Grammar Inference and Statistical Machine Translation

Ye-Yi Wang

CMU-LTI-98-160

School of Computer Science
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Alex Waibel, Chair
Jaime Carbonell
John Lafferty
Wayne Ward
Allen Gorin (AT&T Research Laboratories)

Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

Keywords: Machine Translation, Statistical Machine Translation, Grammar Inference, Translation Modeling, Language Modeling, Decoding Algorithms, Decoder, Search, Alignment, Alignment Models, Word-Based Alignment, Structure-Based Alignment

Abstract

NLP researchers face a dilemma: on one side, it is unarguably accepted that languages have internal structure rather than strings of words. On the other side, they find it very difficult and expensive to write grammars that have good coverage of language structures.
Statistical machine translation tries to cope with this problem by ignoring language structures and using a statistical models to depict the translation process. Most of the translation models are word-based. While the approach has achieved surprisingly good performance comparable to the best commercial systems, many questions remain in the machine translation community. Can the statistical word-based translation still perform well on language pairs with radically different linguistic structures? How would it function with less training data or with spoken languages?
The thesis work investigated these questions. In summary, word-based alignment model is a major cause of errors in German-English statistical spoken language translation. To account for this problem, a structure-based alignment model is introduced. This new model takes advantages of a bilingual grammar inference algorithm, which can automatically acquire shallow phrase structures used by the model. The structure-based model can directly depict the structure difference between English and German spoken languages. It also results in focused learning of word alignment, therefore it can alleviate the sparse data problem. The structure-based model achieved 11 percent error reduction over the state-of-the-art statistical machine translation models.

Acknowledgements

When the disastrous Culture Revolution ended 20 years ago, I just graduated from primary school in China. Education was ignored then. Many good teachers were still in the countryside to be "re-educated" by peasants; no reading materials were available for young students. It was my parents who taught me the importance of education; found for me good teachers for tutoring. My father even hand-copied many books for me to read. I always feel that I have no excuse for not doing my best in the pursuit of excellence. My parents are proud of their children. And I would like to tell them that we are so proud and lucky to have them as our parents.

I wish to thank my advisor, Dr. Alex Waibel for his support, encouragement and advice during my graduate studies. Although he is a busy director of a huge research group, he is always available for discussion. I am also grateful to John Lafferty, from whom I learned a lot about statistical language processing and the IBM machine translation system. He also made detailed comments on my thesis draft. I also wish to thank the other members of my committee, Dr. Jaime Carbonell, Dr. Wayne Ward and Dr. Allen Gorin, for their insightful analysis and valuable comments concerning my work.

I would like to thank my colleagues and friends in the Language Technolodgies Institute and the Interactive Systems Laboratories. I would like to specifically thank Klause Ries, Marsal Gavaldà, Klause Zechner, Bernhard Suhm, Laura Tomokiyo, Jie Yang and Weiyi Yang for their help. I also whsh to thank Debbie Clement, Radha Rao, Sharon Burks and Cathy Morrow for their help.

Most importantly though, I would like to thank my wife Xuebo, and my daughter Susan, for their love, understanding, support and encouragement; for accompanying me through all these difficult but happy years. The life as a graduate student's family is hard. I can still remember that Susan didn't want to talk to me because I failed to be home when she needed me in her sick night. I was so moved and encouraged later when she told her teacher: "I want to be a computer scientist when I grow up, just like my dad." I cannot forget that Xuebo cooked delicious foods and waited for me home until 2AM for many times. My family made my life

these years happy and joyful, but I owed a lot to them. This thesis is dedicated to them.

Contents

Chapter 1

Introduction

In his recent survey article (Knight, 1997) on knowledge acquisition for machine translation, Kevin Knight noted:

"This system (the IBM statistical machine translation system CANDIDE) performs as well as the best commercial systems, with no handbuilt knowledge bases! That's the good news. Where to go from here? It is unclear whether the *outstanding problems can be addressed within word-for-word framework*, via better statistical modeling or more training data. It is also unclear how this method would perform on language pairs like Vietnamese/English, *with radically different linguistic structure* and *less bilingual data* on line."

While acknowledging the great success of statistical machine translation, here Knight also raised three questions about statistical machine translation:

1. How far can the word-for-word statistical machine translation go? — Strictly speaking, statistical machine translation is not word-for-word translation. Although the translation models are more or less based on word/cept alignments, at least language models introduce context constraints into the translation. So from now on I will use *word-based translation model* instead of *word-for-word translation*.

2. What performance can we expect from a statistical translator if the language pair has radically different structures?

3. What kind of performance can we expect if there is less training data?

This thesis investigates these questions about statistical machine translation raised in the machine translation society. It also investigates an additional question about the performance of the word-based translation models for spoken language. Before I go to detailed analysis of the problems, designing of models, description of experiments, and evaluation of performance, I will first review the statistical machine translation approach in the background of different language technologies.

1.1 Background: Rationalist vs. Empiricist Language Technologies

Rationalism and empiricism are two rival approaches in computational linguistics. The philosophical controversy between the two lies in different ways of linguistic knowledge acquisition. The former argues that a significant part of human language knowledge is innate (presumably inherited genetically), while the latter argues that language knowledge derives solely from sensory input and a few elementary operations of association and generalization. As reflected in language technology, the rationalist approach tries to extract human language knowledge and model it formally with the expertise of language engineers. The models are usually expressed with rules in a symbolic system. The empiricist approach, on the other hand, suggests that models of language can be obtained simply by examining a large amount of language data and using procedures (usually statistical or connectionist) of association and inductive generalization.

Because of the philosophical divide, the two approaches differ in the following aspects:

1. The rationalist approaches are generally *theory-based*. The models (grammars) are based on linguistic theories, such as GB, LFG, GPSP, or HPSG. The empiricist approaches are *corpus-based* or more specifically *statistics-based*. By observing a great amount of language data, a model can associate a language construction with a real number representing its likelihood.

2. Since grammar theories generally focus on language competence, the rationalist approaches are therefore *competence-oriented*. On the other hand, the empiricist approaches deal with the performance data directly. Therefore they are intrinsically *performance oriented*.

3. The models in rationalist approaches are generally *engineered* by formally representing human expertise. The models in the empiricist approaches are generally *learnable* from corpora.

Empiricism was dominant in computational linguistics before the 1960's. Between the 60's and the middle of the 80's, the rationalist approach has dominated the field. This was due to the widespread acceptance of Noam Chomsky's theory of innate language faculty. From the late 80's, the empiricist approach resurged as an important language technology, powered by the skyrocketing capacity of modern computers and the increasing availability of large, machine-readable corpora. An example of the resurgence of empiricist approach is statistical machine translation. Although Warren Weaver suggested that the translation problem be attacked with an information theoretic approach as early as 1949, it was not feasible with the computational and storage power of computers at that time. Symbolic machine translation approach thus dominated in the field of machine translation. In recent years, with the increasing availability of on-line corpora and growing computational powers, statistical machine translation is becoming a reality (Brown et al., 1990; Brown et al., 1993; Wang and Waibel, 1997a; Tillmann et al., 1997). Compared to the traditional rationalist approach, statistical machine translation does not require expertise in language knowledge engineering, and it is robust to noise, which is often observed in spoken language data.

1.2 Statistical Machine Translation

Statistical machine translation is based on a channel model. Given a sentence $\mathbf{T}$ in one language (German) to be translated into another language (English), it considers $\mathbf{T}$ to be the target of a communication channel, and its translation $\mathbf{S}$ to be the source of the channel. Hence the machine translation task is to recover the source from the target. Basically every English sentence is a possible source for a German target sentence. If we assign a probability $\Pr(\mathbf{S} \mid \mathbf{T})$ to each pair of sentences $(\mathbf{S}, \mathbf{T})$, then the problem of translation is to find the source $\mathbf{S}$ for a given target $\mathbf{T}$, such that $\Pr(\mathbf{S} \mid \mathbf{T})$ is the maximum. According to the Bayes rule,

$$\Pr(\mathbf{S} \mid \mathbf{T}) = \frac{\Pr(\mathbf{S}) \Pr(\mathbf{T} \mid \mathbf{S})}{\Pr(\mathbf{T})} \tag{1.1}$$

Since the denominator is independent of $\mathbf{S}$, the translation of $\mathbf{T}$ is therefore

$$\hat{\mathbf{S}} = \arg\max_{\mathbf{S}} \Pr(\mathbf{S}) \Pr(\mathbf{T} \mid \mathbf{S}). \tag{1.2}$$

There are three subproblems in statistical machine translation:

- Modeling Problem: How can the process of generating a sentence in a source language be depicted, and what process is used by the channel to generate a target sentence upon receiving a source sentence? The former is the problem of language modeling, and the latter is the problem of translation modeling. They provide a framework to calculate $\Pr(\mathbf{S})$ and $\Pr(\mathbf{T} \mid \mathbf{S})$ in (1.2).

- Learning Problem: Given a statistical language model $\Pr(\mathbf{S})$ and a statistical translation model $\Pr(\mathbf{T} \mid \mathbf{S})$, how can the parameters in these models be estimated from a bilingual corpus of parallel sentences? Moreover, it there a way to automatically learn the structure of the translation model?

- Decoding Problem: With a fully specified (framework and parameters) language and translation model, given a target sentence $\mathbf{T}$, how can the source sentence $\hat{\mathbf{S}}$ that satisfies (1.2) be most efficiently identified.

Remark 1 *Throughout this thesis I will use "source language/sentence" for the language/sentence at the source end of the channel, and use "target language/sentence" for the language/sentence at the target end, i.e., the source language is the one that we are translating to, and the target language is the one that we are translating from. Unfortunately this terminology in statistical machine translation is somewhat confusing: it is opposed to what source/target means in the traditional usage of machine translation. When it is necessary, I will use "input/output language" to refer to the "source/target language" in the traditional machine translation terminology.*

1.2.1 Modeling

Most statistical machine translation systems use ngram (Jelinek, 1990) for language modeling. Translation models rely on the concept of *alignment*. Many alignment translation models assume that a target sentence is generated from a source sentence word by word. Here a word may differ from what "word" traditionally means. It does not necessarily to be a dictionary

entry. Often parallel sentences are preprocessed or transduced, so the source and target sentences may look similar. The transduced sentences may contain a "word" that could be several source words in the original sentence. For example, in the IBM system, *do_not* is treated as a word rather than two separate words in English, and pairs like *ne ... pas, ne ... rien*, etc., are combined into single words in French (Brown et al., 1992b). A target sentence word can be *aligned* with the source sentence word that produces it. In an alignment, each target word can align with only one source sentence word. So far, most of the statistical machine translation systems use word-based alignment model (Brown et al., 1993; Vogel, Ney, and Tillman, 1996; Wang and Waibel, 1997a), and no structure is involved in the alignment. (Brown et al., 1993) introduced five different word-based alignment models for translation modeling. The detailed description of the models can be found in Appendix A. One of the major contributions in this thesis is a structure-based alignment model. Figure 1.1 sketches the difference between different IBM models and the structure-based model.

1.2.2 Parameter Estimation

Since the alignment between a paired source/target sentence is not marked in a parallel training corpus, the maximum likelihood (ML) estimator cannot be directly applied. EM algorithm (Dempster, Laird, and Rubin, 1977) is an effective ML estimator for statistical models with hidden variables, which can be applied to the translation models, where alignments are the hidden variables. (Brown et al., 1993) described how to use the EM algorithm to estimate the parameters in the five translation models in the IBM statistical machine translation system.

1.2.3 Decoding

The decoding algorithm is another crucial part in statistical machine translation. Its performance directly affects translation quality and efficiency. Without a reliable and efficient decoding algorithm, a statistical machine translation system may miss the best translation of an input sentence even if it is perfectly predicted by the model. (Berger et al., 1996) described the stack decoder used by the IBM statistical machine translation system. (Wang and Waibel, 1997a) described a faster stack decoding algorithm for the IBM translation model 2 as well as a simplified model. (Tillmann et al., 1997) used a dynamic programming search algorithm for

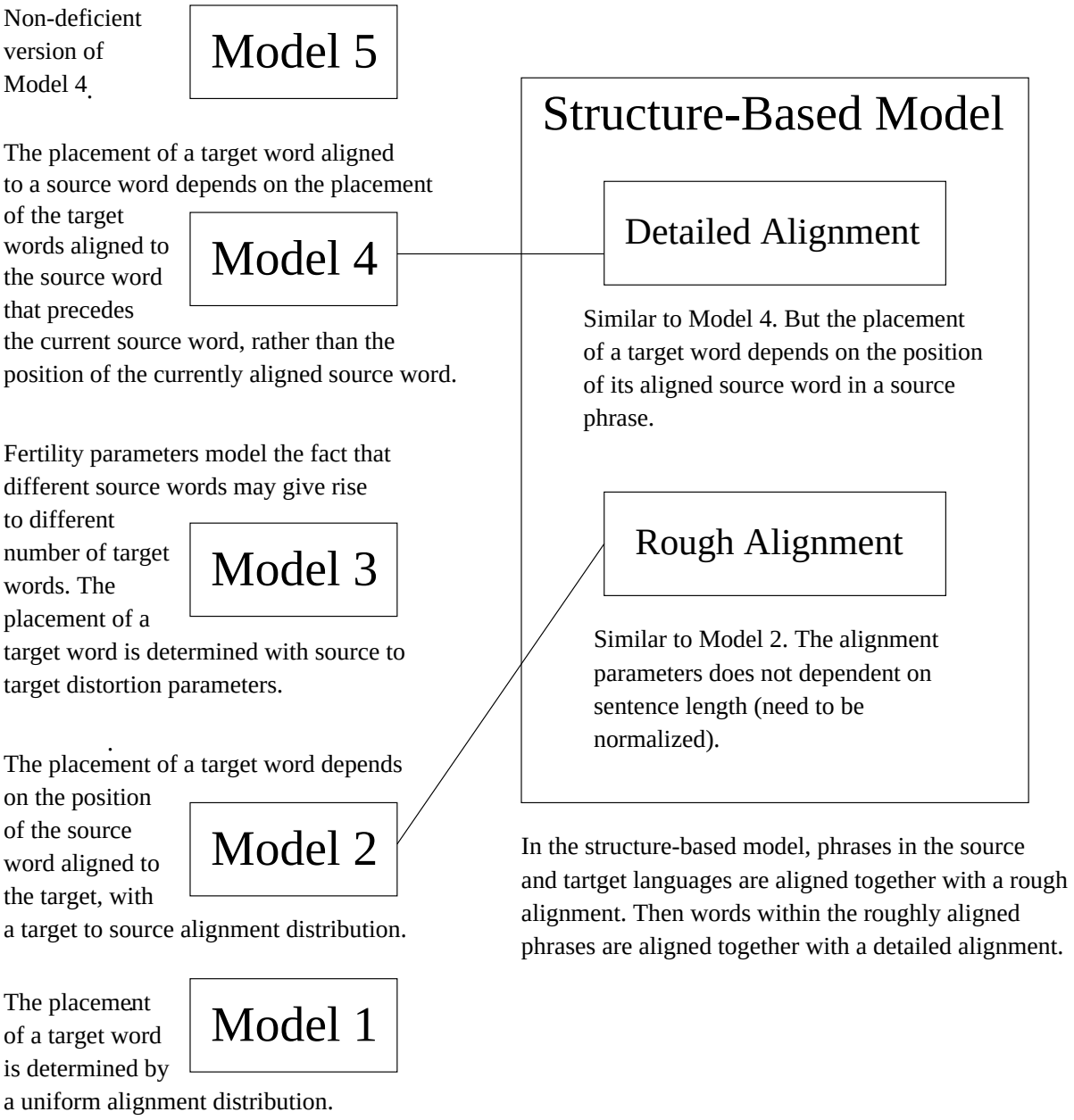

Figure 1.1: Different Translation Models

decoding, which imposes monotonic assignment on alignments.

1.3 Thesis Contribution

The thesis investigated the questions about statistical machine translation raised in the machine translation community. A major criticism to the IBM statistical machine translation system is its requirement of language parallelism — it translated from French to English, two languages that have similar structure, and furthermore, the source/target languages are transduced so the two intermediate languages have even more similar structure. This poses a question on the generality of the approach, as questioned by Knight in his survey paper. Actually, we found that the word-based alignment model used by IBM is a major cause of errors in statistical *spoken language* translation *between a language pair with different structures* and *with less parallel training data.*

The thesis work reported here takes one step forward towards relaxing the parallelism requirement of statistical machine translation. It demonstrates that more sophisticated statistical model is available to account for the structure difference between languages (English and German, which have very different word orders). I consider this to be the most important contribution of this thesis, and I hope that this can clarify some of the questions about statistical machine translation. In the effort towards this major goal, the following detailed contributions were made:

1. Introduction of a new structure-based translation model that improves the performance of spoken language translation between language pairs with different structures.

2. Investigation of novel approaches to grammar inference and bilingual grammar inference that facilitate the structure-based translation model.

3. Statistical dialogue analysis that can divide dialogues and long utterances into basic (shorter) semantic units, which enables a statistical machine translation system to process unsegmented outputs from speech recognizer.

4. Development of efficient decoding algorithms for statistical machine translation. This makes statistical machine translation more practical.

1.4 Related Work

1.4.1 Statistical Analysis of Language Structures

Dialogue Structure Analysis

Dialogue structure provides important information for spoken language understanding. This structure comprises the current topic, discourse state, and speech act, etc. Many researchers use topic information to reduce the perplexity of a task (Young, 1993; Kneser and Steinbiss, 1993). Dialogue analysis segments a dialogue into smaller units and labels the functionality of the units in their contexts. While knowledge-based approaches are widely and successfully used in dialogue structure analysis (Grosz and Sidner, 1986; Litman and Allen, 1990), they require intensive human effort to define linguistic structures and develop grammars to detect the structures.

(Woszczyna and Waibel, 1994) used a statistical approach for dialogue analysis. They modeled dialogue structure with a 6-state Hidden Markov Model. Each state represents a speech act, and it emits words to produce sentences in that speech act. The transition and emission probability can be obtained from labeled data with maximum likelihood estimation:

$$a_{ij} = \Pr(q_t = S_j \mid q_{t-1} = S_i) = \frac{\text{number of transitions from } S_i \text{ to } S_j}{\text{number of transitions from } S_i}$$

$$b_i(k) = \Pr(v_k \text{ at } t \mid q_t = S_i) = \frac{\text{number of times observing } v_k \text{ in } S_i}{\text{number of times in } S_i}$$

Given a new dialogue, its (linear) dialogue structure can be obtained with the Viterbi search algorithm of the Hidden Markov Model.

One deficiency of this model is that it treats words as unrelated items randomly emitted from a state. It does not take into account a much stronger constraint that words must form a legitimate sentence of a speech act. Because of this, the model is inclined to shift among states too often so that the probability of individual word is maximized. In this thesis, I will present a model that uses ngram language models to constrain the word sequences that can be emitted from a dialogue state.

Hidden Understanding Model

(Miller et al., 1994) introduced a statistical understanding model. A sentence is understood by a machine if it can assign a semantic role to each part of the sentence. In (Miller et al., 1994), this was modeled with statistical finite state machines. The statistical model consists of two parts: a semantic language model used for the transitions among semantic roles, and a lexical realization model that specifies the word generation process for each semantic role. Both the semantic language model and the lexical realization model were context dependent ngrams, and both of them were trained with the ML estimator from hand-labeled data. To understand a test sentence, the Hidden Understanding Model carried out Viterbi search for the semantic role sequences that are hidden behind the sentence. This hidden understanding model is very similar to the bracketing model used for dialogue analysis, which is introduced in Chapter 4.

1.4.2 Semantic Parser and Translator

The basic premise for semantic parsing and translation is that the structure of the information to be transmitted is largely independent of the language used to encode it. In semantic parsing like the Phoenix Parser (Ward, 1990), there was no syntactic analysis; instead, speaker utterances were parsed into semantic chunks, which could be strung together without grammatical rules. (Mayfield et al., 1995) used the Phoenix parser to generate language independent interlingua representation from an input sentence. In the interlingua representation, the semantic speech act of a semantic unit was labeled, together with the semantic roles that constitute the speech act. For example, the sentence "I have a meeting on Friday afternoon" can be represented by something like

([**my_unavailability**] ([**temporal**] `Friday afternoon`))

Translation or paraphrasing can be easily produced from the interlingua representation with template-based text generation. Continuing from the above example, paraphrasing of the sentence can be generated by filling the [**my_unavailability**] template "I couldn't do [**temporal**]" with the appropriate paraphrasing of the temporal expression in the original sentence. This results in the sentence "I couldn't do Friday afternoon."

In chapter 8 I will compare the performance of the semantic translation with my statistical translator.

1.4.3 Language Learning

Many different algorithms have been designed, either to acquire linguistic structures automatically from corpora or interactively from users, or to learn the meaning of a sentence by the automatically association between the sentence and the action related to the meaning of the sentence. Here I review just a few of them, which I believe are representative.

(Stolcke and Omohundro, 1994) have described a new technique for inducing the structure of Hidden Markov Models (HMM) from training data. The process begins with a maximum likelihood HMM that directly encodes the training data. Successively more general and compact models are produced by merging HMM states. The method is characterized as follows:

- Data incorporation: Given a body of data X, build an initial model M_0 by explicitly accommodating each data point such that M_0 maximizes the likelihood $\Pr(X \mid M)$. The size of the initial model will thus grow with the amount of data, and will usually not exhibit significant generalization.
- Structure merging: Build a sequence of new models, obtaining model M_{i+1} from M_i by applying a generalization or merging operator that coalesces substructures in M_i.

The merging operation accounts for the data points previously "explained" by separate model substructures with a single, shared structure. The merging process thereby gradually moves from a simple, instance-based model toward one that expresses structural generalizations about the data. The choice of what to merge and when to stop is governed by the Bayesian posterior probability of the model given the data:

$$\Pr(M \mid X) = \frac{\Pr(M)\Pr(X \mid M)}{\Pr(X)} \tag{1.3}$$

The learning procedure is a heuristic search for the HMM structure with the highest $\Pr(M)\Pr(X \mid M)$.

This algorithm, together with the other algorithms in finite state machine induction (Fu and Booth, 1975a; Fu and Booth, 1975b), reveal the gist of grammar inference. Almost every grammar inference algorithm is fulfilling two tasks: data explanation and generalization. In

the case of Bayesian model merging, data incorporation fulfills the data explanation task, and structure merging fulfills the generalization task.

Information-Theoretical Grammar Induction

Another approach to grammar inference is based on information theoretic iterative clustering and sequence finding procedures of words and phrases. (McCandless, 1994) and (Ries, Buø, and Wang, 1995) independently used a similar approach in grammar inference. The idea behind this is that through clustering, the inferred grammar is generalized, and through sequence finding, new structures are introduced into the grammar.

Association-based Understanding

Instead of grammar inference, (Gorin et al., 1991) described a language learning approach along another line. This approach argues that when the language data is vast and semantics of a task are very limited, the traditional grammar parsing might be not as helpful as word/phrase-meaning association. (Gorin et al., 1991) and (Gorin, 1995) introduced information theoretic algorithms for automatic acquisition of the associations and discovery of salient words/phrases that are most important in language understanding. The algorithms were successfully used in tasks like automatically directing incoming telephone calls to appropriate departments. One of their interesting finding is that including the automatically acquired phrases in the association significantly improved the performance. It comports with my observation that automatically acquired phrases can help to improve statistical NLP performance — in my case, statistical machine translation.

1.4.4 Corpus-Based Statistical Machine Translation

(Brown et al., 1993) introduced 5 different models that translated French sentences to English sentences. Of these, Model 2, 3 and 4 were reviewed in detail in Appendix A.

(Tillmann et al., 1997) introduced the monotonic assumption into a translation model to speed up the decoding process in statistical machine translation. While this may work for translation between Spanish and English, (which have similar word orders,) the assumption causes problems when attempting to model the translation between languages with very dif-

ferent word orders, such as English and German. They proposed to moderate the problem by writing grammars to preprocess sentences in order that they would have similar word orders. This greatly curtails the advantages of statistical machine translation.

1.5 Overview of the Thesis

The thesis is organized as follows. In Chapter 2, the task of spoken language translation in the domain of appointment scheduling is described, and the challenge it brings to machine translation is analyzed. Additionally, how the algorithms in the thesis meet the challenge is discussed. The chapters after that are mostly detailed descriptions of the algorithms and models. Chapter 3 gives an overview of the system architecture. In Chapter 4, an algorithm of statistical dialogue analysis is presented. A side-product of the algorithm is the automatic utterance segmentation, which is particularly important in spoken language translation. In Chapter 5, a bilingual grammar inference technique that is able to acquire shallow phrase structures is illustrated. The structures will be used in Chapter 6, where I introduce a novel structure-based alignment model for statistical machine translation. In Chapter 7, decoding algorithms for different translation models are presented. In Chapter 8, performance is evaluated and discussed. And thesis conclusions are presented in Chapter 9.

Chapter 2

Statistical Translation for Spoken Language — The Challenges

2.1 Spoken Language Translation: The Domain of Appointment Scheduling

With the rapid growth of information services and language technology applications in daily life, spoken language processing becomes more appealing. Recently there are intense research activities in speech-to-speech translation (Morimoto and et al, 1994; Roe et al., 1992; Hatazaki et al., 1992; Wahlster, 1993; Kay, Gawron, and Norvig, 1994; Suhm et al., 1995; Waibel, 1996).

Janus (Suhm et al., 1995) and Janus II (Waibel, 1996) are speech-to-speech machine translation systems in the domain of appointment scheduling. The projects aim at multilingual speech recognition and understanding, speech translation for human-to-human communication. The project collected German/English parallel corpus, which were used as part of the training data for the work reported in this thesis. In the mean time, the Verbmobil Project (Wahlster, 1993; Kay, Gawron, and Norvig, 1994) has collected a larger set of parallel sentences in the same domain, and that was another source of the training data for my system. Approximately 15% of my data were from the Janus Projects, and about 85% of the data were from the Verbmobil Project.

2.1.1 The Corpus

The scheduling parallel corpus consists of 569 dialogues in the domain of appointment scheduling. There are around 8,400 English/German parallel utterances in these dialogues. The size of the corpus is around 446,500 (224,000 English + 222,500 German) words. The following is a typical English dialogue with its German translation:

Example 2.1.1 A segment of scheduling parallel corpus

Hello Dr. Noah

Hi Tor let's set up a meeting for a couple hours in the next two weeks when's good for you

Hmm that's a good question um let's see pretty busy um how about Friday the second in the morning

Monday the twenty ninth how about at two thirty I could do it

No I'm only free on mornings of Monday let's see ...

...

Sounds good Tuesday morning nine AM on the thirtieth how's your wife

She's fine why do you ask

Yeah I thought so too well see you then bye

———————————————————

Hallo Dr. Noah

Hallo Tor lassen Sie uns ein Meeting von ein paar Stunden in den nächsten zwei Wochen vereinbaren wann wäre es ihnen recht

Hmm das ist eine gute Frage schauen wir mal ich bin ziemlich beschäftigt wie wäre es denn mit Freitag dem 2 morgens

Wie wäre es mit halb drei am Montag den 29 das ginge bei mir

Nein Montags kann ich nur morgens schauen wir mal ...

...

Dienstag den 30 um neun Uhr morgens hört sich gut an wie geht es denn ihrer Frau

Es geht ihr gut warum fragen Sie

Ja denke ich auch also bis dann tschüs

The Verbmobil part of the data contains the hints for segmenting long utterances into smaller units, while the Janus part does not have that information. The segmentation algorithm described in Chapter 4 was used to presegment the long utterances in the Janus training corpus. The **char_align** alignment algorithm (Church, 1993) was implemented to align the sentences in the parallel corpus.

2.1.2 Why Statistical Machine Translation

Learnability

Symbolic approaches to machine translation take great human effort in language engineering. In knowledge-based machine translation, for example, designers must first find out what kinds of linguistic, general common-sense and domain-specific knowledge are important for a task. Then they have to design an interlingua representation for the knowledge, and write grammars to parse input sentences into and generate output sentences from the interlingua representation. All of these require expertise in language technologies and tedious and laborious work.

The biggest advantage of statistical machine translation is its learnability. As long as a model is set up, it can learn automatically with well-studied algorithms for parameter estimation. Therefore parallel corpus replaces the human expertise for the task.

Coverage and Robustness

Due to its competence-orientation, symbolic approach has encountered great difficulties for spoken language translation. The theory-based approach is good at describing human language competence. However, we are actually processing human language performance when we work with spoken languages.

The difficulties are two-fold. First, it is not realistic to require that a language system designer be able to enumerate all possible language constructions and encode them in his grammar, given the richness of language constructions in spoken language. Therefore the coverage of grammar is a serious problem. (Black, Garside, and Leech, 1993) reported three experiments with several parsers for "general English" sentences taken from the AP newswire, Brown Corpus and Wall Street Journal text. With a very liberal and forgiving standard, they found that "the state of the art in parsing general English is so modest that one could not

even expect a typical parsing system to get a correct analysis for half the sentences input to it — perhaps not even for a third of them." Although the experiment in (Black, Garside, and Leech, 1993) was conducted with parsers for "general English", the performance of parsers is not encouraging with domain-specific tasks either. When I applied the Janus semantic parser and translator to some of my test data, I found that the performance was surprisingly poor. One reason, as suggested by one of the symbolic system developers, was that the borrowed word "Meeting" occurred 6 times in 69 German utterances, but it was not covered in the grammar. As a matter of fact, the translator failed on all the sentences containing the word "Meeting".

The second issue is the robustness. Unlike written languages, spoken languages are hardly ever well-formed. They contain disfluencies, hesitations, repetitions, and false starts. A speech translation system cannot assume any rigid syntactic constraints in an input language. It is also not possible to model the noise with grammar rules due to its unpredictability. A common practice to handle this problem is to write a grammar that does not model the noisy data at all, and build a parser that is able to skip the segments that are not covered by the grammar. This is dangerous, since it will also ignore semantically meaningful segments that are not covered by grammar, like the aforementioned "Meeting" example.

Therefore, to process spoken languages, or language performance, an empirical, frequency-based, data-driven and performance-oriented approach is more desirable. Statistical machine translation is a good candidate that meets these criteria. It can learn to have a good coverage as long as the training data is representative enough. It can statistically model the noise in spoken language, so it does not have to make a binary keep/abandon decision and is therefore more robust to noisy data.

2.2 The Challenges

Our parallel data of spoken language poses many new challenges to statistical machine translation too. The major difficulties, as I perceive them, are related to the questions raised by the machine translation community. As I quoted Kevin Knight's AI magazine survey article at the beginning of this thesis, the difficulties include

1. Language pair with radically different structures

2. Less training data, or sparse data problem

Besides these, other difficulties are related to spoken language. For example, the long utterance problem is specific in our spoken language corpus. Spoken language may also intensify the structure difference between languages. In this section, I discuss these difficulties in detail.

2.2.1 Long Utterance Problem

A spoken dialogue does not consist of sentences in the classical sense. Instead, each utterance (speaker's turn) contains multiple unsegmented sentences. The unsegmented utterances are typically very long. This may cause severe problems for statistical machine translation. First, since the alignment parameters depend on sentence length, the number of alignment parameters is quadratic or cubic in the maximum length of utterances. Therefore a restriction on the acceptable sentence length has to be imposed to limit the number of free parameters. In the IBM translation system, the maximum sentence length is limited to 25. This will be too restrictive for our task if we use utterances as the unit in translation, since the average utterance length of the scheduling data is 24.87. On the other side, our data is much less than the Hansard Corpus used by the IBM system. With longer parallel unit size in the training data, we get much fewer number of parallel units for model training. This will greatly reduce the number of examples per parameter, and make it unrealistic to build a reliable model. The second difficulty lies in decoding algorithms. The longer an utterance is, the more different choices there are in the hypothesis space. In my experience with stack decoder (Wang and Waibel, 1997a), the searching time increases dramatically with the length of utterances. To make the problem tractable, we are forced to keep the searching beam very narrow. This greatly limits the possibility of finding optimal solutions.

