. 19, No. 2, pp. 263–311.

[Casacuberta 2000] F. Casacuberta. 2000. Inference of finite-state transducers by using regular grammars amd morphisms. In *Lecture Notes in Computer Sciences*, 5th Int. Colloqium on Grammatical Inference (ICGI'2000), Springer-Verlag, Vol. 1891, pp. 1–14, Lisboa, Portugal, September.

[Collins et al. 1999] . Collins, J. Hajič, L. A. Ramshaw, and C. Tillmann. 1999. A statistical parser for Czech. In *Proceedings of the 37th Annual Conf. of the Association for Computational Linguistics (ACL'99)* , pp. 505–512, College Park, MD, June.

[Dagan et al. 1993] I. Dagan, K. W. Church, and W. A. Gale. 1993. Bilingual Word Alignment for Machine Aided Translation. In *Proceedings of the Workshop on Very Large Corpora*, pp. 1-8, Columbus, OH.

[Della Pietra et al. 1994] S. Della Pietra, V. Della Pietra, J. Gillett, J. Lafferty, H. Printz, and L. Ures. 1994. Inference and Estimation of a Long-Range Trigram Model. In *Lecture Notes in Artificial Intelligence*, Grammatical Inference and Applications (ICGI'94), Springer-Verlag, pp. 78–92, Alicante, Spain, September.

[Dempster et al. 1977] A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. In *Journal of the Royal Statistical Society*, Vol. 39, No. 1, pp. 1–38.

[Fung et al. 1994] P. Fung and K.W. Church. 1994. K-vec: A New Approach for Aligning Parallel Texts. In *Proceedings of 15th International Conference on Computational Linguistics (COLING'94)*, pp. 1096-1102, Kyoto, Japan.

[Gale et al. 1991] W. Gale and K. Church. 1991. Identifying word correspondences in parallel texts. In *Proceedings of the Forth Darpa Speech and Natural Language Processing Workshop*, pp. 152–157 Pacific Grove, CA.

[Germann et al. 2001] U. Germann, M. Jahr, K. Knight, D. Marcu, and K. Yamada. 2001. Fast Decoding and Optimal Decoding for Machine Translation. In *Proc. of the 38th Annual Conf. of the Association for Computational Linguistics (ACL'2001)*, pp. 228–235, Toulouse,France, July.

[Held,Karp 1962] Held and Karp. 1962. A Dynamic Programming Approach to Sequencing Problems. *J. SIAM*, Vol. 10, No.1, pp. 196–210.

[García et al. 1998a] I. García, F. Casacuberta, and H. Ney. 1998. An Iterative DP-based Search Algorithm for Statistical Machine Translation. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP'98)*, pp. 1135–1139, Sydney, Australia, November.

[García et al. 1998b] I. García-Varea and F. Casacuberta. 1998. Statistical Translation in Limited Domain Tasks. In *Proceedings of the Workshop on Machine Translation*, pp. 33–43, Saarbrücken, Germany, August.

[Horiguchi et al. 1997] K. Horiguchi and A. Franz. 1997. A Formal Basis for Spoken Language Translation by Analogy. In *Proceedings of the Workshop Spoken Language Translation (in conjunction with (ACL/EACL'97)*, pp. 32–39, Madrid, Spain, July.

[Hutchins,Somers 1992] . W. J. Hutchins and H. L. Somers. 1992. *An Introduction to Machine Translation.* Academic Press.

[Jelinek 1976] F. Jelinek. 1976. Speech Recognition by Statistical Methods. In *Proceedings of the IEEE*, Vol. 64, pp. 532-556, April.

[Jelinek 1991] F. Jelinek. 1991. Self-Organized Language Modeling for Speech Recognition. *Readings in Speech Recognition.* A. Waibel and K.F. Lee (eds.), pp. 450–506, Morgan–Kaufmann.

[Jongen,Triesch 1988] H. Jongen and E. Triesch. 1998. Lecture notes "Vorlesung Optimierung B." Augustinus Buchhandlung Aachen.

[Jurafsky,Martin 2000] D. Jurafsky and J. H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recogntion.* Prentice Hall.

[Röscheisen,Kay 1993] M. Kay and M. Röscheisen. 1993. Text-Translation Alignment. *Computational Linguistics*, Vol. 19, No. 1, pp. 121-142.

