Statistical Machine Translation with Cascaded Probabilistic Transducers

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

von

Diplom–Physiker

Stephan Vogel

aus Steinfeld

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A number of different approaches to machine translation have been developed, ranging from linguistically motivated approaches, over example based approaches, to statistical systems. A good overview is given by [Hutchins and Somers 1992]. All these different approaches have their strong and their weak points. Therefore, a tendency is to combine them into one system which hopefully will show better translation performance than each individual component [Nirenburg and Frederking 1994, Wahlster 2000].

In recent years a number of studies on the use of statistical methods for machine translation have proved that this is a competitive approach which shows more robustness than other methods for the translation of spontaneous speech. However, statistical machine translation shows some problems with syntactical correctness of the generated sentences in the target language, and also a number of problems with some specific phenomena, like for example the translation of time expressions.

The goal of this work is to combine ideas from different approaches into one framework. This general framework is provided by weighted finite state transducers which are arranged into a cascade. Example based machine translation can be reformulated as translation with a finite state transducer. Bilingual grammars, when restricted to regular grammars, can be expressed by a cascade of transducers. Using weighted transducers a probabilistic translation model can be formulated. Compared to the standard statistical approach to machine translation the translation model based on cascaded transducers introduces more structure for the translation relation between source and target language.

The following sections will give a short summary of the state of the art regarding the application of statistical and finite state methods to machine translation. This will be followed by an overview of the approach to machine translation on the basis of cascaded transducers.

In the central part of the thesis (Chapters 5 to 7) the details of this approach will be presented. It will be shown how the translation memory approach can be reformulated in terms of transducers. The important advantage of doing so is that the transducer is formulated within the Bayesian framework and constitutes a probabilistic translation model. Cascading finite state transducers is used to achieve generalization.

Training the model as well as formulating the search algorithm both involve the application of the transducers which is essentially bottom-up parsing. This algorithm is described in Chapter 5.

A discussion of the training method for this model follows in Chapter 6. The first step in training is the construction of a top-level transducer. To achieve this an alignment model is used and extended to allow for alignment between two graphs.

A detailed description of the search algorithm follows in Chapter 7. The translation process is characterized as the construction of a translation graph by successive application of transducers from a complete cascade of transducers.

The cascaded transducer approach is not a fully automatic translation method. This raises the question how labor intensive the construction of the transducers is. In Chapter 8 this point is discussed and some methods are explicated which help in the construction of specialized transducers.

Experiments have been performed on two corpora and the results are presented in Chapter 9.

The thesis closes with a summary and a discussion of directions that seem promising for further research.

1.1 Statistical Translation

1.1.1 The Bayesian Approach

The goal is the translation of a text given in some source language into a target language. We are given a source string $f_1^J = f_1...f_j...f_J$, which is to be translated into a target string $e_1^I = e_1...e_i...e_I$. In this article, the term *word* always refers to a *full-form* word. Among all possible target strings, we will choose the string with the highest probability which is given by Bayes' decision rule ([Brown et al. 1993a]):

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

Here, $p(e_1^I)$ is the language model of the target language, and $p(f_1^J|e_1^I)$ is the string translation model. The argmax operation denotes the search problem, i.e. the generation of the output sentence in the target language. The overall architecture of the statistical translation approach is summarized in Figure 1.1.

In general, as shown in this figure, there may be additional transformations to make the translation task simpler for the algorithm. The transformations may range from the categorization of single words and word groups to more complex preprocessing steps that require some parsing of the source string. We have to keep in mind that in the search procedure both the language and the translation model are applied *after* the text transformation steps. However, to keep the notation simple, we will not make this explicit distinction in the subsequent exposition.

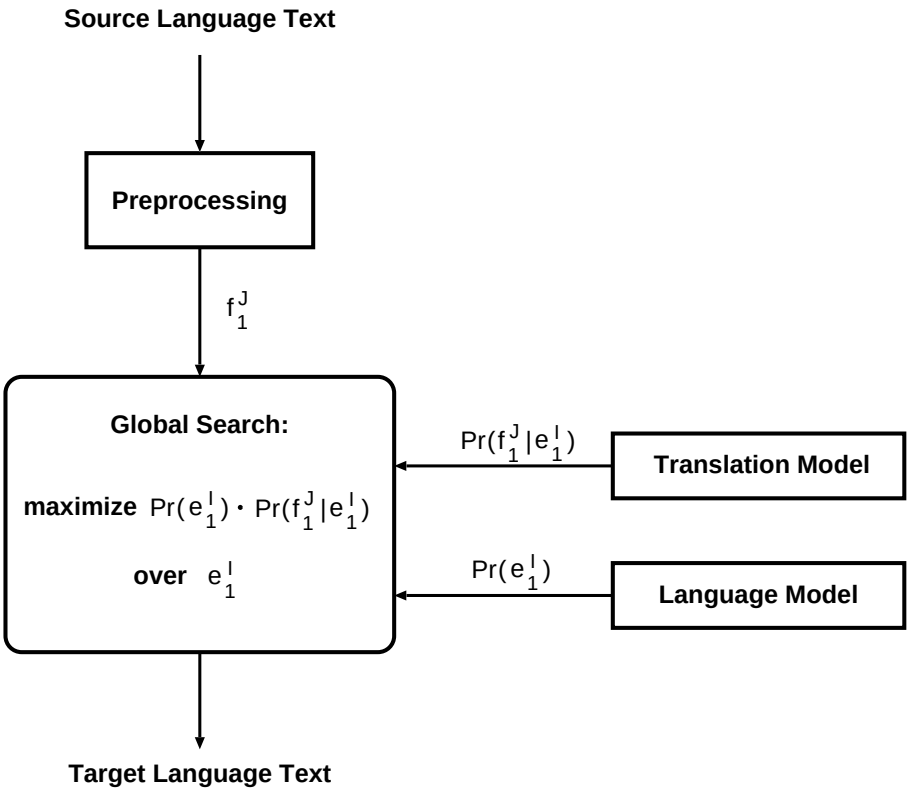

Figure 1.1: Architecture of the translation approach based on Bayes decision rule.

1.1.2 Basic Alignment Models

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

When aligning the words in parallel texts, we typically observe a strong localization effect. Figure 1.2 illustrates this effect for the language pair German–English. In many cases, although not always, there is an even stronger restriction: over large portions of the source string, the alignment is monotone.

To arrive at a quantitative specification, we define the

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

which assigns a word f_j in position j to a word e_i in position $i = a_j$. Now, we can rewrite the probability for the translation model by introducing the 'hidden' alignments

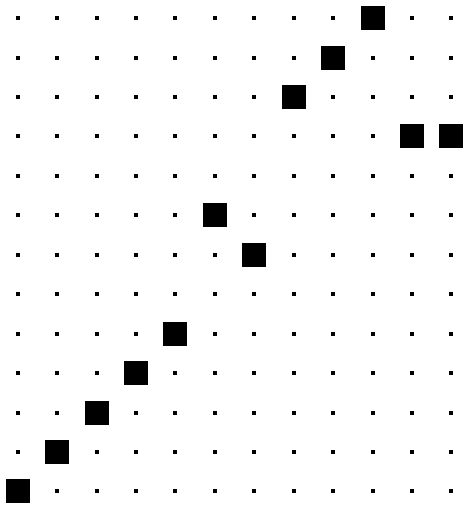

Figure 1.2: Word-to-word alignment.

$a_1^J := a_1 ... a_j ... a_J$ for each sentence pair $(f_1^J; e_1^I)$:

$$p(f_1^J|e_1^I) \;=\; p(J|I) \cdot \sum_{a_1^J} p(f_1^J, a_1^J|e_1^I)$$

where we have included a sentence length probability $p(J|I)$. To structure this probability distribution, we factorize it over the positions in the source sentence and confine the alignment dependencies to a first-order dependence:

$$p(f_1^J|e_1^I) \;=\; p(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} [p(a_j|a_{j-1}, I, J) \cdot p(f_j|e_{a_j})] \;.$$

Here, we have the following probability distributions:

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

By making the alignment probability $p(a_j|a_{j-1}, I, J)$ dependent on the jump width $a_j - a_{j-1}$ instead of the absolute positions a_j, we obtain the so-called homogeneous hidden Markov model, for short HMM ([Vogel et al. 1996]).

We can also use a *zero-order* model $p(a_j|j, I, J)$, where there is only a dependence on the *absolute* position index j of the source string. This is the so-called model IBM-2 ([Brown et al. 1993a]). Assuming a uniform alignment probability $p(a_j|j, I, J) = 1/I$, we arrive at the so-called model IBM-1.

These models can be extended to allow for source words having no counterpart in the translation. Formally, this is incorporated into the alignment models by adding a so-called 'empty word' at position $i = 0$ to the target sentence and aligning all source words without a direct translation to this empty word.

In [Brown et al. 1993a], more refined alignment models are introduced by using the concept of fertility. The idea is that often a word in the target language may be aligned to several words in the source language. This is the so-called model IBM-3. Using, in addition, first-order alignment probabilities along the positions of the source string leads us to model IBM-4. Although these models take one-to-many alignments explicitly into account, the lexicon probabilities $p(f|e)$ are still based on single words in each of the two languages. Search algorithms based on the basic alignment models are described in [Tillmann et al. 1997a], [Nießen et al. 1998], and [Ney et al. 2000].

1.1.3 Current Systems

The first statistical machine translation system was developed at the IBM research center [Brown et al. 1990], [Brown et al. 1993b], [Berger et al. 1994]. A stack decoder was used, but no detailed description can be found in the publications of this research group.

A stack decoder has also been used in [Wang and Waibel 1997] and [Wang 1998].

A different approach to decoding has been developed by [Tillmann et al. 1997a]. Here, a Dynamic Programming approach has been chosen for search. As a rather strong restriction, only monotone alignments were considered during search. That is to say, source sentence and target sentence had essentially the same word order except for the cases where several consecutive source words are translated by one target word and where one additional word is inserted into the target sentence which is not aligned to any of the source sentences. To alleviate this strong restriction a preprocessing step was applied to the source sentence during which the word order was changed to bring it more closely to the word order which is to be expected for the target sentence.

In subsequent work the search algorithm was extended to handle a restricted number of word re-orderings. Different reordering strategies have been developed and compared experimentally [Tillmann 2001].

A search strategy with poses less restriction on word order has been presented in and [Nießen et al. 1998]. The source sentence is constructed word by word and each word is aligned to one or more of the source sentences. To make sure that all words in the target sentence are translated and not only the easy ones, i.e. those with a high lexicon probability, a number of coverage constraints had been incorporated into the search algorithm.

A general shortcoming of the baseline alignment models is that they are mainly designed to model the lexicon dependencies between single words. In ([Och et al. 1999a]) word groups or phrases rather than single words were chosen as the basis for the alignment models . In other words, a whole group of adjacent words in the source sentence may be aligned with a whole group of adjacent words in the target language. As a result, the context of words is taken into account in an explicit manner, and the differences in local word orders between source and target languages is encoded in the alignment information stored with those word group to word group associations.

1.2 Example Based Translation

Example based machine translation (EBMT) is generally traced back to [Nagao 1984]. In this short paper the basic idea of EBMT is formulated: construct the translation for a new sentence from translations encountered earlier - the examples. These examples can be anything, ranging from simple source sentence - target sentence pairs to pairs of partial parse trees which can be used in a transfer based translation approach. Actually, in [Nagao 1984] partial parse trees were stored as examples which were then used to construct parse trees for unseen sentences.

Example based machine translation in its simples form is often referred to as translation memory. In this case, sentences and their translations, originating from human translators are stored. Instead of complete sentences shorter phrases can be made the building block from which to construct new translations [Brown 1996].

Normally, search in the given set of source sentences and their translations is not restricted to exactly matching segments. To get more coverage on new sentences error tolerant matching is used. That is to say, a small number of insertions, deletions, and substitutions is accepted. If those errors are on content words then the translation returned from the system will show semantic errors.

By using word categories the approach has been extended to generalized example based machine translation [Brown 1999].

1.3 Translation with Transducers

Finite state methods have a long tradition in natural language processing. They have been applied to different areas like part of speech tagging, morphology, noun phrase detection and parsing. For translation especially two transducer based approaches have been developed, namely the subsequential transducer approach and the head transducer approach.

1.3.1 Subsequential Transducer

A translation approach which is very close to example based machine translation has been developed based on subsequential transducers [Vidal 1997, Amengual et al. 2000]

Subsequential transducers can be easily constructed from a bilingual corpus: first, a tree transducer is constructed as a prefix tree over the source sentences of the corpus. The translation of each sentence is attached as output of the corresponding final state. Then, in a second step, prefixes of the translations are pushed as far as possible towards the root of the tree. If to pathes starting from a state s generate the two target sequences $e_1, ..., e_i, e_{i+1}, ..., e_I$ and $e_1, ..., e_i, e'_{i+1}, ..., e'_{I'}$ with the common prefix $e_1, ..., e_i$ then this common prefix can already by written to the output on the state transition to state s.

Characteristic features of this approach are:

- For each input sequence there is exactly one output sequence. That is to say, different translations of one sentence - something which frequently can be observed - is not captured by this approach.
- As the translation of a given input sequence is deterministic no transition weights are required. This is true for translation of text input. However, in the case of speech input, where a speech recognizer produces a large number of different word sequences a weighted transducer has been used to play the role of the language model for the speech recognizer.

The subsequential transducer approach has been extended in two ways:

1. Word categories were used for generalization [Amengual et al. 1997]. By replacing words by a category label the amount of training data required could be reduced significantly. First, a transducer is constructed from the labelled bilingual corpus. Second, a number of specialized transducers, one for each category is constructed. Finally, these specialized transducers are inserted into the master transducer to replace all labelled transitions. As this procedure introduces nondeterminism the resulting transducer is again made deterministic.
2. OSTIA: to avoid over-generalization when coupling translation with subsequential a n-gram language model for the source language is used to give preference to those word sequences which are typical for the task. Also, a n-gram language model for the target language is added to produce translations with higher syntactical correctness.
3. Error correcting parsing [Amengual and Vidal 1998]. This is very similar to allowing for insertions, deletions, and substitutions in the case of example based machine translation. Again, the motivation for this extension is to gain greater generalization power.

1.3.2 Head Transducer

Alshawi and co-workers published a number of papers on a translation approach which is intended to capture the hierarchical structure of language and shows some parallels to head driven phrase structure grammars [Alshawi 1996, Alshawi et al. 1998]. This approach can be characterized in the following way: a large number of probabilistic finite state

transducers is associated with head words. Each transducer transcribes the transduction from a head word in the source language to the head word in the target language as well as the transduction of the immediate dependents of the head words.

In one important way these head transducers differ from standard transducers. Whereas for a standard transducer the positions on input and output tape are implicitly changed by reading and writing symbols the head transducer explicitly encodes chances in position in the transitions of the transducer. That is to say, a transition from state s_1 to state s_2 is not only labelled with a symbol f from the input alphabet and a symbol e from the output alphabet but also with two move-position instructions m_f and m_e which give the number of positions to move forward or backward on input and output tape respectively before reading/writing the next symbol.

As the moving of the writing position may lead to a position on the tape already occupied by a symbol written at some earlier state, a heuristics is added: continue to move in the same direction until a free position is found. And any holes which remain on the output tape after reading the complete input sequence are removed in a postprocessing step.

This explicit encoding of positions gives an easy and concise way of introducing word reordering into the transducer. Whereas the subsequential transducer has to postpone writing the output until the word aligned to the first word in the target sentence has been read, the head transducer can write the output synchronously to reading the input: words can be pushed further down the line leaving some gaps to be filled in a later state of processing the input.

Head transducers can be constructed from bilingual alignment and the weights of the state transitions can be collected from such an alignment [Alshawi et al. 1998].

1.4 Cascading Finite State Models

In [Brants 1999a] and [Brants 1999b] cascaded Markov models are used for partial parsing of context-free structures. Each layer is represented by its own Markov Model, and output of a lower layer is passed as input to the next higher layer. A parse tree is constructed layer by layer and for each layer a Markov Model determines the best set of phrases. These phrases are used as input for the next layer.

The Markov Models are used only to filter the best parsing hypotheses on each level. A state on a given level emits a sequence of grammatical tags according to the probabilities of a context free grammar.

The cascaded Markov models as well as the parameters for the context free grammar are trained from annotated data.

1.5 Grammar based Approaches

A different line to direct statistical based machine translation has been taken by Wu in a number of papers [Wu 1994, Wu 1995b, Wu 1995a, Wu 1996]. In this approach bilingual grammars are used. Parsing an input sentence means at the same time writing a target sentence. In this bilingual grammars share some common features with the transducer based approaches.