2.2.2 Sparse Data Problem

Compared to the IBM statistical machine translation system, which used over 1.7 million pairs of sentences, our database has only 8,400 utterances comprising about 200,000 words in each language. Although an utterance may contain several sentences, sentence boundaries in an utterance are not directly available from speech recognition (see the about *long utterance problem*). Even after the statistical dialogue analysis model is used to segment long utterances

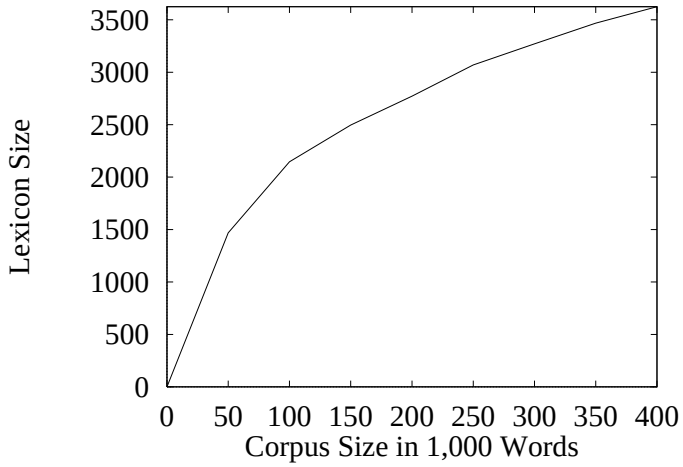

Figure 2.1: Corpus Size vs. Lexicon Size (English)

(Chapter 4), we have only around 30,000 parallel sentences, which is still much less than the amount that IBM had used.

Figure 2.1 and Figure 2.2 show the growth rate of lexicon size in relation to the corpus size. The statistics were collected from monolingual English and German corpora in the scheduling domain. As we can see, at the size of 200,000 words (our parallel corpus size), both corpora still have very high lexicon growth rate. Therefore the parallel corpus is hardly enough to estimate the translation distributions for a stable set of source words.

2.2.3 Modeling Problem: Structure Difference

The languages in our parallel corpus, English and German, have quite difference structure. While the structure difference is not as radical as the difference between English and Vietnamese, it already cause problems with the word-based alignment model.

The structure differences are contributed mostly by two factors: different word orders and deletions in translation.

Word Order

Unlike English and French in the IBM machine translation system, English and German have very different word orders. This observation was also made by another research group in

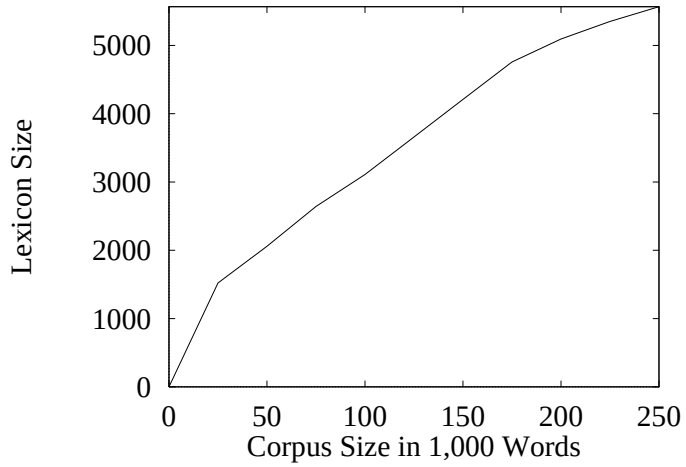

Figure 2.2: Corpus Size vs. Lexicon Size (German)

statistical machine translation (Tillmann et al., 1997). As a typical example shown in Example 2.2.1, time expressions, which most frequently occur in the scheduling data, often appear at different positions in English and German.

Example 2.2.1 Different word (phrase) orders in parallel sentences

I could offer you **Wednesday the twenty fifth** *for the second date in May*

Für der zweiten Termin im Mai könnte ich **den Mitwoch den fünfundzwanzigsten** anbieten.

The word order difference is even more common in spoken language. Since people care less about word orders when they speak, the word order in spoken language is more flexible. The following example shows six English sentences with different word order to express the same meaning. Each of these sentences may be a translation of the German sentence "Ich könnte mich mit Ihnen so um zwei Uhr nachmittags am Mittwoch den dritten treffen."

Example 2.2.2 Flexibility of word orders in spoken language

I could meet with you **at about two PM on Wednesday the third.**

I could meet with you on **Wednesday the third at about two PM.**

On Wednesday the third *I could meet with you* **at about two PM.**

At about two PM *I could meet with you* **on Wednesday the third.**

At about two PM on Wednesday the third *I could meet with you.*

On Wednesday the third at about two PM *I could meet with you .*

Deletions in Translation

Another contributor to the structure difference between the languages in our corpus is deletions in translation. It seems that human translators tend to ignore unimportant parts when they translate spoken languages. As an example of this at extreme, a Japanese interpreter once translated an American diplomat's lengthy joke to "This gentleman just told a very funny joke, please laugh." Deletion can also be a result of erroneous sentence alignment.

Word-based Alignment Model

By far, most (if not all) of the statistical machine translation systems employ a word-based alignment model (Brown et al., 1993; Vogel, Ney, and Tillman, 1996; Wang and Waibel, 1997a), which treats words in a sentence as independent entities and ignores the structural relations among them. While this independence assumption works well in speech recognition, it poses a major problem in our experiments of spoken language translation, with less training data and very different structures between source and target languages.

Let's take IBM alignment Model 2 for example. A majority of the parameters in Model 2 is the translation parameters. Because we have sparse data problem, it is likely that the translation parameters get under-trained. On the other side, the number of alignment parameters is much fewer than the number of translation parameters, therefore the alignment parameters can relatively get over-trained. If this happens, then the model will tend to align a target position to a similar position in the source sentence, regardless of what word appears at that source position. Figure 2.3 shows the Viterbi alignment between a pair of parallel English/German sentences made by IBM Alignment Model 2 (henceforth Model 2), and the 'ideal' alignment of the sentence pair is shown in Figure 2.4. Here the alignment parameters penalize the alignment of English words to their correct German translations, because the translations are far away from those words due to the different phrase orders in the translation. The same alignment problem also happens when there are deletions in translation. Figure 2.5 shows an example of an erroneous alignment (again, made by Model 2) between a pair of parallel sentences with

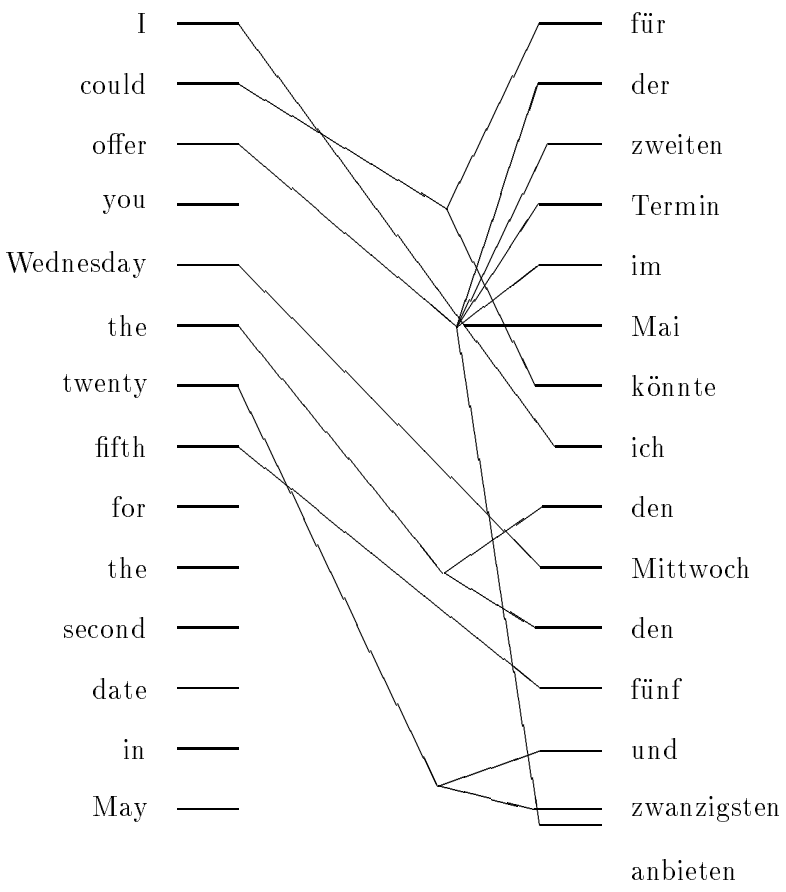

Figure 2.3: Word Alignment of translation with different phrase order: the alignment made by IBM Alignment Model 2.

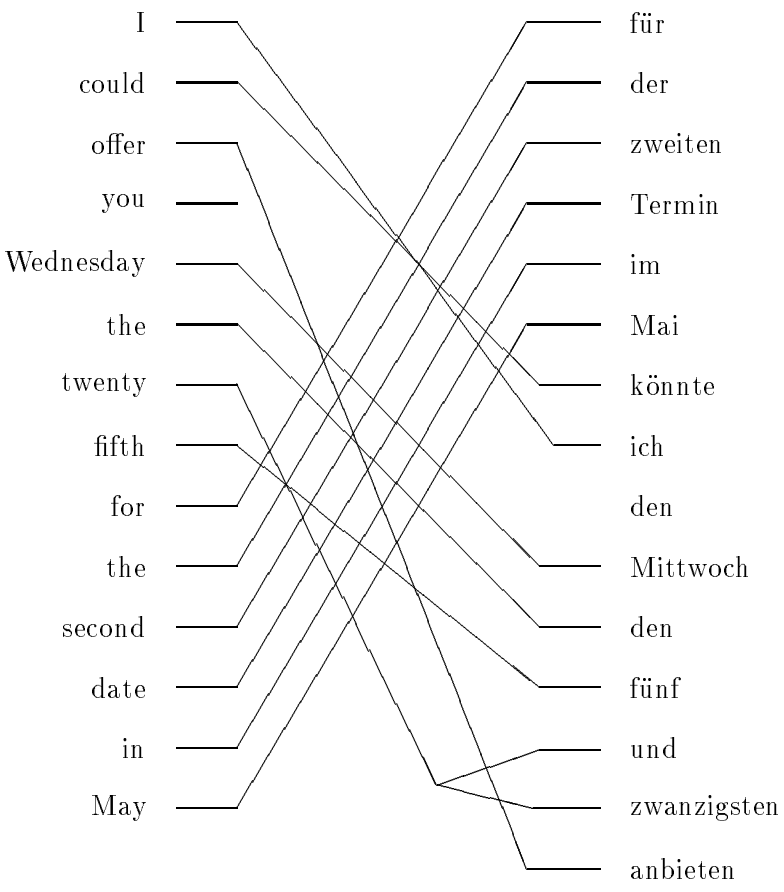

Figure 2.4: Word Alignment of translation with different phrase order: the 'ideal' alignment.

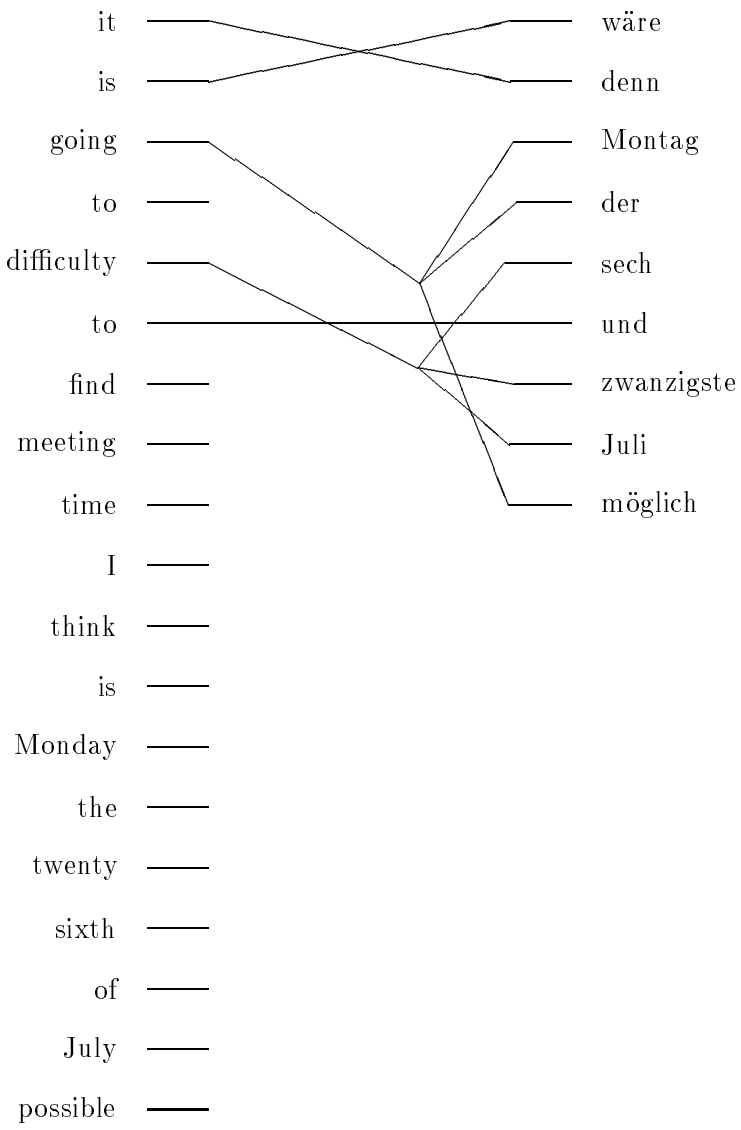

Figure 2.5: Word Alignment with deletion in translation: the alignment made by IBM Alignment Model 2.

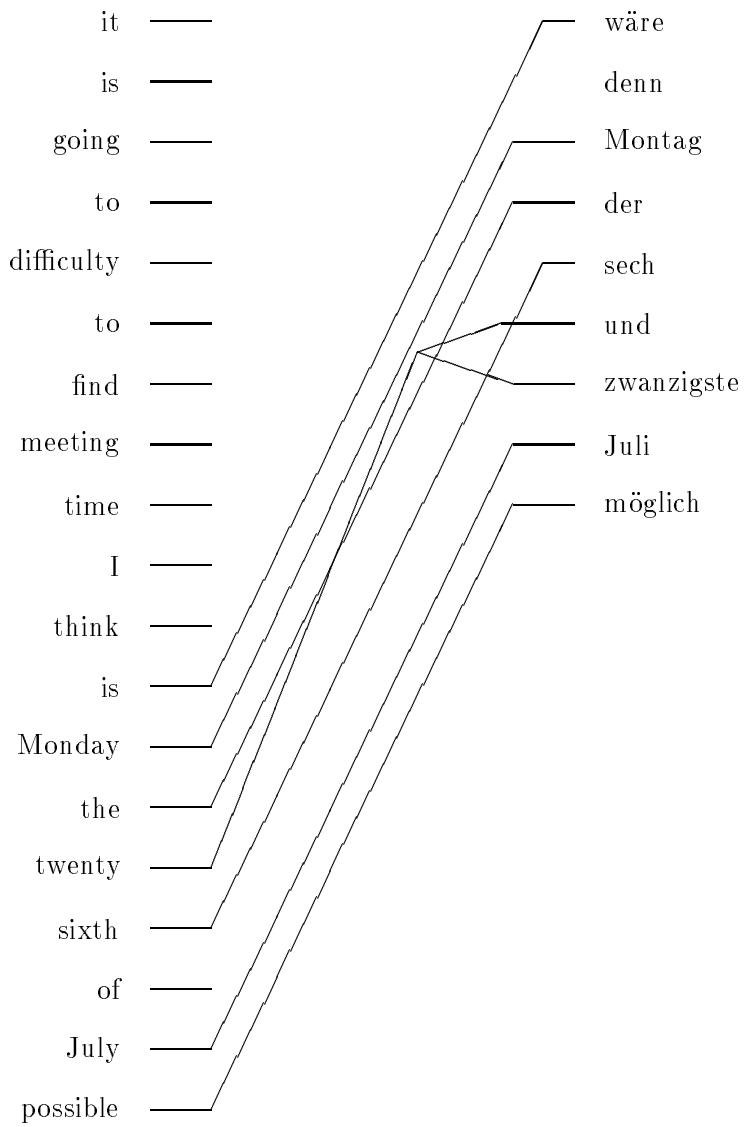

Figure 2.6: Word Alignment with deletions in translation: the 'ideal' alignment.

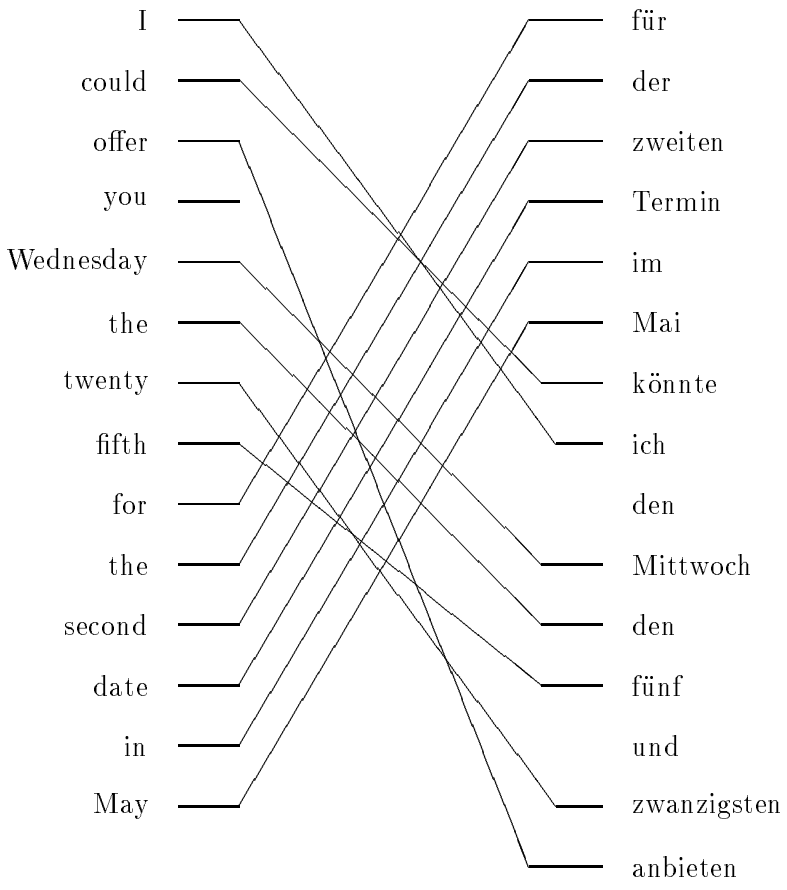

Figure 2.7: Word Alignment with Model 1 for one of the previous examples. Because no alignment probability penalizes the long distance phrase movement, it is much closer to the "ideal" alignment.

deletions. Figure 2.6 shows the 'ideal' alignment.

From these two examples, we can see that Model 2 tends to make mistakes in the Viterbi alignment between parallel sentences when the correct, ideal alignment contains long distance word-to-word correspondence. To see how often this kind of long distance alignment happens in our English/German scheduling conversation parallel corpus (Wang and Waibel, 1997a), an experiment was conducted. The experiment was based on the following observation: IBM Alignment Model 1 (henceforth Model 1, where the alignment distribution is uniform) and IBM Alignment Model 2 (henceforth Model 2) found similar Viterbi alignments when there were no word order differences or deletions; they predicted very different Viterbi alignments when there were many long distance word-to-word alignments between a sentence pair, since the alignment parameters in Model 2 penalized the long distance alignments. Figure 2.7 shows the Viterbi alignment made by Model 1 for the sentences in Figure 2.5 and 2.6. Compared to the alignment made by Model 2, it is much closer to the 'ideal' alignment[1].

I measured the distance between a Model 1 alignment $\mathbf{a^1}$ and a Model 2 alignment $\mathbf{a^2}$ with $\sum_{i=1}^{|\mathbf{g}|} |a_i^1 - a_i^2|$. To estimate how often long distance alignments happen in our parallel corpus, I collected the statistics about the percentage of sentence pairs (with at least five words in a sentence) with the distance between Model 1 and Model 2 alignments greater than $1/4, 2/4, 3/4, \cdots 10/4$ of the target sentence length. By checking the Viterbi alignments made by both models, it is almost certain that there is either a movement or a phrase deletion or both in a sentence pair whenever the distance is greater than $3/4$ of the target sentence length. Figure 2.8 plots this statistics — around 30% of the sentence pairs in our training data contain some degree of long distance alignments. In other orders, Model 2 tends to make alignment errors with around 30% of our training data.

The mistakes in alignments between parallel sentences make the translation model confused about the correct translation of source words. Table 2.1 shows the translation distribution of the English word "I" learned by Model 2. Here only the first word "ich" is a correct translation for "I". It is interesting to note that all the German words in the table often appear at the beginning of a German sentence, the same position where "I" often appears in English sentences.

[1]The better alignment on a given pair of sentences does not mean that Model 1 is a better model. Non-uniform alignment distribution is desirable. Otherwise, language model would be the only constraint that is used to determine the source sentence word order in decoding.

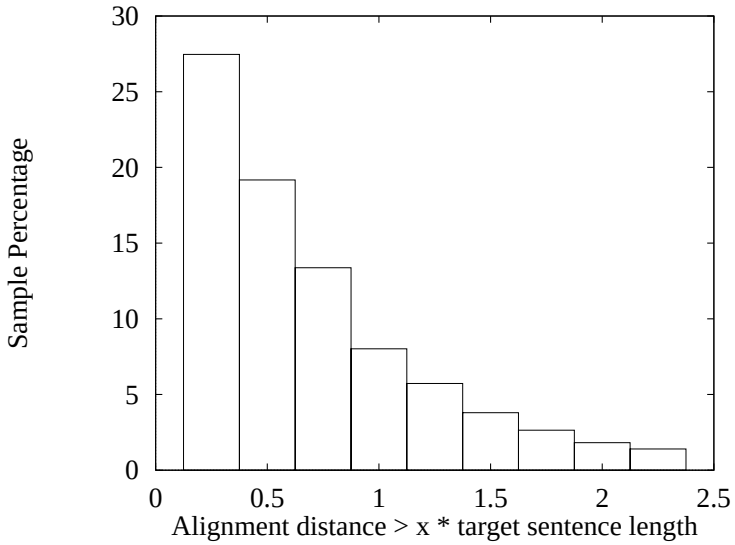

Figure 2.8: Distance of Word Alignment

$t_{M2}(*\mid \text{I})$	
ich	0.708
da	0.104
am	0.024
das	0.022
dann	0.022
also	0.019
es	0.011

Table 2.1: The translation distribution of "I". The biased alignments between parallel sentences forced the association between "I" and those unrelated German words.

It is thus clear now that the biased alignments forced the association between "I" and those unrelated German words.

2.3 Solutions

2.3.1 Long Utterance Problem

I have developed a statistical dialogue analysis system. It segments utterances in a dialogue and labels each segment with a speech act (Wang and Waibel, 1997b). It was used as a preprocessor of the machine translation system.

2.3.2 Modeling Problem

Starting from Model 4, the IBM Statistical Machine Translation System replaced the alignment distributions with the displacement distributions. In this case, the distribution of the aligned position of a source sentence word w depends on the aligned position of the word preceding w, so a long distance phrase movement only gets penalized once for the first words in the phrase. The rest words' alignment depends on the relative distance from the first word of the phrase, instead of their corresponding word position in the source sentence. Hence the probability for the alignment will not get hurt too much.

While this alleviates the symptom, it is not a cure to the disease. It still discourages long distance movement, since at moved phrase boundary, the displacement probability can still get too small. It does not distinguish the context where w and its preceding word appear. A major contribution of the thesis is a structure based model that finds the important phrases, models the structure difference directly by a rough alignment between phrases, and then aligns the words in the corresponding phrases via detailed alignment. The detailed alignment aligns source/target words together with the knowledge of the context (phrase) where the words appear.

2.3.3 Sparse Data Problem

As for the sparse data problem, there is not much for us to do with the data itself, since the collection of parallel corpus is expensive. A common practice is to simplify the statistical model to reduce its complexity. Normally the complexity is measured by the number of free parameters, therefore the per-parameter data increases. However, we cannot go much farther along this line either. IBM Model 1 is probably the simplest model — no translation model can describe the translation process successfully without knowing the probability of the translation from a source word to a target word.

However, the number of free parameters is not the only measure of model complexity. By reducing the model complexity according to information theoretic measures, we can still alleviate the sparse problem. Here the idea is to reduce the uncertainty of the translation of a source word by guiding the training procedure to focus on the correct alignments. The structure-based model achieves this by focusing on the alignment between words inside the roughly aligned

phrases, so the words outside the aligned phrases will not "distract the attention" of a source word.

Additionally, the segmentation algorithm produces more basic parallel translation units, therefore supplies more data for the training of alignment parameters.

Chapter 3

System Overview

Figure 3.1 illustrates the architecture of the statistical machine translation system described in this thesis. The hollow arrows show the data flow of the training process, while the solid arrows display the data flow of the test process.

In the training phase, training data is first manipulated by the preprocessor. It is then passed to the bracketing model for sentence segmentation. The parallel corpus of utterance segments (for the sake of simplicity, I will call "utterance segments" sentences from now on if it does not cause confusion.) are then aligned at sentence level with the **char_align** algorithm (Church, 1993). The source language part of the parallel data is used for the training of language model, and the aligned parallel sentences are first used for grammar inference, and then used together with the inferred phrase structures to build the translation model.

In the test phase, a test target utterance is first preprocessed and segmented, and then each segmented sentence is sent to the decoder as a target. The decoder uses the translation model and language model built in the training phase to find the translation of the target that has the maximum product of language model score and translation model score.

The different processing modules are discussed throughout this thesis. In this chapter I will describe the data preprocessor and the language model. Utterance segmentation is discussed in Chapter 4; grammar inference in Chapter 5; translation model in Chapter 6; decoders in Chapter 7.

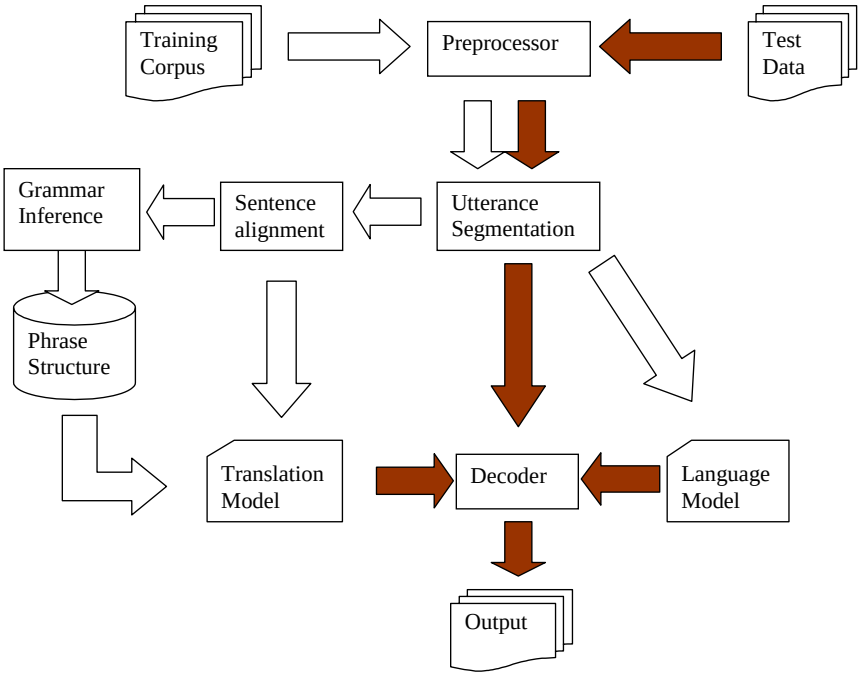

Figure 3.1: System Architecture: the hollow arrows show the data flow of the training phase, the solid arrows show the data flow of the testing phase.

3.1 Resources

The preprocessor and other modules in the system use the following resources of the translation system:

- Lexicons: English and German lexicons have been built from the parallel training corpus. They consist of a list of words, and each word is associated with its frequency in the corpus.
- German Dictionary for the Verbmobil Project: a German dictionary has been developed by the Verbmobil Project. It includes the part-of-speech, pronunciation and compositional (prefixes and suffixes, compound nouns, etc.) information for around 42,500 German words.
- Proper Name List: a list of 84,000 proper nouns, including personal, geographic and holiday names, has been compiled from a variety of internet sources.

3.2 Preprocessing

The purposes of preprocessing include:

1. remove obvious noise in spoken language (Repetition Deletion, Canonization).
2. make two languages in question more similar, so more one-to-one correspondences can be established in word alignments between parallel sentences (Compound Word Decomposition, Language Specific Preprocessing).
3. find the commonality among words to abate the sparse data problem (Proper Noun Replacement, Canonization).

3.2.1 Canonization

This part of the preprocessor is used to standardize the orthography of the transcribed words, as well as correcting common spelling errors. Because the transcription was made by several transcribers, there were many different orthographic forms for the same words, such as (Mr,

Mr., Mister), (dreissigsten, drei"sigsten), (a.m., AM, A.M.), (hmm, hm, /hm), (okay, ok, OK, o.k.). There are also common spelling errors, like "ninteenth" for "nineteenth", "MacDonald's" for "McDonald's" and "zwanzigt" for "zwanzigst".

The canonization preprocessing also decapitalizes the initial word in a sentence if its lowercase form is in the lexicon and it is not in the list of proper names.

3.2.2 Proper Noun Replacement

A great proportion of unseen words includes proper names. For training sentences, if a capitalized word appears in both English and German sentences, then the preprocessor will search for it in the proper name list. It is replaced with a special word "#PROPER#" if a match is found in the list. For test data, if a capitalized word is not in the German lexicon, the preprocessor will search for it in the proper name list and replace it with "#PROPER#" if a match is found.

Additionally, sequences of capital letters are replaced with the special word "#SPELLING#".

3.2.3 Compound Word Decomposition

Compound words are very common in German. This may cause problems in our alignment model. For example, compound words like "Januarwoche" and "sechsundzwanzigsten" should be aligned with their translation "week of January" and "twenty sixth" respectively. However, in an alignment between source and target, each target word is limited to be aligned with at most one source word. Therefore it is desirable to decompose the compound words, so that the above examples can be converted into German word sequences "Januar (January) Woche (week)" and "sechs (six) und (and) zwanzigsten (twentieth)."

The preprocessor decomposes all the compound numbers and uses heuristics to decompose compound nouns. The heuristics work in the following way: if the length of a noun exceeds a threshold (8 was used in my experiment), the preprocessor tries to break it into the form $(\text{word}_1)(\text{word}_2)$ or $(\text{word}_1)\text{s}(\text{word}_2)$ at any breakable position. The breakable positions can be obtained from the German dictionary of the Verbmobil Project. If both word_1 and word_2 are words in the German dictionary, the original word is then decomposed into word_1 and word_2.

3.2.4 Language Specific Preprocessing

I found the following two language specific preprocessing steps are particularly helpful in improving the system performance:

1. Words like "sechsundzwanzigsten" has been decomposed into "sechs (six) und (and) zwanzigsten (twentieth)" in the previous decomposition step. However, the expression is still not in conformity with the English expression "twenty sixth". Therefore we rewrote "sechste (six) und (and) zwanzigsten (twentieth)" as "sechs (sixth) und (and) zwanzig (twenty)." This procedure was applied to all similar expressions.

2. German expresses the time expressions like "two thirty" as "halb (half) drei (three)". This makes translation models confused about the translation of the English word "two." Therefore we rewrote "halb (half) drei (three)" as "halb (half) zwei (two)", "halb (half) vier (four)" as "halb (half) drei (three)", etc.

3.2.5 Repetition Deletion

Repetitions are very common in spoken language, as a form of hesitation. The following is an example taken from the corpus.

Example 3.2.1 Repetition in Spoken Language

I forgot to to mention that that my Fridays are are relatively free.

The repetitions are often ignored in translations, since they are not meaningful in the original sentence. However, this may result in difficulties in alignment models — the repeated part may not have a counterpart in another language, therefore it will be incorrectly aligned with some other parts.

The preprocessor removes any sequence of up to 4 words that exactly duplicates the words before it, with one exception, when the sequence is a list of numbers. This ensures that it will not incorrectly remove a number sequence like a telephone or room number.