[Katz 1987] S.M. Katz. 1987. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 35, pp. 400–401, March.

[Knight 1999] K. Knight. 1999. Decoding Complexity in Word-Replacement Translation Models. *Computational Linguistics*, Vol. 25, No. 4, pp. 607-615.

[Markov 1913] A. A. Markov. 1913. An example of a statistical investigation of the text of "Eugene Onegin" illustrating the dependence between samples in chain. Izvestia Imperatorskoi Akademii Nauk (Bulletin de l'Académie Impériale des Sciences de St.Pétersbourg), Vol. 7, pp. 153-162.

[Lau et al. 1993] R. Lau, R. Rosenfeld, and S. Roukos. 1993. Trigger-Based Language Models: A Maximum Entropy Approach. In *Proceedings IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'93)*, pp. 45–48, Minnesota, MN, April.

[Lau et al. 1993] R. Lau, R. Rosenfeld, and S. Roukos. 1993. Adaptive Language Modeling Using the Maximum Entropy Approach. In *Proceedings of the ARPA Human Language Technology Workshop*, pp. 108-113, Morgan–Kaufmann, March.

[Lavie et al. 1995] A. Lavie, L. Levin, A. Waibel, D. Gates, M. Gavalda, and L. Mayfield. 1995. JANUS: Multi-lingual translation of spontaneous speech in a limited domain. In *Proceedings of the 2nd Conference of the Association for Machine Translation in the Americas*, pp. 252-255, Montreal, Canada, October.

[Ney et al. 1992] H. Ney, D. Mergel, A. Noll, and A. Paeseler. 1992. Data Driven Search Organization for Continuous Speech Recognition in the SPICOS System. *IEEE Transaction on Signal Processing*, Vol. 40, No. 2, pp. 272-281, February.

[Ney 1992] H. Ney. 1992. A Comparative Study of Two Search Strategies for Connected Word Recognition: Dynamic Programming and Heuristic Search. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 5, pp. 586-595, May.

[Ney et al. 1994] H. Ney, U. Essen, and R. Kneser. 1994. On Structuring Probabilistic Dependencies in Language Modeling. *Computer, Speech and Language*, Vol. 8, pp. 1-38.

[Ney et al. 1995] H. Ney, M. Generet, and F. Wessel. 1995. Extensions of Absolute Discounting for Language Modeling. In *Proceedings of the Fourth European Conference on Speech Communication and Technology (Eurospeech'95)*, pp. 1245–1248, Madrid, September.

[Ney 1996] H. Ney. 1996. Lecture notes "Suchverfahren der Spracherkennung". RWTH Aachen.

[Ney 1999] H. Ney. 1999. Speech Translation: Coupling of Recognition and Translation. In *Proceedings of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'99)*, pp. 517-520, Phoenix, AR, March.

[Ney et al. 2000] H. Ney, S. Niessen, F. J. Och, H. Sawaf, C. Tillmann, and S. Vogel. 2000. Algorithms for Statistical Translation of Spoken Language. *IEEE Transactions on Speech and Audio Processing*, Vol. 8, No. 1, pp. 24–36, January.

[Niessen et al. 1998] S. Niessen, S. Vogel, H. Ney, and C. Tillmann. 1998. A DP-Based Search Algorithm for Statistical Machine Translation. In *Proceedings of the 35th Annual Conference of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING/ACL'98)*, pp. 960-967, Montreal, Canada, August.

[Niessen et al. 2000a] S. Niessen and H. Ney. 2000. Improving SMT quality with morpho-syntactic analysis. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING'00)*, pp. 1081–1085, Saarbrücken, Germany, July-August.

[Niessen et al. 2000b] S. Niessen, F.J. Och, G. Leusch, and H. Ney. 2000. An Evaluation Tool for Machine Translation: Fast Evaluation for MT Research. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, pp. 39–45, Athens, Greece, May.

[Och et al. 1998] F. J. Och and H. Weber. 1998. Improving Statistical Natural Language Translation with Categories and Rules. In *Proceedings of the 35th Annual Conference of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING/ACL'98)*, pp. 985-989, Montreal, Canada, August.