A bilingual grammar is a grammar where each rule has two right hand sides. For example:

$$\begin{aligned} NP &\rightarrow DetNN\#DetNN \\ NP &\rightarrow DetADJNN\#DetNNADJ \end{aligned}$$

Word reordering between source and target language is encoded in those grammar rules. In the given example the second rule states that in the target language the adjective comes after the noun whereas in the source language the adjective precedes it. Adding probabilities to the rules converts the grammar into a stochastic context free bilingual grammar.

Training of a bilingual grammar in Chomsky Normal Form on a bilingual corpus can be done using an extension of the well-known Inside-Outside algorithm [Wu 1996]. The time complexity is then $O(J^3I^3)$ where J is the sentence length of the source sentence and I is the sentence length of target sentence.

The simplest bilingual grammar possible is a grammar which uses only one nonterminal. That is to say all rules are of the form:

$$\begin{aligned} A &\rightarrow AA\,\#\,[\,AA\,] \\ A &\rightarrow AA\,\#\,<\,AA\,> \\ A &\rightarrow f\,\#\,e \\ A &\rightarrow f\,\#\,\epsilon \\ A &\rightarrow \epsilon\,\#\,e \end{aligned}$$

The brackets in the first rule mean that the two segments in source and target language are aligned parallel whereas in the second rule the segments are aligned at cross, i.e. the succession of the two segments in the target sentence is inverted with respect to the source sentence. Hence the name 'Stochastic Inversion Bracketing Grammar'.

Chapter 2

Scientific Goals

The main goal of this work is to combine aspects of different machine translation approaches: statistical, example based, finite state technology, bilingual grammars. The resulting translation system is tested in different applications ranging from very small data speech translation tasks to very large data text translation tasks.

In detail the contributions of this thesis are:

- Developing a framework for machine translation based on cascaded finite state transducers. This allows to incorporate word and phrase-level translation pairs extracted automatically from bilingual corpora, as well as specialized transducers which are manually or semi-automatically constructed to embody specific linguistic or domain-specific phenomena.
- Formulating this approach within a Bayesian statistical framework where the transition probabilities in the transducers can be identified as language model and translation model probabilities.
- Developing a training algorithm for this cascaded transducer translation approach. Training of the transducers requires an extension of the standard word-based alignment models to align graphs instead of sentences.
- Development of a search algorithm which constructs a hierarchical translation graph by applying the cascade of transducers. The language model is used to find the best path in the translation graph.
- A method of bilingual labelling on the basis of cascaded transducers and an alignment model, thereby guaranteeing that number and type of the category labels in source and target sentence are equal. This is also the basis for constructing the top-level transducer.
- Applying statistical machine translation to the situation where only a small amount of bilingual training data is available. Very often SMT is criticized as being applicable only when large amounts of training data is available. When this is not the case grammar-based and knowledge-based systems are the only option.

- Applying statistical machine translation to large data applications. In this situation phrase-to-phrase translations extracted from the training data become the most important knowledge source. This has been the stronghold of example based machine translation.

Chapter 3

Translation with Cascaded Transducers

A central issue in data driven approaches to natural language processing is the question how good the method generalizes to new data. A system built on a given corpus of data, called the training corpus, performs generally well on that data or on data which is very similar to it. To have good performance on new data, generally called test data, some capability of generalization is required. Different types of systems employ different strategies of generalization. These will be shortly reviewed in the first section.

In this work generalization is based on cascaded finite state transducer. The first step is to transform a bilingual corpus into a finite state transducer. The starting point for generalization is similar to generalized EBMT, i.e. generalizing source sentence - target sentence pairs to translation patterns by using categorization. However, this is extended by using dedicated bilingual grammars. The relation between the current work and other data driven approaches to machine translations will be taken up again towards the end of this chapter.

This chapter gives an informal description of the cascaded transducer approach to translation. The formalization, i.e. grounding this approach in the Bayesian framework of statistical machine translation will follow in the next chapter.

3.1 Generalization

3.1.1 Approximative matching

In example based machine translation generalization is achieved through error tolerant matching. Usually, the well-known edit distance is used to find the best match. The more insertions, deletions and substitutions are allowed the more new sentences can be matched to one of the stored sentences. However, translation quality will suffer the higher the number of errors in matching. This is especially the case when there is a mismatch

on content words. To alleviate the problem, weighted edit distance can be used, to allow for mismatches on filler and function words but avoid mismatches on content words.

One possibility to cope with matching errors for content words is to use a lexicon to correct them. For substitutions this is straightforward. If for a sentence $f_1...f_j...f_J$ the best match in the database is $f_1...f'_j...f_J$ then the translation e' of f'_j has to be replaced by a translation e of f_j in the translation $e_1...e_I$. This can normally be done without explicit alignment information, just by using the lexicon. The situation is similar in the case of deletions, although deleting a word from the reference translation may harm fluency. For insertions, however, additional information is required from which the best position for the inserted word in the target sentence can be extracted. This can be an explicit word-to-word alignment or a language model.

3.1.2 Segmentation

The longer a sentence the less likely it will match other sentences, even if error tolerant matching is allowed. To improve coverage shorter segments should be used. That is to say a new sentence may be covered by a number of segments from the example database. The translation is then the concatenation of the segment translations. A necessary condition is that

1. each segment is a complete source – target pair;
2. the segment translations can be concatenated.

Segmentation of the training corpus can be

- hard: a sentence pair is split into a number of segment pairs which are then used to translate new sentences;
- soft: no explicit segmentation is used to built the database, instead, each sequence of words in the corpus can be used to match part of a new sentence. The translation of that segment has to be found in the parallel sentence from the bilingual corpus. This is possible using alignment information, as in the alignment template approach [Och and Weber 1998] or on the basis of lexical information as in the Pangloss system [Brown 1996].

Segmentation has also been used in statistical translation systems. In [Och et al. 1999b] the sentence pairs in the training corpus have been segmented, but the effect of segmentation has not been studied in a systematic way. For translation long sentences are segmented at sentence marks or, in case of speech recognition input, at prosodic boundaries [Vogel et al. 2000b].

In [Wang 1998] segmentation is based on a dialog model. Utterances or parts of longer utterances are classified as greetings, suggestions, requests, rejections, etc. A hidden Markov model is trained on annotated data to recognize these dialog acts. Longer sentences are then segmented at the boundaries between dialog acts and translated segment for segment.

3.1.3 Labelling

Instead of correcting substitution errors via an additional lexicon, as described above, words can be clustered into word classes. The translation examples are transformed into translation patterns, where some of the words are replaced by category labels. Translation is then done by the following processing steps: Replace words by their category label; find best matching translation pattern; insert translation of replaced words into the appropriate positions.

An important issue in using category labels for generalization is to have the same number and types of category label in source and target sentence. Otherwise, translated segments will be lost. This requires bilingual labelling, that is to say, labelling is not performed on the two parts of the bilingual corpus independently but jointly. For the automatically generated word classes through clustering this has been done by [Och 1999] and [Amengual et al. 1997]. Often it is desirable to extend labelling beyond replacing individual words with labels.

A different method which is based on applying category transducers to both source and target sentences of a bilingual corpus and aligning the resulting graph structures will be presented in 6.

3.1.4 State Merging

Starting from a tree transducer, which represents the translation database, a generalizing transducer can be constructed by merging states. This transforms the tree into a graph which contains more pathes than the original tree. For translation such an approach has been chosen by [Amengual et al. 2000]. Merging of two states q and q' is possible if they are equivalent with respect to some criterion. The criterion is that each (f, e) sequence which can be generated from state q can also be generated starting in state q' and vice versa.

3.1.5 Smoothing

A widely used approach to achieve generalization is smoothing the probability distributions. In natural language processing there is a large body of literature on this issue. A typical example is the formulation of robust language models for speech recognition. Different smoothing methods have been developed, like linear interpolation and backing off. The basic idea is to resort to less specific models in those cases where the specific model would not be applicable. If for example a tri-gram language model is used and a triplet (w_1, w_2, w_3) has not been seen in training data a backing-off to a bi-gram, uni-gram or even zero-gram is applied.

The question is, how smoothing can help to achieve robust translation. Let f be a source word in a test sentence which has not been seen in training, and let e be the only admissible translation for f. We assume that these words are in the vocabularies V_F and V_E respectively. This can happen when - as in the case of the Verbmobil corpus -

the vocabularies of the speech recognition systems contain words not seen in the training corpus. There are now two possibilities: e is either known or not known from training.

- Case 1: e was not seen in training

  In that case a language model without smoothing would give the probability $p(e|h) = 0$ for any history h, even for the uni-gram language model. But smoothing the language model would allow to use e in the translation. For the translation model without smoothing we also have $p(f'|e') = 0$ if $f' = f$ and $e' = e$. Smoothing the lexicon probabilities means to have $p(f|e) > 0$ for all e and f. For a uniform distribution $p(f|e) = 1/|V_F|$ we have an equal lexicon probability for all target words e' not seen in the training data. That is to say, that neither smoothing in the language model nor smoothing in the translation model would help to select the correct target word.

  With a smoothing method which sets $p(f|e)$ proportional to the count of f the situation would be the same.

- Case 2: e was seen in training

  If now e has been seen in training data, for example by using a large background corpus for estimating the language model probabilities, the language model would allow to use e in the target sentence. Still, the lexicon probabilities $p(f|e')$ would be equal for all e' not seen in the training of the translation model and larger than $p(f|e'')$ for all e'' seen during training. From this follows that in this case also the translation of f would be selected solely on the basis of the language model.

In both cases smoothing does not really help in coming up with a good translation for a word not seen in the training corpus. Generally speaking: whenever the language model probabilities are unreliable due to sparse training data then the lexicon probabilities are also unreliable and vice versa. This situation is different from speech recognition where a beneficial duality between high frequency function words and low frequency content words is observed. The short function words which are predicted unreliably from the acoustic models are on the other side predicted reliably from the language model. And the low frequency but longer content words, although poorly predicted from an n-gram language model can reliably be recognized on the basis of the acoustic models.

3.2 Generalization through Cascaded Transducers

3.2.1 Translation Memory as Finite State Transducer

A translation memory - as it is generally understood - is simply the collection of source – target sentence pairs, i.e. just a bilingual corpus. Building a prefix tree over one part of the bilingual corpus and attaching the corresponding sentences from the other half of the corpus to the final states of this tree results in a tree-transducer which represents exactly the training corpus. Therefore, the translation memory approach can easily be

```
okay dann                          # well then (1)
okay dann Montag                   # okay on Monday then (2)
okay dann Montag bei dir           # okay then on Monday at
                                     your place (3)
okay dann ist das vereinbart       # well that's settled then (4)
okay dann ist das vereinbart       # okay that's fixed then (5)
okay dann machen wir das           # well let us do that (6)
okay dann machen wir das morgen    # okay tomorrow then (7)
okay dann machen wir den           # okay let us fix that date (8)
Termin fest
```

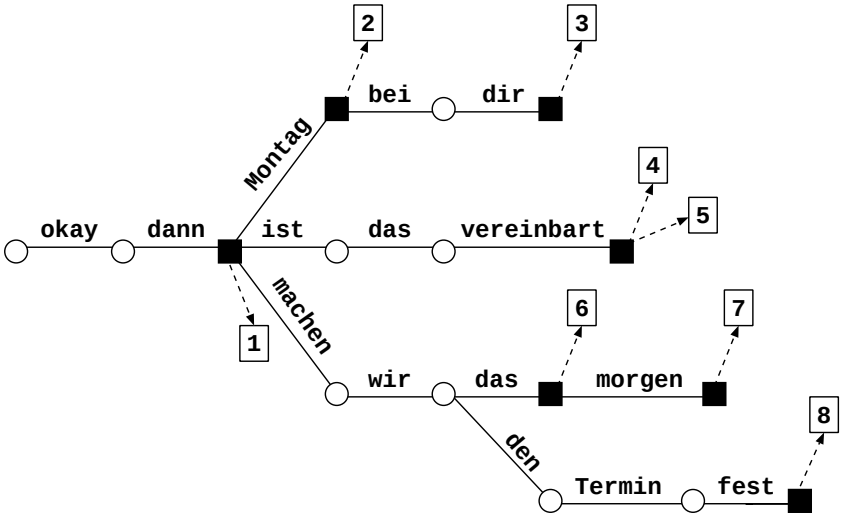

Figure 3.1: Tree-transducer build from bilingual corpus

reformulated within the transducer-based approach to translation. Later it will be shown how error-tolerant matching, which is the preferred method to achieve generalization with a translation memory, can also be incorporated into the transducer-based approach.

The conversion from a simple translation memory into a tree-transducer is shown in Figure 3.1 for a small sample corpus. This example shows a German–English toy corpus and the resulting tree-transducer constructed as prefix tree over the German sentences. Emitting states are shown as dark squares. Emitted word sequences are only indicated by their number. In Section 3.3 more details on the transducers will be given.

Notice, that a similar tree can be constructed over the English sentences.

3.2.2 Hierarchical Translation Memory and Cascaded Transducers

Categorization, i.e. replacing individual words by category labels, makes translation a two-level process. Within a translation memory this means bilingual categorization, i.e. for each word in the source sentence, which is replaced by a category label, the corresponding word in the target sentence is also replaced by the same category label.

This approach can be extended into two directions:

- replace multi-word segments by category labels;

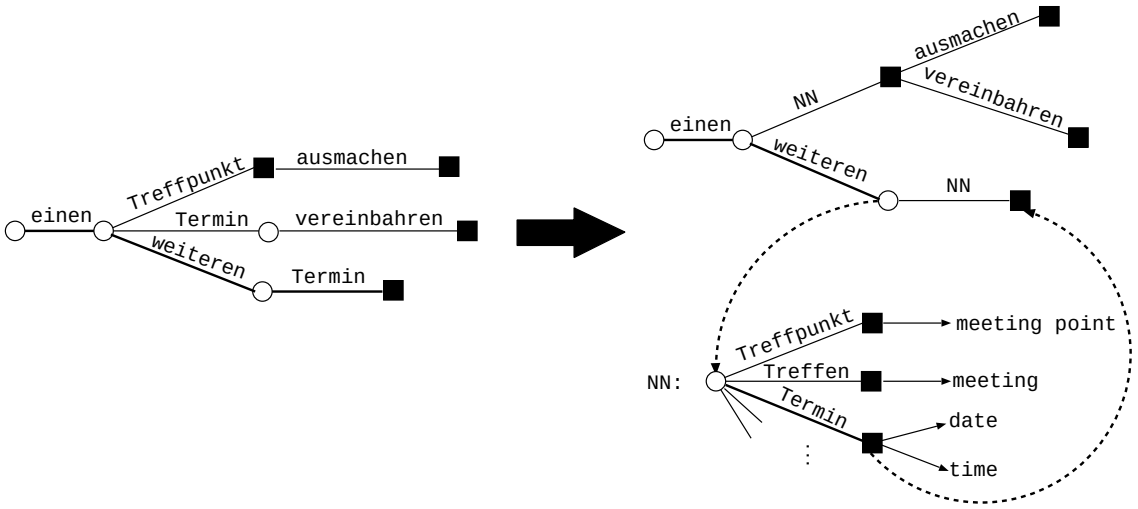

Figure 3.2: Two-level translation with category transducer.

- allow for hierarchical replacement.

The first task does not introduce any new aspect. Here, the question is only how to select those multi-word segment. Candidates are e.g. English word sequences corresponding to German compound nouns.

Now, we consider hierarchical replacement. By this we mean translation patterns like 'NP # DET NN # DET NN'. That is to say a sequence of labels (and words) is replaced by one higher-level label. If this is recast into the language of transducers this means to replace one overall transducer, where the transitions are labelled with words, by a cascade of transducers, where some of the transitions are now labelled with category labels. This is shown in Figure 3.2.

3.3 Different Views on Transducers

Normally, a transducer is understood as a finite state device where the transitions, i.e. the edges between the states are labelled with a symbol from the input vocabulary and one or several symbols from the output vocabulary. A different type of transducer is given when the output is not associated with the transitions but with the states. Such finite state machines are the well-known Hidden Markov Model used in speech recognition or part of speech tagging. In the literature the two types of transducers are known as Mealy and Moore machines.