3.2.6 Removing Low Frequency Words

Words that occur only once in the corpus were replace with the token #UNKNOWN#.

Language	Original	Canonization	Proper Noun	Comp. Word	Rm Low Freq.
English	2782	2746	2422	2422	1372
German	4792	4765	4441	4359	2202

Table 3.1: Lexicon Sizes After Each Step of Preprecessing

3.2.7 Preprocessing Outcome

Table 3.1 shows how the preprocessing affects the lexicon sizes. After preprocessing, all the low frequency words that occur only once in the training corpus were discarded from the lexicon, and they were treated as unknown words.

3.3 Language Modeling

Bigram (Jelinek, 1990) was used for language modeling in the system. The English language model was trained on a monolingual corpus as well as on the English part of the parallel training data. The training data for language modeling consists of 420,000 tokens. The words that appear in the corpus but not in the lexicon were treated as unknown words in language model training.

Deleted Interpolation (Jelinek and Mercer, 1980) was used to smooth the language model with unigram and the uniform distribution.

The bigram perplexity of the text data is 42.2. I also conducted experiments with a class-based bigram for language modeling. Although the class-based language model reduced the test data perplexity slightly to 40.4, it had no noticeable influence on translation performance. Therefore if not mentioned specifically, the language model used in the experiments reported in this thesis was the standard bigram with deleted interpolation for smoothing.

Chapter 4

Segmenting Long Utterances — A Bracketing Model

In this chapter, I introduce a statistical model that copes with the long utterance problem in statistical machine translation. The model can segment a long utterance into small pieces of sentences/clauses/phrases that are dialogue units with their own speech acts. Actually, with this capability, it is a simple dialogue analysis model that can do more than just segmentation. The model parameters can be estimated with supervised learning.

4.1 Dialogue Model

4.1.1 Dialogue and Bracketing Scheme

A *dialogue* consists of a sequence of *utterances*. An utterance consists of a dialogue participant's turn-taking. It may consist of several *segments*. A *speech act* is associated with each segment that represents the segment functionality. A segment can be a sentence, a clause or a phrase with its own speech act.

In an unmarked dialogue, no segment boundary or speech act information is labeled. Only utterance boundaries are available. An example of unmarked dialogue is shown below:

```
A: hello Dr. Noah
B: hi Tor let's set up a meeting for a couple hours in the next
```

```
   two weeks when's good for you
A: let's see how about Friday the second in the morning
B: I'm busy that morning
   ......
```

A bracketing scheme B breaks long utterances in a dialogue into sequences of segments, and labels each segment with its speech act. Below is an example of a bracketing scheme for the previous dialogue:

Example 4.1.1 *Bracketing Scheme*

```
[nicety] (Hello Dr. Noah)
[nicety] (Hi Tor)
[suggest-meeting] (Let's set up a meeting for a couple hours)
[temporal] (in the next two weeks)
[your-availability] (when's good for you)
[interject] (let's see)
[suggest-time] (how about Friday the second in the morning)
[my-unavailability] (I'm busy that morning)
......
```

This is not the only bracketing scheme for the dialogue. For example, the second utterance could also be bracketed as follows:

```
[nicety] (Hi Tor)
[suggest-meeting] (Let's set up a meeting for a couple hours)
[your-availability] (in the next two weeks when's good for you)
```

In a bracketing scheme B for a dialogue D, B_i denotes the i^{th} segment of B, and A_i represents the speech act label for B_i. $|B|$ is the number of segments in B, $|B_i|$ is the number of words in B_i, and $|D|$ is the number of words in dialogue D. B_i's form a partition of B: there is no overlapping between B_i and B_{i+1} for $i = 1, \cdots, |B|$, and $\sum_{i=1}^{|B|} |B_i| = |D|$.

4.1.2 Bracketing Model

The generation of a dialogue D can be depicted by the following process:

1. At time 0, the dialogue is at a null segment with the speech act <d>, the start of the dialogue. For $i = 1, 2, \cdots$, do the following:

2. At time i, generate the speech act A_i according to a distribution $\Pr(A_i \mid A_0^{i-1}, B_0^{i-1})$. If $A_i =$ </d>, which is the **end of dialogue** speech act, then the dialogue generation is complete. Otherwise the next step should be taken.

3. Generate a segment of words B_i with the speech act A_i according to a distribution $\Pr(B_i \mid A_0^i, B_0^{i-1})$.

We can make the following independence assumptions in this model:

1. $\Pr(A_i \mid A_0^{i-1}, B_0^{i-1}) = \Pr(A_i \mid A_{i-1})$.

2. $\Pr(B_i \mid A_0^i, B_0^{i-1}) = \Pr(B_i \mid A_i)$.

And $\Pr(B_i \mid A_i)$ can be modeled with a speech act dependent ngram model:

$$\Pr(B_i \mid A_i) = \sigma(B_i) \prod_{j=1}^{|B_i|+1} \Pr(b_{ij} \mid b_{i(j-N+1)}^{j-1}, A_i) \tag{4.1}$$

here $b_{i0} =$ <s> and $b_{i(|B_i|+1)} =$ </s>. The function $\sigma(B_i) = 1$ when there is no utterance boundary between b_{ij} and $b_{i(j+1)}$ for $j = 1, \cdots, |B_i| - 1$, otherwise $\sigma(B_i) = 0$. $\sigma(B_i)$ guarantees that the bracketing scheme B respects the natural utterance boundaries — no segment can cross the boundary between two utterances.

Hence we can get the likelihood of a dialogue D, which is the sum of the probabilities of generating D over all possible bracketing schemes:

$$\Pr(D, B) = \prod_{i=1}^{|B|} \Pr(A_i \mid A_{i-1}) \Pr(B_i \mid A_i) \times \Pr(\texttt{</d>} \mid A_{|B|}) \tag{4.2}$$

$$\Pr(D) = \sum_{B} \Pr(D, B). \tag{4.3}$$

here $A_0 =$ <d>.

4.1.3 Parameter Estimation

In the bracketing model, there are two types of parameters: a speech act transition distribution $\Pr(A_i \mid A_{i-1})$ depicts the likelihood of switching from one speech act to another; for each speech act A, the speech act dependent ngram distribution $P_A(w_i \mid w_{i-N+1}^{n-1}) = \Pr(w_i \mid w_{i-N+1}, \ldots, w_{n-1}, A)$ models the likelihood of generating a sentence under speech act A. The parameters can be estimated with ML, with a training data set hand-labeled in the format of Example 4.1.1. The speech act sequence in the labeled data can be used to estimate the speech act transition parameters, and the texts for each speech act can be used to train the ngram for that speech act.

4.2 Algorithms for the Bracketing Model

Given a bracketing model, we face two problems. The first is how the likelihood of a dialogue $D = d_1 d_2 ... d_n$, $\Pr(D)$, can be effectively computed; the second is how the most probable bracketing scheme, i.e., the most probable structure of a dialogue, can be identified. Both problems can be solved with dynamic programming algorithms.

We define $Q_{ij}(A)$ to be the likelihood that $d_i d_{i+1} \ldots d_j$ is a complete segment generated in speech act A, regardless of its context. $Q_{ij}(A)$ can be computed with

$$\begin{aligned} Q_{ii}(A) &= P_A(d_i \mid \texttt{<s>}) P_A(\texttt{</s>} \mid d_i) \\ Q_{ij}(A) &= \frac{Q_{i(j-1)}(A) P_A(d_j \mid d_{j-1}) P_A(\texttt{</s>} \mid d_j)}{P_A(\texttt{</s>} \mid d_{j-1})} \times \sigma(d_{j-1} d_j) \end{aligned}$$

To solve the first problem, we define $\alpha(k, A)$ to be the probability that $d_k \cdots d_n$ starts with a segment of speech act A (here $n = |D|$) in all possible bracketing schemes. Then

$$\Pr(D) = \sum_A \Pr(A \mid \texttt{<d>}) \alpha(1, A).$$

$\alpha(k, A)$ has the following recursive relation that licenses the use of dynamic programming:

$$\begin{aligned} \alpha(n, A) &= Q_{nn}(A) \Pr(\texttt{</d>} \mid A) \\ \alpha(k, A) &= Q_{kn}(A) \Pr(\texttt{</d>} \mid A) + \sum_{k \le j < n, C \in \mathcal{S}} \alpha(j+1, C) Q_{kj}(A) \Pr(C \mid A) \end{aligned}$$

Here $\mathcal{S}$ is the set of speech acts.

In the second problem, we would like to bracket a dialogue $D = d_1 d_2 ... d_n$ into a sequence of segments labeled with their speech acts. The bracketing scheme should be the most probable one:

$$\hat{B} = \arg\max_{B} \Pr(D, B) \tag{4.4}$$

where $\Pr(D, B)$ is defined in (4.2). We call bracketing scheme $\hat{B}$ the Viterbi bracketing. It can be found with dynamic programming.

Let $\gamma(k, A)$ be the maximum probability that $d_k \cdots d_n$ starts with a segment of speech act A in a bracketing scheme. Then

$$\begin{aligned}
\gamma(n, A) &= Q_{nn}(A) \Pr(\texttt{</d>} \mid A) \\
\gamma(k, A) &= \max\{Q_{kn}(A) \Pr(\texttt{</d>} \mid A), \max_{k \le j < n, C \in \mathcal{S}} \gamma(j+1, C) Q_{kj}(A) \Pr(C \mid A)\} \\
\xi(k, A) &= \begin{cases} (n, A) & \text{when } \gamma(k, A) = Q_{kn}(A) \Pr(\texttt{</d>} \mid A) \\ \arg\max_{(k < j \le n; C \in \mathcal{S})} \gamma(j+1, C) Q_{kj}(A) \Pr(C \mid A) & \text{otherwise} \end{cases}
\end{aligned}$$

here $\xi(k, A) = (j, C)$ is used to remember the end position j of the first bracket and the speech act of the second bracket that follows the first A bracket. in the bracketing scheme for $d_k \cdots d_n$ that starts with the speech act A and maximizes $\gamma(k, A)$. By choosing $\hat{A} = \arg\max_A \Pr(A \mid \texttt{<d>}) * \gamma(0, A)$ as the speech act of the first bracket and backtracking with ξ (starting from $\xi(0, \hat{A})$), the optimal bracketing and labeling of the whole dialogue can be recovered.

The time complexity of the dynamic programming algorithms is $O(n^2 * |\mathcal{S}|^2)$, where n is the length of a dialogue.

4.3 Performance

The bracketing algorithm was applied to the Janus scheduling data with 19 different speech acts, exclusive of <d> and </d>.

A data set of 96 dialogues was hand bracketed. The dialogues contain 1400 utterances or

Subject	# of Segments	# of Common Segments	Precision	Recall
Human 1	417	343	82.3%	86.2%
Human 2	398	343	88.9%	82.3%

Table 4.1: Human Bracketing Performance: each subject's performance was evaluated with the other's bracketing as reference.

Subject	# of Segments	# of Common Segments	Precision	Recall
Human 1	417	307	—	—
Machine	377	307	81.4%	73.6%

Table 4.2: Machine Bracketing Perfermance with the Reference Made by Subject 1

around 40,000 words. The data were segmented into around 6900 segments and each was labeled with a speech act. A speech act bigram model and 19 speech act dependent word bigram models were trained with the data set. The speech act dependent bigram models were smoothed with a speech act independent model with deleted interpolation (Jelinek and Mercer, 1980). The speech act independent bigram model was trained with a corpus of around 420,000 tokens in the same domain, which is the same language model used in statistical machine translation.

A test set of 8 dialogues (117 utterances) was bracketed with the Viterbi dynamic programming algorithm. It was also bracketed by two human subjects. Here the precision/recall score was used to evaluate the performance of the algorithm. Since dialogue bracketing is a subjective task, different people may come up with different bracketing schemes. The first experiment compared the bracketing made by two human subjects. In doing so, we were able to estimate the performance upper bound of the bracketing algorithm. Table 4.1 shows the precision/recall scores of a human subject, with the other's bracketing as reference.

Table 4.2 and Table 4.3 show the performance of the bracketing algorithm, with the bracketing made by human subject 1 and subject 2 as references.

Since the major purpose of the model is to cope with the long utterance difficulty in machine translation, we are more interested in the segmentation (without speech act labeling)

Subject	# of Segments	# of Common Segments	Precision	Recall
Human 2	398	302	—	—
Machine	377	302	80.1%	75.9%

Table 4.3: Machine Bracketing Perfermance with the Reference Made by Subject 2

Subject	# of Boundaries	# of Common Boundaries	Precision	Recall
Human 1	417	365	87.5%	91.2%
Human 2	398	365	91.7%	87.5%

Table 4.4: Human Segmentation Perfermance: each subject's segmentation was evaluated with the other's segmentation as reference.

Subject	# of Boundaries	# of Common Boundaries	Precision	Recall
Human 1	417	351	—	—
Machine	377	351	93.1%	84.2%

Table 4.5: Machine Segmentation Perfermance with the Reference Made by Subject 1

performance. In this case, mislabeling a segment was not considered an error, since we were only interested in how an utterance could be correctly segmented into smaller units. Table 4.4 shows the precision/recall scores of a human subject's performance on segmentation with the other's segmentation as the reference.

Table 4.5 and Table 4.6 show the segmentation performance of the bracketing algorithm, with the segmentations made by human subject 1 and subject 2 as references. It is clear that segmentation performance is close to human performance.

Another experiment was conducted to check if the dialogue model was helpful in language modeling. The perplexity of a 20 dialogue test set with around 2,400 words was calculated with three different models: a speech act independent bigram model; a bracketing model that computed dialogue probability with (4.4); and a Viterbi bracketing model that computed the probability of a dialogue with its Viterbi bracketing scheme only. Table 4.7 shows the perplexities of these three models. In the Viterbi Bracketing Model, the dialogue likelihood was calculated with only one bracketing scheme, hence it was underestimated. While the test data perplexity was higher, it was very close to the performance of the speech act independent model. The Bracketing Model calculated the dialogue likelihood over all possible bracketing schemes, and it reduced test data perplexity.

Subject	# of Boundaries	# of Common Boundaries	Precision	Recall
Human 2	398	346	—	—
Machine	377	346	91.8%	86.9%

Table 4.6: Machine Segmentation Perfermance with the Reference Made by Subject 2

Model	Perplexity
SA Independent Bigram	42.2
Bracketing Model	39.6
Viterbi Bracketing Model	43.7

Table 4.7: Test Data Perplexities of Different Models.

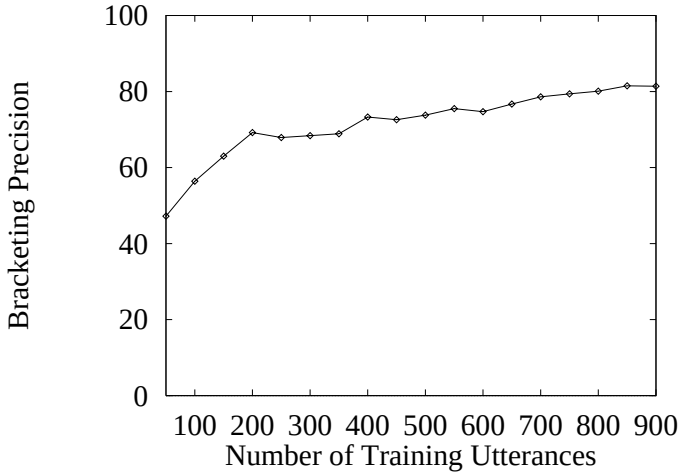

Figure 4.1: Training Data and Bracketing/Segmentation Precision

4.4 Discussion

The bracketing algorithm requires hand made training samples. Although the simplicity of the dialogue structure and a special Emacs editing mode make hand-labeling much easier, it is still a tedious and time-consuming task. However, data labeling can proceed with bootstrapping. We can start with labeling a few hundred utterances, train a system with these initial data, use the trained system to bracket more data, fix the errors in the newly bracketed data, and use them for further training. Figures 4.1 and 4.2 respectively show the relations between the precision/recall scores and the amount of training data. The algorithm was able to achieve adequate performance after the first few hundred samples.

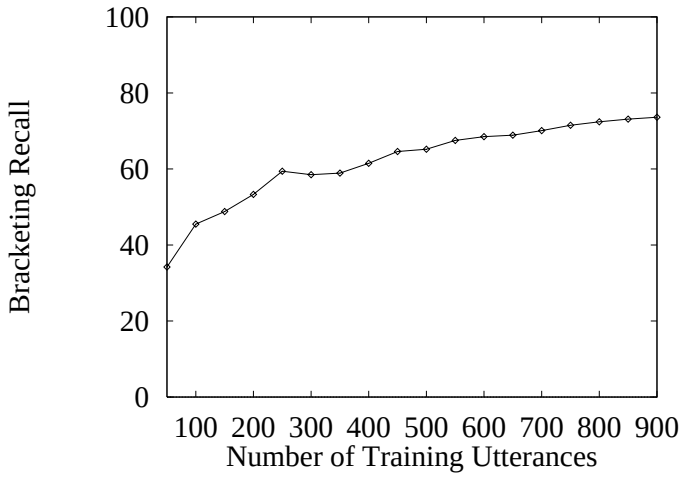

Figure 4.2: Training Data and Bracketing/Segmentation Recall

Chapter 5

Finding Structures for Statistical Machine Translation

As discussed in Chapter 2, the structure-based alignment model is a promising solution for the difficulties in statistical machine translation for spoken language. However, it is of little interest if we have to manually find the language structures and write a grammar for them, since the primary merit of statistical machine translation is to reduce human labor. In this chapter I describe a grammar inference technique that finds the phrases to be used in the structure-based alignment model for statistical machine translation.

5.1 Shallow Phrase Structures

Before introducing the grammar inference algorithm, it is important to know what kind of structures should be acquired, i.e., what kind of structures will facilitate the alignment model without over-complicating the model and thus making it inefficient. The criteria for good structures are:

Simplicity: The structure should be very simple. It can be easily incorporated into an alignment model, and the resulting model should be simple too — we cannot afford a complicated model with too many free parameters or a model that makes source language decoding very inefficient.

Coverage: The structures should have a good coverage of the corpus. If a majority part of the corpus is not covered by the structures, then the structure-based alignment will degenerate to a mostly word-based alignment model.

Learnability: The structure can be easily acquired from a corpus with automatic learning algorithms.

5.1.1 Language Structures

I adopt a very practical definition of language structures. A language structure is a construction that is

1. common enough to be observed frequently in a language
2. general enough to cover multiple instances.

We say that $\mathbf{S} \rightarrow \mathbf{NP\ VP}$ is a language structure in English because **NP VP** is a frequently observed sequence, and it is general enough to cover multiple instances like "David laughed" or "Time flies like an arrow."

The example indicates that a structure can be represented as a sequence of nonterminals. The sequence specifies the components in the structure, while the nonterminals generalize structures to cover multiple instances.

To make structures more suitable for statistical machine translation, I used a simplified definition of structures. Here structures are still sequences of nonterminals, but they are "shallow" ones — they don't allow recursive definition of nonterminals. A nonterminal can only represent a set of terminals. No substructures are allowed in the definition of a nonterminal. A sentence is a linear ordered list of structures. This can be illustrated with the following example:

Example 5.1.1 Shallow Phrase Structures

Suppose that we have the following shallow phrase structures — sequences of word classes. The brackets mark the word classes: words inside a pair of brackets have similar meanings or grammatical functions and thus form a class. Word classes can be referred to with nonterminals.

```
[on at ...] [Sunday Monday...] [afternoon morning...]
[I we ...] [can could ...] [meet get ...]
```

then the sentence "I could meet on Wednesday afternoon" can be parsed into an ordered list of two phrases:

```
(I could meet) (on Wednesday afternoon)
```

The simplicity of structures enables us to effectively build a structure-based alignment model. (See Chapter 6.)

5.2 Grammar Inference with Clustering and Structuring

As mentioned in Chapter 1, the gist of grammar inference algorithms lies in the two tasks they perform: data explanation and generalization. The grammar inference algorithms introduced here are based on the work in (Ries, Buø, and Wang, 1995), which performed the two tasks with the following two operators:

1. **Clustering:** Word/phrases with similar meanings/grammatical functions are clustered into equivalent classes. The mutual information clustering algorithm (Brown et al., 1992a) was used for this purpose. The clustering operator fulfills the generalization task of grammar inference. For example, if we know that "Monday" and "Tuesday" behave similarly and put them into the same equivalent class, then we can generalize the observation of the phrase "Monday morning" in the corpus to infer that "Tuesday morning" is also a legitimate language construction.

2. **Phrasing:** The equivalent class sequence $c_1, c_2, \cdots c_k$ forms a phrase if

$$\Pr(c_1, c_2, \cdots c_k) \log \frac{\Pr(c_1, c_2, \cdots c_k)}{\Pr(c_1)\Pr(c_2)\cdots\Pr(c_k)} > \theta, \tag{5.1}$$

   where θ is a threshold. By changing the threshold, we can obtain a different number of phrases. The phrasing operator fulfills the data explanation task. It "explains" why the components of a phrase, like "Monday" and "morning" can be put together in a sentence, while an arbitrary word like "it" is not often put together with "Monday" in a sentence.

These two operators are applied to the training corpus in alternative steps. The clustering operator puts words or non-terminal symbols having similar usage in the same equivalent

class, and label that class with a new nonterminal node. Rewriting rules are then introduced to rewrite each element in the class as its non-terminal label. For example, if the words `Sunday Monday ... Saturday` are clustered into a class, and the class is labeled as `D_O_W`, then the rules `D_O_W` $\rightarrow$ `Sunday` , ... , `D_O_W` $\rightarrow$ `Saturday` are introduced to the grammar, and all the occurrences of `Sunday Monday ... Saturday` in the training corpus are replaced with `D_O_W`. The phrasing operator finds "sticky" sequences of words or nonterminals with mutual information criterion, and formulate new structural production rules to generate the sequence. As an example of sequence rules, `Temporal`$\rightarrow$ `D_O_W T_O_D` means that the sequence `D_O_W T_O_D` is a meaningful unit and will be rewritten as a new nonterminal, representing a new concept or phrase, say `Temporal`. The structure `Temporal` can decompose into terminal strings like `Friday Morning, Tuesday Afternoon`, etc. For each newly introduced sequence production, the corresponding sequences in the training corpus are also replaced by the left hand side nonterminal of the production rule. This clustering/phrasing procedure iterates until all sentences in the training corpus are reduced to a starting node. Figure 5.1 shows a grammar part that has emerged from this training algorithm for the scheduling database. The algorithm produces structures that frequently correspond to meaningful entities, such as prepositional phrases, semantic concepts or fragments, etc.

The rules obtained in this way are combinations of syntactic and semantic regularities. They reflect domain-specific word usage. As we can see from the previous example, the word "lunchtime" is clustered in the same class as numerals, because those numerals are mostly used in time expressions in the scheduling domain.

When we use this approach to find structures for statistical machine translation, we only apply each operator once to acquire shallow phrase structures.

5.3 Bilingual Grammar Inference

Since the algorithm only uses a monolingual corpus, it often introduces some language-specific structures resulting from biased usage of a specific language. For example, when we applied the clustering algorithm in (Brown et al., 1992a) to scheduling data, we found many cases like the class {`couple few lot message`}, in which the word `message` is out of place. This is due

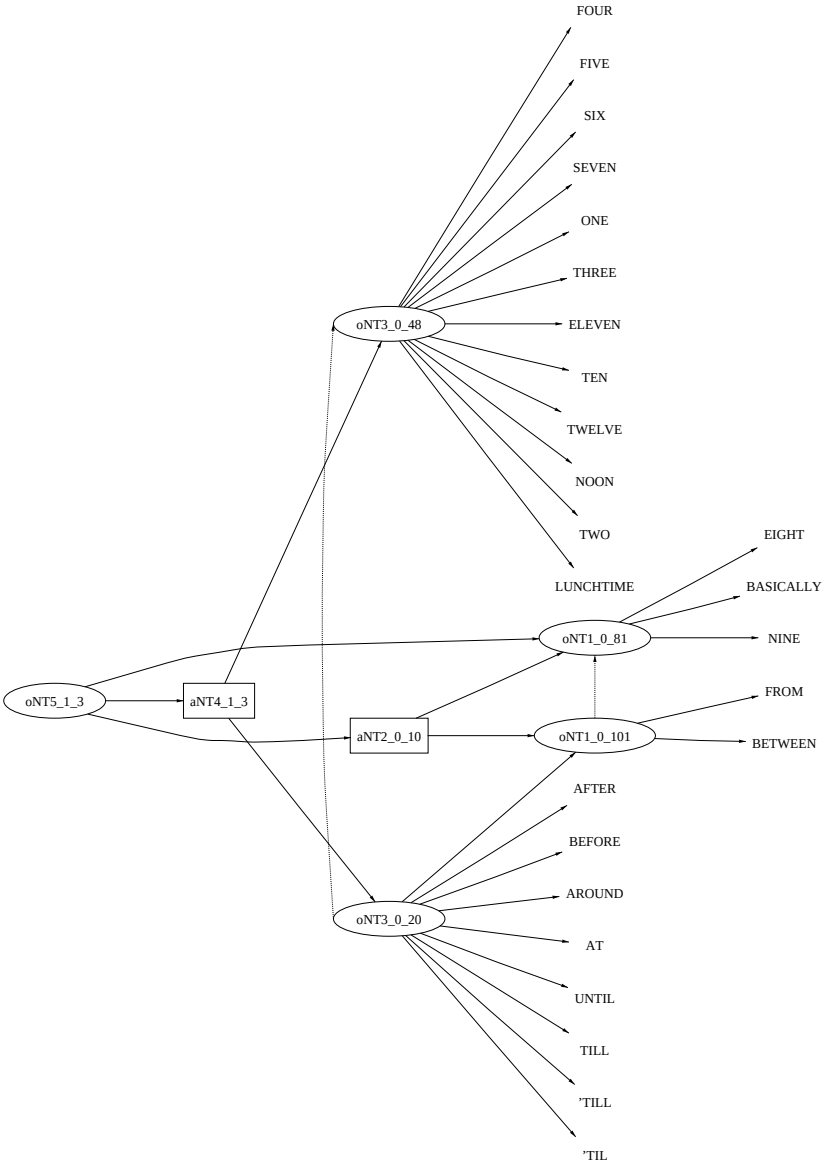

Figure 5.1: Inferred Rules. Ovals represent cluster nodes, their children are words/phrases in the cluster. Rectangles are phrase nodes, their children are the components of the phrase, and the dashed arrows indicate the sequential relations among the phrase components.

to the fact that the clustering technique is based on local information from word bigrams, and each word in this class typically follows the word `a` and precedes the word `of` or `to` in the training corpus.

Instead of language-specific structures, we are more interested in cross-linguistic structures in machine translation. This is similar to the case of interlingua that represents cross-linguistic knowledge in knowledge-based MT. To find structures that are common in two languages, I have to take constraints from both languages. For this purpose, we have introduced the bilingual clustering and bilingual phrasing operators.

5.3.1 Bilingual Word Clustering

To obtain structures that are common in both languages, a bilingual mutual information clustering algorithm (Wang, Lafferty, and Waibel, 1996) was used as the clustering operator. Compared to a clustering algorithm based on a monolingual corpus, a clustering algorithm that takes constraints from a parallel corpus potentially has several advantages. First, training samples in another language provide indirect evidence for a classification. Second, constraints from both languages may help to "wash out" biased language-specific usage, resulting in classes of better quality.

The bilingual clustering algorithm described here is based on this mutual information clustering algorithm (Brown et al., 1992a). It is one of the maximum likelihood classification algorithms (Brown et al., 1992a; Kneser and Ney, 1993), which seeks a classification C such that $\Pr(W \mid C)$, the class-based likelihood of corpus W, is maximized. It was shown in (Brown et al., 1992a) that maximizing the log-likelihood of a corpus with a class-based bigram is equivalent to maximizing the average mutual information $I(C_1, C_2)$ between adjacent classes in text:

$$\frac{1}{n-1} \log \Pr(W \mid C) \approx -H(W) + I(C_1, C_2) \tag{5.2}$$

where $H(W)$ is the entropy of the corpus, which is independent of the clustering. A greedy algorithm was then introduced to find classes that maximize the average mutual information. Initially each word is assigned to a distinct class and the average mutual information between adjacent classes is computed. The algorithm then iterates to merge classes. Each class merge will cause the loss of mutual information between adjacent classes. At each step in the iteration,

the loss in average mutual information that results from merging each candidate pair of classes is computed, and the merge is then carried out for the pair affecting the smallest loss.

The bilingual clustering algorithm described here is based on this mutual information clustering technique. To employ the constraints from a parallel corpus, alignments between pairs of sentences (Brown et al., 1993) were used as a "bridge" between the languages. To be concrete, suppose we have an English corpus E and its parallel German corpus G, and we want to cluster the English words appearing in E. Instead of maximizing the log-likelihood $\log \Pr(E \mid C)$, we seek to maximize the joint log-likelihood of the parallel corpus:

$$\begin{aligned} \frac{1}{n-1} \log \Pr(E, G \mid C) &= \frac{1}{n-1}(\log \Pr(E \mid C) + \log \Pr(G \mid E, C)) \\ &\approx -H(E) + I(C_1, C_2) + \frac{1}{n-1} \log \Pr(G \mid E, C) \end{aligned} \tag{5.3}$$

where

$$\Pr(G \mid E, C) = \sum_{i}^{L} \sum_{A} \Pr(G_i, A \mid E_i, C) \tag{5.4}$$

Here E_i and G_i are the i^{th} pair of utterances in the parallel corpus, L is the number of sentences in the corpus, and A is an alignment between E_i and G_i.

We can initially assign each word to a separate class, and incrementally merge classes using a greedy search algorithm. At the k^{th} step in the algorithm, the decrease in likelihood (5.3) resulting from a merge of classes c_1 and c_2 can be expressed as a sum of two terms: $L_k(c_1, c_2)$, the loss of average mutual information between adjacent classes, and $D_k(c_1, c_2)$, the change in the likelihood of the German corpus when c_1 and c_2 are merged. With clever bookkeeping, one can efficiently find the smallest $L_k(c_1, c_2)$ in time $O(V^2)$, where V is the lexicon size (Brown et al., 1992a). In the following section I describe a method to efficiently calculate $D_k(c_1, c_2)$ using a class-based translation model.

Implementation and Complexity

To model the change in likelihood of the German corpus, I employ a slight modification of IBM Alignment Model 1. This model generates the German corpus from the English corpus using a

simple alignment between word pairs:

$$\Pr(G_i, A \mid E_i) = \frac{\epsilon}{(|E_i|+1)^{|G_i|}} \prod_{j=1}^{|G_i|} t(g_j \mid e_{a_j}). \tag{5.5}$$

Equation (5.5) can be interpreted as follows: Given English sentence E_i, the length of its German translation G_i, $|G_i|$, is first picked according to a fixed probability ϵ. Then the source position a_j corresponding to a position j in G_i is determined by a uniform distribution, i.e., j is aligned to any position in its English translation E_i with the equal likelihood $(|E_i|+1)^{-1}$. The German word at position j, g_j, is then generated from the English word e_{a_j} at the position aligned to j according to the *translation probability* $t(g_j \mid e_{a_j})$. The parameters cab be estimated with EM algorithm.

By "tying" the translations probabilities so that $t(g_j \mid e_{a_j}) = t(g_j \mid c_{e_{a_j}})$, where c_e is the class of English word e, the model can be expressed as

$$\begin{aligned}
\Pr(G_i \mid E_i, C) &= \sum_A \Pr(G_i, A \mid E_i, C) \\
&= \frac{\epsilon}{(|E_i|+1)^{|G_i|}} \sum_{a_1=0}^{|E_i|} \cdots \sum_{a_{|G_i|}=0}^{|E_i|} \prod_{j=1}^{|G_i|} t(g_j \mid c_{e_{a_j}}) \\
&= \frac{\epsilon}{(|E_i|+1)^{|G_i|}} \prod_{j=1}^{|G_i|} \sum_{k=0}^{|E_i|} t(g_j \mid c_{e_k}) \,.
\end{aligned} \tag{5.6}$$

Therefore,

$$\begin{aligned}
D_k(c_1, c_2) &= \sum_i^L \log \Pr(G_i \mid E_i, C(c_1+c_2)) - \sum_i \log \Pr(G_i \mid E_i, C) \\
&= \sum_i^L \sum_{j=1}^{|G_i|} \log \left(\sum_{k=0}^{|E_i|} t'(g_i \mid c'_{e_k}) \,/\, \sum_{k=0}^{|E_i|} t(g_i \mid c_{e_k}) \right)
\end{aligned} \tag{5.7}$$

where C is the classification before the merge of c_1 and c_2, $C(c_1+c_2)$ is the classification after the merge, c_e is the class of e in C, c'_e is the class of e in $C(c_1+c_2)$, and t' is the new translation probability after the merge of c_1 and c_2.