[Och et al. 1999] F. J. Och, C. Tillmann, and H. Ney. 1999. Improved Alignment Models for Statistical Machine Translation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (in conjunction with ACL'99)*, pp. 20–28, College Park, MD, USA, June.

[Och et al. 2000a] F. J. Och and H. Ney. 2000. A Comparison of Alignment Models for Statistical Machine Translation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING'00)*, pp. 1086–1090, Saarbrücken, Germany, July-August.

[Och et al. 2000b] F. J. Och and H. Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of the 38th Annual Conf. of the Association for Computational Linguistics (ACL'00)*, pp.440–447, Hong Kong, China, October.

[Och et al. 2001] F. J. Och, N. Ueffing, and H. Ney. 2001. An Efficient (A)* Search Algorithm for Statistical Machine Translation. In *Proceedings Data-Driven Machine Translation Workshop, 39th Annual Conf. of the Association for Computational Linguistics (ACL'01)*, pp. 55–62, Toulouse, July.

[Ortmanns et al. 1997] S. Ortmanns, H. Ney, and X. Aubert. 1997. A word graph algorithm for Large Vocabulary Continuous Speech Recognition. *Computer, Speech and Language*, Vol. 11,No. 1, pp. 43–72, January.

[Paul et al. 1992] D.B. Paul and J.B. Baker. 1992. The Design for the Wall Street Journal-based CSR Corpus. In *Proceedings of the DARPA SLS Workshop*, pp. 357–361, February.

[Rabiner,Juang 1993] L. Rabiner and B.-H. Junang. 1993. *Fundamentals of Speech Recognition*. Prentice Hall.

[Reithinger et al. 1996] N. Reithinger, R. Engel, M. Kipp, and M. Klesen. 1996. Predicting Dialogue Acts for a Speech-To-Speech Translation System. *DFKI Saarbrücken. Technical Report 151*, Verbmobil Project, August.

[Rosenfeld 1994] R. Rosenfeld. 1994. Adaptive Statistical Language Modeling: A Maximum Entropy Approach. *Ph.D. thesis*, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, CMU–CS–94–138.

[Tillmann et al. 1996a] C. Tillmann and H. Ney. 1996. Selection Criteria for Word Triggers in Language Modeling. In *Lecture Notes in Artificial Intelligence*, Int. Colloquium on Grammatical Inference (ICGI'96), Springer-Verlag, pp. 95-106, Montpellier, France, September.

[Tillmann et al. 1996b] C. Tillmann and H. Ney. 1996. Statistical Language Modeling and Word Triggers. In *Proceedings of the International Workshop "Speech and Computer" (SPECOM'96)*, pp. 22-27, St.Petersburg, Russia, October.

[Tillmann et al. 1997a] C. Tillmann, S. Vogel, H. Ney, and A. Zubiaga. 1997. A DP-based Search Using Monotone Alignments in Statistical Translation. In *Proceedings of the 35th Annual Conf. of the Association for Computational Linguistics (ACL/EACL'97)*, pp. 289-296, Madrid, Spain, July.

[Tillmann et al. 1997b] C. Tillmann and H. Ney. 1997. Word Trigger and the EM Algorithm. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL'97)*, pp. 117-124, Madrid, Spain, July.

[Tillmann et al. 1997c] C. Tillmann, S. Vogel, H. Ney, A. Zubiaga, and H. Sawaf. 1997. Accelerated DP Based Search for Statistical Translation. In *Proceedings of the 5th European Conf. on Speech Communication and Technology (Eurospeech'97)*, pp. 2667-2670, Rhodos, Greece, September.

[Tillmann et al. 2001] C. Tillmann, S. Vogel, H. Ney, and H. Sawaf. 2001. Statistical Translation of Text and Speech: First Results with the RWTH System. Accepted for *Machine Translation Journal*, Kluwer Verlag.

[Tillmann et al. 2000] C. Tillmann and H. Ney. 2000. Word Re-Ordering and DP-based Search in Statistical Machine Translation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING'00)*, pp. 850–856, Saarbrücken, Germany, July-August.

[Vidal et al. 1996] E. Vidal. 1996. Final report of Esprit Research Project 20268 (EuTrans): Example-Based Understanding and Translation Systems.

[Vidal 1997] E. Vidal. 1997. Finite-State Speech-to-Speech Translation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'97)*, pp. 111-114, Munich, Germany, April.