In the current work the different views or ways of speaking are used interchangeably to allow for presentation when focusing on different aspects of the model. Generally, the view taken on transducers is to have emitting states. Even more restricted, only final states emit the sequence of output symbols, as shown in Figure 3.3a. Such a transducer is equivalent to a transducer with only one final state which is constructed from the first one by adding one transition from a (formerly) final state to the new final state. This

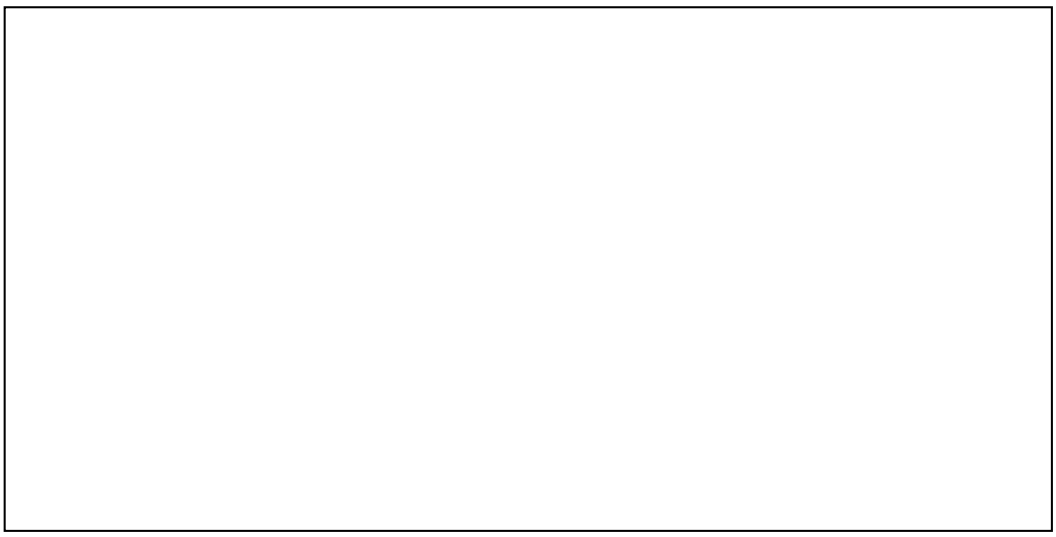

Figure 3.3: Equivalent transducers: emitting states vs. output from the transitions.

transition reads no symbol but emits the complete sequence. All other transitions emit ϵ, as in the first case. This transducer is shown in Figure 3.3b.

The different forms of transducers are equivalent in that they read and generate identical source and target strings. This equivalence should be kept in mind. Normally, the first view is taken. We speak of emitting states and associate the translation probabilities with the emission probabilities. But when traversing cascaded transducers, i.e. embedding the traversal of a specialized transducer into the traversal of the higher level transducer the second picture might be more appropriate. The specialized transducer has then only one start and one final state. This gives a simpler picture of moving into and out of the specialized transducer. When introducing the error model, an even more detailed view is appropriate: the emitted word sequence is also split into a number of transitions as shown in Figure 3.3c. This is then the most complete view on the transducers as used in this work.

3.4 Model, Training, Search: An Overview

In this section an overview of the translation approach based on cascaded transducers will be given. Details will be presented in subsequent chapters.

The three tasks in building a statistical machine translation system are:

- Construction of a model which in this case will be a probabilistic translation model based on cascaded finite state transducers;
- Estimation of the model parameters, i.e. the probability distributions from a given corpus;
- Formulation of an efficient search strategy.

The central goal in this work is to combine ideas from the statistical approach to machine translation and example based machine translation. On one side, longer word sequences are used in the translation process, on the other side probabilities will be attached to these sequences. The probabilities are estimated from a bilingual corpus and used in the search process to find the translation with the highest probability.

3.4.1 The Model

The translation model is a cascade of weighted finite state transducers. The weights attached to the transitions will be interpreted as language model and string translation probabilities as used in statistical translation approach presented in 1.1.1.

Categorization of words and word sequences is used to achieve generalization. This is extended to multi-level or hierarchical categorization similar to chunk parsing and bilingual grammars. This requires the transition to cascaded transducers.

Although hierarchical categorization is the main road to generalization the concept of error tolerant matching as used in example based machine translation is incorporated into the translation model. This is done by adding transitions to the transducers corresponding to insertions, deletions, and substitutions.

3.4.2 Parsing with Transducers

Training the transducers as well as searching for the optimal translation for a given new sentence involves the application of transducers. Therefore, this will be described previous to dealing with the training procedure and the search algorithm.

Application of a transducer means matching part of a given sentence with a word sequence given by a path in the transducer from the start state to some emitting state. Applying a cascade of transducers - starting with the most specific category transducers and working towards the top-level transducer - is then bottom up parsing. This will be done in a way which is essentially a chart parsing algorithm. Specific to the algorithm presented in 5 is that the sentence to be parsed is viewed as a graph. For closed hypotheses new edges are added to this graph.

Efficient matching of word sequences in the source sentences requires a restructuring of all transducers: transitions are not labelled with target words but with source words and target word sequences are emitted and used as partial translations from which an optimal translation is constructed.

3.4.3 Parameter Estimation

The transition and emission probabilities of the transducers are estimated from a bilingual corpus using an extended alignment. The standard alignment model presented in 1.1.2 is extended in the following directions:

- Not only word-to-word alignment is possible, but also word group to word group. That is to say, not each word in the source sentence is required to have a direct alignment to a target word.
- An alignment on the basis of a cascade of transducers is a hierarchical alignment.

The algorithms implemented to find the optimal alignment applies the cascade of transducers to the source sentences of the bilingual corpus and the reverted transducers to the target sentences. As has already been mentioned, a sequence of translation patterns can be transformed into a tree-transducer as a prefix tree over the source parts as well as a prefix tree over the target parts - the reverted transducer.

So, from source and from target sentences graphs are constructed by applying the transducers resp. reverted transducers. The extended alignment model is then used to find the best alignment between the two graphs, guaranteeing that it is compatible with a path through the cascade of transducers.

3.4.4 Search for the Best Translation

The final task is to develop an efficient search algorithm which find the best translation for a given source sentence. Parsing a sentence with a cascade of transducers, thereby generating the translation graph, is already the first part of the search algorithm. But now, the language model is used as a second knowledge source. The language model probabilities are taken into account while running left to right over all pathes in the translation graph. This is similar to language model re-scoring in speech recognition. The best translation is then simply the target language word sequence read of the resulting optimal path.

Chapter 4

Cascaded Transducers as Probabilistic Translation Model

In this chapter the translation model based on cascaded weighted finite state transducers will be described in detail. Firstly we will show how the transducer approach can be formulated in the Bayesian framework of statistical machine translation. Secondly we will introduce categorization, and we will move on to a hierarchical alignment model through cascaded transducers. Finally we will formulate an error model to capture approximate matching within the transducer framework.

4.1 Probabilistic Transducer in a Bayesian Framework

Let us recall the basis of standard statistical translation starts from Bayes' decision rule:

$$\begin{aligned} \hat{e}_1^I &= \underset{e_1^I}{\text{argmax}} \{Pr(e_1^I|f_1^J)\} \\ &= \underset{e_1^I}{\text{argmax}} \frac{\{Pr(e_1^I) \cdot Pr(f_1^J|e_1^I)\}}{Pr(f_1^J)} \quad . \end{aligned}$$

$Pr(e_1^I)$ is the language model of the target language, whereas $Pr(f_1^J|e_1^I)$ is the string translation model. The argmax operation denotes the search problem which will be discussed in Chapter 7.

For a given sentence f_1^J the denominator does not influence the result. Therefore:

$$\hat{e}_1^I = \underset{e_1^I}{\text{argmax}} \{Pr(e_1^I) \cdot Pr(f_1^J|e_1^I)\} \tag{4.1}$$

$$= \underset{e_1^I}{\text{argmax}} \; p(e_1^I, f_1^J) \quad . \tag{4.2}$$

For the reverse translation we have:

$$\hat{f}_1^J = \underset{f_1^J}{\operatorname{argmax}} \{Pr(f_1^J) \cdot Pr(e_1^I|f_1^J)\} \tag{4.3}$$

$$= \underset{f_1^J}{\operatorname{argmax}} \; p(e_1^I, f_1^J) \quad . \tag{4.4}$$

So the joint probability could be used for both translation directions. However, it is often an advantage to use 4.1 instead of 4.2, as the two models, the language model $p(\mathbf{e})$ and the translation model $p(\mathbf{f}|\mathbf{e})$ can be trained separately.

The two probability distributions can be modelled directly with a tree-transducer which is constructed as a prefix tree over the target strings.

- Language Model:
  Each final state is uniquely characterized by a word sequence $e_1...e_I$, and the probability of reaching this state is given by:

$$p(\mathbf{e}) = p(e_1^I) = \prod_{i=1}^{I} p(e_i|e_1..e_{i-1}) \quad . \tag{4.5}$$

  The language model probabilities are given by the product of the transition probabilities in the transducer. Of course this language model allows only the strings seen in the training data or any prefix thereof. Again, generalization becomes an issue.

- Translation Model:
  The lexicon model in this simplest case is given by the relative frequency of the source string with respect to a given target string:

$$p(\mathbf{f}|\mathbf{e}) = \frac{N(\mathbf{f}, \mathbf{e})}{\sum_{\mathbf{f}'} N(\mathbf{f}', \mathbf{e})} \quad . \tag{4.6}$$

So far no additional structure for the language model and the translation model has been introduced. This is necessary to allow for generalization from training data to unseen data. In the standard alignment models the mapping from word sequences $\mathbf{f}$ to word sequences $\mathbf{e}$ is modelled through word-to-word alignments with the additional restriction that each word in the source sentence is aligned to exactly one word in the target sentence. This is already a severe restriction of the general alignment concept which allows for an arbitrary mapping from source to target sentences.

Using transducers as translation model allows for $n : m$ alignment. Using categorization and finally cascaded transducers will lead to a hierarchical alignment which allows for word-to-word alignments, but also for aligning entire word groups.

4.2 Generalization through Categorization

Now we have to turn to generalizing the transducer. In the first step we will introduce word classes. Word classes can be parts of speech, automatically learned word classes or just specifically chosen categorization for only part of the vocabulary, e.g. numbers and proper names. Categorization is frequently used as a means of generalization. In a translation memory system this generalizes the database: a translation template can be expanded to a number of translation pairs by inserting all words from the category. In terms of transducers this means that some transitions are not labelled with proper words but with category labels.

4.2.1 Example

The transition from using one overall transducer to using cascaded transducers was already displayed in Figure 3.2. On the left hand side a very simple transducer was given which has been constructed from a bilingual corpus consisting of only four sentence pairs. The right hand side of the diagram showed how this transducer is transformed into two transducers, a top-level transducer and a special transducer, which in this example collected the nouns.

We calculate the probability for 'another date' as the translation of 'einen weiteren Termin'. In the left transducer the overall probability is given by following the path, multiplying all transition probabilities and the emission probability in the final state. For the two-level transducer we also have to multiply all transition and emission probabilities along the chosen path. In this case the path runs over 'einen' 'weitern' 'NN' 'Termin'. The probability to switch to the specialized transducer NN is set equal to the transition probability over the edge labelled with 'NN'. We have two emitting states: one in the NN-transducer and one in the top-level transducer. In total we get:

$$\begin{aligned} Pr(\mathbf{e}|\mathbf{f}) &= Pr(\mathbf{e})Pr(\mathbf{f}|\mathbf{e}) \\ &= p(\text{einen}) \cdot \\ &\quad p(\text{weiteren}|\text{einen}) \cdot \\ &\quad p(\text{NN}|\text{einen weiteren}) \cdot \\ &\quad p(\text{Termin}|\text{NN}) \cdot \\ &\quad p(\text{date}|\text{NN}, \text{Termin}) \cdot \\ &\quad p(\text{another NN}|\text{einen weiteren NN}) \quad . \end{aligned}$$

The transition in the top-level transducer for the edge 'NN' is $p(\text{NN}|q_i) \cdot p(\text{Termindate}|\text{NN})$. The first factor is the probability of entering the sub-level transducer, the second factor is the probability of one chosen path in that transducer.

4.2.2 Formalization

To distinguish plain word sequences from sequences which contain category labels we use $\tilde{\mathbf{f}}$ for the latter. The target sequence with labels is denoted by $\tilde{\mathbf{e}}$.

We write:

$$Pr(\tilde{\mathbf{f}}, \tilde{\mathbf{e}}) = Pr(\tilde{f}_1 = f_1, ..., \tilde{f}_j = L, ..., \tilde{f}_J = f_j, \tilde{e}_1 = e_1, ..., \tilde{e}_i = L, ..., \tilde{e}_I = e_I) \quad . \tag{4.7}$$

Now we have for the language model:

$$Pr(\tilde{\mathbf{e}}) = Pr(\tilde{e}_1^I) = \prod_{i=1}^{I} Pr(\tilde{e}_i | \tilde{e}_1 ... \tilde{e}_{i-1}) \quad . \tag{4.8}$$

We define:

$$p(\tilde{e}_i | \tilde{e}_1 ... \tilde{e}_{i-1}) = \begin{cases} p(e_i | \tilde{e}_1 ... \tilde{e}_{i-1}) & \text{for} \quad \tilde{e}_i = e_i \\ p(L_i | \tilde{e}_1 ... \tilde{e}_{i-1}) & \text{for} \quad \tilde{e}_i = L_i \end{cases} .$$

$$p(e_i | \tilde{e}_i) = \begin{cases} p(e_i | L_i) & \text{for} \quad \tilde{e}_i = L_i \\ 1 & \text{for} \quad \tilde{e}_i = e_i \end{cases} .$$

With

$$p(\tilde{e}_i | ...) \cdot p(e_i | \tilde{e}_i) = p(e_i | ...)$$

we get:

$$\begin{aligned} Pr(\mathbf{e}) &= Pr(e_1^I) \\ &= \prod_{i=1}^{I} Pr(e_i | e_1 ... e_{i-1}) \\ &= \prod_{i=1}^{I} p(e_i | \tilde{e}_1 ... \tilde{e}_{i-1}) \\ &= \prod_{i=1}^{I} p(\tilde{e}_i | \tilde{e}_1 ... \tilde{e}_{i-1}) p(e_i | \tilde{e}_i) \\ &= p(\tilde{\mathbf{e}}) \prod_{i=1}^{I} p(e_i | L_i) \\ &= p(\tilde{\mathbf{e}}) \prod_{\tilde{e}_i = L_i} p(e_i | L_i) \quad . \end{aligned}$$

In the last line the product runs over all positions in $\tilde{e}$ where no proper words but category labels are given.

To calculate the translation probability $p(\mathbf{f}|\mathbf{e})$ we expand the transition with label $L = \tilde{e}_i$ by the transition in the specialized transducer L.

$$p(\mathbf{f}|\mathbf{e}) = p(\tilde{\mathbf{f}}|\tilde{\mathbf{e}}) \prod_{\tilde{e}_i = L_i} p_L(\tilde{f}_j | \tilde{e}_i) \quad , \tag{4.9}$$

with

$$p_L(\tilde{f}|\tilde{e}) = \frac{N_L(\tilde{f}, \tilde{e})}{\sum_{\tilde{e}'} N_L(\tilde{f}, \tilde{e}')} \quad . \tag{4.10}$$

Thus we get the overall translation probability by multiplying the probability of generating the labelled sentence pair with the probabilities of generating word pairs from the labels.

4.2.3 Extended Labelling

It is only a minor step to extend the model to include the labelling of word sequences by only one category label. This is important, as even simple cases like numbers may be one word in one language but several words in another language. German compound nouns are a second example where a sequence of words has to be replaced by one category label, e.g. 'NN → Zahnarzttermin # dentist appointment'. And, of course, this extension allows to replace whole phrases by one category, like 'GREETING → freut mich Sie zu sehen # nice to see you'.

The language and translation probabilities, $p(\mathbf{e})$ and $p(\mathbf{f}|\mathbf{e})$ are calculated exactly as in the former case, by summing over all paths which generate the word sequences $\mathbf{e}$ and $\mathbf{f}$. The transition and emission probabilities along these lines are again multiplied. The only difference is that the segmentation of the sentences on the level of categories is now different from the segmentation on the word level. This makes the notation more involved and cumbersome.

Let $f_1, ..., f_J \# e_1, ..., e_I$ denote a sentence pair (we drop the index for indicating different sentence pairs). Segmentations are made in accordance with the word categories. That is to say, there are segment pairs $f'_j \# e'_i$ which are elements of a (bilingual) category set. Therefore, we can rewrite the segmented sentences as sequences of words and category labels which again are subsumed under the notation $\mathbf{f}$ and $\mathbf{e}$.

$$\tilde{f}_l = \begin{cases} f_l & \text{if no category} \\ C_l & \text{if } f_{j_{l-1}+1} ... f_{j_l} \epsilon \mathcal{C}_l \end{cases} .$$

No words are grouped together which can not be replaced by a category label.