Although (5.7) can be used to calculate the likelihood change of the second language corpus, it is not practical for implementation. To estimate the likelihood change of the German corpus after a merge, we would in principle need to know the new parameters t'. Since they are estimated with EM training, and all of the parameters could be affected by a single merge, the

bookkeeping method that works for monolingual clustering is not applicable in the bilingual case.

To reduce the computational demands, the following approximating assumptions have been made:

1. The merge of classes c_1 and c_2 will not affect the translation distributions for classes other than c_1 and c_2. That is, $t(g \mid c)$ will remain unchanged, for $c \neq c_1$,and $c \neq c_2$. For the merged class $c1+c2$, its translation distribution can be approximated without re-training:

$$\begin{aligned} t(g \mid c1+c2) &\approx t(g \mid c1)\Pr(c1 \mid c1+c2) + t(g \mid c2)\Pr(c2 \mid c1+c2) \\ &= \frac{t(g \mid c1)\Pr(c1) + t(g \mid c2)\Pr(c2)}{\Pr(c1)+\Pr(c2)} \end{aligned} \tag{5.8}$$

2. The translation distributions will not change significantly for at least M merges.

3. The best potential merge pair c_1, c_2 is within the top N merge candidates with lowest $L_k(c_1, c_2)$ identified by the monolingual clustering algorithm.

With approximation 1, we do not need to retrain the parameters for each potential merge. Similarly, with approximation 2, we can avoid reestimating the parameters after each merge is actually carried out. With approximation 3, we only have to calculate $D_k(c_1, c_2)$ for N pairs. Figure 5.2 illustrates the average percentage of agreement between the Viterbi alignments of the parallel corpus with the approximated parameters and with the re-estimated parameters, as a function of the number of merging steps. It shows that approximation (1) and (2) are reasonable up to $M = 5$.

With these simplifying assumptions, we obtain the following algorithm:

Algorithm 5.3.1 Bilingual Clustering

1. Initialization: assign a distinct class to each word e. Compute $L_V(c_1, c_2)$ and the other variables used in monolingual clustering for all pairs of English classes c_1, c_2.

2. Alignment: Train the parameters $t(g \mid c)$ of the class-based translation model using the EM algorithm.

3. Set **no-reestimation-count** to 0.

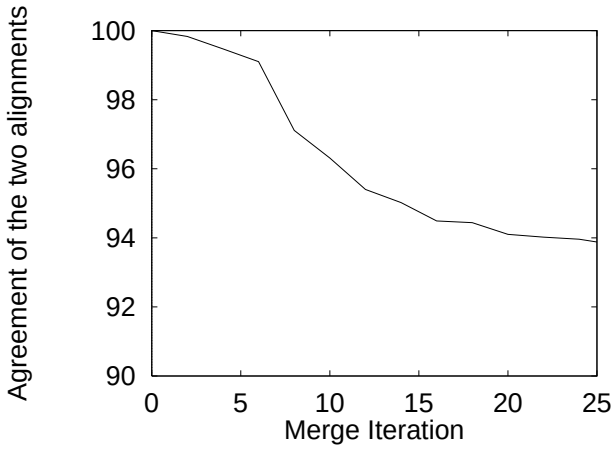

Figure 5.2: The average agreement of the Viterbi alignments of the parallel corpus with the approximated parameters and the re-trained parameters.

4. Repeat the followings:

(a) With the monolingual clustering technique, find the N pairs c_1, c_2 with the smallest $L_k(c_1, c_2)$.

(b) For each pair c_1, c_2 of the N merge candidates, compute $D_k(c_1, c_2)$. Re-score the pair c_1, c_2 with $L_k(c_1, c_2) + D_k(c_1, c_2)/(n-1)$, where n is the number of words in the English corpus.

(c) Merge the pair c_1, c_2 with the lowest score.

(d) Increase **no-reestimation-count** by 1.

(e) If **no-reestimation-count**$> M$, reestimate the translation probabilities with EM algorithm, and set **no-reestimation-count** to 0.

Pragmatic Issues

The performance of clustering relies heavily on the amount of training data. Since we have a larger monolingual corpus available in addition to the parallel corpus, I modified the above algorithm to select a pool of merge candidates with the monolingual corpus, i.e., find the top N merge candidates that will result in the smallest loss of mutual information. The bilingual constraints were then used to pick the best pair from the pool.

An additional constraint was used in clustering to get around with the sparse data problem. The constraint required that words in the same class have at least one common potential part-of-speech.

I also manually pre-classified four classes of words — 12 month names, 7 weekday names, ordinal numbers and cardinal numbers. Those words appear frequently in the scheduling corpus. Normally the automatic clustering algorithm can group these words together, maybe into several different classes. For example, one of my experiments found two classes for weekday names: {`Monday, Thursday, Saturday`} and {`Tuesday, Wednesday, Friday, Sunday`}. By manually assigning them into a unique equivalent class, the resulting structures become more compact.

Words that occur fewer than 5 times did not participate in the mutual information clustering. They were classified in the following way:

1. Low frequency words were classified into different classes according to their morphological suffixes. For example, words ended with "-able" were put into the same class, and words ended with "-ness" were put into another class;

2. The rest of the low frequent words were classified according to their part-of-speeches.

5.3.2 Bilingual Phrasing

Bilingual heuristics were used to filter out the sequences obtained by the phrasing operator that may not be common in multiple languages. The heuristics include:

1. **Average Translation Span:** Given a phrase candidate, its average translation span is the distance between the leftmost and the rightmost target positions aligned to the words inside the candidate, averaged over all Model 1 Viterbi alignments of sample sentences. A candidate is filtered out if its average translation span is greater than the candidate's length multiplied by a threshold. Basically this criterion states that the words in the translation of a phrase must be close enough to form a phrase in another language.

2. **Ambiguity Reduction:** a word occurs in a phrase should be less ambiguous than in other random contexts. Therefore a phrase should reduce the ambiguity (uncertainty) of

```
[Sunday Monday...][afternoon morning...]
[Sunday Monday...][at by...][one two...]
[Sunday Monday...][the every each...][first second third...]
[Sunday Monday...][the every each...][twenty depending remaining]
[Sunday Monday...][the every each...][eleventh thirteenth...]
[Sunday Monday...][in within...][January February...]
[January February...][first second third...][at by...]
[January February...][first second third...]
[January February...][the every each...][first second third...]
[I he she itself][have propose remember hate...]
[eleventh thirteenth...][after before around][one two three...]
```

Figure 5.3: Example of Acquired Phrases.

the words inside it. The ambiguity or uncertainty of word translations can be measured with translation entropy: for each source language word class c, its translation entropy is defined as $\sum_g t(g \mid c) \log(g \mid c)$. The average per source class entropy reduction induced by the introduction of a phrase P is therefore

$$\frac{1}{|P|} \sum_{c \in P} [\sum_g t(g \mid c) \log t(g \mid c) - \sum_g t(g \mid c, P) \log t(g \mid c, P)] \tag{5.9}$$

which is the amount of uncertainty taken away by the introduction of P. A threshold was set up for minimum entropy reduction.

Figure 5.3 shows some of the phrases learned with the bilingual grammar inference algorithm. The whole set of phrases can be found in Appendix **??**.

5.4 Two Languages are More Informative than One

Several experiments were carried out with the bilingual grammar inference algorithm.

The first experiment was conducted with the bilingual clustering algorithm. I built two class-based bigram models, with classes obtained from monolingual and bilingual mutual information clustering algorithm[1]. The perplexity of the language model with monolingual classification was 36.9, while the perplexity of the bilingual trained model was 35.2 (measured with the

[1]The experiment was conducted without manual pre-classification of month and weekday names.

training data). While this improvement was not significant, it appeared that the new clustering algorithm found classes of higher quality. Table 5.1 and Table 5.2 list some of the classes discovered by the monolingual and bilingual algorithms.

```
say +re
are unless days times
fact May January November July having department case Wean
after around before between
or +ah+ afternoons
out fine
free clear available open
and however otherwise idea Patty through
day weekend right Mark
good perfect space nice great better away
pretty completely totally real
half m date conference cream bit
what afterwards why
couple few lot message
```

Table 5.1: Example Word Classes Discovered with Monolingual Mutual Information Clustering

```
are +re
January May November July fact
one noon
it early
or through
after before between
hours weeks days times
all
still had certainly may completely totally
well yeah unfortunately John Patty Mark
fine great better perfect nice
what when where
third sixteenth eleventh lounge thirtieth fifteenth
couple little bit lot half
```

Table 5.2: Example Word Classes Discovered with Bilingual Mutual Information Clustering

In Table 5.1, the "month name" class {`fact May January November July having department case Wean`} is mixed with what might be considered "noise" words, which appear because of

various biased, language-specific usage of words — in this specific case, many of those words frequently follow the preposition `in`. This is improved in Table 5.2. The same effect occurs in many other classes.

How does bilingual clustering achieve this improvement? This can be explained as follows. The alignment model will assign some probability mass not only to the correct translations of the classes, but also to words that appear frequently in the same sentences with the correct translations. This spreading of the probability is less harmful if the classes contain semantically similar words. Since semantically similar words usually appear in similar contexts (in this case, sentences), although the class-based probability may reduce the probability of the correct translation of a word, it may raise the probability of other words in the context of the correct translation. If a class contains words of distinct meanings, because those words generally occur in different contexts, the translation probabilities can become much more spread out over different contexts, hence the overall sentence translation probability will be reduced significantly.

To be more precise, we define the *ϵ-mirror* of an input language class C_i as the set of all possible translations of C_i in another language with translation probability greater than ϵ:

$$C_i^\epsilon = \{s : t(s \mid C_i) > \epsilon\} \tag{5.10}$$

The average size of an ϵ-mirror indicates the extent to which the translation probability is spread out. With $\epsilon = 0.05$, the bilingual clustering has an average ϵ-mirror size of 3.46 words for the classes discovered by the mutual information clustering (i.e., classes of words with more than 5 occurrences in the corpus), while the monolingual clustering has an average size of 4.31.

The spreading of the translation distributions can be also measured with the conditional entropy

$$H(G \mid C_E) = -\sum_{c_E} \Pr(c_E) \sum_{g} t(g \mid c_E) \log t(g \mid c_E) \tag{5.11}$$

over all classes of words occurred more than 5 times in the corpus. This measure reflects the uncertainty of the target German word given a source English class. The conditional entropy is 2.52 with the bilingual-trained classes, and 2.60 with the monolingual trained classes.

The third experiment is about the direct impact of the bilingual grammar inference in the

translation performance of the structure-based system. I will discuss this in Chapter 8 after I introduce the structure-based translation model.

5.5 Phrase Structure Parsing

Given a set of phrases, a sentence can be deterministically parsed into a sequence of phrases by replacing the leftmost unparsed substring with the longest matching phrase.

Chapter 6

The Structure-based Alignment Model

In the previous chapter, I presented an algorithm to find the shallow phrase structures from a corpus. The structures are going to be used in a structure-based alignment described in this chapter.

6.1 The Model

For German to English translation, we have to use a stochastic process to model the translation from an English sentence $\mathbf{e}$ to a German sentence $\mathbf{g}$, so we can know the probability $\Pr(\mathbf{g} \mid \mathbf{e})$, and then use a decoding algorithm to find the English sentence that maximizes $\Pr(\mathbf{g} \mid \mathbf{e}) \Pr(\mathbf{e})$. In the structure-based translation model, the translation from an English sentence $\mathbf{e} = e_1 e_2 \cdots e_l$ to its German counterpart $\mathbf{g} = g_1 g_2 \cdots g_m$ can be modeled with the following process:

1. Parse $\mathbf{e}$ into a sequence of phrases E, so

$$\begin{aligned} E &= (e_{11}, e_{12}, \cdots, e_{1l_1})(e_{21}, e_{22}, \cdots, e_{2l_2}) \cdots (e_{n1}, e_{n2}, \cdots, e_{nl_n}) \\ &= E_0 E_1 E_2 \cdots E_n, \end{aligned}$$

   where E_0 is a null phrase. The parsing can be done deterministically with the algorithm described in the previous section.

2. With the probability $\Pr(q \mid \mathbf{e}, E)$, determine q, the number of phrases in $\mathbf{g}$. Let $G_0 \cdots G_{q-1}$ denote these q phrases. Unlike English phrases, words in a German phrase do not have to form a consecutive sequence. So $\mathbf{g}$ may be expressed with something like $\mathbf{g} = g_{11}g_{12}g_{21}g_{13}g_{22}\cdots$, where g_{ij} represents the j^{th} word in the i^{th} phrase of $\mathbf{g}$.

3. For each German phrase $G_i, 0 \leq i < q$, with the distribution $\Pr(r_i \mid i, r_0^{i-1}, q, \mathbf{e}, E)$, align it with an English phrase E_{r_i}. Each source phrase can be aligned with at most one target phrase.

4. For each German phrase $G_i, 0 \leq i < q$, if it is not aligned to the null word source phrase, determine its beginning position b_i in $\mathbf{g}$ with the distribution $\Pr(b_i \mid i, b_0^{i-1}, r_0^{q-1}, q, \mathbf{e}, E)$. Call this position the *anchor point* of G_i

5. Now it is time to generate the individual words in the German phrases through *detailed alignment*. For each word e_{ij} in phrase E_i, $i > 0$, its fertility ϕ_{ij} has the distribution $\Pr(\phi_{ij} \mid i, j, \phi_{i1}^{i(j-1)}, \phi_0^{i-1}, b_0^{q-1}, r_0^{q-1}, q, \mathbf{e}, E)$.

6. For each word e_{ij} in the phrase E_i, $i > 0$, it generates a tablet $\tau_{ij} = \{\tau_{ij1}, \tau_{ij2}, \cdots \tau_{ij\phi_{ij}}\}$ by generating each of the words in τ_{ij} in turn, with the following probability for the k^{th} word in the tablet:

$$\Pr(\tau_{ijk} \mid \tau_{ij1}^{k-1}, \tau_{i1}^{j-1}, \tau_0^{i-1}, \phi_0^l, b_0^{q-1}, r_0^{q-1}, q, \mathbf{e}, E). \tag{6.1}$$

7. For each element τ_{ijk} in the tablet τ_{ij}, $i > 0$, the permutation π_{ijk} determines its position in the target sentence according to the distribution

$$\Pr(\pi_{ijk} \mid \pi_{ij1}^{k-1}, \pi_{i1}^{j-1}, \pi_0^{i-1}, \tau_0^l, \phi_0^l, b_0^{q-1}, r_0^{q-1}, q, \mathbf{e}, E). \tag{6.2}$$

8. Determine the fertility and translations of the null word and place the translation in the target sentence in the way described late in this chapter.

Obviously it is not realistic to use the above probabilities directly, because it would result in an enormous number of parameters. Instead the following independent assumptions are made:

1. The number of target sentence phrases depends only on the number of phrases in the source sentence:

   $\Pr(q \mid \mathbf{e}, E) = p_n(q \mid n)$

2. $\Pr(r_i \mid i, r_0^{i-1}, q, \mathbf{e}, E) = a(r_i \mid i) \times \prod_{0 \leq j < i}(1 - \delta(r_i, r_j))$

   i.e., $\Pr(r_i \mid i, r_0^{i-1}, q, E)$ depends on i and r_i. It also depends on r_0^{i-1} with the factor $\prod_{0 \leq j \leq i}(1 - \delta(r_i, r_j))$ to ensure that each English phrase is aligned with at most one German phrase.

3. The anchor point of a target phrase depends on its distance from the anchor point of its preceding phrase, as well as the length of the source phrase aligned to that preceding phrase:

   $\Pr(b_i \mid i, b_0^{i-1}, r_0^{q-1}, q, \mathbf{e}, E) = \alpha(b_i - b_{i-1} \mid |E_{r_{i-1}}|) = \alpha(\Delta_i \mid |E_{r_{i-1}}|)$

4. The fertility of a source word depends only on the word:

   $\Pr(\phi_{ij} \mid i, j, \phi_{i1}^{i(j-1)}, \phi_0^{i-1}, b_0^{q-1}, r_0^{q-1}, q, \mathbf{e}, E) = n(\phi_{ij} \mid e_{ij})$

5. The translation tablet of a source word only depends on the word:

   $\Pr(\tau_{ijk} \mid \tau_{ij1}^{k-1}, \tau_{i1}^{j-1}, \tau_0^{i-1}, \phi_0^l, b_0^{q-1}, r_0^{q-1}, q, \mathbf{e}, E) = t(\tau_{ijk} \mid e_{ij})$

6. The leftmost position of the translations of a source word, henceforth the *pin point* of the source word, depends on its distance from the anchor point of the target phrase aligned to the source phrase that contains that source word. It also depends on the identification of the source phrase, and the position of the source word in that source phrase:

   $$Pr(\pi_{ij1} \mid \pi_{i1}^{j-1}, \pi_0^{i-1}, \tau_0^l, \phi_0^l, b_0^{q-1}, r_0^{q-1}, q, \mathbf{e}, E) = d_1(\pi_{ij1} - b_i \mid E_i, j) \tag{6.3}$$

   For a target word τ_{ijk} other than the leftmost τ_{ij1} in the translation tablet of the source word e_{ij}, its position depends on its distance from the position of the tablet word $\tau_{ij(k-1)}$ closest to its left, as well as the class of the current target word:

   $$\Pr(\pi_{ijk} \mid \pi_{ij1}^{k-1}, \pi_{i1}^{j-1}, \pi_0^{i-1}, \tau_0^l, \phi_0^l, b_0^{q-1}, r_0^{q-1}, q, \mathbf{e}, E) = d_2(\pi_{ijk} - \pi_{ij(k-1)} \mid \mathcal{G}(\tau_{ijk})) \tag{6.4}$$

here $\mathcal{G}(g)$ is the equivalent class for g, which can be obtained with the bilingual clustering algorithm described in section 5.

7. The treatment of the hypothetical null word in the source sentence is the same as in Model 3. The fertility for the null word at the position 0 of the source sentence, ϕ_0, is determined on the assumption that each target word generated from its aligned source word requires an extraneous word with probability p_1, and this extraneous word must be aligned to the null word. The probability that exactly ϕ_0 of the extraneous words are required by the $\sum_{i=1}^{n}\sum_{j=1}^{l_n}\phi_{ij}$ target words that are generated from the non-null source words $e_1e_2...e_l$, can be determined with the binomial distribution:

$$\begin{aligned}\Pr(\phi_0 \mid \phi_1^n, \mathbf{e}) &= n_0(\phi_0 \mid \sum_{i=1}^{n}\sum_{j=1}^{l_n}\phi_{ij}) \\ &= \left(\begin{array}{c}\sum_{i=1}^{n}\sum_{j=1}^{l_n}\phi_{ij} \\ \phi_0\end{array}\right)(1-p_1)^{\sum_{i=1}^{n}\sum_{j=1}^{l_n}\phi_{ij}-\phi_0}p_1^{\phi_0}\end{aligned}$$

The target word aligned with the null word are placed in the target sentence one by one after all the other target words have been placed. We assume that the k^{th} translation of the null word is placed at a vacant position with a uniform distribution. Because the 1^{st}, 2^{nd} ... and $(k-1)^{th}$ target words have already be placed at the moment, there are $(\phi_0 - k + 1)$ vacant positions available, therefore the contribution of the placement of the k^{th} translation of the null word is $1/(\phi_0 - k + 1)$, and the contribution of the placement of all the target words aligned to the null source words is $1/\phi_0!$.

Therefore we have

$$\begin{aligned}\Pr(q, r, b, \tau, \pi \mid \mathbf{e}, E) &= p_n(q \mid n) \\ &\times \left(\begin{array}{c}\sum_{i=1}^{|E|}\sum_{j=1}^{|E_j|}\phi_{ij} \\ \phi_0\end{array}\right)(1-p_1)^{\sum_{i=1}^{|E|}\sum_{j=1}^{|E_j|}\phi_{ij}-\phi_0}p_1^{\phi_0}\times\prod_{i=1}^{\phi_0}t(\tau_{0i} \mid \phi) \\ &\times \prod_{i=0,r_i\neq 0}^{q-1} a(r_i \mid i)\prod_{0\leq j<i}(1-\delta(r_i, r_j)) \\ &\times(1+(1-\delta(i,0))\times(\alpha(b_i - b_{i-1} \mid |E_{r_{i-1}}|)-1))\end{aligned}$$

$$\times \prod_{j=1}^{|E_i|} n(\phi_{ij} \mid e_{ij}) \prod_{k=1}^{\phi_{ij}} t(\tau_{ijk} \mid e_{ij})$$
$$\times d_1(\pi_{ij1} - b_i \mid E_i, j)$$
$$\times \prod_{k=2}^{\phi_{ij}} d_2(\pi_{ijk} - \pi_{ij(k-1)} \mid \mathcal{G}(\tau_{ijk}))$$
$$\times (1 - \prod_{j=1}^{|E_i|} (1 - \delta(b_i, \pi_{ij1}))) \tag{6.5}$$

Here $\delta(x, y) = 1$ if $x = y$, otherwise $\delta(x, y) = 0$. The factor $(1 - \prod_{j=1}^{|E_i|}(1 - \delta(b_i, \pi_{ij1})))$ guarantees that the anchor point of each German phrase is actually occupied by a word. Without this restriction, the anchor point is arbitrary and the translation model is not well defined. ϕ is the (source) null word, τ_0 is the tablet for ϕ, and the factors in the second line are the contributions from the target words aligned to the null word. The factor $(1 + (1 - \delta(i, 0)) \times (\alpha(b_i - b_{i-1} \mid |E_{r_{i-1}}|) - 1))$ states that if target phrase i is the first phrase of the sentence, then its anchor point position is fixed as the first word of the sentence with probability 1, otherwise it is determined by the α parameters.

Normally for each possible $\langle \mathbf{g}, \mathbf{a} \rangle$ ($\mathbf{a}$ is a word to word alignment) that aligns positions between source and target and generates $\mathbf{g}$, there might be many $(q, r, b, \tau, \pi)'$s that are in conformity with $\langle \mathbf{g}, \mathbf{a} \rangle$, since there are tablets that contain the same words in a difference sequence. So

$$Pr(\mathbf{a}, \mathbf{g} \mid \mathbf{e}) = \sum_{(q,r,b,\tau,\pi) \in \langle \mathbf{g}, \mathbf{a} \rangle} Pr(q, r, b, \tau, \pi \mid E, \mathbf{e}) \tag{6.6}$$

However, in this specific structure-based model, since we require that subsequent words from τ_{ij} be placed in order — the second word from τ_{ij} has to lie to the right of the first, the third to the right of the second (not necessarily adjacent,) and so on, there is only one arrangement of the words in τ_{ij} that is in conformity with $\langle \mathbf{g}, \mathbf{a} \rangle$. With this arrangement we can calculate $P(\mathbf{a}, \mathbf{g} \mid \mathbf{e})$, as will be illustrated in a later example.

6.1.1 Constraints on the Model

To limit the number of parameters in the model, I added the following constraints to the structure-based model:

- The maximum number of words in a phrase from grammar inference **max_phrase_len** = 6;
- The maximum sentence length **max_sen_len** = 20;
- The maximum number of phrases in a sentence **max_phrase_num** = 15;
- The maximum fertility of a non-null word **max_fertility** = 3;
- The maximum distance between the anchor points of two adjacent phrases, i.e., the maximum disstance allowed in the α parameters, **max_phrase_distance** = **2**$\times$**max_phrase_len** = 12;
- The maximum distance between the pin point of a source word s and the anchor point of the target phrase containing the translation of s (the distance in the d_1 parameters) $\mathbf{max\_d}_1 = 2 \times$ **max_phrase_len** = 12;
- The maximum distance between positions of two target words aligned to the same source word (the maximum distance in the d_2 parameter) is $\mathbf{max\_d}_2 = 2 \times$ **max_fertility** = 6;

6.2 Parameter Estimation

EM algorithm was used to estimate the eight types of parameters: p_n, p_1, a, α, ϕ, τ, d_1 and d_2. It is impossible to used the full-blown EM algorithm, since the number of possible alignments is exponential in the sentence length. Instead, a subset of probable alignments was used in the EM learning. To define the subset unambiguously, I follow the terminology in (Brown et al., 1993): we say that two alignments $\mathbf{a}$ and $\mathbf{a}'$ differ by a move if there is one and only one value j for which $a_j \neq a'_j$. They differ by a swap if $a_j = a'_j$ for every j except at two values, j_1 and j_2, for which $a_{j_1} = a'_{j_2}, a_{j_2} = a'_{j_1}$. Two alignments are neighbors if they are identical or if they differ by a move or a swap. And we use $\mathcal{N}(\mathbf{a})$ to denote the set of all neighbors of an alignment $\mathbf{a}$. We use $b(\mathbf{a})$ to denote the neighbor of $\mathbf{a}$ for which the likelihood $P(b(\mathbf{a}) \mid \mathbf{e}, \mathbf{g})$ is the greatest among the neighbors of $\mathbf{a}$. Suppose that ij is pegged for $\mathbf{a}$, among the neighbors of $\mathbf{a}$ for which ij is also pegged, $b_{i \leftarrow j}(\mathbf{a})$ is the one with the greatest likelihood. The sequence of $\mathbf{a}, b(\mathbf{a}), b(b(\mathbf{a}))$ ... , converges in a finite number of steps to an alignment that we write as $b^{\infty}(\mathbf{a})$.

Similarly, $\mathbf{a}, b_{i\leftarrow j}(\mathbf{a}), b_{i\leftarrow j}(b_{i\leftarrow j}(\mathbf{a})), ...$, converges to $b^{\infty}_{i\leftarrow j}(\mathbf{a})$. In other words, $b^{\infty}(\mathbf{a})$, found with hill-climbing search from $\mathbf{a}$, is the alignment with the greatest likelihood, and $b^{\infty}_{i\leftarrow j}(\mathbf{a})$ is the ij pegged alignment with the greatest likelihood.

Given a sentence pair $(\mathbf{e}, \mathbf{g})$, the alignment subset used to accumulate the counts for parameter estimation, $\mathcal{S}(\mathbf{e}, \mathbf{g})$, was defined as

$$\begin{aligned}\mathcal{S}(\mathbf{e}, \mathbf{g}) &= \mathcal{N}(b^{\infty}(V(\mathbf{g} \mid \mathbf{e}; 1))) \bigcup \mathcal{N}(b^{\infty}(V(\mathbf{g} \mid \mathbf{e}; 2))) \\ &\quad \bigcup \cup_{ij} \mathcal{N}(b^{\infty}_{i\leftarrow j}(V(\mathbf{g} \mid \mathbf{e}; 1))) \bigcup \cup_{ij} \mathcal{N}(b^{\infty}_{i\leftarrow j}(V(\mathbf{g} \mid \mathbf{e}; 2)))\end{aligned} \tag{6.7}$$

which is the union of the neighboring sets of the most probable alignments found with hill-climbing search from the Model 1 and Model 2 Viterbi alignments between $\mathbf{e}$ and $\mathbf{g}$. Here $V(\mathbf{g} \mid \mathbf{e}; 1)$ is the Viterbi alignment between $\mathbf{e}$ and $\mathbf{g}$ according to IBM Alignment Model 1, and $V(\mathbf{g} \mid \mathbf{e}; 2)$ is the Viterbi alignment according to Model 2. I chose to include the Model 1 Viterbi alignment here because the Model 1 alignment is closer to the "ideal" when long distance word-to-word alignments exist between a sentence pair.

The expected counts of the parameters from a sample sentence pair $(\mathbf{e}, \mathbf{g})$ are shown below, where r_i is the source phrase position aligned with the target phrase G_i. a_i is the source word position aligned with the target word g_i. b_i is the anchor point of the target phrase G_i. $I_e(i, k)$ is the position of the k^{th} word of E_i in sentence e. $A(i, j)$ is the position of the j^{th} target word aligned with the source sentence word e_i. $\mathcal{G}(w)$ is the equivalent class that target word w is in.

$$c_{p_n}(j \mid i; \mathbf{e}, \mathbf{g}) = \sum_{\mathbf{a}\in\mathcal{S}(\mathbf{e},\mathbf{g})} P(\mathbf{a} \mid \mathbf{e}, \mathbf{g}) \delta(|E|, i) \delta(|G|, j)$$

$$c_{p_1}(0; \mathbf{e}, \mathbf{g}) = \sum_{\mathbf{a}\in\mathcal{S}(\mathbf{e},\mathbf{g})} P(\mathbf{a} \mid \mathbf{e}, \mathbf{g})(m - 2\phi_0) \tag{6.8}$$

$$c_{p_1}(1; \mathbf{e}, \mathbf{g}) = \sum_{\mathbf{a}\in\mathcal{S}(\mathbf{e},\mathbf{g})} P(\mathbf{a} \mid \mathbf{e}, \mathbf{g})\phi_0 \tag{6.9}$$

$$c_a(j \mid i; \mathbf{e}, \mathbf{g}) = \sum_{\mathbf{a}\in\mathcal{S}(\mathbf{e},\mathbf{g})} P(\mathbf{a} \mid \mathbf{e}, \mathbf{g}) \delta(r_i, j) \tag{6.10}$$

$$
\begin{aligned}
c_\alpha(\Delta \mid l; \mathbf{e}, \mathbf{g}) &= \sum_{\mathbf{a} \in \mathcal{S}(\mathbf{e},\mathbf{g})} P(\mathbf{a} \mid \mathbf{e}, \mathbf{g}) \sum_{i=2}^{|G|} \delta(b_i - b_{i-1}, \Delta)\delta(|E_{r_{i-1}}|, l) \\
c_n(\phi \mid w_e; \mathbf{e}, \mathbf{g}) &= \sum_{\mathbf{a} \in \mathcal{S}(\mathbf{e},\mathbf{g})} P(\mathbf{a} \mid \mathbf{e}, \mathbf{g}) \sum_{i=0}^{|\mathbf{e}|} \delta(w_e, e_i)\delta(\phi, \sum_{j=1}^{|\mathbf{g}|} \delta(i, a_j)) \\
c_t(w_g \mid w_e; \mathbf{e}, \mathbf{g}) &= \sum_{\mathbf{a} \in \mathcal{S}(\mathbf{e},\mathbf{g})} P(\mathbf{a} \mid \mathbf{e}, \mathbf{g}) \sum_{i=1}^{|\mathbf{g}|} \delta(w_g, g_i)\delta(w_e, e_{a_i}) \\
c_{d1}(\Delta \mid \mathcal{E}, k; \mathbf{e}, \mathbf{g}) &= \sum_{\mathbf{a} \in \mathcal{S}(\mathbf{e},\mathbf{g})} P(\mathbf{a} \mid \mathbf{e}, \mathbf{g}) \sum_{i=1}^{|G|} \delta(E_{r_i}, \mathcal{E})\delta(\underset{1 \le x \le |g|}{\arg\min}(a_x = I_e(r_i, k)) - b_i), \Delta) \\
c_{d2}(\Delta \mid \mathcal{G}; \mathbf{e}, \mathbf{g}) &= \sum_{\mathbf{a} \in \mathcal{S}(\mathbf{e},\mathbf{g})} P(\mathbf{a} \mid \mathbf{e}, \mathbf{g}) \qquad (6.11) \\
&\quad \times \sum_{i=0}^{|\mathbf{e}|} \sum_{j=2}^{\phi_i} \delta(\Delta, A(i,j) - A(i,j-1))\delta(\mathcal{G}(g_{A(i,j)}), \mathcal{G}) \qquad (6.12)
\end{aligned}
$$

Here the probability of an alignment is

$$
P(\mathbf{a} \mid \mathbf{e}, \mathbf{g}) = \frac{P(\mathbf{a}, \mathbf{g} \mid \mathbf{e})}{P(\mathbf{g} \mid \mathbf{e})} \approx \frac{P(\mathbf{a}, \mathbf{g} \mid \mathbf{e})}{\sum_{A \in \mathcal{S}(\mathbf{e},\mathbf{g})} P(\mathbf{a}, \mathbf{g} \mid \mathbf{e})}. \qquad (6.13)
$$

where $P(\mathbf{a}, \mathbf{g} \mid \mathbf{e})$ can be calculated with (6.6).