[Vidal et al. 2000] E. Vidal. 2000. Final report of Esprit Research Project 30268 (EuTrans): Example-Based Language Translation Systems.

[Vilar et al. 1996] J. M. Vilar, E. Vidal, and J. C. Amengual. 1996. Learning Extended Finite State Models for Language Translation. In *Proceedings of the 12th European Conference on Artificial Intelligence*, Budapest, Hungary.

[Vogel et al. 1996] S. Vogel, H. Ney, and C. Tillmann. 1996. HMM Based Word Alignment in Statistical Translation. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, pp. 836-841, Copenhagen, Denmark, August.

[Vogel et al. 2000] S. Vogel and H. Ney. 2000. Translation with Cascaded Finite-State Transducern. In *Proceedings of the 38th Annual Conf. of the Association for Computational Linguistics (ACL'00)*, pp. 23–30, Hong Kong, October.

[Wahlster 2000] Wahlster, W. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation.* Springer Verlag, Berlin.

[Wang et al. 1997] Y.-Y. Wang and A. Waibel. 1997. Decoding Algorithm in Statistical Translation. In *Proceedings of the 35th Annual Conference of the Association for Computational Linguistics (ACL/EACL'97)*, pp. 366-372, Madrid, Spain, July.

[Wang et al. 1998a] Y.-Y. Wang and A. Waibel. 1998. Modeling with Structures in Statistical Machine Translation. In *Proceedings of the 36th Annual Conference of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING/ACL'98)*, pp. 2775–2778, Sydney, Australia, November.

[Wang et al. 1998b] Y.-Y. Wang and A. Waibel. 1998. Fast Decoding For Statistical Machine Translation. In *Proceedings of the International Conference on Speech and Lanaguage Processing (ICSLP'99)*, pp. 1357-1363, Montreal, Canada, August.

[Wessel et al. 1998] F. Wessel, W. Macherey, and R. Schlüter. 1998. Using Probabilities as Confidence Measures. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'98)*, pp. 225-228, Seattle, WA, May.

[Wu 1996] D. Wu. 1996. A Polynomial-Time Algorithm for Statistical Machine Translation. In *Proceedings of the 34th Annual Conference of the Association for Computational Linguistics (ACL'96)*, pp. 152-158, Santa Cruz, CA, June.

[Wu et al. 1998] D. Wu and H. Wong. 1998. Machine Translation with a Stochastic Grammatical Channel. In *Proceedings of the 36th Annual Conference of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING/ACL'98)*, pp. 1408–1414, Montreal, Canada, August.

Lebenslauf

Name:	Christoph Tillmann
Adresse:	Wolfshovener Str. 133
	52428 Jülich
Geburtstag:	11. November 1967
Geburtsort:	Jülich
Eltern:	Hubert Tillmann
	Ursula Tillmann, geb. Bongartz

Schulbildung

1974-1978	Besuch der Grundschule in Jülich-Welldorf
1978-1984	Besuch der Realschule in Jülich
1984-1987	Besuch des Gymnasiums Haus Overbach in Jülich-Barmen

Wehrdienst

1987-1988	Wehrdienst im Frontnachrichten Aufklärungsbatallion in Köln-Wahn

Beruflicher Werdegang

Oktober 1988	Beginn des Studiums an der RWTH Aachen
September 1990	Vordiplom in Informatik
September 1991- August 1992	DAAD-Stipendiat an der staatlichen Universität in St.Petersburg, Rußland
April 1993	Heirat mit Lada Kornilzewa
April 1991 - März 1995	Stipendiat des Cusanuswerks, der Studienstiftung der deutschen katholischen Bischöfe
Januar 1996	Diplom in Informatik
November 1996	Geburt des Sohnes Maxim Tillmann
Juli-August 1998	Language Engineering Workshop an der Johns Hopkins Universität in Baltimore
März 1996 - Oktober 2000	Promotionsstudent am Lehrstuhl für Informatik VI an der RWTH Aachen
Mai 2001	Doktorprüfung
November 2001	Geburt der Tochter Valentina Tillmann
November 2000 - Heute	Post Doc am IBM T.J. Watson Research Center

Fremdsprachen: Fließend in Deutsch, Russisch und Englisch. Gute Französischkenntnisse.