4.3 Generalizing to a Cascade of Transducers

The final step now is to allow for more than two levels of transducers. This is a step towards bilingual grammars as advocated by [Wu 1995b, Wu 1996]. But here not as context free grammars but more restricted as regular grammar. This extension is brought about by allowing that the category transducers themselves have transitions which are labelled not with a proper word e but with a category label C. Therefore, the category transducer has to be treated in the same manner as the top-level transducer in the previous section.

To calculate the language model and translation model probabilities we have, as before, to sum over all pathes which have as their upper projection the word sequence $\mathbf{e}$. The translation probability along one of those paths is the product of all emission probabilities along this path.

4.4 Independence Assumptions

In the small example given earlier a number of independence assumptions have already been mentioned. They are summarized in the following statements:

Independence assumptions for translation model probabilities $p_C(\tilde{f}|\tilde{e})$ in a transducer $\mathcal{T}_C$:

- The translation probability in a category transducer depends only on the final state in that transducer and not on the position where this category transducer is embedded in a higher level transducer.
- The translation probability does not depend on the translations for any category labels C which are encountered in $\tilde{e}$ and have to be translated by category transducer $\mathcal{T}_C$.

Independence assumptions for language model probabilities:

- The language model probability $p_C(\tilde{e})$ for a sequence of words/category labels through a category transducer $\mathcal{T}_C$ depends only on the category label C and the history within this transducer, but not on the sequence of words seen before this category transducer was entered.
- The language model probability $p_C(\tilde{e})$ for a sequence of words/category labels through a category transducer $\mathcal{T}_C$ does only depend on the category labels C seen in the sequence $p_C(\tilde{e})$, but does not depend on the words/category labels seen when traversing the transducer $\mathcal{T}_C$.

These independence assumptions make sense for the translation probabilities. The specialized transducers are chosen along these lines. E.g. in a sentence pair like 'das ist in Ordnung # that is okay 'in Ordnung has to be translated as a fixed expression and not on a word-by-word basis. Therefore, this expression with the proper translation has to be encoded in a transducer. In other sentences 'Ordnung may be taken as a noun which can be translated irrespectively of the surrounding words. (E.g. as 'order' like in 'ein Polynom höherer Ordnung # a higher order polynomial.)

Often, words have more than one translation. Then it is left to the language model to select the best translation on the basis of the surrounding context. This is the reason why the independence assumptions stated for the language model probabilities are more problematic. Many category transducers, especially those for simple labelling, encode translation pairs where the target part is only one or two words long. For those the independence assumptions mean using a uni-gram or bi-gram language model. It is known that longer histories give improved results in statistical machine translation as in speech recognition. However, from the decomposition of the model into a language model and a translation model on the basis of Bayes decision rule 4.1 it does not follow that the language model as given by the cascaded transducers has to be used. Rather, the decomposition has the advantage that the specific structures introduced into the two models

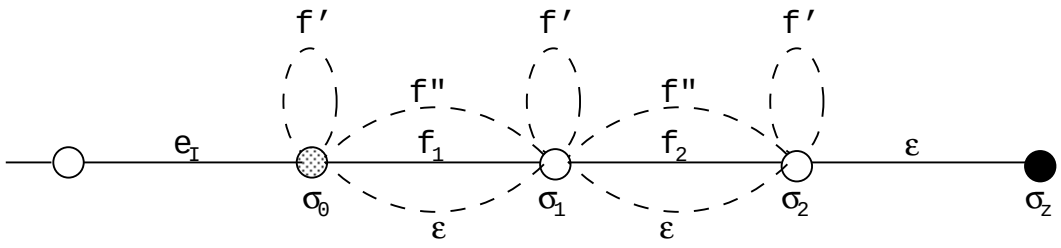

Figure 4.1: Error model.

can be chosen independently of each other. This allows to use a different language model, one which takes longer histories into account, instead of the language model given by the cascaded transducers. In the experiments reported in Chapter 9 a standard n-gram language model trained on plain text will be used.

4.5 Error Model

A frequently chosen approach to generalization is through error tolerant match. In the model proposed here this means that the final state s which is reached through a path $e_1, ..., e_I$ not only emits the sentences $\mathbf{f} = f_1, ..., f_J$ but also sentences $\mathbf{f}'$ which are similar to $\mathbf{f}$. This poses a number of questions:

- What do we mean by 'similar'?
- Which probability do we assign to those sentences $\mathbf{f}'$, i.e. what is $p(\mathbf{f}'|\mathbf{e})$?
- How does this affect the probability $p(\mathbf{f}|\mathbf{e})$?

Assume we have a final state with only one word sequence $f_1, ..., f_J$ being emitted. This has now to by replaced be a graph as given in Figure 4.1. Instead of one transition a number of transitions between the states are possible. For each state except the last one there is:

- One ε-transition to the successor state; this allows for words to be skipped, i.e. for deletions.
- A number of transitions from the state back to itself, labelled with words $f' \in V_F$. These transitions allow insertions of words. (In the figure only one transition is shown.)
- A number of transitions to the successor state labelled with words $f'' \neq f$ from the source vocabulary V_F. These transitions allow for substituting f with some other word f'. (Again, only one transition is shown.)

In the last state, i.e. the one which is reached through the transition labelled with f_J, only transitions back to the state, that is, only insertions are possible.

For a vocabulary V_F these are $|V_F|-1$ substitutions, $|V_F|$ insertions, and one deletion, totaling $2 * |V_F|$ additional transitions for all states except the last one, which has $|V_F|$ additional transitions.

To make these additional transitions possible the probability mass has to redistributed. For each state σ_{j-1} except the final state we set:

$$p(\tilde{f}|\sigma_{j-1}) = \begin{cases} 1-\delta & \text{for } f = f_j \\ \frac{\delta}{2|V_F|} & \text{else} \end{cases}.$$

For the final state only $\delta/2$ is discounted from the prime transition labelled with f_J and redistributed over the $|V|$ insertion transitions.

Normally, we do not want to have insertions, deletions, or substitutions of content words. Therefore, instead of having transitions for all words from the vocabulary only a small number of words is taken. Let $\mathcal{I} = \{f_i\}$ denote the set of words which can be inserted, $\mathcal{D} = \{f_d\}$ the set of words which can be deleted, and $\mathcal{S} = \{(f_k, f_l)\}$ the set of word pairs, where f_k can be substituted for f_l. Then we get:

$$p(\tilde{f} : \sigma_{j-1} \to \sigma') = \begin{cases} 1-\delta & \text{if } \tilde{f} = f_j \wedge s' = s_j \\ \frac{\delta}{N_f} & \text{if } \begin{cases} f_j \in \mathcal{D} & \wedge \ \sigma' = \sigma_j \\ \tilde{f} \in \mathcal{I} & \wedge \ \sigma' = \sigma_{j-1} \\ (f_j, \tilde{f}) \in \mathcal{S} & \wedge \ \sigma' = \sigma_j \end{cases} \end{cases}$$

The case of several sentences $\mathbf{f}_a, \mathbf{f}_b, \mathbf{f}_c, ...$ emitted from one state introduces nothing new. There are two possibilities how to structure the transducer which are shown in Figure 4.2

- The transitions leaving the state reached by e_I are labelled with the first words from the different sentences $\mathbf{f}_a, ...$ For each of these arcs one ϵ-transition and all substitution transitions are added. But there is only one set of insertion transitions.
- A number of ϵ-transition is inserted. The translation probability for each following sentence is then already attached to that ϵ-transition. Starting from states τ_k the situation is then as if only one translation were available.

4.6 Cascaded Transducers and Hierarchical Alignment

In Section 1.1.2 word-based alignment models have been discussed. These alignment models view a sentence as a sequence of words and pay no attention to the internal structure. The structure of language is a hierarchical one which linguists try to capture

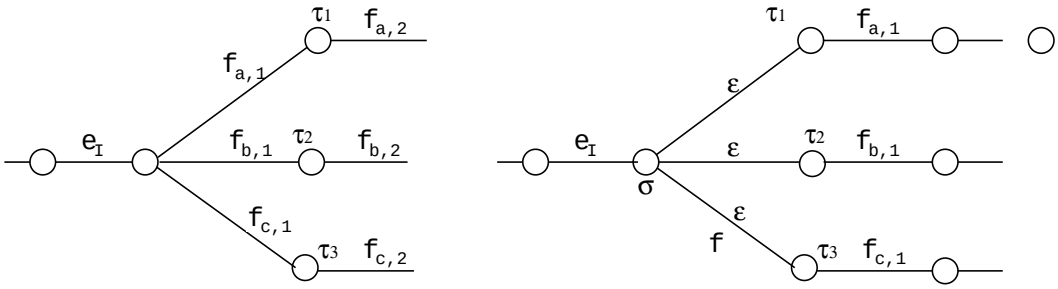

Figure 4.2: Error model.

with their grammar theories. Even shallow parsing strategies allow for several strata to describe the structure of a sentence.

Given a sentence pair in different languages, being translations of each other, both can be structured hierarchically on the basis of same grammar. The alignments should reflect this.

The cascade of transducers can be viewed as a bilingual grammar which imposes a hierarchical structure on both, the source and the target sentence in such a way that corresponding segment are aligned to each other.

4.7 Cascaded Transducers and Bilingual Grammars

In Section 1.5 translation based on bilingual grammars has been described. The cascaded transducer approach shows some similarities.

- The translation patterns encoded in the cascaded transducers are of the same form as the rules of the bilingual grammar.
- Both try to capture - though to different degrees - the recursive structure of language.
- Both approaches require some manual effort for the construction of the grammar respectively the transducers.

The differences:

- Bilingual grammars as formulated by Wu are stochastic context free grammars which are more powerful than the finite state transducers. But as for parsing, where it has been shown that finite state technology is as powerful in practical applications as context free grammars, the same seems to be true for translation.
- The training algorithm for bilingual grammars, a modification of the Inside-Outside Algorithm has a higher computational complexity than the training algorithm presented in Chapter 6.

In a way, translating with cascaded transducers stands in a similar relation to translation with a bilingual grammar as chunk parsing stand to full parsing.

Chapter 5

Parsing with a Cascade of Transducers

Training the transducers as well as translating new sentences on the basis of a cascade of probabilistic transducers requires the application of transducers. In this chapter this common basis for the following two chapters, i.e. Training and Search, will be developed.

The working of the transducers can be described as the construction of a translation graph. That is to say, the sentence to be translated is viewed as a graph which is traversed from left to right. For each matching source pattern, as stored in the transducers, a new edge is added to the graph. The edge is labelled with the category label of the translation pattern. The translation and the translation score are attached to the edge. In this way a graph is constructed, which is called the translation graph. In those cases, where a source pattern has several translations, one edge for each translation is added to the graph.

This idea will be explained in more detail in the subsequent sections.

5.1 Re-Organization of Transducers

5.1.1 Organization of Transducers for Search

The translation model developed in Section 4 is not well suited for the search process. The transducers are essentially prefix trees over the target language. For efficient search it is paramount to have fast access to the translation probabilities required for the translation of a given sentence $\mathbf{f}$.

With the simple tree-structured transducers it is easy to re-organize them as prefix trees over the source language. The translation probabilities $p(\mathbf{f}|\mathbf{e})$ are now attached to each $\mathbf{e}$. It should be stressed that the result is not a proper probabilistic transducer:

- The transition probabilities are dropped altogether, as these transition probabilities would be language model probabilities for the source, not the target language. The

language model probabilities for the target language are provided through a separate language model.

- The sum of the emission probabilities in the final states need not be equal to one. The probabilities $p(\mathbf{f}|\mathbf{e})$ are attached to the emission $\mathbf{e}$ in the final state reached through the word sequence $\mathbf{f}$. Therefore, the emission probabilities are $\sum_n p(\mathbf{f}|\mathbf{e}_n)$.

This should be kept in mind during the following description of the search process: when the term transducer is used actually this re-organized version is meant.

5.1.2 Merging Transducers

In the translation model developed so far each category label gives rise to one transducer which encodes all bilingual translation patterns for this category. Many of these transducers are independent of each other. For a given cascade of transducers the transducers can be grouped into a small number of levels such that the translation patterns encoded in the transducers at each level only use category labels introduced by transducers at lower level.

The parsing algorithm described in this chapter is based on applying each transducer in turn. To increase efficiency all transducers from one level are merged into one transducer. For example, all lexical transducers, i.e. those which have no category labels on the transitions, can be merged into one transducer. And all higher level transducers which are independent of each other can also be merged. Thereby, a large number of transducers can be merged into a small number of transducers, typically three to six.

The different category labels of the different transducers, i.e. the left hand sides of the translation patterns are then attached to the emission. For example, the translation patterns 'NUM # ein # one' and 'DET # ein # a' form only one path in the merged transducer, but now with two emissions: 'NUM # one' and 'DET # a'.

Merging transducers has no effect on the emission probabilities. Normalization in a merged transducer is separate for each category label as required by the model described in Chapter 4.

5.1.3 Cascade of Transducers versus One Transducer

The approach described in this work advocates the use of a cascade of transducers. Other researchers, also using a transducer based approach to translation and categorization as one means of generalization, insert the category transducers back into the top-level transducer. The result is then one transducer where all transitions are labelled with proper words, not category labels. What are the advantages and disadvantages of these two alternatives.

First of all, there is no difference in expressive power. Cascading transducers does not go beyond finite state. Even stronger, the cascade of transducers can be transformed

into an equivalent transducer in the sense, that for each path through the cascade of transducers there is one path through the expanded transducer and vice versa. So, the difference is rather a difference in implementation. But this may make a difference in memory requirement and time complexity of the search algorithm.

Memory Requirement of the Model

It is clear that the size of one all-including transducer is much larger than a cascade of transducers for all but very simple applications. Whereas it is possible to insert category transducers for simple categorization (e.g. names, number) this is no longer the case for truly hierarchical structures like noun phrases, prepositional phrase or complex time and date expressions. In these cases the size of the overall transducers grows considerable.

Each transducer grows by a number of transitions which is given by the product of the transitions with carry a category label with the size of the category transducer for that category. If —T— denotes the size of a transducer, i.e. the number of transitions, N_C the number of transitions in a transducer labelled with category label C, then the size of the expanded transducer $\mathcal{T}_x$ is given by

$$|\mathcal{T}_x| = |\mathcal{T}| + \sum_C (N_C - 1) \cdot |\mathcal{T}_C| \tag{5.1}$$

Going to a cascade of transducers this formula has to be applied to each level. This results in an exponential growth of the transducer.

Memory Requirement during Search

Memory requirement during search means the additional memory necessary to store competing search hypotheses. Therefor, time complexity and memory requirements are closely related as they are both dependent on the size of the search space.

The memory requirement during search depends on the transducer being deterministic or non-deterministic with respect to the source language. If there is only one path through the transducer generating the source sentence $\mathbf{f}$ then it is possible to organize the transducer in such a way that it is deterministic over the source language. Then only one search hypothesis exists at any step in the search procedure. This gives minimal time complexity $O(J)$, with J the length of the sentence to be translation, and also minimal memory requirement.

However, this is an unrealistic scenario as it would restrict the set of sentences which can be translated too strongly. Error tolerant match, a frequently used generalization strategy, introduces the necessity to hold competing search hypotheses in parallel, as does true non-determinism when one sentence can have different translations.

Time Complexity

It has already been mentioned that time complexity essentially depends on the search space. The organization of the translation model into a cascade of transducers versus multiplying it all into one transducer makes no difference with respect to search space. There is only some overhead in propagating partial translations from transducers further down the line towards the top-level transducer. In the worst case this number is $O(J)$, for

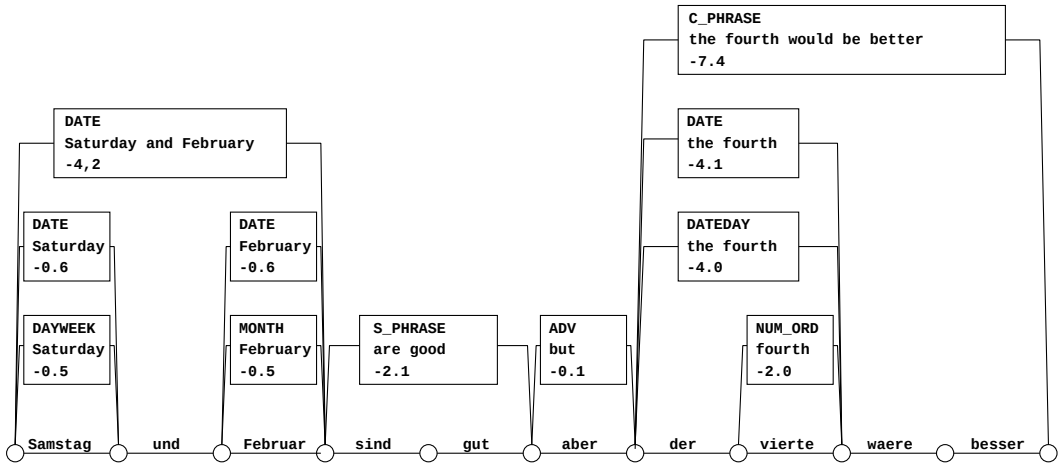

Figure 5.1: Translation example.

a sentence of length J. But on average, a much smaller number of matches from category transducers is observed.