With the counts, we can reestimate the parameters with the following formulae:

$$
\begin{aligned}
p_n(j \mid i) &= \beta_i^{-1} \sum_{s=1}^{S} c_{p_n}(j \mid i; \mathbf{e}^{(s)}, \mathbf{g}^{(s)}) \qquad (6.14) \\
p_1 &= \frac{\sum_{s=1}^{S} c_{p_1}(1; \mathbf{e}^{(s)}, \mathbf{g}^{(s)})}{\sum_{s=1}^{S} [c_{p_1}(1; \mathbf{e}^{(s)}, \mathbf{g}^{(s)}) + c_{p_1}(0; \mathbf{e}^{(s)}, \mathbf{g}^{(s)})]} \qquad (6.15) \\
a(j \mid i \ ) &= \beta_i^{-1} \sum_{s=1}^{S} c_a(j \mid i; \mathbf{e}^{(s)}, \mathbf{g}^{(s)}) \qquad (6.16) \\
\alpha(j \mid i) &= \gamma_l^{-1} \sum_{s=1}^{S} c_\alpha(\Delta \mid l; \mathbf{e}^{(s)}, \mathbf{g}^{(s)}) \qquad (6.17) \\
n(\phi_e \mid w_e) &= \zeta_{w_e}^{-1} \sum_{s=1}^{S} c_n(\phi_e \mid w_e; \mathbf{e}^{(s)}, \mathbf{g}^{(s)}) \qquad (6.18) \\
t(w_g \mid w_e) &= \eta_{w_e}^{-1} \sum_{s=1}^{S} c_t(w_g \mid w_e; \mathbf{e}^{(s)}, \mathbf{g}^{(s)}) \qquad (6.19)
\end{aligned}
$$

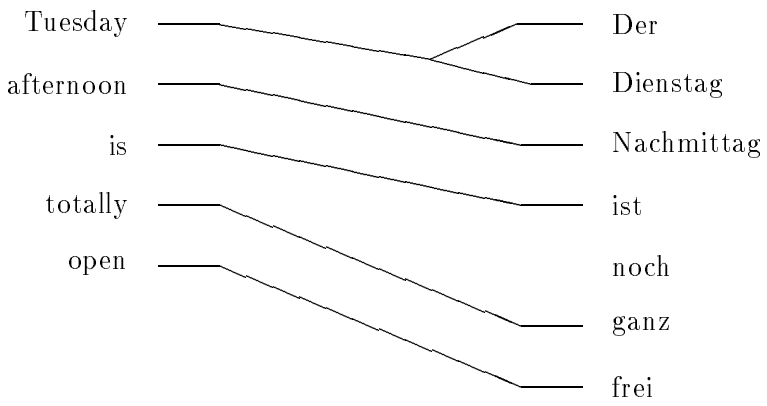

Figure 6.1: The Viterbi Alignments with both model 1 and model 2 for the parallel sentences

$$d1(\Delta \mid \mathcal{E}, k) = \theta_{\mathcal{E}k}^{-1} \sum_{s=1}^{S} c_{d1}(\Delta \mid \mathcal{E}, k; \mathbf{e}^{(s)}, \mathbf{g}^{(s)}) \tag{6.20}$$

$$d2(\Delta \mid \mathcal{G}) = \lambda_{\mathcal{G}\phi}^{-1} \sum_{s=1}^{S} c_{d2}(\Delta \mid \mathcal{G}; \mathbf{e}^{(s)}, \mathbf{g}^{(s)}) \tag{6.21}$$

Here S is the number of sentences in the parallel corpus, and $\beta_i^{-1}, \gamma_l^{-1}, \zeta_{w_e}^{-1}, \eta_{w_e}^{-1}, \theta_{\mathcal{E}k}^{-1}$,and $\lambda_{\mathcal{G}\phi}^{-1}$ are the normalization factors.

6.3 An Example

In this section I provide an example to demonstrate how the learning algorithm works. Suppose that we have the following sentence pair in the parallel corpus:

Example 6.3.1 Parallel Sentences

$\mathbf{e}$ = `Tuesday afternoon is totally open`

$\mathbf{g}$ = `Der Dienstag Nachmittag ist noch ganz frei`

To collect the expected count, we first have to parse the source sentence to obtain the phrase sequence of the sentence. We get the following for English:

$E_0E_1E_2E_3 = (\phi)$`(Tuesday afternoon) (is) (totally open).`

We then find the Model 1 and Model 2 Viterbi alignments for the sentence pair, and they are both the same as shown in Figure 6.1.

With this alignment $\mathbf{a}$ available, we can find the phrase sequence of the target sentence:

$G_0G_1G_2G_3$ = (Der Dienstag Nachmittag) (ist) (noch) (ganz frei)

With both the source and target structures at hand, we can now calculate the probability of the Model 1 and Model 2 Viterbi alignment according to the structure-based model:

$$
\begin{aligned}
P(\mathbf{a},\mathbf{g} \mid \mathbf{e}) &= P(\mathbf{a},\mathbf{g} \mid \mathbf{e},\mathbf{E}) \\
&= p_n(4 \mid 4) \times \binom{6}{1} (1-p_1)^5 p_1 \times t(\texttt{noch} \mid \phi) \\
&\quad \times a(1 \mid 0) \times \alpha(1 \mid 1) \\
&\qquad \times n(2 \mid \texttt{Tuesday}) \\
&\qquad\quad \times t(\texttt{Der} \mid \texttt{Tuesday}) \times d_1(0 \mid E(\texttt{Tuesday afternoon}), 0) \\
&\qquad\quad \times t(\texttt{Dienstag} \mid \texttt{Tuesday}) \times d_2(1 \mid G(\texttt{Dienstag}), 2) \\
&\qquad \times n(1 \mid \texttt{afternoon}) \\
&\qquad\quad \times t(\texttt{Nachmittag} \mid \texttt{afternoon}) \times d_1(2 \mid E(\texttt{Tuesday afternoon}), 1) \\
&\quad \times a(2 \mid 1) \times \alpha(3 \mid 2) \\
&\qquad \times n(1 \mid \texttt{is}) \\
&\qquad\quad \times t(\texttt{ist} \mid \texttt{is}) \times d_1(0 \mid E(\texttt{is}), 1) \\
&\quad \times a(3 \mid 3) \times \alpha(1 \mid 0) \\
&\qquad \times n(1 \mid \texttt{totally}) \\
&\qquad\quad \times t(\texttt{ganz} \mid \texttt{totally}) \times d_1(0 \mid E(\texttt{totally open}), 0) \\
&\qquad \times n(1 \mid \texttt{open}) \\
&\qquad\quad \times t(\texttt{frei} \mid \texttt{open}) \times d_1(1 \mid E(\texttt{totally open}), 1)
\end{aligned}
$$

We then use a greedy algorithm (Algorithm 6.3.1) to search for the alignment that has the greatest likelihood according to the structure-based model among those that are reachable from $\mathbf{a}$ with the *neighbor* binary relation:

Algorithm 6.3.1 search for the most likely alignment reachable from $\mathbf{a}$.

1. find $\mathcal{N}(\mathbf{a})$,the set of the neighboring alignments of $\mathbf{a}$;

2. for each $\mathbf{a}' \in \mathcal{N}(\mathbf{a})$, calculate $P(\mathbf{a}', \mathbf{g} \mid \mathbf{e})$ with (6.6);

3. $\mathbf{a}_{\max} \leftarrow \arg\max_{\mathbf{a}' \in \mathcal{N}(\mathbf{a})} P(\mathbf{a}', \mathbf{g} \mid \mathbf{e})$[1]*;*

4. if $(\mathbf{a}_{\max} = \mathbf{a})$, return $\mathbf{a}$ as the alignment with the greatest likelihood;

5. otherwise $\{\mathbf{a} \leftarrow \mathbf{a}_{\max}$; goto *1*;$\}$

With the above example, the greedy algorithm finds that $\mathbf{a}$ itself happens to be the alignment that has the greatest likelihood according to the structure-based model, among those that are reachable from $\mathbf{a}$. Therefore, given the sentence pair $(\mathbf{e}, \mathbf{g})$**,** $\mathbf{a}$ and its neighbors in $\mathcal{N}(\mathbf{a})$ are the alignments used for collecting the counts in the EM learning in (6.11).

[1] We are actually looking for the alignment with the greatest likelihood $P(a' \mid e, g) = P(a', g \mid e)/P(g \mid e)$. Since the factor $P(g \mid e)$ is the same for all alignments, we can look for $\arg\max_{a' \in \mathcal{N}(a)} P(a', g \mid e)$ instead.

Chapter 7

Decoding Algorithms

Decoding algorithms are crucial in statistical machine translation. Their performance directly affects translation quality and efficiency. Without a reliable and efficient decoding algorithm, a statistical machine translation system may miss the best translation of an input sentence even if it is perfectly predicted by the model. There are two important issues in decoding:

1. Optimality: Can the decoding algorithm find the optimal translation as predicated by the model?

2. Speed: Can the decoding algorithm identify the optimal translation in a time efficient way?

We often have to have some trade-offs between the optimality and decoding speed. If we exhaust the hypothesis space, we can certainly find the optimal one as predicated by the model. However, since the hypothesis space is exponential in the maximum sentence length allowed, it is impractical to enumerate all possible hypotheses. Often we have to prune hypotheses to speed up the decoder, and this may sacrifice optimality.

In this chapter, I describe two decoding algorithms for the structure-based model. One is based on the IBM stack decoder for IBM Model 3, and the other is based on a fast decoder with best-first nature. The first decoder is more accurate, but it is slow and it fails on many long sentences. The second decoder is fast and robust, and its accuracy is lower than the first one on those successfully decoded sentences.

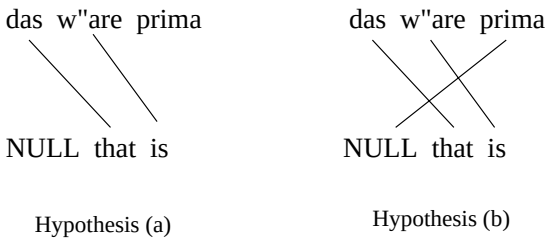

Figure 7.1: Hypotheses in the IBM Stack Decoder

7.1 IBM Stack Decoder

Stack decoding algorithms are widely used in speech recognition systems. The IBM statistical machine translation group used stack decoder for their translation system (Berger et al., 1996). In this section, I briefly review the algorithm, since the experiments for performance comparison between IBM Model 3 and the structure-based model were conducted with the decoder, and the algorithm was originally published in a US patent rather than widely available journals or conference proceedings.

7.1.1 IBM Stack Decoder

In the IBM Stack Decoder (Berger et al., 1996), a hypothesis is comprised of a source sentence prefix string and an alignment between the prefix string and the target (input) sentence. Each hypothesis is associated with a score. For example, Figure 7.1 shows two hypotheses with the same source prefix string but different alignments between the source prefix and the target sentence.

For a given target sentence to be translated, each subset of the words in the target sentence is associated with a priority queue. A hypothesis is put into a priority queue according to the target words that have been accounted for by the hypothesis. For example, hypothesis (a) in Figure 7.1 is put into the queue corresponding to the subset {das, w"are}, while hypothesis (b) is put into the queue that corresponds to the subset {das, w"are, prima}. A hypothesis is open if the last word in the hypothesis is allowed to be aligned to additional target words. A hypothesis is closed if the last word is not allowed to be aligned to any more target words.

The search proceeds iteratively. In the initial step, an empty source word sequence is entered into the queue corresponding to the empty subset of the target words. That means, the initial

hypothesis is an empty string that does not account for any word in the target sentence.

In the following steps, the decoder set up a threshold for each of the priority queues. Then it looks in each of the priority queues for the hypotheses that scored higher than the threshold of that queue. A selected hypothesis is then extended by the decoder to account for a target word t that is not aligned to any source word in the hypothesis — the extension is repeated for each of the target words that are yet to be accounted for. The extension is carried out in the following ways:

1. If the hypothesis to be extended is open, then

   (a) Aligned the last word of the hypothesis to t, and keep the new extended hypothesis open.

   (b) Aligned the last word of the hypothesis to t, and close the new extended hypothesis.

2. If the hypothesis to be extended is closed, then

   (a) Append a source word s to the hypothesis, align s to t, make the new hypothesis open.

   (b) Append a source word s to the hypothesis, align s to t, close the new hypothesis.

   (c) Aligned the null word to the target word to be accounted for.

   (d) Append two source words $s_1 s_2$ to the hypothesis, align s_2 to t, make the new hypothesis open.

   (e) Append two source words $s_1 s_2$ to the hypothesis, align s_2 to t, close the new hypothesis.

To speed up the decoding process, we do not have to enumerate over all possible s and $s_1 s_2$ sequences. Rather we can only pick s and $s_1 s_2$ sequences that can significantly increase the likelihood of t, the target word that is to be accounted for, and fit in the current hypothesis well with a high ngram likelihood. Details about this can be found in (Berger et al., 1996).

The score of a newly extended hypothesis can be incrementally calculated from the score of its parent hypothesis. Details about score calculation, as well as the threshold calculation for the priority queues, can be found in (Berger et al., 1996) too.

It is clear that the number of stacks in the IBM decoder, i.e., the number of the subsets of the target words in a target sentence, is exponential in the target sentence length. In practice, not all the queues have to be created. Instead, the queues can be created on the fly, i.e., a queue is created only when a hypothesis accounting for the target word subset corresponding to that queue is created. In the IBM system, since the English and French data are preprocessed in such a way that they have similar word order, the model may strongly prefer the hypotheses that align similar positions in the source/target sentences together. Therefore a hypothesis that has long distance alignment may never get a chance to have a score high enough to be extended, hence many queues may not be created by the decoder. However, since our English/German parallel sentences have radically different word orders, the distortion parameters in Model 3 do not strongly prefer the hypotheses that align similar source/target positions together. This means that a majority of the exponential number of priority queues may be created. This has two adversary effects:

1. A long target sentence will result in huge number of priority queues, hence too much memory space. When an input target sentence is longer than 15 words, the decoder can allocated more than 1 GB memory.

2. Exponential number of priority queues implies that exponential number of hypotheses being generated by the decoder. Therefore the decoder is extremely slow when the target sentence is long.

In my experiments, I used the IBM Model 3 decoder to search for the translations of our test input sentences up to 20 words long. Whenever the decoder allocated more than 750 MB memory, it stopped searching and registered a failure. Table 7.1 shows the number of sentences that IBM decoder had failed.

7.1.2 IBM Decoder for Structure-Based Model

The aforementioned IBM decoder for Model 3 can be slightly modified so it can be used for Model 1, Model 2, and Model 4. The modifications are mostly with the computation of hypothesis scores and queue thresholds. Unfortunately, it is not clear how the algorithm can be

Sentence Length	Total Sentence #	Failed Sentence #	Failure Percentage	Average Extended Hypothesis #
1-4	81	0	0%	16
5-8	128	0	0%	2,705
9-12	101	0	0%	11,660
13-16	41	19	46.3%	63,472
17-20	16	9	56.3%	145,768
All (1-20)	367	28	7.6%	—

Table 7.1: Decoding Performance for the IBM Model 3 Decoder: Input target sentences are grouped together according to their lengths. The first column lists these groups. The second column lists the total number of sentences in each group. The third column lists the number of sentences failed by the decoder in each group. The fourth column lists the failure percentage. The fifth column lists the average number of hypotheses being extended by the decoder — The statistics were collected from those successfully decoded input sentences in each group.

applied to the structure-based model. Unlike the word-based models, in which we can effectively compute the score of a hypothesis from the score of its parent hypothesis, the score of a hypothesis in the structure-based model depends on the words that are yet to be appended to the current hypothesis. This is mostly due to the following two reasons: (a) the hypothesis may contain a prefix of a phrase whose identification is unknown before the phrase is fully extended; and (b) the likelihood of the alignment between the hypothesis and the target sentence, as well as the identifications of the phrases in the target sentence, depend on the phrases in the source sentence.

One possible solution to this problem is to make a hypothesis comprise of not only a prefix string and an alignment between the prefix string and the target sentence, but also the number of source phrases as well as the identification of those phrases. However, this will complicate the decoding algorithm further and make it even more inefficient.

To overcome this difficulty, the hypothesis reshuffling algorithm was introduced for the structure-based model.

7.2 Hypothesis Reshuffling for Structure-Based Model

The idea of hypothesis reshuffling was based on the observation that the IBM Stack Decoder often found, in the top N translation candidates, the correct translations, or almost correct translations — translations that have the correct bags of words arranged in wrong orders.

	Sentences	M3 Score	SM Score
Input	Ich habe ein Meeting von halb elf bis um zw"olf	—	—
Preprocessed	ich hab ein Meeting von halb zehn bis um zw"olf	—	—
Reference	I have a meeting from ten thirty to twelve	2.54316e-17	8.03062e-20*
Hypotheses	I have a meeting from ten to twelve thirty	3.21979e-17*	3.62686e-24
	I have a meeting from ten thirty to twelve	2.54316e-17	8.03062e-20*
	I have a meeting from ten thirty until twelve	1.11035e-17	3.29499e-20
	I have a meeting from ten thirty till twelve	9.65641e-18	2.94287e-20
	I have got a meeting from ten to twelve thirty	3.70554e-18	5.36138e-25
	I have got a meeting from ten thirty to twelve	2.42777e-18	1.31520e-20
	I have a meeting from twelve thirty to ten	2.61774e-18	3.13659e-24
	I have a meeting from ten thirty till noon	2.29252e-18	2.63906e-20
	I have got a meeting from ten thirty until twelve	1.05997e-18	5.39632e-21
	I have a meeting from ten thirty until noon	1.75381e-18	1.96589e-20

Table 7.2: Top 10 hypotheses from Model 3 Stack Decoder. The hypotheses are followed by the Model 3 scores and the structure-based model scores. The scores with an attached * are the highest scores among the hypotheses with respect to the corresponding models.

Table 7.2 lists the top 10 translations for an input German sentence, discovered by the IBM Stack decoder for Model 3.

The hypothesis reshuffling algorithm uses the stack decoder to find top N hypotheses with a simple model, say Model 3, and then in the neighborhood of those candidate hypotheses, search for the translation that has the highest score according to a more advanced model with a hill-climbing algorithm. We define the following terminology to describe the algorithm:

Definition 7.2.1 *Word move*

Two hypotheses $H = e_1e_2e_3...e_n$ and $H' = e'_1e'_2e'_3...e'_n$ differ by a word move if there exist $1 \leq i \leq j \leq n$ such that either of the following holds:

$(e_1...e_{i-1} = e'_1...e'_{i-1}) \wedge (e_i = e'_j) \wedge (e_{i+1}...e_j = e'_i...e'_{j-1}) \wedge (e_{j+1}...e_n = e'_{j+1}...e'_n)$ or

$(e'_1...e'_{i-1} = e_1...e_{i-1}) \wedge (e'_i = e_j) \wedge (e'_{i+1}...e'_j = e_i...e_{j-1}) \wedge (e'_{j+1}...e'_n = e_{j+1}...e_n)$

Definition 7.2.2 *Phrase move*

Two hypotheses $H = e_1e_2e_3...e_n = E_1E_2...E_m$ and $H' = e'_1e'_2e'_3...e'_n = ...E'_m$ [1] differ by a phrase move if there exist $1 \leq i \leq j \leq m$ such that either of the following holds:

$(E_1...E_{i-1} = E'_1...E'_{i-1}) \wedge (E_i = E'_j) \wedge (E_{i+1}...E_j = E'_i...E'_{j-1}) \wedge (E_{j+1}...E_n = E'_{j+1}...E'_m)$ or

[1] E_i and E'_i are phrases in H and H'.

$$(E'_1...E'_{i-1} = E_1...E_{i-1}) \wedge (E'_i = E_j) \wedge (E'_{i+1}...E'_j = E_i...E_{j-1}) \wedge (E'_{j+1}...E'_n = E_{j+1}...E_m)$$

Definition 7.2.3 *Word swap*

Two hypotheses $H = e_1e_2e_3...e_n$ and $H' = e'_1e'_2e'_3...e'_n$ differ by a word swap if $e_k = e'_k$ *holds for all* $1 \leq k \leq n$ *except for* $1 \leq i \leq j \leq n$, for which we have $(e_i = e'_j) \wedge (e_j = e'_i)$.

Definition 7.2.4 *Phrase swap*

Two hypotheses $H = e_1e_2e_3...e_n = E_1E_2...E_m$ and $H' = e'_1e'_2e'_3...e'_n = ...E'_m$ [2] differ by a phrase swap if $E_k = E'_k$ *holds for all* $1 \leq k \leq m$ *except for* $1 \leq i \leq j \leq m$, for which we have $(E_i = E'_j) \wedge (E_j = E'_i)$.

Definition 7.2.5 *Neighbor hypothesis*

Two hypotheses H and H' are neighbors iff H and H' differ by a word move, a phrase move, a word swap or a phrase swap.

The search process can be described with the following algorithm:

Algorithm 7.2.1 Decoding with IBM Stack Decoder and Hypothesis Reshuffling (IBM+R)

Input:	target sentence $T = t_1t_2 \cdots t_n$.
Output:	source sentence $S = s_1s_2 \cdots s_m$.
Data Structures:	a priority queue **Q** for hypotheses.
Models:	a base model M_1 using IBM Stack Decoder;
	a model M_2 for re-scoring the candidate hypotheses and
	their neighbors.

1. Using the IBM Stack decoding algorithm for the base alignment model M_1, find the top N hypotheses. Score the hypotheses with the alignment model M_2 and then add these hypotheses to Q.

2. Repeat Step *3-7*, until there is no change of the top K hypotheses in Q:

3. For each H in the top K hypotheses in Q

4. $\mathcal{N}_H \leftarrow$ **neighbor_set**(H);

[2] E_i and E'_i are phrases in H and H'.

Model	Decoder	Total Sentence #	Correct Translations	Okay Translations	Incorrect Translations	Translation Accuracy
Model 3	IBM	339	191	68	80	66.4%
SModel	IBM+R	339	202	77	60	70.9%

Table 7.3: IBM Decoder with Reshuffling. The first row is the performance of IBM Model 3 with stack decoder (without reshuffling). The second row is the performance of the structure-based model, with reshuffling in the decoding algorithm.

5. for each $H' \in \mathcal{N}_H$

6. score H' with the alignment model M_2

7. $\mathrm{Q} \leftarrow H'$

8. Report the hypothesis with the highest score in $\mathbf{Q}$ as the translation of T.

The choice of the value N and K is a trade-off between speed and accuracy. A large N and K make the hill-climbing search in the hypothesis neighborhood less likely to stop at a local maximum, while the reshuffling process takes much more time. In experiments reported here, $N = 12$ and $K = 6$ were selected by trial and error. Table 7.3 compares the performance between Model 3 using the IBM stack decoder and the structure-based model using the IBM decoder with reshuffling.

Two different kinds of errors occur in statistical machine translation. A *modeling error* occurs when the model assigns a higher score to an incorrect translation than all correct translations. We cannot do anything about this with the decoder. A *decoding error* or *search error* happens when the search algorithm fails to identify a correct translation that has a higher score.

When evaluating a decoding algorithm, being able to identify how many errors are caused by the decoder is useful. Unfortunately, this is not attainable. The following explains the reason. Suppose that we are going to translate a German sentence $\mathbf{g}$, and we know from the sample that $\mathbf{e}$ is one of its possible English translations. The decoder outputs an incorrect $\mathbf{e}'$ as the translation of $\mathbf{g}$. If the score of $\mathbf{e}'$ is lower than that of $\mathbf{e}$, we know that a search error has occurred. On the other hand, if the score of $\mathbf{e}'$ is higher, we cannot decide if it is a modeling error or not, since there may still be other legitimate translations with a score higher than the score of $\mathbf{e}'$ — we just do not know what these better translations are.

Model	Total Errors	$Score_{\mathbf{e}} > Score_{\mathbf{e}'}$	$Score_{\mathbf{e}} \leq Score_{\mathbf{e}'}$
Model 3	148	17 (11.5%)	131 (88.5%)
SModel	137	18 (13.1%)	119 (86.9%)

Table 7.4: Reference Translations vs. Machine-Made Translations. $Score_{\mathbf{e}}$ is the score of reference translation, $Score_{\mathbf{e}'}$ is the score of the machine made translation. When $Score_{\mathbf{e}} > Score_{\mathbf{e}'}$, we know, for sure, that a decoding error has occurred

Although we cannot distinguish a modeling error from a search error, the comparison between the score of a decoder output and that of a reference translation can still reveal some information about the performance of the decoder. If we know that the decoder can find a sentence with a higher score than the "correct" reference translation, we will be more confident that the decoder is less prone to cause errors. Table 7.4 shows how many errors are knowingly caused by the decoder.

7.3 Fast Stack Decoder for Model 1 and Simplified Model 2

While the IBM stack decoder and the reshuffling decoding algorithm based on it have low known decoding error rate, (around 5% with preprocessed English/French data and 11.5% with our English/German data.) its high failure rate and almost intolerably slow speed greatly limit the applicability of statistical machine translation. In this section, I describe a fast stack decoding algorithm for IBM Model 1 and a simplified version of Model 2. This algorithm can later be used as the base decoding algorithm for the reshuffling algorithm for Model 3 and the structure-based model.

The high failure rate and the slow speed of the IBM stack decoder were due to the same reason — retaining the alignment between a source sentence prefix and the target sentence in a hypothesis. This is mostly due to the fact that no efficiently algorithm is available for Model 3 to calculate $\Pr(T \mid S)$, the *a posteriori* probability of the target sentence given a source sentence over all possible alignments. Instead, we have to make an assumption about the likely alignment between a hypothesis and the target and compute the *a posteriori* probability of the target sentence with that alignment, $\Pr(T, A \mid S)$, given the source hypothesis. Because the number of possible alignments is exponential in sentence length, this results in exponential number of priority queues and hypotheses. Therefore the algorithm is too expensive with

respect to both time and space complexities.

On the other side, efficient algorithms are available for Model 1 and Model 2 to calculate $\Pr(T \mid S) = \sum_A \Pr(T, A \mid S)$, the *a posterior* probability of a target sentence T given a source S, over all possible alignments A between S and T. Therefore we do not have to make assumptions about the alignment between a hypothetical source sentence prefix string and the target sentence. Instead, a hypothesis can be just a prefix string of the source sentence, whose score is the likelihood of the target sentence summed over all possible alignments. In doing so, we greatly reduced the size of hypothesis space and make the decoding more efficient.

To be specific, here I introduce a fast stack decoder for a simplified version of IBM alignment Model 2. The decoder for Model 1 can be simply derived from it, since Model 1 is a special case of Simplified Model2 when the alignment distribution is uniform.

7.3.1 Simplified Model 2

In IBM alignment Model 2 (see detailed description in Appendix A), the alignment parameters depend on the source and target sentence length l and m. This causes the following difficulties:

1. There are too many parameters and therefore too few training data per parameter. This may not be a problem when massive training data are available. However, in our application, this is a severe problem. Figure 7.2 plots the length distribution for the English and German sentences. When sentences are longer, very few training data are available.

2. The search algorithm has to make multiple hypotheses of different source sentence length. For each source sentence length, it searches through almost the same prefix string and finally settles on a sentence length. This is a very time consuming process and makes the decoder very inefficient.

To alleviate the problems, I introduced a simplified Model 2 in (Wang and Waibel, 1997a), in which the alignment parameters are independent of the sentence length l and m:

$$\Pr(i \mid j, m, \mathbf{e}) = \Pr(i \mid j, l, m) = a(i \mid j) \tag{7.1}$$

Here $i, j < L_m$, and L_m is the maximum sentence length allowed in the translation system. A slight change to the EM algorithm was made to estimate the parameters.

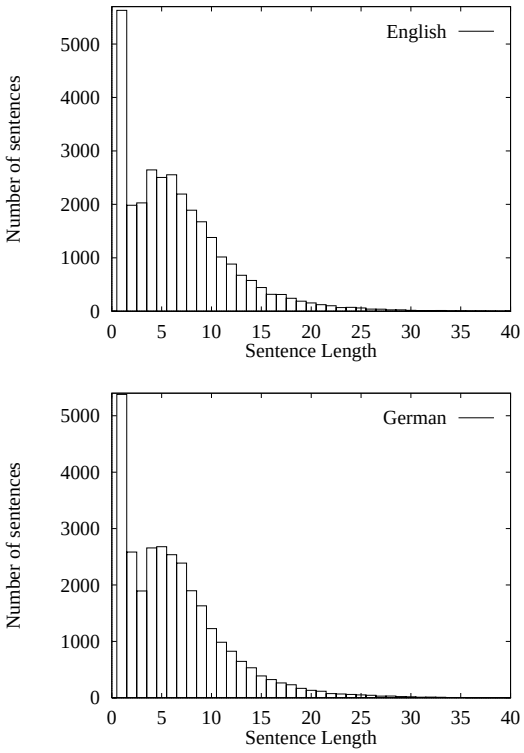

Figure 7.2: Sentence Length Distribution

There is a problem with this model: given a sentence pair **g** and **e**, when the length of **e** is smaller than L_m, then the alignment parameters do not sum to 1:

$$\sum_{i=0}^{|\mathbf{e}|} a(i \mid j) < 1. \tag{7.2}$$

We handle this problem by using normalized a_l instead of a for a hypothesis of source sentence with length l:

$$a_l(i \mid j) = \frac{a(i \mid j)}{\sum_{k=0}^{l} a(k \mid j)} \tag{7.3}$$

A hypothesis in the simplified model can be expressed as $H = e_1, e_2, \cdots, e_k$, and $|H|$ is used to denote the length of the sentence prefix of the hypothesis H, in this case, k.

7.3.2 Fast Stack Decoder: Scoring a Hypothesis

To calculate the probability of a target given a source in Simplified Model 2, (A.1) can be modified as in (7.4) for Simplified Model 2:

$$\begin{aligned} \Pr(\mathbf{g} \mid \mathbf{e}) &= \epsilon \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(g_j \mid e_{a_j}) \times a_l(a_j \mid j) \\ &= \epsilon \prod_{j=1}^{m} \sum_{i=0}^{l} t(g_j \mid e_i) \times a_l(i \mid j) \end{aligned} \tag{7.4}$$

Although (7.4) was obtained from the alignment model, it would be easier for us to describe the scoring method of the fast decoder if we interpret the last expression in the equation in the following way: each word e_i in the hypothesis contributes the amount $\epsilon\, t(g_j \mid e_i) \times a_l(i \mid j)$ to the probability of the target word g_j.

Given a target sentence $G = g_1 g_2 \cdots g_m$, assume that the source sentence length is l at this moment. The hypothesis $H_l = l : e_1, e_2, \cdots, e_k$ has hypothesized k words as the prefix of the source sentence of length l. Then the probability mass contributed by the source word e_i ($0 \le i \le k$, with e_0 being the null word) to the target word g_j is $\epsilon t(g_j \mid e_i) \times a_l(i \mid j)$. For the rest source positions $k < i \le l$, since the source word at that position has not been introduced

into the prefix string yet, its contribution to the target word g_j is averaged over all possible source words, which is $\epsilon a_l(i \mid j) \times \sum_{n=0}^{|L|} \Pr(w_n) \times t(g_j \mid w_n)$. Here $|L|$ is the size of the source word lexicon, w_n is the n^{th} word in the source lexicon, and $\Pr(w_n)$ is the prior probability of the source word w_n, which is obtained with maximum likelihood estimator. Therefore, if we use $\tau_{kl}(j \mid i, H_l)$ to denote the contribution of the i^{th} source position of H to the probability mass of the j^{th} target word, then we have

$$\tau_{kl}(j \mid i; H_l) = \begin{cases} \epsilon a_l(i \mid j) \times t(g_j \mid e_i) & 0 \leq i \leq k \\ \epsilon a_l(i \mid j) \times \sum_{n=0}^{|L|} \Pr(w_n) \times t(g_j \mid w_n) & k < i \leq l \end{cases} \tag{7.5}$$

The translation model score of the hypothesis H_l is therefore

$$\tau(H_l) = \prod_{i=1}^{m} \sum_{j=0}^{l} \tau_{kl}(j \mid i; H_l) \tag{7.6}$$

In practice, since we do not make any assumption of the source sentence length, the score of the hypothesis $H = e_1, e_2, \cdots, e_k$ has to be averaged over all possible sentence lengths:

$$\tau(H) = \sum_{i=k}^{L_m} \Pr(k \mid m) \times \tau(H_i) \tag{7.7}$$

here $\Pr(k \mid m)$ are source sentence length distributions conditioned on target sentence length, which are modeled with Poisson distributions. L_m is the maximum sentence length allowed in the system.

Because our objective is to maximize $P(\mathbf{e}, \mathbf{g})$, we have to include in the score the ngram language model likelihood of the hypothesis, therefore the score of H is

$$Score_H = \tau(H) \times \prod_{i=1}^{k} P(e_i \mid e_{i-N+1} \cdots e_{i-1}). \tag{7.8}$$

Because of the different number of factors in the language score, hypotheses of different prefix lengths are not directly comparable. Therefore hypotheses of different prefix lengths are stored in different priority queues. This results in the following algorithm:

Algorithm 7.3.1 Fast Stack Decoding for Simplified IBM Model 2 (FSD)

Input:	target sentence $T = t_1 t_2 \cdots t_n$.
Output:	source sentence $S = s_1 s_2 \cdots s_m$.
Data Structures:	a collection of priority queues $\mathbf{Q}_0 \mathbf{Q}_1 \cdots \mathbf{Q}_{L_m}$ for hypotheses. L_m is the maximum sentence length allowed

1. Initialize with a null hypothesis (with prefix string length 0) H_0, compute $S(H_0)$ with (7.5), (7.6), (7.7) and (7.8).
2. $Q_0 \leftarrow H_0$
3. For each $\mathbf{Q} \in \{\mathbf{Q}_0\mathbf{Q}_1 \cdots \mathbf{Q}_{L_m}\}$
4. Set the threshold for $\mathbf{Q}$
5. For each $H \in \mathbf{Q}$
6. If $Score_H > Threshold(\mathbf{Q})$
7. For each promising source word s
8. $H' = \text{append}(H, s)$
9. Score H' with (7.5), (7.6), (7.7) and (7.8).
10. $\mathbf{Q}_{|H'|} \leftarrow H'$
11. Exit the loop if N complete source sentences are available in $\mathbf{Q}$'s.
12. Report the hypothesis with the highest score in $\mathbf{Q}$'s as the translation of T.

In the algorithm, a hypothesis is complete if its prefix string ends with the "end of sentence" symbol </s>.

7.3.3 A* Search: Scoring a Hypothesis

In the A* algorithm(Nilsson, 1971), the score of a hypothesis H consists of two parts: the prefix score for the part that is already in the hypothesis, and the heuristic score that estimates the potential score improvement introduced by new words that are yet to be appended to H to complete the source sentence. A good heuristic score is essential for the optimality of stack decoding. To guarantee an optimal search result, the heuristic function must be an upper-bound of the potential score improvement introduced by every possible extension to a hypothesis (Nilsson, 1971). In other words, the benefit of extending a hypothesis should never

be under-estimated. Otherwise the search algorithm may prematurely conclude with a non-optimal hypothesis. However, if the heuristic function over-estimates the merit of extending a hypothesis too much, the search algorithm will waste a huge amount of time after it hits the optimal result to safeguard the optimality. On the other side, if we do not use any heuristic function, then what we get is the Best-first algorithm. In that case, the algorithm is much faster. However, it is very likely for the algorithm to miss the optimal solution.

The aforementioned algorithm actually has the best-first nature. Although we use the average probability mass for the heuristics as the contribution of the future source words to the target words, this average underestimate the merit of extending a hypothesis. For the sake of simplicity, I will call the algorithm the BF algorithm. Please bear in mind that this is not the pure best-first algorithm, since we still have heuristics for the future extension to a hypothesis. The algorithm is some way in between the best-first and the A* algorithm.

To make the decoder an A* algorithm, we can make the following modification:

$$\tau^*_{kl}(j \mid i; H_l) = \begin{cases} \epsilon a_l(i \mid j) \times t(g_j \mid e_i) & 0 \leq i \leq k \\ \epsilon a_l(i \mid j) \times \max_{0<n\leq|L|} t(g_j \mid w_n) & k < i \leq l \end{cases} \tag{7.9}$$

Here we assume that each future (yet to be appended) source word is going to make the maximum contribution to all the target word simultaneously.

The translation model score of a hypothesis with a hypothetical source sentence length l, H_l, is therefore

$$\tau^*(H_l) = \prod_{i=1}^{m} \sum_{j=0}^{l} \tau^*_{kl}(j \mid i; H_l) \tag{7.10}$$

And again the score for the hypothesis $H = e_1, e_2, \cdots, e_k$ has to be averaged over all possible sentence lengths:

$$\tau^*(H) = \sum_{i=k}^{L_m} \Pr(k \mid m) \times \tau^*(H_i) \tag{7.11}$$

Including language model contribution, the score of a hypothesis H is

$$Score^*_H = \tau^*(H) \times \prod_{i=0}^{k} P(e_i \mid e_{i-N+1} \cdots e_{i-1}). \tag{7.12}$$

Algorithm 7.3.1 can still be used for this A* version of decoding algorithm, with score calculation being replaced with (7.9), (7.10),(7.11) and (7.12).

Decoder	**Model**	**Total Sentence #**	**Failed Sentence #**	**Failure Percentage**
A*	IBM Model 1	367	3	0.8%
BF	IBM Model 1	367	2	0.5%
A*	Simplified Model 2	367	7	1.9%
BF	Simplified Model 2	367	5	1.4%

Table 7.5: Decoding Performance with A* Decoder and Best-First Decoder.

Decoder	Model	Total	Correct	Okay	Incorrect	Accuracy
A*	IBM Model 1	364	161	74	129	55.1%
BF	IBM Model 1	365	154	79	132	53.0%
A*	Simplified Model 2	360	197	46	117	61.1%
BF	Simplified Model 2	362	148	68	146	50.3%

Table 7.6: Translation Accuracy with A* Decoder and Best-First Decoder.

7.3.4 Performance Comparison: A* vs. Best-First Stack Decoder

Table 7.5 shows the number of sentences failed by the A* and the Best-first stack decoder for Model 1 and Model 2, which happens after the decoder has extended too many (5,000) hypotheses.

Table 7.6 compares the performance for the best-first stack decoder and the A* stack decoder for IBM Model 1 and Simplified Model 2. The accuracy was calculated by giving a correct translation 1 point and an okay translation 1/2 point[3].

Table 7.7 shows the comparison between the scores of the outputs from the decoder and the scores of the reference translations when the outputs are not correct translations, i.e., when the outputs are either okay or incorrect translations.

[3]The scoring method will be introduced in chapter 8.

Decoder	Model	Total Errors	$Score_{\mathbf{e}} > Score_{\mathbf{e}'}$	$Score_{\mathbf{e}} \leq Score_{\mathbf{e}'}$
A*	IBM Model 1	203	24 (11.8%)	179 (88.2%)
BF	IBM Model 1	211	26 (12.2%)	185 (87.8%)
A*	Simplified Model 2	163	15 (9.2%)	148 (90.8%)
BF	Simplified Model 2	214	102 (47.7%)	112 (52.3%)

Table 7.7: A* and BF algorithms: Reference Translations vs. Machine-Made Translations. $Score_{\mathbf{e}}$ is the score of reference translation, $Score_{\mathbf{e}'}$ is the score of the machine made translation.

Observations

We have the following observations from Table 7.6 and Table 7.7:

1. While switching from the A* stack decoder to the best-first search caused a big performance drop with Simplified Model 2, the impact was not as serious in Model 1.

2. For IBM Model 1, most of the erroneous translations generated by the best-first search had higher scores than the reference translations; for Simplified Model 2, a majority of the erroneous translations had lower scores than the reference translations (Known decoding errors).

The reason for the above observations can be attributed to the different behaviors of the two models. Assume that for a target sentence, the source word e should appear in position i of the source (the translation), but the decoder reaches a hypothesis in which e is not at i. For IBM Model 1, since the only constraint on word order is from the language model, there is a great chance that e will appear in another position as long as it can significantly increase the likelihood of a target sentence word and it is in the right context to have a high language model score. This misplacement of a correct word at a wrong position will not affect the translation model score very much, since in Model 1 the alignment distribution is uniform and independent of the word position. Because a correct word occurring in the extension of the current hypothesis can still result in a high heuristic score, it is less likely for the A* decoder to return to another hypothesis with e at the right position i. Therefore, the A* search behaves similarly to the best-first search, which is consistent with the first observation. It also implies that both the A* decoder and the best-first search tend to find a correct bag of words, but arrange the words incorrectly. This is consistent with the second observation that the erroneous translation has high scores, since in Model 1 the translation model score depends only on the bag of words, and the placement of the words is irrelevant. On the other hand, in Simplified Model 2, when e is missing from i, because of the penalty from the alignment parameter $a(i' \mid j)$, it does not help much to introduce e to another position i' in the source sentence. With the stack search, the decoder will finally return to a hypothesis with e at the right position. The best-first search, however, due to its hill-climbing nature, is less likely to backtrack to the right hypothesis. Therefore a big performance gap exists between the A* search and the best-first search for

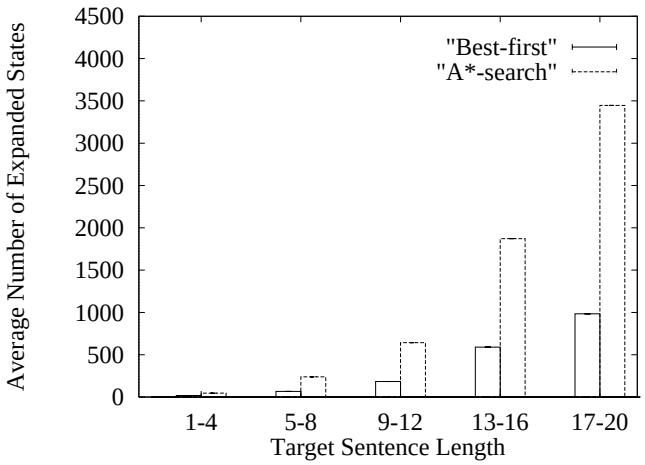

Figure 7.3: Extended States vs. Target Sentence Length

Simplified Model 2. This also explains why, in the best-first search, a majority of the erroneous translations have lower scores than the reference translations.

I examined the incorrect translations made by the best-first and A* search with IBM alignment Model 1, and found that around 85% of those translations identified correct or nearly correct bags of words.

7.3.5 Decoding Speed

Another important issue is the decoder efficiency. Figure 7.3 plots the average number of states being extended by the decoders with Simplified Model 2. It is grouped according to input sentence length. On average the best-first search takes only about 1/4~1/3 decoding time of the stack decoder.

7.3.6 Fast Stack Decoder with Reshuffling for Structure-Based Model

If we use the fast stack decoder (A* or Best-First) as the base decoder to search for the hypothesis candidates and then apply the reshuffling procedure, we can get the following algorithm:

Algorithm 7.3.2 Fast Stack Decoding with Hypothesis Reshuffling (FSD+R)

Input:	target sentence $T = t_1 t_2 \cdots t_n$.
Output:	source sentence $S = s_1 s_2 \cdots s_m$.
Data Structures:	a priority queue **Q** for hypotheses.
Models:	a base alignment model M_1 using FSD to find candidate hypotheses; an alignment model M_2 for re-scoring the candidate hypotheses and their neighbors. M_1 can be the same model as M_2.

1. Using the fast decoding algorithm (FSD) for the base alignment model, find the top N source sentence hypotheses of length from $\lfloor n/2 \rfloor$ to $\lceil 3 * n/2 \rceil$. Score the hypotheses with the alignment model M_2 and then add these hypotheses to Q.

2. Repeat Step 3-7, until there is no change of the top K hypotheses in Q :

3. for each hypothesis H in the top K hypotheses in Q

4. $\mathcal{N}_H \leftarrow$ **neighbor_set**(H);

5. for each $H' \in \mathcal{N}_H$

6. score H' with the alignment model M_2

7. **Q** $\leftarrow H'$

In experiments reported here, $N = 20$ and $K = 6$ were selected by trial and error.

When we apply the algorithm, we often let M_1 "borrow" the translation parameters from M_2, because in general the translation distribution of a source word in a more advanced model is less ambiguous and more accurate (see next chapter).

Because both BF decoder and A* decoder tended to find the correct bag of words with Model 1 (see the analysis above), and BF and A* performed similarly but BF works faster, we used BF decoder for Model 1 as the base model decoder in the FSD+R algorithm in the following experiments reported in this chapter. The decoder is therefore named the BFD+R decoder.

Model	Decoder	Total	Failed	Corr.	Okay	Incorr.	Accuracy	Accuracy*
Model 3	IBM	367	28	191	68	80	66.4%	61.3%
Model 3	BFD+R	367	2	188	74	103	61.6%	61.3%
SModel	IBM+R	367	28	202	77	60	70.9%	65.5%
SModel	BFD+R	367	2	203	76	86	66.0%	65.7%

Table 7.8: Performance Comparison between IBM (IBM+R) and BFD+R. The first row is the performance of IBM Model 3 with the stack decoder (without reshuffling). The second row is the performance of IBM Model 3 with BF decoder and hypothesis reshuffling algorithm. The third row is the performance of the structure-based model with the IBM stack decoder and hypothesis reshuffling, and the fourth row is the performance of the structure-based model with BF decoder and hypothesis reshuffling. The "Failed" column lists the number of sentences for which the search aborted. "Accuracy" is calculated with respect to the successfully decoded sentences, and "Accuracy*" is calculated with respect to the total number of input sentences (367).

7.4 Performance Comparison: IBM Stack Decoder vs. Best-First with Reshuffling

Two experiments were conducted to evaluate the performance of the IBM decoder with the BFD+R decoder. In the first experiment, we compared the performance of the BFD+R algorithm[4] with that of the IBM stack decoder for IBM Model 3. In the second experiment, we compare the performance of two different reshuffling algorithms for our structure-based model: the first used IBM Model 1 as based model and applied the Best-first decoding algorithm to find the hypothesis candidates (BFD+R) as the starting point for hill-climbing reshuffling; the second used IBM Model 3 as the base model and applied the IBM stack decoder to find the hypothesis candidates (IBM+R). Table 7.8 compares the performance: among those successfully decoded sentences, the IBM decoder and IBM+R decoder have higher accuracy (Accuracy column). However, the different algorithms performed similarly if the accuracy was calculated among all input sentences (Accuracy* column). This is because the IBM decoder failed on more sentences, and usually those sentences were difficult ones and likely to result in errors with the BFD+R algorithm.

For those erroneous (okay and incorrect) translations, Table 7.9 compares their model scores

[4]The BFD+R for Model 3 is slightly different from the BFD+R for the structure-based model — only word move and word swap are used in finding the neighbors of a hypothesis. The concept of phrase does not exist in Model 3.

Model	Decoder	Total Errors	$S(\mathbf{e}) > S(\mathbf{e}')$	$S(\mathbf{e}) \leq S(\mathbf{e}')$
Model 3	IBM	148	17 (11.5%)	131 (88.5%)
Model 3	BFD+R	177	26 (14.7%)	151 (85.3%)
SModel	IBM+R	137	18 (13.1%)	119 (86.9%)
SModel	BFD+R	162	28 (17.3%)	134 (82.7%)

Table 7.9: Reference vs. Machine-Made Translations. $S(\mathbf{e})$ is the score of reference translation, $S(\mathbf{e}')$ is the score of the machine made translation. When $S(\mathbf{e}) > S(\mathbf{e}')$, we know, for sure, that a decoding error has occurred.

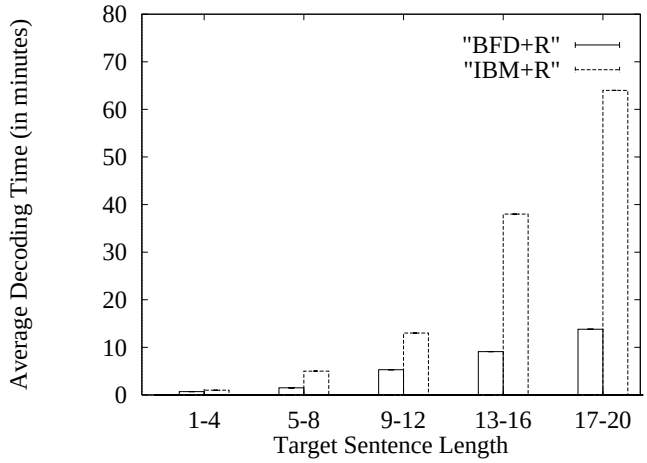

Figure 7.4: Decoding Time: BFD+R vs. IBM+R

with that of the reference translations. Here the BFD+R decoder had higher known decoding error rate than the IBM decoder (Model 3) or the IBM decoder with reshuffling (IBM+R for the structure-based model) had. However, since BFD+R decoded more sentences, and the IBM decoder failed on those extra sentences, it was likely that the BFD+R decoder made more mistakes on these difficult sentences and resulted in higher decoding error rate.

While the new algorithm did not improve the translation accuracy, its biggest advantage is its decoding speed. Using fast stack decoder for the base model, we do not have to differentiate hypotheses with the same prefix string but different alignments. Therefore we reduce the number of hypotheses dramatically. Although we need extra time in the reshuffling phase, we found the new decoder worked 4-5 times faster than the IBM decoder did for Model 3. The speed advantage was more evident for the structure-based model. Since the IBM decoder was not directly applicable to this model, it had to be coupled with the reshuffling algorithm anyway.

Therefore the time reduction resulting from switching from the IBM decoder to the fast stack decoder was fully observed. The IBM+R decoder could spend hours on long sentences, while the BFD+R decoder normally found a translation within 15 minutes (Figure 7.4).

Another advantage of the new decoding algorithm is its generality. Base model decoding plus reshuffling according to an advanced model provides a general framework for any complicated models.

Chapter 8

Performance Evaluation

In this chapter, I evaluate the performance of the structure-based statistical machine translation system. The evaluation was conducted to investigate the following:

1. Does the structure-based model improve the translation over the word-based alignment models?

2. How does the structure-based model perform compared to the semantic-based symbolic translation?

3. What is the impact of different grammar inference algorithms (monolingual vs. bilingual) on the translation model?

4. How does segmentation error influence translation performance?

8.1 Evaluation Method

Unlike in speech recognition, "accurate translations" is difficult to define in machine translation. In speech recognition an output can be compared to the transcription of the test data, and the word error rate can be used for performance measurement. In machine translation, (Tillmann et al., 1997) used the same method to evaluate statistical machine translation performance. However, word error rate is not an appropriate performance measure for machine translation, since a sentence may have several legitimate translations. It makes little sense to say that one

of them (the reference) is perfectly correct and the rest are not. Word error rate measure is also biased against semantic based translations, for the translations made by the semantic translator are usually non-literal, as shown in the following example:

Example 8.1.1 Semantic Translation

Input Sentence:	`Wir sehen uns dann`
Literal Translation:	`See you then`
Semantic Translation:	`Wonderful`

"Wonderful" is a good translation of "Wir sehen uns dann" in the scheduling domain, because both of them are used by conversation participants to express their agreement on a proposed time for an appointment. Since the parallel corpus contains literal translations, the comparison between semantic translations and the parallel data will greatly underestimate the performance of the semantic-based translator.

Since there is no suitable automatic way to evaluate the machine translation performance, I used human subjects to judge the machine-made translations. A translation is classified into one of the following three categories[1]:

1. Correct translations: Translations that are grammatical and preserve the original meaning of the inputs.

2. Okay translations: Translations that convey the same meaning but with small grammatical mistakes that do not affect correct understanding or translations that convey most but not the entire meaning of the input.

3. Incorrect translations: Translations that are ungrammatical or convey little meaningful information, or the information is different from the input.

Examples of correct, okay, and incorrect translations are shown in Table 8.1. When scoring translations, a correct translation is given one credit; an incorrect one gets 0 credit; and an okay translation gets 1/2 credit.

[1] This is roughly the same as the classification in IBM statistical translation, except we do not have "legitimate translation that conveys different meaning from the input" — we did not observe this case in our outputs.

Correct	German	ich habe ein Meeting von halb zehn bis um zwölf
	English (reference)	I have a meeting from nine thirty to twelve
	English (output)	I have a meeting from nine thirty to twelve
	German	wir sollten es vielleicht mit einem anderen Termin versuchen
	English (reference)	we might want to try for some other time
	English (output)	we should try another time
Okay	German	ich glaube nicht däs ich noch irgend etwas im Januar frei habe
	English (reference)	I do not think I have got anything open in January
	English (output)	I think I will not free in January
	German	ich glaube wir sollten ein weiteres Meeting vereinbaren
	English (reference)	I think we have to have another meeting
	English (output)	I think we should fix a meeting
Incorrect	German	schlagen Sie doch einen Termin vor
	English (reference)	why don't you suggest a time
	English (output)	why you an appointment
	German	ich habe Zeit für den Rest des Tages
	English (reference)	I am free the rest of it
	English (output)	I have time for the rest of July

Table 8.1: Examples of Correct, Okay, and Incorrect Translations: for each translation, the first line is an input German sentence; the second line is the human made (reference) translation for that input sentence; and the third line is the output from a translator.

Model	Decoder	Total	Fail	Corr.	OK	Incor.	Accuracy	Accuracy*
S. Model 2	IBM	367	21	169	91	86	61.9%	58.4%
Model 3	IBM	367	28	191	68	80	66.4%	61.3%
Model 4	IBM	367	34	176	71	86	63.5%	57.6%
Str-Based	IBM+R	367	28	202	77	60	70.9%	65.5%

Table 8.2: Translation Accuracy: a correct translation gets one credit; an okay translation gets 1/2 credit; an incorrect one gets 0 credit. Accuracy was calculated with respect to those successfully decoded sentences, and Accuracy* was calculated with respect to all input sentences.

8.2 Structure-base vs. Word-based Alignment

8.2.1 Translation Accuracy

Table 8.2 shows the end-to-end translation performance of Simplified Model 2, Model 3 and the structure-based translation model. The structure-based model achieved around 11% error reduction over Model 3, the best performing word-based model. The result reported here are for the translation with the IBM stack decoder (plus the reshuffling algorithm for the structure-based model). Here I focus on the comparison among difference models. The comparison among different decoding algorithms can be found in Chapter 7.

Why Model 4 underperformed Model 3

Model 4 is a more advanced model than Model 3. However, we found that Model 3 performed better than Model 4. One reason is that in Model 4, we often get underflow when we compute the likelihood of an alignment between parallel sentences. Figure 8.1 shows an alignment example that was found by Model 4 hill-climbing search — actually it was the same alignment found by Model 3 that served as the starting point of the Model 4 search. Since Model 4 had underflow likelihood for all its neighbors, this one was taken as the "best" alignment for Model 4. Figure 8.2 shows the Model 4 score computation, which gets an underflow score for the alignment.

In the example, several parameters had value smaller than 10^{-8}, and was replaced with 10^{-8} in the model. The first such parameter is `d1(7-2 | A(suggestion),B(könnte))`. It states that it is very unlikely to place "`könnte`", the translation of "`+d`", `5` words away from the center[2] of

[2]The center of a source word is the ceiling of the average position of the target words that are aligned to that source word.

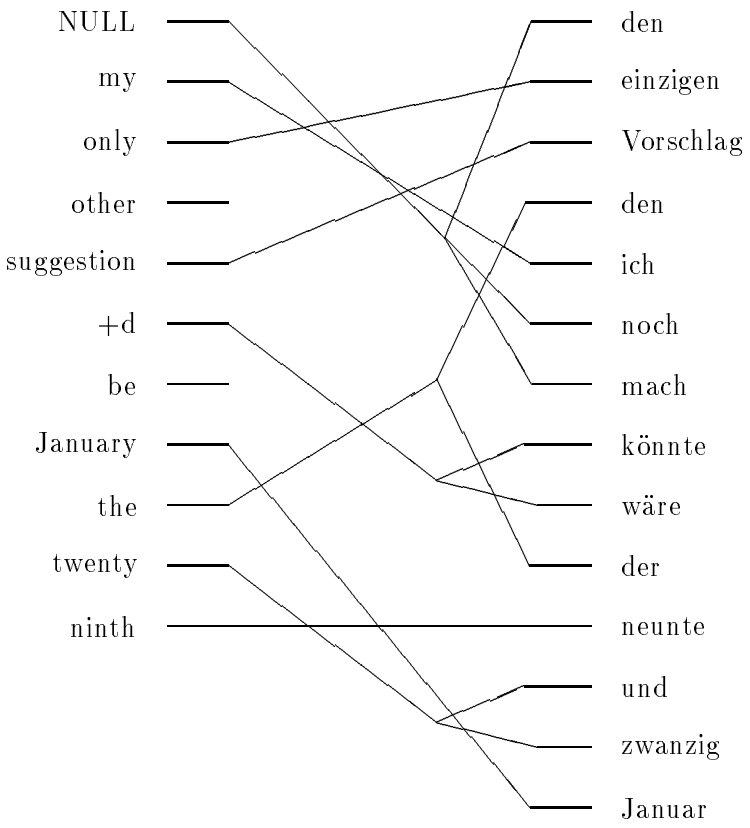

Figure 8.1: Alignment for a Parallel Sentence. It is the "best" alignment between the sentences that was found by the Model 4 hill-climbing search.

```
NULL word alignment score=0.177569
    t(den | NULL)=0.0230913
    t(noch | NULL)=0.0286865
    t(mach | NULL)=0.01478
n(1 | my)=0.903444
    t(ich | my)=0.0780847          d1(4-0 | A(#NULL#),B(ich))=0.0254732
n(1 | only)=0.907803
    t(einzigen | only)=0.195031    d1(1-4 | A(my),B(einzigen))=0.023548
n(0 | other)=0.421362
n(1 | suggestion)=0.669533
    t(Vorschlag | sugg...)=0.39558  d1(2-1 | A(only),B(Vorschlag))=0.36933
n(2 | +d)=0.00239905
    t(könnte | +d)=0.0590251       d1(7-2 | A(sugg...),B(könnte))= 1e-8
    t(w''are | +d)=0.115425        d2(8-7 | B(w''are))=0.908741
n(0 | be)=0.889456
n(1 | January)=0.939069
    t(Januar | January)=0.948094   d1(13-8 | A(+d),B(Januar))=0.361803
n(2 | the)=0.00232391
    t(den | the)=0.318586          d1(3-13 | A(January),B(den))=1e-8
    t(der | the)=0.249571          d2(9-3 | B(der))=1e-8
n(2 | twenty)=0.864041
    t(und | twenty)=0.294376       d1(11-6 | A(the),B(und))=1e-8
    t(zwanzig | twenty)=0.671504   d2(12-11 | B(zwanzig))=0.985792
n(1 | ninth)=0.954483
    t(neunte | ninth)=0.463206     d1(10-12 | A(twenty),B(neunte))=0.6881
Translation Model Score:  0
```

Figure 8.2: Model 4 Score Calculation for the Previous Alignment.

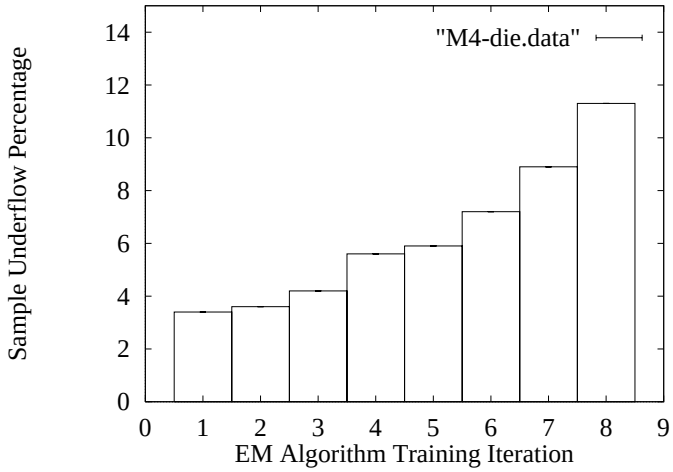

Figure 8.3: EM Training Iterations vs.Percentage of Samples with Underflow Model 4 Score

"`suggestion`", the word that precedes "`+d`". In other words, because in English "`+d`" is close to "`suggestion`", it is not likely that their translations in German will be far away. This is still the effect of the long distance alignment problem.

Other small parameters include `d1(3-13 | A(January),B(den))`, `d2(9-3 | B(der))`, and `d2(9-3 | B(der))`. They demonstrate another problem with Model 4. In English and German, many function words like "`the`", "`der`", "`den`" may not be translated into the other language. If a function word in the target sentence does not have its correspondence in the source, a simpler model like Model 2 or Model 3 may align it to another source function word. In the example here, the German function word "den" does not have a translation in the source sentence. Instead of aligning it to the null word, Model 3 aligned it to the English word "`the`". This effectively shifted the center of "`the`" from `9` to `6`. The problem is that Model 4 is too sensitive to this kind of alignment errors, since it changes the center of a source word significantly and results in much different placement likelihood. On one side, this discriminative power is good, since it can isolate the "bad" alignments. On the other side, an alignment with mistakes will be so isolated that all its neighbors have underflow likelihood. If this alignment is the one found by a simpler model to serve as the starting point of the Model 4 hill-climbing search for the "best" alignment, then the search will stop at a local maximum at very beginning, since there is no gradient around the starting point. As an effect of this, many parallel sentences have underflow scores and do not play any role in model training.

j	0	1	2	3	4	5	6	7	8	...
$a_{M2}(j \mid 1)$	0.04	0.86	0.054	0.025	0.008	0.005	0.004	0.002	3.3×10^{-4}	...
$a_{SM}(j \mid 1)$	0.003	0.29	0.25	0.15	0.07	0.11	0.05	0.04	0.02	...

Table 8.3: The alignment distribution for the first German word/phrase in Simplified Model 2 and the structure-based model. The second flatter distribution reflects the high possibility of different phrase orders in translations.

Figure 8.3 shows the percentage of the training data that had underflow alignment likelihood in each iteration of EM parameter estimation. With around 10% of data not used for model training, the trained model is certainly biased and not as reliable as Model 3.

Because Model 4 underperformed model 3, in the following evaluation part we will focus on the comparison between Model 3 and the structure-based model.

8.2.2 Word Order and Alignment Distributions

Table 8.3 shows the alignment distribution for the first German word/phrase in Simplified Model 2 and the structure-based model. The distribution is much flatter in the structure-based model, reflecting the fact that English and German have different phrase orders. On the other hand, the word based model tends to align target sentence words with source words at similar positions, which results in many incorrect alignments, and makes the word translation probability t distributed over many incorrect target words, as shown in 8.2.3.

8.2.3 Model Complexity

The structure-based model has 3,081,617 free parameters, an increase of about 2% over 3,022,373 free parameters of Simplified Model 2. This small increase does not cause over-fitting, as the performance on test data suggests. On the other hand, the structure-based model is more accurate. This can be illustrated with an example of translation probability distribution of the English words "I" and "that". Tables 8.4 and 8.5 compare the translation distribution for these two words with probability greater than 0.01, among Model 2, Model 3 and the structure-based model. It is clear that the structure-based model "focuses" better on the correct translations.

From Table 8.4 and 8.5, we can see that the word-based model is more uncertain about the correct translations of those two source words. The questions is, is there a way to quantitatively

$t_{M2}(*\mid \text{I})$		$t_{M3}(*\mid \text{I})$		$t_{SM}(*\mid \text{I})$	
ich	0.708	ich	0.944	ich	0.988
da	0.104	mir	0.023	mich	0.010
am	0.024	also	0.012		
das	0.022				
dann	0.022				
also	0.019				
es	0.011				

Table 8.4: The translation distribution of "I", learned by Model 2, Model 3 and the structure-based model. It is more uncertain in the word-based alignment model because the biased alignment distribution forces the associations between unrelated English/German words.