Resume

With respect to memory requirement and time complexity of the search algorithm there is no fundamental difference between applying a cascade of transducers and applying one all-including transducer. One the other side, cascaded transducers have a much smaller static memory requirement. So, for larger vocabularies and larger corpora the cascaded transducer approach is the preferred choice.

5.2 Applying the Transducers: Bottom-up Parsing

5.2.1 Applying one Transducer

The left–right traversal of the graph is organized in such a way that all paths are traversed in parallel and the patterns stored in the transducer are matched synchronously. For each node n and each edge e leading to that node all patterns in the transducer starting with the word or category label of e are attached to n. This gives a number of hypotheses describing partially matching patterns. Already started hypotheses are expanded with the label of the edge running from the previous node to the current node.

As an example, the translation graph for the sentence ‘Samstag und Februar sind gut, aber der vierte wäre besser’ is shown in Figure 5.1. Actually, the graph is much bigger. In the figure, only those edges are shown which contributed to the construction of the best path.

5.2.2 Search Hypothesis and Back-Trace Information

A parsing hypothesis $h = (j_1, j_2, \sigma, Q)$ contains information about:

- the segment in the source sentence, given by start position j_1 and current position j_2, which is spanned;
- the current state σ in the transducer, which has been reached while matching the source words between j_1 and j_2;
- the accumulator Q for the translation probability for $e_1...e_i$ as translation of $f_{j_1}...f_{j_2}$.

Actually, the algorithm is implemented in such a way, that the search hypotheses are stored with respect to the current node j_2. Therefore, j_2 is not stored explicitly in the search hypothesis.

For initializing the search algorithm the empty search hypothesis is used. For each node in the lattice, except the last one, a hypothesis $h_0 = (j, j, \sigma_0, 0)$ is used, where j is the position of the node and σ_0 is the start state in the transducer.

Back-trace information is required at two stages:

- When closing a hypothesis, i.e. when constructing an edge which is added to the graph. In this case it is necessary to trace back over the path in the graph and collect the edges which were traversed in constructing this hypothesis. To do so, the predecessor hypothesis and the edge, over which this predecessor hypothesis was expanded to generate the current hypothesis, is stored.
- When reconstructing the best parse. To do this, each edge carries pointers to the generating edges and the generating transducer item. During training this information can be used to update the counts for the transducer items.

5.2.3 Expansion of a Search Hypothesis

The matching between a path through a transducer and part of a sentence can start at each position in the sentence. As the algorithm is formulated using a graph the positions are the nodes between the words. Therefore, an initial hypothesis $(j, j, \sigma_0, 0)$ is set for each position $j = 0, ..., J - 1$.

Expansion of hypotheses in a graph can be organized over the outgoing edges of the nodes or over the incoming edges. In accordance with the implementation the second alternative will be described. Selecting one way over the other is largely a matter of taste.

Expanding hypotheses over incoming edges is then structured in the following way: Let n be a node in the translation graph, $\mathcal{I}(n)$ be the set of incoming edges, and let $n^<(e)$ denote the start node of an edge. Then, for each incoming edge $e \in \mathcal{I}(n)$ all hypotheses h in $n^<(e)$ are expanded with the word f or the category label C attached to e. That is to say, if σ is the transducer state of hypothesis h and $\tilde{f}$ denoted the word or category label, then $\{\sigma'\}$ is the set of transducer states which can be reached from σ over transitions labelled $\tilde{f}$. At the moment only exact match is considered and then either one or no successor state in the transducer is possible. Later, when the error model is used, i.e.

when insertions, deletions and substitutions are allowed, more then one successor state can exist.

If expansion is possible then a new hypothesis is generated:

$$h = (j, n^{<}(e), \sigma, Q) \rightarrow h = (j, n, \sigma', Q')$$

The accumulator for the translation probability collects the probabilities from the edges labelled with a category label.

$$Q' = \begin{cases} Q \cdot Q(\tilde{f}) & \text{if} \tilde{f} \text{carries category label} \\ Q & \text{else} \quad . \end{cases}$$

5.2.4 Creating Edges

When an emitting state can be reached in the transducer new edges are created and added to the translation graph - one edge for each translation emitted from that state. The edge is labelled with the category label associated with the emission. Start node and end node of the new edges are the nodes as given in the search hypothesis.

Additional information attached to the edge are:

- The partial translation due to the target pattern emitted in the emitting state. This target pattern can be a single word or category label, or a sequence of words and category labels. Category labels are replaced by the partial translations associated with those labels which can be collected by back-tracing the path in the translation graph. That is to say, partial translations are propagated to the higher levels of the parsing tree.
- The translation score Q.
- The list of edges in the translation graph which where traversed while traversing the transducer from its start state to the emitting state. This is the back-trace information already mentioned and which is used in estimating the transducer probabilities in training.
- The transducer and the emitted transducer item. Notice, as an emitting state can emit several translations the state would not be enough. In training the counts for the different translations have to be collected to estimate the translation probabilities.

5.2.5 Recombination of Hypotheses

Recombination of hypotheses can be performed for open and for closed hypotheses. It makes sense to differentiate the two situations as they require different tests if two hypotheses are equal under the optimization criterion and can therefore be recombined.

Two open hypotheses are equal when they span the same segment in the source sentence and have the same transducer state. This can happen if there are different parses possible for this sequence of words. As the transducer state fully determines which final states can be reached, i.e. which category labels and partial translations will be emitted, both hypotheses will generate the same set of closed hypotheses upon further expansion. So, only the hypothesis with the better score needs to be retained.

Actually, for the parsing and translation of sentences equal open hypotheses are not very frequent. The test for equality between hypotheses requires more computation time than expanding occasional superfluous hypothesis.

For closed hypotheses the situation is somewhat different. To be equal, they also have to be equal when taking the language model into account. That is to say that the same target word sequence is generated from both hypotheses. But different final states can emit the same category labels and translations. For example, the two translation patterns:

$$\begin{array}{cccccc} C & \# & f_1 f_2 & \# & e_1 e_2 \\ C & \# & f_1 f_2 f_3 & \# & e_1 e_2 \end{array}$$

will have different final states in the transducer for category label C but emit in both states the category label C and the word sequence $e_1 e_2$. Allowing error tolerant match both final states could be reached. In such a case two different hypotheses are equal as far as the parsing and translation of the sentence is concerned. Therefore, only for the hypothesis with the better score will an edge be generated and added to the translation graph.

Notice that recombination of closed hypotheses catches also those hypotheses which are already identical due to identical transducers state and which would be eliminated when recombination on open hypotheses would be used.

5.3 Why Construction of a Graph

Applying a cascade to a sentence has been described as chart parsing but with the difference that the explicit construction of a graph is part of the parsing algorithm. Of course, for parsing a sentence and using a bilingual grammar this means also translating a sentence this explicit graph construction is not necessary. So the question is, what is advantage of doing so.

To a large extent it is simply a question of using one out of many possible ways to implement the parsing algorithm. The graph structure serves as a concise interface between different steps in the training and search algorithm.

In training not the best parses for source and target sentence are required but those parses which can be matched with each other. In the next chapter a training algorithm based on finding an optimal alignment between two graphs will be given.

In search two knowledge source are used to find the best translation of a given sentence, the translation model, which in our case is given by the cascade of transducers, and the

language model. Again, the graph can be used as an interface between the two processing steps: The translation model generates the translation graph and the language model is used to find the first-best path in this graph.

Chapter 6

Training of Transducers

In the previous chapter a translation model on the basis of cascaded transducers has been developed. This model is a statistical translation model with two components: the language model and the string translation model. In this chapter the estimation of the parameters of these models will be discussed. Actually, only the training of the translation model will be developed in detail as for the experiments reported in Chapter 9 a standard n-gram language model has been used instead of the language model given by the transition probabilities in the transducers. The reason for this has already been given in Section 4.4.

The transition and emission probabilities of the transducer can be collected from a given bilingual training corpus. The idea is to find the optimal path through the cascade of transducers which gives an alignment for a sentence pair. In general, there may be several paths. However, normally one path is the preferred one. This allows us to use simpler Viterbi style training instead of the full forward-backward training.

6.1 Problem Formulation

The problem with training the cascade of transducers is that the top-level transducer is not given. Only the manually or semi-automatically constructed category transducers are available. The top-level transducer results from applying the category transducers to the bilingual corpus to replace sequences of words $f_{j_1}...f_{j_2}$ and $e_{i_2}...e_{i_2}$ by category labels. This replacement it subject to the condition that the generalized translation pattern have the same category labels on source and target side. Labelling independently will - except for simple sentence pairs - not guarantee this. Here, we will rely on alignment for this purpose. From all possible replacements of word sequences by category labels only those will be used which are compatible with an alignment given by an extended alignment model as described in this chapter.

Training the cascaded transducer therefore implies more than only collecting counts for the parameters of the model. It entails also finding the structure for part of the overall translation model. It is well-known that the induction of structure, i.e grammatical

inference is a very difficult problem when attempted in its general sense. Here, the task is a more restricted one. Only part of the structure is induced as a kind of residual which is left over after applying all category transducers.

The training of the cascaded transducer translation model divides into the following steps:

1. Apply the category transducers to the bilingual corpus.
2. Extract the structure of the top-level transducer.
3. Estimate the model parameters of the full translation model.

6.2 Construction of Top-Level Transducer

In Chapter 5 it has be shown how the application of a transducer to a sentence can be viewed as constructing a graph where each edge spans a sequence of words $\mathbf{f}$ which have been accepted by the transducer. These translation graphs can also be used for training. In this case the translations attached to the edges are not required, only which translation pattern was applied in the creation of this edge.

So, for both source and target sentence graphs are generated giving a compact representation of all word sequences compatible with the transducers. The goal now is to generalize the standard alignment models to the alignment of those graphs. To do so, the basic features of the word-based HMM-style alignment model will be recounted.

6.2.1 Word-based HMM-style Alignment

The HMM-style alignment model as introduced in [Vogel et al. 1996] is a first order alignment model where the probability for aligning the word f_j at position j in the source sentence to word e_i at position i in the target sentence depends on the alignment of the previous word f_{j-1} at the position $j-1$ in the source sentence to some word $e_{i'}$ at position i' in the target sentence.

The motivation for this type of alignment model is that we typically observe a strong localization effect in aligning the words in parallel texts (for language pairs from Indo-European languages): the words are not distributed arbitrarily over the sentence positions, but tend to form clusters.

We can rewrite the probability by introducing the 'hidden' alignments $a_1^J := a_1 ... a_j ... a_J$ for a sentence pair $[f_1^J; e_1^I]$

$$\begin{aligned} Pr(f_1^J|e_1^I) &= \sum_{a_1^J} Pr(f_1^J, a_1^J|e_1^I) \\ &= \sum_{a_1^J} \prod_{j=1}^{J} Pr(f_j, a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) \end{aligned}$$

We now assume a first-order dependence on the alignments a_j only:

$$\begin{aligned} & Pr(f_j, a_j | f_1^{j-1}, a_1^{j-1}, e_1^I) \\ &= p(f_j, a_j | a_{j-1}, e_1^I) \\ &= p(a_j | a_{j-1}, I) \cdot p(f_j | e_{a_j}) \end{aligned}$$

where, in addition, we have assumed that the translation probability depends only on e_{a_j}. Putting everything together, we have the following HMM-based model:

$$Pr(f_1^J | e_1^I) = \sum_{a_1^J} \prod_{j=1}^{J} \left[p(a_j | a_{j-1}, I) \cdot p(f_j | e_{a_j}) \right] \tag{6.1}$$

with the following components:

- HMM alignment probability: $p(i|i', I)$ or $p(a_j|a_{j-1}, I)$;
- translation probability: $p(f|e)$.

In addition, we assume that the HMM alignment probabilities $p(i|i', I)$ depend only on the jump width $(i - i')$. Using a set of non-negative parameters $\{s(i - i')\}$, we can write the HMM alignment probabilities in the form:

$$p(i|i', I) = \frac{s(i - i')}{\sum_{l=1}^{I} s(l - i')} \quad . \tag{6.2}$$

This form ensures that for each word position i', $i' = 1, ..., I$, the HMM alignment probabilities satisfy the normalization constraint.

We use maximum approximation:

$$Pr(f_1^J | e_1^I) \cong \max_{a_1^J} \prod_{j=1}^{J} \left[p(a_j | a_{j-1}, I) \cdot p(f_j | e_{a_j}) \right] \tag{6.3}$$

The task of finding the optimal alignment is straight forward by using a dynamic programming approach for which we have the following typical recursion formula:

$$Q(i, j) = p(f_j | e_i) \max_{i'=1,..,I} \left[p(i|i', I) \cdot Q(i', j - 1) \right]$$

Here, $Q(i, j)$ is a sort of partial probability for an alignment path which runs over $f_1 ... f_j$ and ends in position i of the target sentence aligned to f_j.

6.2.2 Extending the HMM Alignment to Graph Alignment

The HMM-style alignment model will now be extended by allowing also word groups to be aligned to each other. Which word groups to consider in the alignment is given by the transducers.

Such an alignment can be realized as an alignment between two graphs. Each graph is the result of applying the cascade of transducers to source respectively target sentence. Each graph is a compact representation of all possible parses given the special transducers.

6.2.2.1 Restrictions to the Alignment

In the most general alignment it would be possible that a source word sequence labelled with some label L is aligned to one target word e, or that a word sequence labelled as L_1 is aligned to a word sequence labelled as L_2. Such alignments would violate the alignment model on the basis of cascaded transducers and are therefore forbidden.

Actually, only those edges can be aligned which are generated by the same sequence of bilingual translation patterns. It does not suffice that the edges $\vec{f}$ and $\vec{e}$ have been generated by the same translation pattern. For the edges $\vec{f_j}$ and $\vec{e_i}$which have been traversed in the construction of $\vec{f}$ and $\vec{e}$ and which have been constructed from lower level transducers such matches have to be found. That is to say, the graph alignment is compatible with a path through the cascade of transducers.

6.2.2.2 String translation probabilities

When training the cascade of transducers the following problem arises: probabilities for edges generated from more general transducers are smaller than probabilities for edges generated from the transducers further down in the hierarchy. Each level contributes an additional factor ≤ 1 to the overall translation probability according to the translation model developed in the previous chapter. Therefore, the best alignment path would normally run over the edges generated by lower level transducers. This is especially the case when a transducer reads and emits only one category label. That is to say, a translation pattern of the form: $L2 \rightarrow L1\#L1$ has been used. Such a rule on its own will not change the translation of any sentence. But it can effect the overall performance if there is also a translation rule which involves $L2$ but no corresponding rule which differs only in that $L2$ is replaced by $L1$.

There is an additional problem in finding the alignment between two sentences after categorization. Applying category transducers to a sentence often generates edges with the same category label, where one edge is spanning only over part of the words which are covered by the other edge.

If we had the top-level transducer from the start its application would ensure that the hierarchical alignment on all levels is used. The top-level transducer would generate (at least) a pair of edges spanning over the complete source sentence on one side and over the complete target sentence on the other side. Tracing back the construction of these edges would recover the complete hierarchical alignment as well as the complete sequence of translation patterns. This training is possible only in a second step.

As this top-level is not given some other means are required to drive the alignment towards aligning edges generated by higher level transducers.

Those alignment pathes should be preferred which

- align edges in the two translation graphs generated from transducers higher up in the cascade of transducers;

- align longer edges.

To realize the first condition each transducer is assigned to a level in the hierarchy of all transducers. We denote the transducers level by l and the number of all levels by L.

When calculating the optimal alignment path the alignment criterion should ensure that the longer edges in source and target language are selected. For the source sentence this is generally the case as longer edges mean a fewer number of position alignments resulting in a higher overall alignment probability. For the target sentence this is not the case. Here, going over shorter edges may produce a higher overall score.