$t_{M2}(*\mid \text{that})$		$t_{M3}(*\mid \text{that})$		$t_{SM}(*\mid \text{that})$	
das	0.637	das	0.787	das	0.789
da	0.057	daβ	0.052	daβ	0.068
daβ	0.055	diese	0.041	da	0.035
also	0.033	da	0.031	diese	0.014
dann	0.025	also	0.031	so	0.011
ist	0.020	bei	0.010		
so	0.014	es	0.010		
es	0.014				
ja	0.012				
der	0.011				

Table 8.5: The translation distribution of "that", learned by Model 2, Model 3 and the structure-based model. It is more uncertain in the word-based alignment model because the biased alignment distribution forces the associations between unrelated English/German words.

measure a model's uncertainty of source word translation overall? The answer is yes. Here we define the *average translation entropy* as

$$\sum_{i=1}^{m} P(e_i) \sum_{j=1}^{n} -t(g_j \mid e_i) \log t(g_j \mid e_i). \tag{8.1}$$

Here m, n are English and German lexicon sizes. The average translation entropy is a direct measure of word translation uncertainty. The average translation entropy of Simplified Model 2 is 3.01 bits per source word, 2.74 bits per source for IBM Model 3, while it is 2.50 bits per source word in the structure-based model.

This may explain how the structure-based model alleviates the sparse data problem. Normally people cope with the sparse data problem by reducing the complexity of a model. The complexity is often measured by the number of free parameters. For example, in speech recognition, we used class-based ngram to reduce the number of free parameters in language models, semi-continuous HMM (Huang and Jack, 1989) and triphone clustering (Lee, 1990) to reduce the number of free parameters in acoustic models. In neural networks, we used weigh sharing, like the time delay neural networks (Waibel et al., 1989) to reduce the number of parameters.

However, there is not much we can do along this line to solve the sparse data problem in machine translation, since we have to know the translation distribution of each individual word, and the translation parameters comprise a majority of parameters in all models. Techniques like source word clustering will cause the loss of critical information for translation.

Actually the number of free parameters is not the unique measure of model complexity. The translation entropy here is a information theoretic measure of model complexity. By focusing the learning of word translation on the roughly aligned phrases, the complexity of the structure-based model is effectively reduced.

8.2.4 How Structure-based Model Outperforms Word-based Models

In this subsection, I use some detailed examples to demonstrate how the structure-based model avoid the translation errors made by IBM Model 3.

Example 8.2.1 The following table lists an input German sentence, its reference English translation, and the translation made by Model 3 and the structured-based model.

Input (T):	ich wei"s nicht recht
Reference (R):	I do not know
Model 3 ($\mathbf{S}_{M3}$):	I will not know
Structure-based ($\mathbf{S}_{SM}$):	I do not know

Figure 8.4 illustrates the parameters involved in computing $P_{M3}(\mathbf{T}, \mathbf{A}_1 \mid \mathbf{S}_{M3})$ with Model 3, and it is compared with Figure 8.5, which shows the parameters involved in computing $P_{M3}(\mathbf{T}, \mathbf{A}_2 \mid \mathbf{S}_{SM})$ with Model 3. Here $\mathbf{A}_i$'s are the best alignments between the target and the source sentences that could be found by Model 3 with the hill-climbing search, and they are shown in the upper part of the two figures: The number in the parenthesis following a word shows the index (starting from 0) of the aligned position for that word in the translation. The lower part of the figures illustrates the parameters involved in calculating the likelihood of the alignment. `np(y|x)` is the fertility probability for the source word `x` to have fertility `y`. `t(y | x)` is the translation probability for translating the source word `x` to the target word `y`. `d(y | x)` is the distortion probability for aligning the source position `x` to the target position `y`. Appendix A explains what these parameters are for in detail. The last line shows the product of the translation model likelihood and the language model likelihood (the channel model score).

The only difference in translation model score in Figure 8.4 and Figure 8.5 is the fertility parameter `np(0|will)`=0.623619 versus `np(0|do)`=0.728261. Therefore the translation model slightly prefers the correct translation. However, this preference is offset by the stronger language model preference on the Model 3 translation (7.62136e-06 versus 5.9468e-06.) Therefore Model 3 picks the wrong translation.

On the other side, Figure 8.6 illustrates the parameters involved in computing $P_{SM}(\mathbf{T}, \mathbf{A}_3 \mid \mathbf{S}_{M3})$ with the structure-based model, and it is compared with Figure 8.7, which shows the parameters involved in computing $P_{SM}(\mathbf{T}, \mathbf{A}_4 \mid \mathbf{S}_{SM})$ with the structure-based model. Again, $\mathbf{A}_i$'s are the best alignments between the target and the source sentences that could be found by the structure-based model with the hill-climbing search, and they are shown in the upper part of the two figures: The number in the parenthesis following a word shows the index (starting from 0) of its aligned position in the translation. Each word in the German (target) sentence is preceded by a bracket `[x.y]`. Here `x` represents the index of the phrase that the

Alignment A_1:

```
ich (1) wei''s (4) nicht (3) recht (0)
#NULL# (3) I (0) will () not (2) know (1)
```

Score:

```
NULL word alignment score = 0.310632
    t(recht | NULL) = 0.000643527
np(1|I)=0.855647                    1!= 1
    t(ich | I) = 0.954941           d(0 | 1) = 0.763699
np(0|will)=0.623619                 0!= 1
np(1|not)=0.808492                  1!= 1
    t(nicht | not) = 0.760451       d(2 | 3) = 0.321565
np(1|know)=0.896904                 1!= 1
    t(wei''s | know) = 0.415161     d(1 | 4) = 0.11319
Translation Model Score:  6.48203e-14
Language Model Score:  7.62136e-06
TMS x LMS = 6.48203e-14 x 7.62136e-06 = 4.94019e-19
```

Figure 8.4: Model 3 Score for Model 3 Made Translation: the upper part shows the alignment between the source hypothesis and the target sentence. The number in the parenthesis following a word shows the index (starting from 0) of the aligned position for that word in the translation. The lower part of the figure illustrates the Model 3 parameters involved in calculating the likelihood of the alignment.

Alignment A_2:

```
ich (1) wei''s (4) nicht (3) recht (0)
#NULL# (3) I (0) do () not (2) know (1)
```

Score:

```
NULL word alignment score = 0.310632
    t(recht | NULL) = 0.000643527
np(1|I)=0.855647                    1!= 1
    t(ich | I) = 0.954941           d(0 | 1) = 0.763699
np(0|do)=0.728261                   0!= 1
np(1|not)=0.808492                  1!= 1
    t(nicht | not) = 0.760451       d(2 | 3) = 0.321565
np(1|know)=0.896904                 1!= 1
    t(wei''s | know) = 0.415161     d(1 | 4) = 0.11319
Translation Model Score:  7.56974-14
Language Model Score:  5.9468e-06
TMS x LMS = 7.56974-14 x 5.9468e-06 = 4.50155e-19
```

Figure 8.5: Model 3 Score for Structure-Based Model Made Translation: the upper part shows the alignment between the source hypothesis and the target sentence. The number in the parenthesis following a word shows the index (starting from 0) of the aligned position for that word in the translation. The lower part of the figure illustrates the Model 3 parameters involved in calculating the likelihood of the alignment.

word is in, and `y` is the index of the word in the sentence (starting from 0). The angle brackets mark the phrases in the English (source) sentence. The number that immediately follows an opening angle bracket (starting point of a phrase) is the index of the phrase in the sentence (starting from 0 for the null word phrase). The number in the bracket preceding a word in the source sentence is the index of the word position in the sentence (starting from 0 for the null word). The lower part illustrates the parameters involved in calculating the likelihood of the alignment. `ap(y | x)` is the rough alignment probability for aligning target phrase `x` with the source phrase `y`. `np(y | x)` is the fertility probability for the source word `x` to have fertility `y`. `t(y | x)` is the translation probability for translating the source word `x` to the target word `y`. `alpha(y | x)` is the probability for the anchor point of a target phrase to be `y` words away from the anchor point of its preceding phrase, provided that the source phrase aligned to the preceding target phrase is `x` words long. `d1(y | p,x)` is the probability for placing the leftmost translation (pin point) of the $\mathtt{x}^{th}$ word of the source phrase `p` at a position that is `y` words away from the anchor point of the target phrase roughly aligned to the source phrase `p`. Here `p` is the identification of the source phrase rather than the index of the phrase in the sentence. The last line shows the product of the translation model likelihood and the language model likelihood (the channel model score).

Here "I do not know" is treated as a single phrase in the structure-based model, and "I will not know" is treated as two phrases ("I will not" + " know") because the grammar inference algorithm did not find that "I will not know" is a frequently observed sequence as "I do not know". Because of this, more parameters (the rough alignment parameter `ap(2 | 1)` and the anchor point parameter `alpha(1-0 | 3)` for the phrase "know") are involved in computing $P_{SM}(\mathbf{T}, \mathbf{A}_1 \mid \mathbf{S}_{M3})$ than in computing $P_{SM}(\mathbf{T}, \mathbf{A}_2 \mid \mathbf{S}_{SM})$. Therefore the translation model's preference on the correct translation is magnified to offset the bias of the language model.

Example 8.2.2 The following table shows another example of an input German sentence, its reference English translation, and the translation made by Model 3 and the structured-based model.

Alignment A_3:

```
[0.0] ich (1) [1.1] wei''s (4) [0.2] nicht (3) [2.3] recht (0)
<0:[0] #NULL# (3)><1:[1] I (0) [2] will () [3] not (2)><2:[4] know (1)>
```

Score:

```
Target Phrase 0:
    ap(1 | 0) = 0.719307
    np(1 | I) = 0.860038
        tp(ich | I) = 0.94953          d1(0-0 | 305,0) = 0.64144
    np(0 | will) = 0.612335
    np(1 | not) = 0.799499
        tp(nicht | not) = 0.75429      d1(2-0 | 305,2) = 0.263653
Target Phrase 1:
    ap(2 | 1) = 0.41841                alpha(1-0 | 3) = 0.313435
    np(1 | know) = 0.877584
        tp(wei''s | know) = 0.425503
Target Phrase 2:
    NULL word alignment score = 0.243
    tp(recht | NULL) = 0.00089768
Translation Model Score:  3.91871e-07
Language Model Score:  7.62136e-06
TMS x LMS = 3.91871e-07 x 7.62136e-06 = 2.98659e-12
```

Figure 8.6: Structure-Based Model Score for Model 3 Made Translation: the upper part shows the alignment between the source hypothesis and the target sentence. The number in the parenthesis following a word shows the index (starting from 0) of its aligned position in the translation. Each word in the German (target) sentence is preceded by a bracket [x.y]. Here x represents the index of the phrase that the word is in, and y is the index of the word in the sentence (starting from 0). The angle brackets mark the phrases in the English (source) sentence. The number that immediately follows an opening angle bracket is the index of the phrase in the sentence (starting from 0 for the null word phrase). The number in the bracket preceding a word in the source sentence is the index of the word (starting from 0 for the null word). The lower part illustrates the structure-based model parameters involved in calculating the likelihood of the alignment.

Alignment A_4:

```
[0.0] ich (1) [0.1] wei''s (4) [0.2] nicht (3) [1.3] recht (0)
<0:[0] #NULL# (3)><1:[1] I (0) [2] do () [3] not (2) [4] know (1)>
```

Score:

```
Target Phrase 0:
    ap(1 | 0) = 0.719307
    np(1 | I) = 0.860038
        tp(ich | I) = 0.94953           d1(0-0 | 115,0) = 0.864257
    np(0 | do) = 0.725129
    np(1 | not) = 0.799499
        tp(nicht | not) = 0.75429       d1(2-0 | 115,2) = 0.677895
    np(1 | know) = 0.877584
        tp(wei''s | know) = 0.425503    d1(1-0 | 115,3) = 0.804824
Target Phrase 1:
    NULL word alignment score = 0.243
    tp(recht | NULL) = 0.00089768
Translation Model Score:  9.86591e-06
Language Model Score:  5.9468e-06
TMS x LMS = 9.865916-06 x 5.9468e-06 = 5.86706e-11
```

Figure 8.7: Structure-Based Model Score for Structure-based Model Made Translation: the upper part shows the alignment between the source hypothesis and the target sentence. The number in the parenthesis following a word shows the index (starting from 0) of its aligned position in the translation. Each word in the German (target) sentence is preceded by a bracket [x.y]. Here x represents the index of the phrase that the word is in, and y is the index of the word in the sentence (starting from 0). The angle brackets mark the phrases in the English (source) sentence. The number that immediately follows an opening angle bracket is the index of the phrase in the sentence (starting from 0 for the null word phrase). The number in the bracket preceding a word in the source sentence is the index of the word (starting from 0 for the null word). The lower part illustrates the structure-based model parameters involved in calculating the likelihood of the alignment.

Input (T):	ich hab ein Meeting von elf bis um eins
Reference (R):	I have a meeting from eleven to one
Model 3 ($\mathbf{S}_{M3}$):	I have a meeting from one to eleven
Structure-based ($\mathbf{S}_{SM}$):	I have a meeting from eleven to one

Figure 8.8 illustrates the parameters involved in computing $P_{M3}(\mathbf{T}, \mathbf{A}_5 \mid \mathbf{S}_{M3})$ with Model 3, and it is compared with Figure 8.9, which shows the parameters involved in computing $P_{M3}(\mathbf{T}, \mathbf{A}_6 \mid \mathbf{S}_{SM})$ with Model 3.

The differences in translation model score in Figure 8.8 and Figure 8.9 are the distortion parameters for the alignment between "eleven" and "elf" (`d(5 | 8)` = 0.148687 versus `d(5 | 6)` = 0.20641), and the alignment between "one" and "eins" (`d(8 | 6)` = 0.0479222 versus `d(8 | 8)`= 0.108858). Because of the nature of different word order between English and German, the differences in distortion probabilities are not very distinctive. Although the translation model prefers the correct translation slightly (1.85226e-17 versus 5.84093e-17), the language model bias (1.49928e-09 versus 4.13439e-10) again makes Model 3 pick the incorrect translation.

On the other side, Figure 8.10 illustrates the parameters involved in computing $P_{SM}(\mathbf{T}, \mathbf{A}_7 \mid \mathbf{S}_{M3})$ with the structure-based model, and it is compared with Figure 8.11, which shows the parameters involved in computing $P_{SM}(\mathbf{T}, \mathbf{A}_8 \mid \mathbf{S}_{SM})$ with the structure-based model. In these two alignments, the placement of the German words "elf" and "eins" has very different likelihood. In Figure 8.10, which is for the incorrect translation, placing "eins" at a position four words away from the anchor point (`4`) of the phrase is quite unlikely (`d1(8-4 | 113,1)` = 0.00678217), because the corresponding source word, "one", is only the second word in phrase 113 (the <from> ... <to> ... phrase). In Figure 8.11, on the other hand, placing "eins" at four words away from the anchor point (`4`) of the target phrase is very likely (`d1(8-4 | 113,3)` = 0.603562), since the model learned that the fourth word in source phrase 113 is often aligned to the position that is four words away from the anchor point of the corresponding target phrase. For the same reason, the structure-based model penalizes the alignment between "elf" and "eleven" in the incorrect translation. This results in the huge difference in translation model preference (7.71893e-13 versus 1.39528e-08), which is far more than enough to offset the language model bias.

Alignment A_5:

```
ich (1) hab (2) ein (3) Meeting (4) von (5) elf (8) bis (7)
    um (0) eins (6)
#NULL# (7) I (0) have (1) a (2) meeting (3) from (4) one (8)
    to (6) eleven (5)
```

Score:

```
NULL word alignment score = 0.389679
    t(um | NULL) = 0.0209076
np(1|I)=0.855647                        1!= 1
    t(ich | I) = 0.954941               d(0 | 1) = 0.763699
np(1|have)=0.72464                      1!= 1
    t(hab | have) = 0.748826            d(1 | 2) = 0.579236
np(1|a)=0.631654                        1!= 1
    t(ein | a) = 0.869895               d(2 | 3) = 0.321565
np(1|meeting)=0.910993                  1!= 1
    t(Meeting | meeting) = 0.261291     d(3 | 4) = 0.269056
np(1|from)=0.885466                     1!= 1
    t(von | from) = 0.708435            d(4 | 5) = 0.228576
np(1|one)=0.811801                      1!= 1
    t(eins | one) = 0.247491            d(8 | 6) = 0.0479222
np(1|to)=0.313078                       1!= 1
    t(bis | to) = 0.930271              d(6 | 7) = 0.185022
np(1|eleven)=0.990371                   1!= 1
    t(elf | eleven) = 0.935053          d(5 | 8) = 0.148687
Translation Model Score:  1.85226e-17
Language Model Score:  1.49928e-09
TMS x LMS = 1.85226e-17 x 1.49928e-09 = 2.77704e-26
```

Figure 8.8: Model 3 Score for Model 3 Made Translation: the upper part shows the alignment between the source hypothesis and the target sentence. The number in the parenthesis following a word shows the index (starting from 0) of the aligned position for that word in the translation. The lower part of the figure illustrates the Model 3 parameters involved in calculating the likelihood of the alignment.

Alignment A_6:

```
ich (1) hab (2) ein (3) Meeting (4) von (5) elf (6) bis (7)
    um (0) eins (8)
#NULL# (7) I (0) have (1) a (2) meeting (3) from (4) eleven (5)
    to (6) one (8)
```

Score:

```
NULL word alignment score = 0.389679
    t(um | NULL) = 0.0209076
np(1|I)=0.855647                        1!= 1
    t(ich | I) = 0.954941               d(0 | 1) = 0.763699
np(1|have)=0.72464                      1!= 1
    t(hab | have) = 0.748826            d(1 | 2) = 0.579236
np(1|a)=0.631654                        1!= 1
    t(ein | a) = 0.869895               d(2 | 3) = 0.321565
np(1|meeting)=0.910993                  1!= 1
    t(Meeting | meeting) = 0.261291     d(3 | 4) = 0.269056
np(1|from)=0.885466                     1!= 1
    t(von | from) = 0.708435            d(4 | 5) = 0.228576
np(1|eleven)=0.990371                   1!= 1
    t(elf | eleven) = 0.935053          d(5 | 6) = 0.20641
np(1|to)=0.313078                       1!= 1
    t(bis | to) = 0.930271              d(6 | 7) = 0.185022
np(1|one)=0.811801                      1!= 1
    t(eins | one) = 0.247491            d(8 | 8) = 0.108858
Translation Model Score:  5.84093e-17
Language Model Score:  4.13439e-10
TMS x LMS = 5.84093e-17 x 4.13439e-10 = 2.41487e-26
```

Figure 8.9: Model 3 Score for Structure-Based Model Made Translation: the upper part shows the alignment between the source hypothesis and the target sentence. The number in the parenthesis following a word shows the index (starting from 0) of the aligned position for that word in the translation. The lower part of the figure illustrates the Model 3 parameters involved in calculating the likelihood of the alignment.

Alignment A_7:

```
[0.0] ich (1) [1.1] hab (2) [1.2] ein (3) [1.3] Meeting (4) [2.4] von (5)
    [2.5] elf (8)[2.6] bis (7) [3.7] um (0) [2.8] eins (6)
<0:[0] #NULL# (7)><1:[1] I (0)><2:[2] have (1) [3] a (2) [4] meeting (3)>
     <3:[5] from (4) [6] one (8) [7] to (6) [8] eleven (5)>
```

Score:

```
Target Phrase 0:
    ap(1 | 0) = 0.719307
    np(1 | I) = 0.860038
        tp(ich | I) = 0.94953                d1(0-0 | 305,0) = 0.64144
Target Phrase 1:
    ap(2 | 1) = 0.41841                      alpha(1-0 | 1) = 0.802678
    np(1 | have) = 0.717237
        tp(hab | have) = 0.736251            d1(1-1 | 283,0) = 0.951571
    np(1 | a) = 0.640636
        tp(ein | a) = 0.86964                d1(2-1 | 283,1) = 0.348292
    np(1 | meeting) = 0.905556
        tp(Meeting | meeting) = 0.257002     d1(3-1 | 283,2) = 0.356933
Target Phrase 2:
    ap(3 | 2) = 0.25607                      alpha(4-1 | 3) = 0.244793
    np(1 | from) = 0.874983
        tp(von | from) = 0.736739            d1(4-4 | 113,0) = 0.970412
    np(1 | one) = 0.806936
        tp(eins | one) = 0.231316            d1(8-4 | 113,1) = 0.00678217
    np(1 | to) = 0.329776
        tp(bis | to) = 0.916636              d1(6-4 | 113,2) = 0.935702
    np(1 | eleven) = 0.987117
        tp(elf | eleven) = 0.937257          d1(5-4 | 113,3) = 0.00450878
Target Phrase 3:
    NULL word alignment score = 0.382637
    tp(um | NULL) = 0.0215548
Translation Model Score:  7.71893e-13
Language Model Score:  1.49928e-09
TMS x LMS = 7.718932-13 x 1.49928e-09 = 1.15728e-21
```

Figure 8.10: Structure-Based Model Score for Model 3 Made Translation: the upper part shows the alignment between the source hypothesis and the target sentence. The number in the parenthesis following a word shows the index (starting from 0) of its aligned position in the translation. Each word in the German (target) sentence is preceded by a bracket [x.y]. Here x represents the index of the phrase that the word is in, and y is the index of the word in the sentence (starting from 0). The angle brackets mark the phrases in the English (source) sentence. The number that immediately follows an opening angle bracket is the index of the phrase in the sentence (starting from 0 for the null word phrase). The number in the bracket preceding a word in the source sentence is the index of the word (starting from 0 for the null word). The lower part illustrates the structure-based model parameters involved in calculating the likelihood of the alignment.

Alignment A_8:

```
[0.0] ich (1) [1.1] hab (2) [1.2] ein (3) [1.3] Meeting (4) [2.4] von (5)
    [2.5] elf (6) [2.6]bis (7) [3.7] um (0) [2.8] eins (8)
<0:[0] #NULL# (7)><1:[1] I (0)><2:[2] have (1) [3] a (2) [4] meeting (3)>
     <3:[5] from (4) [6] eleven (5) [7] to (6) [8] one (8)>
```

Score:

```
Target Phrase 0:
    ap(1 | 0) = 0.719307
    np(1 | I) = 0.860038
        tp(ich | I) = 0.94953
Target Phrase 1:
    ap(2 | 1) = 0.41841                    alpha(1-0 | 1) = 0.802678
    np(1 | have) = 0.717237
        tp(hab | have) = 0.736251          d1(1-1 | 283,0) = 0.951571
    np(1 | a) = 0.640636
        tp(ein | a) = 0.86964              d1(2-1 | 283,1) = 0.348292
    np(1 | meeting) = 0.905556
        tp(Meeting | meeting) = 0.257002   d1(3-1 | 283,2) = 0.356933
Target Phrase 2:
    ap(3 | 2) = 0.25607                    alpha(4-1 | 3) = 0.244793
    np(1 | from) = 0.874983
        tp(von | from) = 0.736739          d1(4-4 | 113,0) = 0.970412
    np(1 | eleven) = 0.987117
        tp(elf | eleven) = 0.937257        d1(5-4 | 113,1) = 0.915819
    np(1 | to) = 0.329776
        tp(bis | to) = 0.916636            d1(6-4 | 113,2) = 0.935702
    np(1 | one) = 0.806936
        tp(eins | one) = 0.231316          d1(8-4 | 113,3) = 0.603562
Target Phrase 3:
    NULL word alignment score = 0.382637
    tp(um | NULL) = 0.0215548
Translation Model Score:  1.39528e-08
Language Model Score:  4.13439e-10
TMS x LMS = 1.39528e-08 x 4.13439e-10 = 5.76863e-18
```

Figure 8.11: Structure-Based Model Score for Structure-based Model Made Translation: the upper part shows the alignment between the source hypothesis and the target sentence. The number in the parenthesis following a word shows the index (starting from 0) of its aligned position in the translation. Each word in the German (target) sentence is preceded by a bracket [x.y]. Here x represents the index of the phrase that the word is in, and y is the index of the word in the sentence (starting from 0). The angle brackets mark the phrases in the English (source) sentence. The number that immediately follows an opening angle bracket is the index of the phrase in the sentence (starting from 0 for the null word phrase). The number in the bracket preceding a word in the source sentence is the index of the word (starting from 0 for the null word). The lower part illustrates the structure-based model parameters involved in calculating the likelihood of the alignment.

Model	Decoder	Total	Failed	Correct	OK	Incorrect	Accuracy*
Semantic	—	367	4	199	8	156	55.3%
Structure-Based	BFD+R	367	2	203	76	86	65.6%
Structure-Based	IBM+R	367	28	202	77	60	65.5%

Table 8.6: Translation Accuracy of Semantic-Based Translator and the Structure-Based Statistical Translator: a correct translation gets one credit; an okay translation gets 1/2 credit; an incorrect one gets 0 credit. Accuracy* was calculated with respect to all input sentences.

8.3 Statistical vs. Symbolic Machine Translation

As mentioned in Chapter 1, a semantic-based symbolic machine translator was developed in the same domain. I hesitated to do direct comparison between statistical machine translation with the symbolic one, because it did not seem to be fair: the symbolic translation took several man-years to develop the grammar and the interlingua representation, while the statistical translator learned the translation automatically.

It is reported that the semantic translator achieved over 80% translation accuracy. (The accuracy was calculated by giving full credit to both correct and okay translations.) However, the first experiment with my own test data showed that the statistical machine translator outperformed the semantic translator. The performance data is listed in Table 8.6.

Unlike in the previous tables, here the accuracy is reported with respect to the total number of input sentences (367) rather than the number of sentences successfully decoded by a decoder.

8.3.1 Statistical Machine Translation is More Robust

So what was wrong with the semantic translator? By examining the test data and the parsed interlingua representation with the designer of the semantic translator, it seemed that unknown words and language constructions were major causes of errors. As stated by the parser designer: "It's just bad luck that the word 'meeting' appears in 6 of the 69 utterances and it's not in the grammar — I guess the grammar was developed when Germans still spoke German..."

Actually the grammar was developed with the original monolingual Janus scheduling data, while the test data was drawn from the Verbmobil corpus. Although they are both in the same domain of appointment scheduling, there are differences between the two corpora. One example was described above — in the Verbmobil data, the English word "meeting" was frequently used

in the German utterances, while in the original Janus data, the occurrence of "meeting" was not frequent enough to attract the grammar developers' attention.

While it is legitimate to argue that the performance evaluation was not fair to the semantic translator, it reveals the disadvantage of the approach: it is not very robust to unseen or noisy data.

The poor robustness performance can be attributed to the semantic translation algorithm. The translator tries to figure out what the speech act of a sentence is, and then uses a template to generate a target sentence for that speech act, with the necessary arguments of the template translated from the input sentence. For example, it may guess that "my_unavailability" is the speech act of the sentence "Ich habe nämlich ein Seminar ab zehn," and use the output sentence template "I couldn't do + **time_expression**" to generate the output sentence with the speech act "my_unavailability." The argument **time_expression** can be obtained by translating the time expression "ab zehn", which will result in the translation "I couldn't do after ten." Normally the parser does very well in parsing time expressions, since a great amount of effort was devoted to frequently-observed time expressions in the scheduling data. On the other hand, the frequency of time expressions also implies that they can occur in a variety of speech acts. Therefore a time expression alone is not a good indicator of speech acts. Instead we have to use other constructions to guess the speech act of sentences. Unfortunately, as discussed in Chapter 2, it is not realistic to have a grammar with good coverage for quite arbitrary oral language. The strategy, used by the semantic parser to handle this problem, was to skip unparsable segments. However, this may have disastrous effects on the translations it generated: the skip often caused incorrect guesses about speech acts, and as a result, the sentences were translated with a wrong template for a different speech act, or translated as a dangling time expression that could only confuse the listener. This was the most frequently observed type of mistake in semantic translation, as illustrated in Table 8.7.

The correct/okay/incorrect translation distribution of the semantic translator in Table 8.6 is consistent with the analysis: there were not as many okay translations as the statistical translator. Most translations are either correct (with correct guess of speech-act or its semantic functionality) or incorrect (with incorrect guess of speech act). As a matter of fact, the translator failed to accurately translate all sentences containing the word "meeting".

Input Sentence:	Ich habe ein weiteres Meeting von drei bis um vier am Freitag.
Semantic Translation:	From three o'clock to four o'clock Friday.
Correct Translation 1:	I have another meeting from three o'clock to four o'clock Friday.
Correct Translation 2:	I am busy from three o'clock to four o'clock Friday.

Table 8.7: Skip causes incorrect translation: because the sentence translator did not know the word "meeting", it skipped the beginning part of the input sentence and translated the time expression only, which is not considered to be incorrect since it is confusing and most listeners would take it as "suggesting_time".

Experiment		**Total**	**Correct**	**OK**	**Incorrect**	**Accuracy**
"Meeting" Sentences	Control	27	14	9	4	68.5%
	Meeting →**unknown**	27	10	11	6	57.4%

Table 8.8: Translation Accuracy for sentences with unknown words: a correct translation gets one credit; an okay translation gets 1/2 credit; an incorrect one gets 0 credit.

To see how the statistical translator performed with unseen data, I replaced the word "Meeting" in the test sentences with the token for unknown word, and used the statistical translator to find the translations. In the 367 test sentences, there were 27 occurrences of "Meeting". Table 8.8 shows the performance of the 27 translations.

The result in Table 8.8 shows that the performance of the 27 translations decreased from 64.8% to 57.4%. While evaluating the performance with unknown words, translations with the correct speech act and appropriate replacements of the unknown words were scored as correct translations. For example, with the input sentence "Ich habe ein weiteres **unknown** (Meeting) von drei bis um vier am Freitag", any translation like "I have another meeting/class/seminar from three to four on Friday" is given a full credit. Compared to the fact that the semantic translator failed on all the "meeting" sentences, the statistical translator is much more robust. Actually for this example, the translator correctly generated the translation "I have another meeting from three to four on Friday." This is because of the fact that "meeting" often follows "another" in the corpus, and "another meeting" is a phrase in the phrase structure. Adding the phrase to the translation would greatly increase the likelihood of "weiteres" and other words in the sentence.

Model	Total	Correct	OK	Incorrect	Accuracy
Semantic	300	164	29	107	59.5%
Structure-Based	300	171	43	86	64.2%

Table 8.9: Translation Accuracy with Balanced Test Data: a correct translation gets one credit; an okay translation gets 1/2 credit; an incorrect one gets 0 credit.

8.3.2 Statistical Machine Translation is More Accurate

To have a fair performance comparison, I drew another set of test data consisting of 300 sentences, half of which were from the original Janus data and half of which were from the Verbmobil data. Table 8.9 shows the performance of the semantic and statistical translators.

8.3.3 Statistical Machine Translation is More Natural

Another advantage of the statistical machine translator is that the translations it generates are more natural or close to the literal translation of the input sentence, while the semantic translator makes terse translations that lack variations. While it is considered to be correct to translate "Wir sehen uns dann" to "Wonderful", and the translation receives a full credit in performance evaluation, it is not very natural to translate everything that expresses agreement about a proposed appointment time as "Wonderful", as the semantic translator did. Table 8.10 demonstrates some examples of correct translations made by the statistical and semantic translators.

8.4 Monolingual Grammar Inference vs. Bilingual Grammar Inference

Table 8.11 shows the performance of the translation model based on structures inferred from monolingual and bilingual corpora. The decoder used were the BFD+R algorithm.