To avoid this situation an alignment heuristics could be chosen to prefer longer edges in source and target sentence: Let $\vec{f}$ be an edge in the source translation graph starting at position j_1 and ending at position j_2 and let $\vec{e}$ be an edge in the target translation graph starting at position i_1 and ending at i_2. The string translation score for aligning the edge $\vec{f}$ to the edge $\vec{e}$ is then:

$$S(\vec{f},\vec{e}) = \frac{j_2 - j_1}{J} \cdot \frac{i_2 - i_1}{I} \cdot \frac{l}{L} \quad .$$

A different approach, one which is more in line with the probabilistic translation model, is the following. At each position (j,i) in the trellis only those alignments are allowed which have been generated by the transducers higher up in the cascade of transducers. To be more precisely: let $\vec{f}_n$ and $\vec{e}_m$ be the incoming edges into nodes n_j and n_i in the two graphs. Then only those pairs $(\vec{f}_n, \vec{e}_m)$ are considered for the alignment for which the following conditions hold:

- The two edges have been generated by the same sequence of translation patterns.

$$\mathcal{T}(\vec{f}_n) = \mathcal{T}(\vec{e}_n)$$

- There exists no pair of edges $(\vec{f}_{n'}, \vec{e}_{m'})$ which can be matched on the first condition and which has a higher transducer level $\mathcal{L}$.

$$\mathcal{T}(\vec{f}_{n'}) = \mathcal{T}(\vec{e}_{n'}) \rightarrow \mathcal{L}((\vec{f}_{n'}) \leq \mathcal{L}(\vec{f}_n)$$

As the hierarchy of the transducers induce only a partial ordering on the set of all transducers the second condition could not be replaced by referring to the absolute level of the transducer.

Not all words in the source and target sentence are covered by some edge generated by a transducer. Therefore, not all grip points in the trellis can be reached using higher level transitions. For those grid points where this is not the case the translation probabilities from a word-to-word lexicon can be used. Such a lexicon is nothing but an additional transducer of the form shown in Figure 6.1. On the left, the lexicon transducer is shown as a standard transducer where the input and the output symbols are both attached to the transitions. On the right, the transducer is structured as a Moore finite state machine.

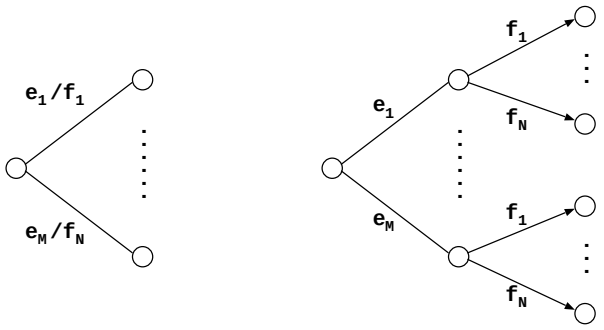

Figure 6.1: Statistical lexicon as standard transducer (left) or as transducer with emissions in final states (right). M is the size of the target vocabulary, N the size of the source vocabulary.

The string translation probabilities are now given as:

$$p(\vec{f}|\vec{e}) \quad = \quad \begin{cases} p_L(f|e) & \text{if} \quad \mathcal{L}(\vec{f}) = \mathcal{L}(\vec{e}) = 0 \\ p_T(f|e) & \text{if} \quad \mathcal{T}(\vec{f}) = \mathcal{T}(\vec{e}) \\ 0 & \text{else} \end{cases} \tag{6.4}$$

Here, p_L denotes the probability from the statistical lexicon and p_T denotes the probability from transducer T, which may involve recursive calculation of the emission probabilities from lower level transducers.

6.2.2.3 Position Alignment

In word-based HMM alignment the positions in the sentences are attached to the words. Now, the alignment is formulated in terms of the nodes between words. The position alignment is detached from the individual words insofar, as several edges can lead to one node and this node will feature in the alignment irrespectively which edge has been traversed to reach the node. We want to reformulate the position alignment to take this point of view into account.

Advancing over an edge in the graph is associated with advancing the position in the sentence. This means that no jumps in the position alignment are required for strict monotone alignment. Notice that aligning the same target word e_i to two adjacent source words f_j and f_{j+1} would then require a jump backwards which is counterintuitive. So, to stay as close as possible to the formulations in the basic model each transition in the graph is interpreted as a jump in position.

Aligning edges which cover a number of words means that fewer jump probabilities are involved.

Having fewer position alignment probabilities favors the alignment of longer edges. For the case where one word is replaced by a label or one label is replaced by a label from a higher level transducer, the same number of jump probabilities is involved for both high level and low level alignment. That is to say, the position alignment probabilities give a bias towards higher level, more generalizing transducers only in those cases where edges spanning more than one edge are generated.

6.2.3 Viterbi Alignment

To calculate the Viterbi alignment the shortest path through a trellis (j, i), $0 \leq j \leq J$ and $0 \leq i \leq I$, has to be calculated. A path through this trellis represents an alignment of nodes in the source translation graph to nodes in the target translation graph. What is required, however, is an alignment between edges in source and target graph, as edges - not nodes - represent words and word groups. The edges, over which the node (j, i) has been reached has to be stored in the back-trace information.

To calculate the best partial path ending in (j, i) we have to consider all possibilities, how this grid point can be reached. This is possible by joint transition into node n_j and node n_i which in turn means by a joint transition over a pair of edges $(\hat{e}, \hat{f})$, where $\hat{f}$ is taken from the edges running into node n_j and $\hat{e}$ is taken form the edges running into node n_i. The number of possible transitions into (j, i) is number of edges with node n_j as end point times number of edges with node n_i as end point.

We use $Q(j, i)$ as accumulator of the score for a partial alignment running from position $(0, 0)$ to position (j, i) in the trellis.

$$Q(j,i) = \max_{\vec{f}} \max_{\vec{e}} \max_{i'} S(\vec{f}, \vec{e}) \cdot Q(j_1, i') \cdot p(i' - i'') \tag{6.5}$$

6.2.4 Bilingual Labelling

The best alignment is used not only to collect the counts for the alignment model. It is also used to write a bilingual corpus with word sequences replaced by category labels. For edges in the trellis which correspond to words in source and target sentence just this words are written. For aligned edges with category labels the word sequences corresponding to those edges has to be recovered. This involves a recursive descend as this edge may have been constructed from other edges also carrying category labels.

However, to construct the labelled target sentence it is not possible to trace back the best alignment path and write - from back to front - the words associated with the edges. An alignment is only guaranteed to cover the complete source sentence and to be a function from source to target sentence. For the target sentence

- there may be words not hid by the best alignment path;
- there may be words, which are hid several time;
- the word order of the target sentence would normally be changed.

Therefore, writing the labelled target proceeds the following way. First, all edges with labels from the target sentence graph lying on the best alignment path are recorded and stored according to the node to which these edges lead. Then, running backwards over the target sentence any remaining gaps are filled by the word at the given position.

6.2.5 Re-normalizing the Transducers

The transducers are re-normalized after each iteration. For each final state the relative frequencies of the source strings are taken.

$$p(\mathbf{f}|\mathbf{e}) = \frac{N(\mathbf{f},\mathbf{e})}{\sum_{\mathbf{f}'} N(\mathbf{f}',\mathbf{e})} \quad . \tag{6.6}$$

As simple smoothing strategy is used to avoid zero probabilities in training. All counts are incremented by a small constant. Therefore:

$$p(\mathbf{f}|\mathbf{e}) = \frac{N(\mathbf{f},\mathbf{e}) + c}{\sum_{\mathbf{f}'}(N(\mathbf{f}',\mathbf{e}) + c)} \quad . \tag{6.7}$$

Typically, a value of $c = 0.5$ for the smoothing constant is chosen. Notice that smoothing affects only the translations actually given in the transducer. As a result, a pair $(\mathbf{f},\mathbf{e})$ encoded in the transducer will have a non-zero probability, even if it has not be seen in the training corpus. But no probability mass is distributed over pairs $(\mathbf{f}',\mathbf{e}')$ which are not in the transducer. This is in contrast to statistical lexicon models where smoothing is used to assign a non-zero probability to all $f \in V_F$ for any $e \in V_E$. To give non-zero probability to unseen word sequences is left to the error model introduced in Section 4.5. This error model is not incorporated into the training scheme.

6.2.6 Deficiency of the Model Approximation

The model used in the graph alignment to construct the top-level transducer is not the full model introduced in the previous chapter but only an approximation. For the alignments generated according to the method proposed there remains a problem. It is not excluded that a sequence of words $\mathbf{e}$ is aligned to $\mathbf{f}$ and a sequence of words $\mathbf{e}'$ which is contained in $\mathbf{e}$ or partially overlapping with $\mathbf{e}$ is aligned to $\mathbf{f}'$ which is disjoint from $\mathbf{f}$. This is a situation similar to the deficient models IBM3 and IBM4. To avoid those alignments bookkeeping over already aligned positions would be required, resulting in an enormous increase in the time complexity of the alignment algorithm.

6.3 Training the Complete Model

The result of the training described so far is a bilingual labelled corpus. This can now be used to construct the top-level transducer as a tree-transducer over the target side of the corpus. This step completes the cascade of transducers.

The parameters of this complete model, i.e. the transition and emission probabilities of all transducers, can now be estimated from the original bilingual corpus in a similar way as the training described so far. The main difference being that for all sentence pairs the translation graphs will have at least one edge which is covering the complete sentence. And the best alignment is selected only from those alignments with complete coverage in one step.

6.3.1 Calculating the alignment

The alignment between source and target sentence is built in a bottom-up fashion. All transducers in the cascade are applied in turn, starting with the low-level transducers and working up to the top-level transducer.

The alignment costs are now calculated in a different way. The actual translation probabilities as given by the transducers are used. That is to say, probabilities are propagated from bottom to top and accumulated to give the complete translation probability. Position alignment probabilities are no longer taken into account as they only were part of the approximation to the correct model.

We use again Dynamic Programming for efficient calculation of the alignment probability and use the following accumulator:

$Q(j_1, j_2, i_1, i_2, C)$ is the probability for aligning the source word sequence from position j_1 to j_2 to the target word sequence from position i_1 to i_2 by the category transducer C.

For initialization we have to use all category transducers which are terminal, i.e. which have only words but no category labels for transitions or emissions.

$$Q(j_1, j_2, i_1, i_2, C) = p_C(\vec{f}_{j_1}^{j_2}, \vec{e}_{i_1}^{i_2}). \tag{6.8}$$

Only some of the points in this 5-dimensional matrix are set during initialization, as normally only part of source and target sentences are covered by the terminal transducers.

The recursion step is:

$$Q(j_1, j_2, i_1, i_2, C) = \max_{\vec{f} \epsilon \mathcal{E}(j_1, j_2)} \max_{\vec{e} \epsilon \mathcal{E}(i_1, i_2)} p_C(\vec{f}_{j_1}^{j_2}, \vec{e}_{i_1}^{i_2}) \prod_{f_k = C^l} Q(j_1^k, j_2^k, i_1^k, i_2^k, C^l) \quad . \tag{6.9}$$

The maximization is over all edges spanning from position j_1 to position j_2 in the source sentence and from position i_1 to i_2 in the target sentence. $p_C(\vec{f}_{j_1}^{j_2}, \vec{e}_{i_1}^{i_2})$ is the emission probability from the transducer C. The recursive descent is given by the product which runs over over all generating edges $\vec{f}_k$ which carry a category label C^l. $\vec{f}_k$ spans from j_1^k to j_2^k and is aligned to edge $\vec{e}_k$ which spans from i_1^k to i_2^k. As the generating edges for the the edges $\vec{f}$ and $\vec{e}$ are known from the construction of the two translation graphs, no optimization over the generating edges is required.

Termination:

$$Q(0, J, 0, I) = Q(0, J, 0, I, C) \tag{6.10}$$

The best overall alignment is given by the best alignment of the edges generated from the top-level transducer. Due to the maximization in the recursion step only one pair of edges spanning the complete sentences is aligned.

6.3.2 Back-Tracing and Updating Counts

To reconstruct the best alignment path the local decisions, i.e. the edges $\vec{f_b}, \vec{e_b}$ for the locally best alignment have to be stored:

$$B(j_1, j_2, i_1, i_2, C) =< \vec{f_b}, \vec{e_b} >= \max_{\vec{f}\epsilon\mathcal{E}(j_1,j_2)} \max_{\vec{e}\epsilon\mathcal{E}(i_1,i_2)} Q(j_1, j_2, i_1, i_2, C) \tag{6.11}$$

Together with the information about the generating edges for $\vec{f}$ the complete alignment can be reconstructed by recursive decent.

To update the counts only the parse tree for $\vec{f}$ or $\vec{e}$ has to be traversed. In that case it is not important to know, where the aligned edges are positioned within the sentences. If e.g. the word pair 'Montag - Monday' occurs twice in the sentence pairs, then both are aligned using the same transducer item. To increment the count for that transducer item it is not necessary to know if the first Montag is aligned to the first or the second occurrence.

6.3.3 Hierarchical Alignment

The hierarchical alignment generated be the cascaded transducer alignment model is also recovered during the back-trace. Alignment is on the basis of the words, i.e. the lexical items. However, not only the transducers on the lowest level read and emit words. This is possible by each transducer on any level. But there is no direct alignment information given for the lexical items. That is to say, in a translation pattern

$$\mathrm{C} \rightarrow f_1 f_2 f_3 \ \# \ e_1 e_2$$

each f_j is aligned with each e_i. Following the distinction between possible (p) and required or sure (s) alignments, these alignments have to be set as possible alignments. For alignment patterns, where both right hand sides are one word only, these alignments are sure alignments.

For the purpose of word alignment and lexicon generation this would not suffice as only for that part of the vocabulary, which is covered by the category transducers, a word-to-word alignment is generated. If word alignment is required an alignment as given by the HMM alignment model used for the generation of the top-level transducer is more appropriate.

Chapter 7

Search

Translating a new source language sentence using the translation model and the language model for the target language is a maximization over all possible target language sentences using the model probabilities. In other words, a search for the best translation has to be performed.

7.1 Search Strategy

Different search strategies are possible and have been used in different publications.

- The source sentence is traversed from left to right and possible segments of the target sentence are generated [Tillmann et al. 1997b, Tillmann et al. 1997c]. This approach suffers from the fact that different permutations of these segments are competing hypotheses and have to be kept in parallel. This approach is feasible only if additional restrictions are imposed on the number of possible word re-orderings.

- The target sentence is constructed word for word while more and more words in the source sentence are covered [Nießen et al. 1998]. This search strategy has the advantage that the language model for the target language can be used at each stage in the construction of the hypotheses for partial translations.

- In parsing bottom up construction of the parsing tree is one of the preferred methods. In translation this amounts to the bottom up construction of growing segments of the source sentence translated. Word reordering is introduced implicitly by considering permutations of the segments or explicitly – as in the cascaded transducer approach – where the relative order of the segments is given by the translation patterns.

It is this last search strategy which is used in the present work. The transducers are applied to the source sentence in a bottom up manner, parsing and thereby translating longer and longer segments of the source sentence. The details of this approach will be given in the subsequent sections.

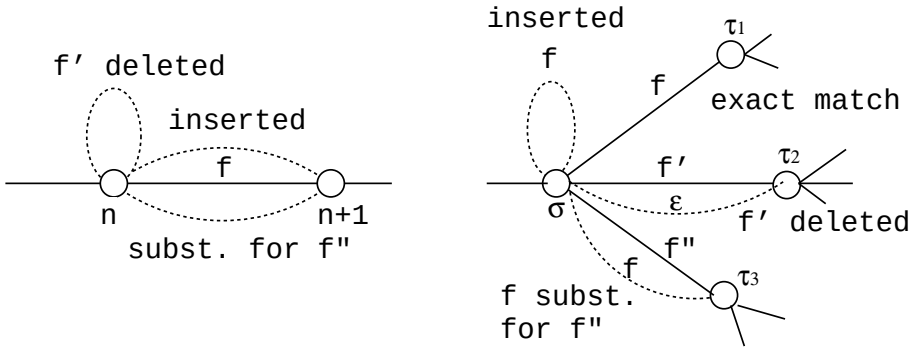

Figure 7.1: Structure of error model for decoding.

7.2 Error-tolerant match

In the original transducer-based translation model the source language word sequences are emitted from final states. In Section 4.5 the incorporation of an error model has been described. The different organization of the transducers for efficient search requires that the error model is modified too. The situation during search is shown in Figure 7.1. On the left hand side part of the input sentence (converted into a graph) is shown. The word f between the nodes n and $n+1$ is shown. On the right hand side part of a transducer is displayed. The transitions which are actually in the transducer are those with solid lines whereas the transitions resulting from the error model are characterized with dashed lines.