8.5 Hand-Made vs. Automated Utterance Segmentation

All the above evaluations used the manually presegmented utterances, i.e., every long utterance had already been manually segmented into smaller semantic units. Although I had developed

Input Sentence:	Ich am Freitag in den Urlaub gehe.
Semantic Translator:	I couldn't do Friday.
Statistical Translator:	Friday I am on vocation.
Input Sentence:	An dem morgen bin ich schon verplant.
Semantic Translator:	I couldn't do on that in the morning.
Statistical Translator:	I am busy on the morning.
Input Sentence:	Ich habe nämlich ein Seminar ab zehn.
Semantic Translator:	I couldn't do starting ten o'clock
Statistical Translator:	I have class from ten.
Input Sentence:	Das geht bei mir.
Semantic Translator:	Wonderful.
Statistical Translator:	That suits me.
Input Sentence:	Wir sehen uns dann.
Semantic Translator:	Wonderful.
Statistical Translator:	See you then.
Input Sentence:	Das hört sich gut.
Semantic Translator:	Wonderful.
Statistical Translator:	That sounds good.

Table 8.10: Some Corrected Translations Made by the Semantic and Statistical Translators. The semantic translations are terse and lack variations.

GI Algorithm	Total	Failed	Correct	OK	Incorrect	Accuracy*
Monolingual	367	2	195	73	97	63.1%
Bilingual	367	2	203	76	86	65.6%

Table 8.11: Translation Accuracy vs. Grammar Inference Algorithms: a correct translation gets one credit; an okay translation get 1/2 credit; an incorrect one get 0 credit.

Segmentation	Total	Failed	Correct	OK	Incorrect	Accuracy
Automatic	342	4	177	67	94	61.5%
Hand Made	367	2	203	76	86	65.6%

Table 8.12: Translation Accuracy vs. Segmentation: a correct translation gets one credit; an okay translation gets 1/2 credit; an incorrect one gets 0 credit.

a segmentation algorithm, the performance of the algorithm is not 100% correct. This experiment tries to answer the question about the impact of incorrect segmentations. Table 8.12 compares the translation performance between the automatic segmentations and the manual segmentations. The error rate increased by 11.9% relatively with the automatic segmentations.

8.6 Error Analysis

Figure 8.12 shows different causes od translation errors. Here around 37% of target sentences that had incorrect or okay translations contained words that occurred fewer than 3 times in the training data [`LowFreqW`]. Around 28% of the incorrect or okay translations had grammatical errors [`Ungrammatical`]. For those grammatical errors, a more powerful language model may be very helpful. Around 11% of the translations missed some close class words like articles or prepositions [`Missing CCW`]. Around 8% sentences contained out of valcalbury words [`OOV`], and 7% sentences were out of domain [`OOD`]. Around 3 percent had incorrectly translated closed class words [`Wrong CCW`]. And around 3% of errors were morphology related. Table 8.13 shows examples of some of those translation errors.

I have time all afternoons were translated as I am free every time in the afternoon.

From the above error analysis, we can see that the sparse data problem is still a major cause of translation errors. `LowFreqW`, `OOV` and `OOD` are directly related to te sparse data problem. `Ungrammatical` is indirectly related: we used bigram instead of trigram for language modeling because we did not have enough training data. Therefore, the language model was not restrictive enough (compared to the trigram model) to avoid some grammatical errors. It is interesting to know if more data will be helpful to improve the performance of the structure-based model. Without actual data, we can not answer this question definitely. However, we can project from history to get some knowledge about the future. Figure 8.13 plots the accuracy of the translator as a function of the amount of the training data — it recorded the performance when 10,000,

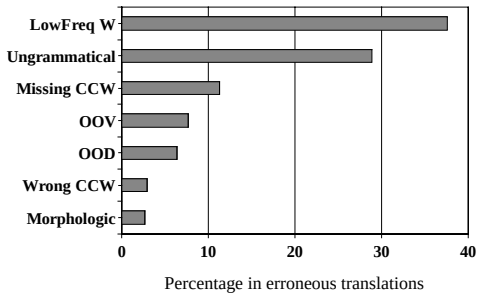

Figure 8.12: Causes of Translation Errors

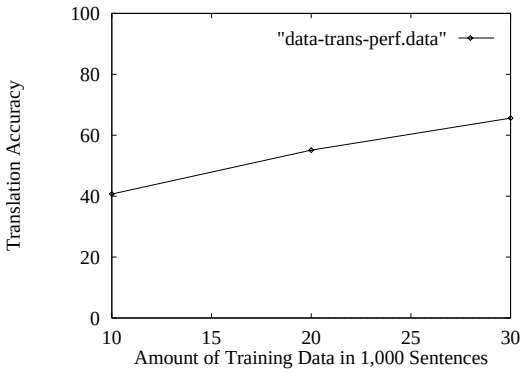

Figure 8.13: Translation Accuracy vs. Amount of Training Data for the Structure-Based Model

`Ungrammatical`	Input	ich glaub nicht da"s ich das einrichten k"onnen
	Output	I guess I can do not possible
	Ref	I do not think I can do that
`Missing CCW`	Input	am Mittwoch hab Sie da Zeit
	Output	on Wednesday you have time
	Ref	do you have time on Wednesday
`Wrong CCW`	Input	das ja ich wei"s liegt genau zwischen zwei Meetings
	Output	that I already know is exactly from two meetings
	Ref	that I know is exactly between two meetings
`Morphological`	Input	ich hab ein Meeting von drei bis um vier an diese Tag
	Output	I have a meeting from three to four days
	Ref	I have a meeting from three to four that day

Table 8.13: Examples of Different Types of Errors.

20,000 and 30,000 parallel sentences were used for model training. Clearly the curve did not reach a plateau yet. We can still expect performance improvement with more training data.

Chapter 9

Summary

This final chapter reviews the accomplishments of the thesis. I have examined the problem of statistical machine translation for spoken language between a language pair that has different structures. It is a difficult task since spoken languages are hardly ever well-formed. A speech translation system cannot assume any rigid syntactic constraints in an input language. It is also not practical to model the noise with grammar rules due to its unpredictability. While the empiricist approach like statistical machine translation is intrinsically advantageous for spoken languages, the structure difference between the language pair (different word orders and deletions in translations) limited the power of the conventional word-based alignment model for statistical machine translation.

Is the parallelism/structure similarity a must for statistical machine translation? If so, the generality of the approach is quite questionable. This is actually the most poignant criticism towards statistical machine translation.

The thesis work presented here advanced the state of the art statistical machine translation techniques by applying the approach to a spoken language pair with different structures, and therefore demonstrated that sophisticated statistical model is available to handle language pairs with more structure difference. Although the structure difference between English and German is moderate, and we still do not know if the model described in this thesis is applicable to language pairs with more radically difference structure, the work reported here is the first step towards the investigation of the generality of statistical machine translation.

The following key ideas are essential in the thesis:

1. Statistical machine translation is promising for spoken language translation because of its performance orientation. It is modeled directly from performance data rather than from human language competence. It is also desirable because of its learnability.

2. We have to tackle the problems specific to spoken languages and specific to some language pairs in order to make statistical machine translation generally applicable. These include noisy data, long unsegmented utterances, structure difference, etc. We have to deal with the sparse data problem that puzzles most empirical approaches.

3. Structure-based statistical machine translation can solve or alleviate some of these problems. The rough phrase alignment can directly model the structure difference between English and German. It also results in a more focused automatic learning; therefore, it alleviates the noisy data and sparse data problems.

4. Automatic grammar inference is desirable for the structure-based translation model. If we have to manually find useful structures, then all the advantages of statistical machine translation will vanish: laborious human involvement is required for language knowledge engineering; the structure is a reflection of the designers' understanding of the data rather than the best fit of the data, so it is not necessarily beneficial for statistical machine translation.

9.1 Contributions

In the effort towards the achievement of applying statistical machine translation to a spoken language pair with different structures, the following

contributions were made in the thesis:

1. Introduction of a new structure-based translation model that improves the performance on spoken language or language pairs with different structures. Including structures in the statistical machine translation model enables us to directly model the difference word/phrase orders and deletions in translation. It enables a more focused learning and results in an information-theoretically simpler model.

2. Investigation of novel approaches to grammar inference and bilingual grammar inference that facilitate the structure-based translation model. While automatic grammar inference allows us to stay in the automatic empirical framework, the bilingual grammar inference algorithm makes the grammar inference techniques more suitable for machine translation.

3. Development of a statistical dialogue analysis model that can divide dialogues and long utterances into basic (shorter) semantic units, which enables statistical machine translation systems to process unsegmented speech outputs. While the algorithm was used for segmentation only, it has potential application to many other different tasks related to dialogue analysis, such as topic-dependent language modeling. My preliminary experiment has shown that the speech act information can be used to reduce language model perplexity.

4. Investigation of different decoding algorithms for statistical machine translation and development of a faster BFD+R algorithm that fastens the decoding speed without loss of translation quality.

9.2 Conclusions

We can draw the following conclusions from the experiments in this thesis:

At least in a limited domain, statistical machine translation is promising for spoken language pairs with more radical structure difference than the transduced English and French in the IBM system.

1. For spoken language translation, especially between a language pair with different word orders, structure information is very important. Statistical machine translation needs structures, and the structure-based model improves translation performance. The structure-based statistical machine translation model is more precise than the word-based alignment models. It reduced the error rate by 11% relatively over the state of the art, best performing word-based alignment model.

2. Compared to semantic-based symbolic translation, structure-based statistical translation is more robust, accurate and natural. It is more robust to noisy data. It is able to generate

satisfying outputs when unknown words exist in the input sentences. Its performance exceeds the performance of the symbolic translator, and the translations made by the statistical translator are more literal and natural, not like the terse translations made by the semantic translator.

3. Automatic grammar inference is not only possible, but also very promising for statistical machine translation. It played a very important role in the structure-based model. The bilingual grammar inference algorithm was more suitable for machine translation, and it improves translation performance.

4. Statistical dialogue analysis can achieve close-to-human performance in segmenting long utterances, while bracketing (segmentation + labeling) performance has room for improvement.

5. While statistical machine translation achieved better performance, there are limitations to the approach: it is data hungry, and to make it worse, it requires parallel corpus. The decoding speed is still slow. Improvement is necessary for any real time or near real time applications without sacrificing accuracy.

9.3 Future Directions

While the structure-based statistical translation has achieved better performance than semantic translation and word-alignment based statistical translation, there is still much room for performance improvement. Several theoretical issues also need to be clarified. The future directions for statistical machine translation, as I perceive them, include

- **Combination of statistical and symbolic techniques in machine translation.**

  While this work is the first one that includes language structures into a statistical translation model, there are many different ways to find and use structures. In this thesis the structure is automatically learned. The shallow phrase structures contain much noise. Here we found that statistical significance does not imply linguistic significance. It is of great interest to combine the statistical and symbolic techniques in machine translation,

especially when we have less training data. For example, we can take advantage of existing symbolic systems and used statistical approach to smooth the symbolic model.

- **Automatic Detection of preprocessing procedures:** According to my experience in this thesis work, the preprocessing procedures played a very important role in translation performance improvement. Two most important preprocessing steps improved the performance by almost 5 percent. It is of great interest if the model can detect or suggest promising preprocessing procedures. Some prominent preprocessing steps can be identified with machine learning techniques. For example, if the model has learned that the word "one" can be translated to "eins (one)" and "zwei (two)" in German, and it is translated to "zwei" if and only if when "zwei" follows the word "halb (half)", then it can introduce the preprocessing step to rewrite "halb zwei" as "halb eins", so "one" can be consistently translated to "eins" in the model.

- **Sophisticated Decoding Algorithms:** One of the biggest problem of statistical machine translation is source sentence decoding. The IBM stack decoder was not efficient to handle long input sentences, and it also failed on many long sentences. My best-first with reshuffling algorithm worked much faster and failed on fewer input sentences. However, its accuracy on those successfully decoded sentences was lower than that of the IBM decoder. Basically the sentence failed by the IBM stack decoder are mostly difficult ones, and the faster decoding algorithm often generate incorrect translations for these difficult sentences. So the two decoding algorithms had similar accuracy if it was measured with respect to the total number of input sentences.

A possible and promising approach to speed up decoder is to identify the equivalent prefix strings. For example, if we have find that "I have a meeting", "I have got a meeting" and "I have one meeting" are all equivalent, then we can speed up decoding by extending only the most prominent hypothesis in the equivalent class.

- **Incorporating Bilingual Dictionaries:** bilingual dictionaries is a very important source of information that can be used to alleviate the sparse data problem. IBM CANDIDE system used bilingual dictionaries in a weird way. It treated dictionary entries as data and used them to estimate the parameters in a translation model. As a more

intuitive way to incorporate dictionary into the model, we can use a bilingual dictionary to get the prior translation distribution of source words, and then use bilingual data to estimate the *a posteriori* translation distribution.

- **Application to a larger domain with more data:** it is interesting to see how the algorithms described in the thesis can be scaled up to deal with more complicated tasks. While this structure-based model is introduced for the scheduling domain, I perceive nothing that prevents its application to other domains except that the languages in that domain may contain too many structures that will make the structure-based model too complicated. Experiments can be conducted to find out how big a task can be such that it can be handled effectively by the structure-based model.

- **Application to language pairs with more radically different structures**. Although the structure-based model is one step forward towards statistical machine translation for languages with different structure, the structure difference between English and German is moderate. We would like to know how much the structure-based model can help with languages with more radically different structures, like English and Vietnamese.. We would like to know how the structure difference other than different word orders and deletions in translation might affect the performance of statistical machine translation.

- **Evaluation method.** Last but not least, we need a better evaluation method. The method should be more objective and ideally automatic or semi-automatic. Of course, this task is not just for statistical machine translation. It's a task for the whole machine translation society.

Appendix A

Translation Models

A.1 IBM Translation Models

Most statistical machine translation systems use ngram (Jelinek, 1990) for language modeling. Translation models rely on the concept of *alignment*. An alignment translation model assumes that a target sentence is generated from a source sentence word by word. A target sentence word can therefore be *aligned* with the source sentence word that produces it. In an alignment, each target word can align with only one source sentence word. So far, most of the statistical machine translation systems use word-based alignment model (Brown et al., 1993; Vogel, Ney, and Tillman, 1996; Wang and Waibel, 1997a), and no structure is involved in the word-by-word alignment. (Brown et al., 1993) introduced five different word-based alignment models for translation modeling, and I briefly review model 2, 3, and 4 here, because in later chapters I will heavily analyze the problems of these models, compare the different behavior and performance between these models and my structure-based alignment model.

A.1.1 Model 1 and Model 2

IBM alignment model 2 is a typical example of word-based alignment. Assume that a sentence $\mathbf{e} = e_1, \cdots, e_l$ is at the source of a channel, it picks a length m for the target sentence $\mathbf{g}$ [1] with the distribution $\Pr(m \mid \mathbf{e}) = \epsilon$, where ϵ is a small, fixed number. Then for each position

[1] I use $\mathbf{e}$ and $\mathbf{g}$ here for source and target sentence because in this thesis the translation is performed from German (target language) to English (source language).

i $(0 < i \leq m)$ in $\mathbf{g}$, it finds its corresponding position a_i in $\mathbf{e}$ according to an *alignment* distribution $\Pr(a_i \mid i, a_1^{i-1}, m, \mathbf{e}) = a(a_i \mid i, m, l)$. Finally, it generates a word g_i at the position i of $\mathbf{g}$ from the source word e_{a_i} at the aligned position a_i, according to a *translation* distribution $\Pr(g_i \mid a_1^m, g_1^{i-1}, \mathbf{e}) = t(g_i \mid e_{a_i})$.

Therefore, the likelihood of generating a target $\mathbf{g}$ from a source $\mathbf{e}$, $\Pr(\mathbf{g} \mid \mathbf{e})$, is the sum of the probabilities of generating $\mathbf{g}$ from $\mathbf{e}$ over all possible alignments A, in which the position i in $\mathbf{g}$ is aligned with the position a_i in $\mathbf{e}$:

$$\begin{aligned} \Pr(\mathbf{g} \mid \mathbf{e}) &= \epsilon \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(g_j \mid e_{a_j}) a(a_j \mid j, m, l) \\ &= \epsilon \prod_{j=1}^{m} \sum_{i=0}^{l} t(g_j \mid e_i) a(i \mid j, l, m) \end{aligned} \tag{A.1}$$

The successive models after Model 2 try to incorporate the fact that different source words may produce different numbers of target words (the fertility of source words in Model 3, 4, and 5), as well as the fact that the multi-word translation of a source word often moves as a whole unit in the target sentence (Model 4 and 5).

Model 1 is a special case of Model 2, in which the alignment parameters are replaced by uniform distributions..

A.1.2 Model 3

In Model 3, a fertility distribution $n(\phi_i \mid e_i)$ is introduced for each source word e_i in the source sentence $\mathbf{e} = e_1 e_2 ... e_l$, so it can be used to statistically determine the number of target words that can be generated from e_i. The fertility for the hypothesized null word at the position 0 of the source sentence, ϕ_0, is determined differently. It is based on the assumption that each target word generated from its aligned source word requires an extraneous word with probability p_1, and this extraneous word must be aligned to the null word. The probability that exactly ϕ_0 of the extraneous words are required by the $\sum_{i=1}^{l} \phi_i$ target words that are generated from the source words $e_1 e_2 ... e_l$, can be determined with the binomial distribution:

$$\Pr(\phi_0 \mid \phi_1^l, \mathbf{e}) = n_0(\phi_0 \mid \sum_{i=1}^{l} \phi_i) = \left(\begin{array}{c} \sum_{i=1}^{l} \phi_i \\ \phi_0 \end{array} \right) (1-p_1)^{\sum_{i=1}^{l} \phi_i - \phi_0} p_1^{\phi_0} \tag{A.2}$$

Model 3 can be described generatively with the following process to produce **g** or *failure* from **e** $=e_1 e_2 ... e_l$:

1. For each $i = 1, 2, ..., l$, with the probability $n(\phi_i \mid e_i)$, determine the fertility ϕ_i of the source word e_i.

2. Choose the fertility ϕ_0 for the hypothesized null word at the position 0 of the source sentence, according to the distribution $n_0(\phi_0 \mid \sum_{i=1}^{l} \phi_i)$.

3. Let the target sentence length $m = \phi_0 + \sum_{i=1}^{l} \phi_i$.

4. For each $i = 0, 1, 2, ..., l$, creates a list of ϕ_i target words (*tablet*) $\tau_i = \{\tau_{i1} \tau_{i2} ... \tau_{i\phi_i}\}$ as the translations for the source word e_i. Each of the target word τ_{ik} in the tablet can be determined statistically with a translation distribution $t(\tau_{ik} \mid e_i)$.

5. For each $i = 1, 2, ..., l$ and $k = 1, 2, ..., \phi_i$, according to a distortion distribution $d(\pi_{ik} \mid i, m, l)$, choose a position π_{ik} from $1, 2, ..., m$, and place the translation τ_{ik} there in the target sentence **g**.

6. If any target position has been chosen more than once then return *failure*.

7. For each $k = 1, 2, ..., \phi_0$, choose the position π_{0k} from the $\phi_0 - k + 1$ remaining vacant positions in $1, 2, ..., m$, according to the uniform distribution.

Therefore, the probability for **g** to be a translation of **e**, is the sum of the probabilities of generating **g** from **e** with all possible alignments:

$$\Pr(\mathbf{g} \mid \mathbf{e}) = \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} \Pr(\mathbf{g}, \mathbf{a} \mid \mathbf{e})$$

$$= \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \binom{m-\phi_0}{\phi_0} (1-p_1)^{m-2\phi_0} p_1^{\phi_0} \prod_{i=1}^{l} \phi_i! n(\phi_i \mid e_i) \times \prod_{j=1}^{m} t(g_j \mid e_{a_j}) d(j \mid a_j, m, l).$$

In the equation the factorial $\phi_i!$ is introduced because there are $\phi_i!$ different sequences to generate the ϕ_i target words in the translation tablet for e_i, and then place them in the target sentence with different π_{ik}. $\phi_0!$ is not in the equation because it is cancelled by the probability of sequentially placing the ϕ_0 target words (that are aligned to the null word) into the vacant target positions with the uniform distribution, which is $1/\phi_0!$.

A.1.3 Model 4

Model 4 works similarly to Model 3, except for the placement of the translations of a source word e_i in its tablet τ_i: the distortion parameters, $d(j \mid i, m, l)$, are replaced by two sets of parameters: one placing the *head* of τ_i, the words in τ_i for which the position in the target string the leftmost, and one for placing the remaining words in τ_i.

After determine the fertility and the tablets for each of the source words, the head of a tablet for the source word e_i is placed with the distribution

$$\Pr(\Pi_{i1} = j \mid \pi_1^{i-1}, \tau_0^l, \phi_0^l, \mathbf{e}) = d_1(j - \odot_{i-1} \mid \mathcal{A}(e_{i-1}), \mathcal{B}(g_j)). \tag{A.3}$$

Here $\odot_{i-1}$is the *center* of the source word e_{i-1}, i.e., the ceiling of the average value of the positions in the target string of the words in the tablet of e_{i-1}. $\mathcal{A}$ and $\mathcal{B}$ are functions that respectively map source/target words into the equivalent classes that contain the source/target words. The equivalent classes are obtained with a mutual information clustering algorithm (Brown et al., 1992a).

The rest words in the tablet are then placed with the distribution

$$\Pr(\Pi_{ik} = j \mid \pi_{i1}^{k-1} \pi_1^{i-1}, \tau_0^l, \phi_0^l, \mathbf{e}) = d_{>1}(j - \pi_{i(k-1)} \mid \mathcal{B}(g_j)) \tag{A.4}$$

Here the position of a target word is relative to the position of the previously placed target word in the same tablet. This allows the model to account for the fact that the translations of a source word often move together in the target sentence.

The target generation precess of Model 4 can be described as follows:

1. For each $i = 1, 2, ..., l$, with the probability $n(\phi_i \mid e_i)$, determine the fertility ϕ_i of the source word e_i.

2. Choose the fertility ϕ_0 for the hypothesized null word at the position 0 of the source sentence, according to the distribution $n_0(\phi_0 \mid \sum_{i=1}^{l} \phi_i)$.

3. Let the target sentence length $m = \phi_0 + \sum_{i=1}^{l} \phi_i$.

4. For each $i = 0, 1, 2, ..., l$, choose a list of ϕ_i target words (*tablet*) $\tau_i = \{\tau_{i1}\tau_{i2}...\tau_{i\phi_i}\}$ as the translations for the source word e_i. Each of the target word τ_{ik} in the tablet can be determined statistically with a translation distribution $t(\tau_{ik} \mid e_i)$.

5. For each $i = 1, 2, ..., l$ and $k = 1, 2, ..., \phi_i$, choose a position π_{ik} from $1, 2, ..., m$, and place the translation τ_{ik} there in the target sentence $\mathbf{g}$. The choice of the position π_{ik} is made according to the following:

   (a) If $k = 1$, the choose π_{i1} according to the distribution $d_1(j - \odot_{i-1} \mid \mathcal{A}(e_{i-1}), \mathcal{B}(\tau_{i1}))$

   (b) If $k > 1$, the choose π_{ik} according to the distribution $d_{>1}(\pi_{ik} - \pi_{i(k-1)} \mid \mathcal{B}(\tau_{ik}))$

6. If any target position has been chosen more than once then return *failure*.

7. For each $k = 1, 2, ..., \phi_0$, choose the position π_{0k} from the $\phi_0 - k + 1$ remaining vacant positions in $1, 2, ..., m$, according to the uniform distribution.

Therefore, the probability for $\mathbf{g}$ to be a translation of $\mathbf{e}$, is the sum of the probabilities of generating $\mathbf{g}$ from $\mathbf{e}$ with all possible alignments:

$$\Pr(\mathbf{g} \mid \mathbf{e}) \quad = \quad \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} \Pr(\mathbf{g}, \mathbf{a} \mid \mathbf{e})$$

$$
\begin{aligned}
= \quad & \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \binom{m-\phi_0}{\phi_0} (1-p_1)^{m-2\phi_0} p_1^{\phi_0} \\
& \times \prod_{i=1}^{l} n(\phi_i \mid e_i) \times \prod_{j=1}^{m} t(g_j \mid e_{a_j}) \\
& \quad \times [\delta(\arg\min_k(a_k = a_j), j) \\
& \qquad \times d_1(j - \left\lceil \frac{\sum_{k=1}^{m} \delta(a_k, a_j - 1) \times k}{\phi_{a_j - 1}} \right\rceil \mid \mathcal{A}(e_{a_j}), \mathcal{B}(g_j)) \\
& \quad + (1 - \delta(\arg\min_k(a_k = a_j), j)) \\
& \qquad \times d_{>1}(j - \arg\max_k(a_k = a_j \wedge k < j) \mid \mathcal{B}(g_j))] \qquad \text{(A.5)}
\end{aligned}
$$

here $\delta(i,j) = 1$ if $i = j$, otherwise $\delta(i,j) = 0$. The factor $\delta(\arg\min_k(a_k = a_j), j)$ ensures that position j is the leftmost position in the target sentence for a translation of e_{a_j}, and $(1 - \delta(\arg\min_k(a_k = a_j), j))$ ensures that j is not such a position, so $d_{>1}$, rather than d_1, should be used to determine the placement of the translation.

References

Berger, A. L., P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, J. R. Gillett, J. D. Lafferty, R. L. Mercer, H. Printz, and L. Ures. 1996. *Language Translation Apparatus and Method Using Context-Based Translation Models.* United States Patent No. 5,510,981.

Black, E., R. Garside, and G. Leech. 1993. *Statistically-Driven Computer Grammar of English: The IBM/Lancaster Approach.* Rodopi B. V., Amsterdam - Atlanta.

Brown, P. F., J. Cocke, S. Della-Pietra, V. J. Della-Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85.

Brown, P. F., S. A. Della-Pietra, V. J Della-Pietra, and R. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

Brown, P. F., V. J. Della-Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. 1992a. Class-Based N-gram Models of Natural Language. *Computational Linguistics*, 18(4):467–479.

Brown, P. F., S. A. Della Pietra, V. J. Della Pietra, J. D. Lafferty, and R. L. Mercer. 1992b. Analysis, Statistical Transfer, and Synthesis in Machine Translation. In *Proceedings of the fourth International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 83–100.

Church, K. W. 1993. Char_align: A Program for Aligning Parallel Texts at the Character Level. In *Proceedings of the 31th Annual Meeting of the Association for Computational Linguistics*, pages 1–8.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Loyal Statistical Society*, B. 39.

Fu, King-Sun and Taylor L. Booth. 1975a. Grammatical Inference: Introduction and Survey — Part I. *IEEE Transaction on Systems, Man, and Cybernetics*, SMC-5(1):95–111.

Fu, King-Sun and Taylor L. Booth. 1975b. Grammatical Inference: Introduction and Survey — Part II. *IEEE Transaction on Systems, Man, and Cybernetics*, SMC-5(4):408–423.

Gorin, A. L. 1995. On Automated Language Acquisition. *Journal of Acoustical Society of America*, 97(6):3441–3461.

Gorin, A. L., S. E. Levinson, A. N. Gertner, and E. Goldman. 1991. Adaptive Acquisition of Language. *Computer Speech and Language*, 5:101–132.

Grosz, B. J. and C. J. Sidner. 1986. Attention, intention, and the structure of discourse. *Computational Linguistics*, 12 (3).

Hatazaki, K., J. Noguchi, A. Okumura, K. Yoshida, and T. Watanabe. 1992. INTERTALKER: An Experimental automatic interpretation system using conceptual representation. In *ICSLP '92*.

Huang, X. D. and M. A. Jack. 1989. Semi-Contiunous Hidden Markov Model for Speech Recognition. *Computer Speech and Language*, 3(3).

IEEE. 1995. *ICASSP '95*.

Jelinek, F. 1990. Self-Organized Language Modeling for Speech Recognition. In A. Waibel and K-F. Lee, editors, *Readings in Speech Recognition*. Morgan Kaufmann.

Jelinek, F. and E. L. Mercer. 1980. Interpolated Estimation of Markov Source Parameters from Sparse Data. In D. Gelsema and L. Kanal, editors, *Pattern Recognition in Practice*. North-Holland.

Kay, Martin, Jean M. Gawron, and Peter Norvig. 1994. *Verbmobil: A Translation System for Face-to-Face Dialog*. CSLI Lecture Notes; No. 33, Stanford, CA.

Kneser, R. and V. Steinbiss. 1993. On the Dynamic Adaptation of Stochastic Language Models. In *ICASSP '93*, volume 2, pages 586–589. IEEE.

Kneser, Reinhard and Herman Ney. 1993. Improved Clustering Techniques for Class-Based Statistical Language Modelling. In *Proceedings of European Conference on Speech Recognition*, volume 2, pages 973–976.

Knight, K. 1997. Automating Knowledge Acquisition for Machine Translation. *AI Magazine*, 18(4):225–242.

Lee, K. F. 1990. Context Dependent Phonetic Hidden Markov Model for Continous Speech Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, April.

Litman, Diane J. and James F. Allen. 1990. Discourse processing and commonsense plans. In *Intentions in Communications*.

Mayfield, Laura, Marsal Gavaldà, Wayne Ward, and Alex Waibel. 1995. Concept-based speech translation. In ICASSP-95 (ICA, 1995).

McCandless, M. 1994. Automatic acquisition of language models for speech recognition. Master's thesis, MIT.

Miller, Scott, Robert Bobrow, Robert Ingria, and Richard Schwartz. 1994. Hidden Understanding Models of Natural Language. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 25–32, Las Cruces, New Mexico.

Morimoto, T. and et al. 1994. ATR's Speech Translation System: ASURA. In *Proceedings of EUROSPEECH 1993*, page 1295.

Nilsson, N. 1971. *Problem-Solving Methods in Artificial Intelligence*. McGraw Hill, New York, New York.

Ries, Klaus, Finn Dag Buø, and Ye-Yi Wang. 1995. Improved Language Modelling by Unsupervised Acquisition of Structure. In ICASSP-95 (ICA, 1995). corrected version available via `http://www.cs.cmu.edu/ries/icassp_95.html`.

Roe, D. B., F. C. N. Pereira, R. W. Sproat, and M. D. Riley. 1992. Efficient Grammar Processing for a Spoken Language Translation System. In *ICASSP '92*, volume 1. IEEE.

Stolcke, A. and S. M. Omohundro. 1994. Best-first Model Merging for Hidden Markov Model Induction. Technical Report TR-94-003, International Computer Science Institute, Berkeley, California.

Suhm, B., P.Geutner, T. Kemp, A. Lavie, L. Mayfield, A. McNair, I. Rogina, T. Schultz, T. Sloboda, W. Ward, M. Woszczyna, and A. Waibel. 1995. JANUS: Towards multilingual spoken language translation. In *Proceedings of the ARPA Speech Spoken Language Technology Workshop, Austin, TX, 1995.*

Tillmann, C., S. Vogel, H. Ney, and A. Zubiaga. 1997. A DP-based Search Using Monotone Alignments in Statistical Translation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL/EACL'97)*, pages 313–320, Madrid, Spain.

Vogel, S., H. Ney, and C. Tillman. 1996. HMM-Based Word Alignment in Statistical Translation. In *Proceedings of the Seventeenth International Conference on Computational Linguistics: COLING-96*, pages 836–841, Copenhagen, Denmark.

Wahlster, W. 1993. First Results of Verbmobil: Translation Assistance for Spontaneous Dialogues. In *ATR International Workshop on Speech Translation.*

Waibel, A., T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. 1989. Phoneme Recognition Using Time-Delay Neural Networks. In *icassp89*, May.

Waibel, Alex. 1996. Interactive Translation of Conversation Speech. *Computer*, 29(7).

Wang, Y., J. Lafferty, and A. Waibel. 1996. Word Clustering with Parallel Spoken Language Corpora. In *Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP'96)*, Philadelphia, USA.

Wang, Y. and A. Waibel. 1997a. Decoding Algorithm in Statistical Machine Translation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL/EACL'97)*, pages 366–372, Madrid, Spain.

Wang, Y. and A. Waibel. 1997b. Statistical Analysis of Dialogue Structure. In *Proceedings of the 5th European Conference On Speech Communication and Technology (EUROSPEECH'97)*, Rhodes, Greece.

Ward, W. 1990. The cmu air travel information service: Understanding spontaneous speech. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 127–129.

Woszczyna, M. and Alex Waibel. 1994. Inferring Linguistic Structure in Spoken Language. In *ICSLP 1994*.

Young, S. 1993. Dialog Structure and Plan Recognition in Spontaneous Spoken Dialog. In *EUROSPEECH 1993*, volume 2, pages 1169–1172.