Assume we want to expand a search hypothesis h in node n with word f. Remember, a search hypothesis describes a possible translation of some part of the source sentence by giving start position and current position in the sentence and the current state σ in the cascade of transducers. The following possibilities are given:

- Exact match: this corresponds to the transition from transducer state σ to τ with no additional probability involved, as translation probability for the exactly matching word sequence $\mathbf{f}$ given the target string $\mathbf{e}$ is attached to the final state. In Figure 7.1 this corresponds to moving from node n to node $n+1$ in the translation graph and from state σ to state τ_1 in the transducer.

- Substitutions: for all transitions $f_n : \sigma \rightarrow \tau_n$, $f_n \neq f$ it has to be tested, wether f_n can be substituted for f. If this is the case, then the state τ_n is added to the set of successor states. The probabilities for these substitutions are $p^S(f_n, f)$ as given in Section 4.5. In Figure 7.1 only for one transition a substitution is possible. This is the transition from σ to τ_3 where f is substituted for f''.

- Deletion: for all transitions $f_n : \sigma \rightarrow \tau_n$ it has to be tested, if f_n can be deleted, i.e. if $f_n \epsilon \mathcal{D}$. If this is the case, then the state τ_n is added to the set of successor states. The probabilities for these deletions are $p^D(f_n)$. In the sentence to be translated a deletion means to stay at node n, as indicated in Figure 7.1 with the loop at that node.

- Insertion: it has to be tested, if f can be inserted, i.e. if $f \epsilon \mathcal{I}$. If this is the case, then the state σ is itself a successor state and has to be added to the set of all successor states. The probabilities for the insertions are $p^I(f_n)$. In Figure 7.1 this is indicated by the loop in the transducer at state σ, which is labelled with f.

In Table 7.1 the resulting steps in the source sentence respectively. translation graph and the transducer are summarized.

Table 7.1: Error-tolerant matching.

Type	Position in Sentence	Position in Transducer	Additional Probability
Exact	successor node	successor state	none
Insertion	successor node	same state	$p^I(f)$
Deletion	same node	successor states	$p^D(f)$
Substitution	successor node	successor states	$p^S(f, f')$

The deletion error differs from the other cases in that no step forward in the translation graph is associated with it. This requires a different handling: for each node n in the translation graph and for each transducer all search hypotheses $h(n)$ in n have to be expanded without consuming any word. For each transducer state which is reached from the state $\sigma(h(n))$ via an ϵ-transition a new search hypothesis is created and added to node n.

To allow for two or more deletions following right one after another this expansion without advancing in the translation graph has to be repeated until no successor states are returned. This will eventually be the case, as the transducer is finite and all ϵ-transitions in the transducers involve a change in state. For the most general error model where each word has a positive probability to be translated all states located in the tree with σ as its roots would be returned. As the probabilities for insertions are small the number of consecutive deletions can be set to some small integer. If only words from a small subset of the vocabulary can be deleted this is not necessary.

Actually, the error model is not implemented by adding all those transitions to the transducer for the insertions, deletions and substitutions. Rather, this is captured in an algorithmic way.

7.3 Applying a Language Model

The application of the transducers to a given source sentence yield a large number of target sentences. These are scored according to the cumulative scores of the applied translation patterns. As an independent and direct model of the likelihood of the target sentences a language model is applied.

As presented in Section 4 the transition probabilities of of the transducers are the language model probabilities for the target language and could be used as such.

In the experiments reported in this work a different approach is taken. We use a word-based n-gram language model [Sawaf et al. 2000]. This model is based on a suffix tree implementation thereby allowing for variable length history. As an upper bound 5-grams are used as longer histories are rarely effective on test data.

The application of the language model is implemented as a re-scoring of the translation graph comparable to the re-scoring of a word hypothesis graph in speech recognition.

The logarithm of the language model probabilities is added to the transducer scores when the best path through the translation graph is extracted. A scaling factor allows for a bias on the effect of the language model.

Chapter 8

Construction of Cascaded Transducers

8.1 Semi-Automatic Construction of Category Transducers

8.1.1 From Alignment

The category transducers, which have also been called specialized transducers, are dedicated to encode characteristic phenomena of the domain under consideration. It has not been the goal to develop a method for fully automatic construction of such transducers. Instead, the idea was to incorporate linguistic knowledge to some extend.

For the Verbmobil domain parts of speech have been used for a number of category transducers. A semi-automatic construction for these transducers was devised using part of speech tagging and lexical information extracted from automatically generated alignments.

The Verbmobil corpus has been tagged using the Stuttgart-Tübinger tag set for the German part of the corpus and the well-known Penn tree-bank tag set for the English part. A standard HMM alignment model was the trained. Using the Viterbi alignment triplets of the form

Tag # German word # English Word

were collected. If the Viterbi path aligned two ore more consecutive source words to the same target word and these source words were all labelled with the same part of speech tag, then the complete word sequence was written as one triplet of the form:

Tag # German word sequence# English Word

or in the notation used previously:

Tag # f_1, ..., f_n # e

This was specially for the German compounds.

Of course, due to tagging errors and alignment errors the resulting set of translation patterns could not be used unmodified. It was corrected manually. Still, this semi-automatic procedure gave a considerable time saving.

8.1.2 From Lexicon

Not all words are seen in the training corpus. But even for those words missing part of speech tags was made available by the Verbmobil partners in Stuttgart. For those words possible translations had been extracted from online dictionaries. Joining the word-to-tag and word-to-translation lists generated an addition set of triplets which was added to those resulting from the alignment-based approach.

8.1.3 From Corpus

For some of the transducers in the Verbmobil corpus special corpus annotations could be utilized. In that corpus proper names and numbers are marked as such and therefore can easily be collected.

8.2 Induction of Bilingual Grammar

The information obtained from aligning the sentences in a bilingual corpus can be used to help in the construction of bilingual grammars. The idea is to extract high frequency translation patterns and use them to build higher level transducers.

It is not the purpose of this dissertation to develop a fully automatic method for the construction of bilingual grammars. This would be even more difficult than monolingual grammar inference.

We do not want do construct a grammar on purely syntactic considerations. For translation the grammar should be based on the domain for which the translation system is designed for. For example the names for the days of the week or the names of the months are nouns in standard tagging. But it may be more appropriate to subsume those words under a special grammar for date expressions.

One additional difficulty with the construction of the bilingual grammar is that a word class may feature in different specialized grammars. For example, numbers are part of time and date expressions, are used in prices, as room numbers or as part of noun phrases, each of which calls for its own dedicated grammar.

To help with the construction of the grammars candidates for bilingual translation patterns can be selected from the alignment. Tracing back the alignment path segments

from the source sentences with the corresponding segments in the target sentences are collected according to the following selection criteria.

1. The first condition is the condition of aligning category labels: category labels can only be aligned if they are generated by the same derivation from the cascade of transducers.

2. The source sequence of the extracted pattern should contain some category labels. If l is the length of the source pattern and n the number of category labels then $n \geq c \cdot l$ with a constant $c \leq 1$. A typical value of c is 0.5, that is to say, at least half of the symbols in the source patterns should be category labels.

3. The aligned position in the target sentence should be not to far apart. It is not required that the aligned positions form a consecutive sequence. But the gaps should be not too large as this indicates that the word sequence forms at least two distinct phrases.

4. Words or category labels in the target sentence aligned to a sequence of words or category labels in the source sentence should have no additional alignment to some source words outside of this sequence.

Sorting the candidates according to their coverage in the training corpus is a good indication, which patterns to select into the grammars.

8.3 Fully Automatic Construction of Cascaded Transducers

A very simple method has been implemented for fully automatic generation of cascaded transducers. The basic idea is to replace short sequences of words, for which the translations are known in longer sentence pairs. Sentence pairs made up of two short sentences are used as starting point. Converting these sentence pairs into a transducer allows to find and replace them bilingually in the rest of the corpus. This is done in exactly the same way as described in Chapter 6. that is to say, the transducer is treated exactly as the category transducers. The category label can be set arbitrarily, e.g. L1 to indicate that is is the level 1 transducer.

Applying the transducer to the training corpus will result in a modified corpus, where some sentence pairs have now sequences of words replaced by this category label L_1. Multi-word translation patterns will shorten the sentences in the labelled corpus, as several words are replaced by one category label.

this process, selecting the sentence pairs with the shortest sentences, converting them into a transducer, performing bilingual categorization with this transducer, can be repeated several times. At each iteration, the shortest sentence pairs may be purely on the word level, with no category labels, they may contain any lower level category labels.

As shorter segments are helpful in obtaining higher coverage on unseen data, the following modification can be introduced: If after bilingual categorization with transducer L_n source and target pattern begin or end with this category label L_n this category label is removed.

Look at the following example:

buongiorno	good morning
buongiorno telefono da Roma	good morning I'm calling from Rome

And further assume that the first sentence pair has already been used in the construction of the first transducer, for which the category label L_1 is used. so, after applying this transducer the second sentence pair is converted into:

L_1 telefono da Roma	L_1 I'm calling from Rome

This translation pattern would allow to translate the original sentence or any sentence, which starts with some other word sequence labelled as L_1. However, it would not allow to translate 'telefono da Roma alone, unless this word sequence appears somewhere else in the training corpus as a complete sentence.

On the other side, removing the leading category label – together with adjacent punctuation characters – from source and target sentence will improve the situation: all what could be translated before still can be translated, even if the translation has to be made up of several segments. But in addition, the remaining sentence can be translated.

And perhaps even more importantly: if the remaining sentence pair is short enough to qualify for inclusion into a category transducer in one of the next iterations, it is more likely to match than the complete pattern with the leading category label.

For tailing category labels exactly the same story can be told.

In summary, the overall procedure is as follows:

Fully Automatic Construction — Struktogramm

set B_0 equal training corpus
FOR l iterations
select n shortest sentence pairs from B_{l-1}
convert into transducer T_l
apply T_l to $B_{l-1} \rightarrow B'_l$
remove label-only sentence pairs in $B'_l \rightarrow B''_l$
remove labels at head and tail $B''_l \rightarrow B_l$

Chapter 9

Experiments and Results

In this section, we will give some results obtained with the cascaded transducer approach. Experiments were performed on two corpora:

- the FUB corpus collected within the EuTrans project [Vidal et al. 2000];
- the Verbmobil corpus collected within the Verbmobil project [Wahlster 2000].

Both corpora a spoken language corpora. In the Verbmobil project the goal was to develop a dialog system. Therefore, the corpus is a collection of recorded dialogs with the typical characteristics of spoken language like false starts, disfluencies, hesitations, etc. The FUB corpus is a collection of phone calls to a hotel reception making reservation, asking for services, or making complaints. More details of the two corpora will be given further down.

The FUB corpus has been used especially to investigate the amount of labor required in the construction of a cascade of transducers to improve translation quality over a simple translation memory approach. In connection with this segmentation and the construction of dedicated grammars has been studied.

Finally, a comparison with two other translation approaches, the single word based approach and translation with alignment templates will be given.

But before presenting the results the evaluation methodology will be outlined.

9.1 Evaluation Methodology

Evaluation of translation quality is a difficult and controversial task. What constitutes a good translation of a given sentence is – at least to some degree – a matter of taste. Two translations may differ in lexical choice and word order and both could be equally acceptable translations. This makes it somewhat problematic to use fully automatic and yet reliable evaluation methods as in speech recognition where Levenshtein distance is the method of choice.

Subjective evaluations have the drawback, that there is no generally accepted methodology which is used by different research groups. Sometimes evaluation differentiates between different dimensions in quality like fluency and adequacy [White and O'Connell 1994] or syntactic and semantic correctness [Jekat et al. 1999]. In other cases an overall quality judgement by human evaluators is given by classifying the quality of translations into a small number of classes.

In this work, the translations are evaluated according to two measures [Nießen et al. 2000]:

- Word Error Rate (WER): The WER is calculated as the edit distance (minimum number of insertions, deletions and substitutions) between the produced translation and a predefined reference translation. The edit distance has the advantage that it can be calculated automatically. The disadvantages of the WER are that it depends heavily on the choice of the reference translation and that it does not take into account whether and how much the various differences to the reference translation affect the meaning of the translation.

- Subjective Sentence Error Rate (SSER): The translations are classified, by one or several human experts, into a small number of quality levels that range from "perfect" to "absolutely wrong". In comparison with the WER, the SSER is more meaningful and conveys more information, but its measurement is rather expensive in terms of human manpower.

To support the human experts in the assignment of the subjective error scores and to guarantee a certain degree of consistency, an evaluation tool has been developed. For each test sentence in the source language and its current translation, this tool displays to the human expert previously evaluated translations of the same sentence. In addition, the tool is able to automatically compute an estimate of the SSER by finding nearest matches to former evaluated translations stored in a database [Nießen et al. 2000, Vogel et al. 2000a].

9.2 FUB Corpus

9.2.1 The Corpus

To investigate the amount of manual labor required for the construction of the transducers the FUB corpus was used. This is a speech translation corpus which has been collected in the EuTrans-II project. The domain of this corpus is phone calls to a hotel reception. The collected sentences contain queries, request and complaints. The sentences have been transliterated and translated and used to study different approaches to machine translation. Some examples are given in Table 9.2.1.

The next table, i.e. Table 9.2.1 gives details of the corpus. The number of words given includes the punctuation marks, whereas 'Proper Words' is the number of words in the corpus without punctuation marks. The size of the vocabulary is the number of full word

Table 9.1: Examples from the FUB corpus.

buon giorno , vorrei prenotare una stanza tripla , per il periodo dal ventitre' al ventinove dicembre con servizi in camera , frigorifero e televisore , grazie .
good morning , I would like to reserve a triple room , for the period from twenty-third to the twenty-ninth of December with services in room , minibar and television set , thanks .
buonasera , senta , sono all' aeroporto . io ho prenotato un stanza da voi . mi chiamo Toscano , non arrivo piu' alle otto e mezzo ma alle dodici e cinquanta , grazie .
good evening , listen , I am at the airport . I reserved a room a t yours , my name is Toscano , I will not arrive at eight thirty any longer but at twelve fifty , thank you .

Table 9.2: Training and test conditions for the FUB task.

		Italian	English
Train:	Sentences	3 038	
	Words	55 302	65 446
	Proper Words	47 606	57 588
Vocabulary	Size	2 459	1 701
	Singletons	1 118	662
Test:	Sentences	278	
	Words	5 930	7 000
	Proper Words	5 129	6 189
	Out of Vocabulary	100	
	Trigram Perplexity		10.06

forms seen in the corpus. The Italian vocabulary is nearly double the size of the English vocabulary due to greater morphological variation.

As can be seen, this is a rather small corpus. The type–token ratio for proper words is 1 : 19.4 for Italian and 1 : 33.9 for English. The percentage of word types seen only once in the corpus is 45% for the Italian vocabulary and 39% for the English vocabulary. These are rather typical numbers. Of course, these singletons account for only 2% resp. 1% of the word tokens in the corpus. These numbers give an indication of out-of-vocabulary words to be expected in unseen test data and coincide well with the 100 new word types in the test corpus.

9.2.2 Segmentation

The turns in the FUB corpus are rather long. The average sentence length of the Italian sentences is 18.2 and 21.4 for the English sentences. In Figure 9.2.2 the sentence length distributions for the Italian and the English sentences are given. They are somewhat

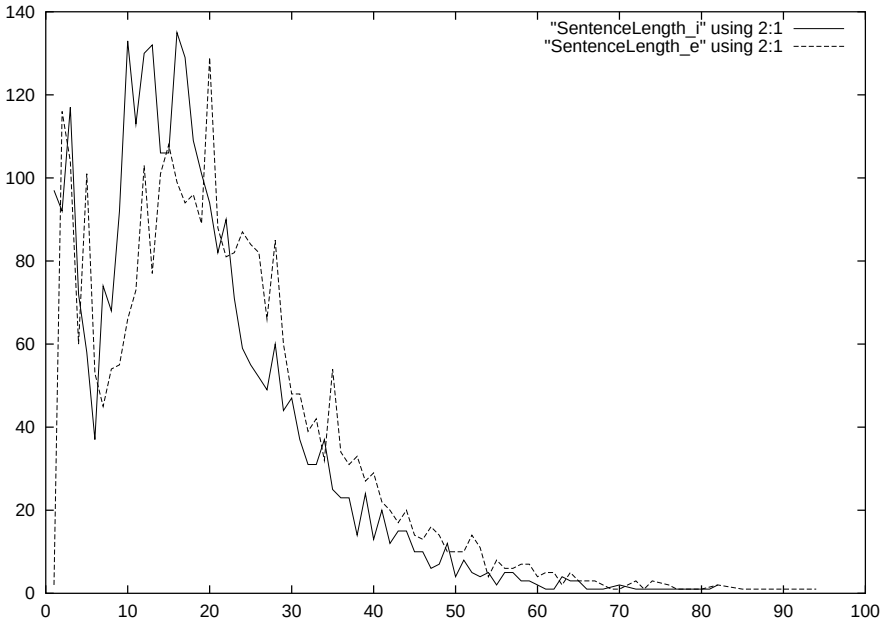

Figure 9.1: Sentence length distribution of FUB corpus.

untypical in that they have two peeks each, one at around 2-4 words and one ranging from 10-25 words. This is due to the fact that the corpus contains a number of very short phrases, like proper names and short date expressions, which have been added to the corpus to provide data for training a speech recognition system.

To gain higher generalization power the sentence pairs have been automatically segmented. Punctuation marks were considered to be potential segmentation points. Those at sentence final position do not constitute possible segmentation point. The number of punctuation marks which might be used as segmentation point are given in Table 9.2.2.

Table 9.3: Segmentation results for different segmentation conditions.

Punctuation mark	Italian	English
.	958	936
!	85	88
?	328	327
:	101	101
;	41	41
,	3601	3537
Total	5114	5030

Punctuation marks are less parallel than these numbers would suggest. It is often the case that not all of the punctuation marks in the source sentence have corresponding punctuation marks in the target sentence. For example, there are 15 sentences with non-matching '?'. And for commas, the mismatch is even stronger.

To avoid incompatible segmentation of source and target sentence only those splits were performed which gave the highest gain in terms of perplexity from a given translation

model. To split the sentences of the FUB corpus it was required that there is a punctuation mark in source and target sentence. For each sentence pair the pair of punctuation marks was selected for segmentation. A segmentation was only done, when it would result in a lower corpus perplexity using the IBM1 alignment model.

The resulting corpus contained 7 325 segment pairs. For ease of terminology, we will in general not distinguish between sentence and segment. The average sentence length has been cut down to 7.4 and 8.8. It is also interesting to see the number of sentences and sentence pairs which occur more than once in the original and the segmented corpus.

9.2.3 Training with Standard Alignment Models

The probabilities for the cascaded transducers estimated using an extended HMM type alignment model. For comparison, and to establish a reference, a standard word-based alignment was generated first. This was done by running the training for the IBM-1 alignment model for 10 iterations and using the resulting lexicon as initialization for the HMM alignment model. The training for this model was then run for 5 iterations. The resulting perplexities are given in Table 9.2.3. Besides the total perplexity those contributions from the lexicon model and the position alignment model are shown. Of course, for the IBM1 model the perplexity from the position alignment model stays constant over different iterations.

Table 9.4: Perplexity as function of iteration for training with IBM-1 and HMM alignment model.

Model	Iteration	Lexicon	Alignment	Total
IBM1	1	3.39	26.66	90.2
	2	2.19	26.66	58.3
	3	1.75	26.66	46.5
	4	1.57	26.66	41.9
	5	1.49	26.66	39.7
	6	1.45	26.66	38.6
	7	1.42	26.66	37.9
	8	1.41	26.66	37.5
	9	1.40	26.66	37.2
	10	1.39	26.66	37.1
HMM	1	2.51	6.89	17.3
	2	2.77	5.29	14.7
	3	2.78	4.61	12.8
	4	2.81	4.27	12.0
	5	2.82	4.11	11.6

The lexicon perplexity for the HMM alignment model is higher than for the IBM-1 alignment model. This is first of all due to the fact that the HMM alignment model has been trained in the maximum approximation. That is to say, only the best, the so-called

Viterbi alignment is calculated. A second reason is that sharpening the position alignment probabilities pulls the lexicon probabilities away from their optimal value. Actually, the lexicon perplexity rises slightly from the first to the second iteration. Still, total perplexity drops and is considerably lower than for the IBM1 alignment model.

9.2.4 The Transducers

For the FUB task a number of special transducers were constructed. On the basic level these transducers were selected to give essentially a semantic classification for part of the vocabulary. Although some transducers conform to part of speech tags (adjectives, adverbs, ...) most of the transducers give a semantic classification, e.g. animals, things in the hotel room or in the bath. Building these transducers was done using the methods described in Section 8.1.

In Table 9.2.4 the categories are listed for which transducers have been constructed and which form the first level in the cascade of transducers. Given is:

- The number of translation pattern, which in this case are simply pairs of words or word groups.
- The number of transitions as an indicator of the size of the transducers. This gives also - when compared to the number of words in the corpus - the compression achieved by this organization.
- The number of final states, which is equal to the number of different target sentences in the corpus.

The values given are for the re-structured transducers which are constructed as prefix tree over the source language and have the target part of the bilingual corpus as their emissions.

For the basic category transducers these numbers are interesting only insofar as they show the number of translation pairs going into each one of those transducers. For most categories we have simple word-to-word associations and therefore the numbers are very close to each other. Exceptions are the categories Greeting, Numbers and VP (verb phrases). Greeting show a certain variability, both in lexical choice and in orthography, as can be seen in Table 9.2.4. For numbers the main source is that ordinal and cardinal numbers are different in English but not in Italian. Finally, for the verbs, we have the effect of richer morphology on the Italian side but observe also that verb groups often have different translations.

9.2.5 Induction of Bilingual Grammar

The first level categories are numbers, the names of the months and special time expressions.

Following the Viterbi alignment path all translation patters have been collected for which the following requirements were fulfilled:

Table 9.5: First level category transducers for FUB task.

Name	States	Final States	Emissions
Adjectives	184	183	210
Adverbs	40	39	47
Animals	12	11	13
Articles	11	10	18
Clothes	11	10	12
Family	9	8	12
Food	34	33	38
Greeting	12	9	21
Holiday	19	18	20
InBath	11	10	12
Meals	4	3	3
Month	13	12	13
Names	207	206	206
Nouns	224	222	240
Numbers	180	179	217
People	11	10	17
PlacesHotel	12	11	13
Service	20	16	18
TimeOfDay	25	24	27
VP	121	71	102

- the length of the source sequence is 2 to 4 words/labels;
- at least half of the positions in the source sequence are occupied by category labels;
- the length of the target language sequence aligned to the source sequence does not exceed the source sequence length by more than two.

In Table 9.9 the most frequent translation patterns are given sorted according to their coverage
(length of source sequence + length of target sequence) * frequency

Specialized grammars:

- Room with numbers;

  Example:

  chiamo dalla stanza trecentocinque # I call from room three hundred and five

- Room descriptions, i.e. what is in a room.

  Example:

Table 9.6: Examples from first level category transducers for FUB task.

Category		Italian		English
GREETING	#	arrivederci	#	bye
GREETING	#	arrivederci	#	bye-bye
GREETING	#	arrivederci	#	good bye
GREETING	#	arrivederci	#	good-bye
GREETING	#	arrivederci	#	goodbye
NUMBER	#	ventinove	#	twenty-ninth
NUMBER	#	ventinove	#	twenty-nine
NUMBER	#	trecentoventicinque	#	three hundred and twenty-five
NUMBER	#	trecentoventicinque	#	three hundred twenty-five
VP	#	aveva	#	has
VP	#	aveva	#	it has
VP	#	aveva	#	it had
VP	#	avevo chiesto	#	I asked for
VP	#	avevo chiesto	#	I required
VP	#	disdirla	#	cancel it
VP	#	disdirla	#	to cancel it
VP	#	disdirlo	#	cancel it
VP	#	disdirlo	#	to cancel it

Table 9.7: First level transducers for time and date expressions.

Name	Patterns
Numbers	217
Month	12
Time	27

vorrei prenotare una stanza tripla , con vista , con televisione a colori e con doccia per non fumatori con frigobar , senza aria condizionata . # I wish to book a triple room , with view , with colour tv and with shower for non smokers with minibar , without air conditioning .

- Date expressions;

  Example:

  una stanza dal ventitre' al venticinque settembre , # a room from the twenty-third to the twenty-fifth of September ,

Table 9.8: Examples for specific time expressions.

TIME # antimeridiane # in the morning
TIME # domani # tomorrow
TIME # ieri # yesterday
TIME # mattina # morning
TIME # mezzogiorno # midday
TIME # mezzogiorno # noon
TIME # notte # night

Table 9.9: Examples for specific time expressions.

2124	354	NUMBER MONTH # NUMBER of MONTH
1932	322	NUMBER e NUMBER # NUMBER NUMBER
1428	238	dal @NUMBER # from the NUMBER
1328	332	e NUMBER # NUMBER
1200	150	NUMBER al NUMBER # NUMBER to the NUMBER
1160	145	al NUMBER MONTH # the NUMBER of MONTH
985	197	al NUMBER # the NUMBER
936	117	alle NUMBER e NUMBER # at NUMBER NUMBER
920	184	alle NUMBER # at NUMBER
918	153	NUMBER al # NUMBER to the

9.2.6 Effect of Language Model

A standard n-gram language model was trained on the English sentences from the training corpus. Using the suffix array implementation allows for longer histories than in the normally used bi- or tri-grams. Normally only a very small fraction of the longer histories are seen in the training corpus. Therefore, the effective length of the histories used in the translation of a test corpus is generally smaller than the history length of the chosen language model. When using for example a tri-gram model backing of to bi-grams or even uni-grams will frequently occur.

In Table 9.2.6 the language model perplexities on training and test corpus are given for the FUB corpus. As can be seen, going beyond a four-gram does not result in further improvement of the test set perplexity. This suggest that going beyond a four-gram will not result in higher translation quality.

Table 9.10: Language model perplexity on training and test set for the FUB task.

Level	LP	USP	WER

The effect of the language model was studied along two lines:

- How much do longer histories affect translation quality?
- How does translation quality depend on the language model scaling factor λ, when the best translation $\hat{e}$ is chosen according to

$$\hat{e}_1^I = \underset{e_1^I}{\operatorname{argmax}} \{p^\lambda(e_1^I) \cdot p(f_1^J|e_1^I)\} \quad .$$

  Due to the approximations made especially in the translation model a scaling factor $\lambda \neq 1$ might give some improvement.

In the first experiment the history length was varied from zero to four, corresponding to using a uni-gram up to using a five-gram. For this experiment all transducers described in the previous section were used.

The results are in Table 9.2.6. The translation quality is shown in terms of word error rate (WER) and position independent error rate (PER). In addition, the language model perplexity (PP) and the average length of the actually used n-grams ($\bar{n}$) are given for the hypothesized translations and - for comparison - for the reference translation. In the final column the processing time in seconds for the 278 test sentences is given to show, how using a higher order language model affect translation time.

What this table shows is that the order of the language model has a influence on translation quality, but only up to a tri-gram. Beyond that there is even a slight degradation, which may arise from over-fitting the model on the training data.

A second experiment was conducted to investigate to what extend the translation quality is influenced by giving more or less weight to the language model. A tri-gram language model was used for this experiment and

Table 9.11: Effect of language model on translation quality for the FUB task.

n	WER	PER	PP	$\bar{n}$	PP_r	$\bar{n}_r$	Time[s]
1	49.8	39.1	180.5	0.98	127.0	0.99	176
2	34.8	26.3	28.2	1.80	14.1	1.91	263
3	34.7	26.2	24.6	2.40	10.1	2.66	308
4	35.3	26.8	26.2	2.82	10.1	3.24	338
5	35.4	27.0	27.1	3.12	10.5	3.66	362

Table 9.12: Effect of language model scaling factor on translation quality for the FUB task.

LM-scale	WER	PER	PP	$\bar{n}$
0.0	45.0	33.9	60.7	2.66
0.1	36.8	27.9	31.8	2.34
0.2	34.7	26.1	26.0	2.39
0.3	34.7	26.0	26.0	2.38
0.4	34.9	26.3	25.4	2.39
0.5	34.7	26.2	24.6	2.40
0.7	34.9	26.5	23.5	2.40
1.0	37.7	29.3	23.1	2.37
2.0	40.8	33.1	21.9	2.31
5.0	42.6	35.4	20.8	2.27

Language model perplexity and average language model order were calculated using the resulting hypothesized translations. The table shows to things. First, there is a clear dependency of the translation quality, as measured by word error rate, on how strong the language model features in selecting the best hypothesis. Starting from a high word error rate when using no language model at all, approaching a minimum with a language model scaling factor at around 0.4, and than moving again to higher error rates, when the language model scaling factor goes up. So, the language model is necessary in selecting the best translation from all possible translations generated by the cascade of transducers. On the other side, if the language model influence is to strong it selects those string which are smoother with respect to the syntactical constraints of the target language, but with are probably not as faithful a translation. And this is the second observation which can be made in Table 9.2.6: the perplexity of the generated translations decreases continuously with higher language model scale. The two measures, word error rate and perplexity are independent measures of translation quality.

9.2.7 Recursive Labeling

In Section 8 the fully automatic construction of a cascade of transducers has been described. In two experiments the effectiveness of this method has been tested. First, it

was applied to the plain corpus, second, it was combined with the use of the special transducers.

The training procedure was as follows: In each iteration the 200 shortest segments were used as new translation patterns and transformed into a transducer. This transducer was used in training, estimating the translation probabilities $p(f|e)$ for this transducer and transforming the bilingual corpus into a labeled bilingual corpus. (Nice picture!)

In Table 9.13 the results are given. LP: Number of translation patterns with lower level category labels.
USP: Number of remaining unique sentence pairs in training corpus

Table 9.13: Automatic construction of transducers for the FUB task.

Level	LP	USP	WER
1	0	5266	90.1
2	8	4970	62.6
3	12	4744	49.7
4	17	4582	43.8
5	16	4462	40.9
6	26	4385	37.9
7	61	4342	36.7
8	65	4302	35.5

In the second experiment the manually built transducers were applied first, transforming the bilingual corpus into a labeled corpus. Then, the automatic construction of a cascade of transducers followed. The results are given in Table 9.14

Table 9.14: Automatic construction of transducers for the FUB task.

Level	LP	USP	WER
1	0	4899	42.9
2	23	4709	37.9
3	22	4617	36.4
4	55	4592	34.8
5	77	4565	34.1
6	104	4531	33.5

9.2.8 Translation Examples

Table 9.15: Examples from the automatically generated transducers

@L1 # annullarla # cancel it # 0.11
@L1 # disdirla # cancel it # 0.57
@L1 # disdirlo # cancel it # 0.32
@L3 # @L1 e' # it's @L1 # 1
@L3 # @L2 @ADJ # @ADJ @L2 # 1
@L3 # @L2 @INROOM # @INROOM @L2 # 1
@L3 # @L2 @TOD # @TOD @L2 # 1
@L3 # @NN @L2 # @L2 @NN # 1
@L3 # @NN fotocopie # photocopy @NN # 1
@L3 # @NN gravissima # serious @NN # 1
@L3 # abbia @L2 # @L2 has # 1
@L3 # avere @L1 # @L1 have # 1
@L6 # mi @VP dire # @VP tell me # 0.89
@L6 # mi @VP indicare # @VP tell me # 0.11
@L6 # per @L1 @NUMBER # @L1 @NUMBER o'clock # 1
@L6 # per @L1 @NUMBER di # @L1 @NUMBER # 0.54
@L6 # per @L1 @NUMBER e # @L1 @NUMBER # 0.46

Table 9.16: Translation examples from the FUB test corpus. S = source sentence, R = reference translation, H = translation hypothesis.

S	buongiorno l' albergo Excelsior di Parigi ? volevo sapere se per la settimana di Natale c' e' disponibilita' di una camera matrimoniale con doccia , con frigobar e con aria condizionata .
H	good morning the Excelsior hotel in Paris ? I would like to know if for the week of Christmas there is availability for a double bed room with shower , minibar and air conditioning .
R	good morning is it the Excelsior hotel in Paris ? I would like to know if for the week of Christmas it is available a double bed room with shower , with minibar and with air conditioning .
S	telefono dalla stanza uno zero uno , volevo disdire l' asciugacapelli che avevo prenotato , grazie .
H	I am calling from room one o one , I would like to cancel the hairdryer that I reserved , thank you .
R	I'm calling from room one o one , I would like to cancel the hairdryer I booked , thank you .
S	vorrei richiedere il servizio di babysitter alle ore cinque e dieci .
H	I would like to require a babysitter service at five ten .
R	I would like to require the babysitter service at five ten .
S	sono spiacente , il vestito che ho mandato a pulire e' sporco .
H	I spiacente , a dress I mandato to clean it is dirty .
R	I am sorry , the cloth that I sent to clean is dirty .
S	buongiorno , chiamo dalla stanza sei quattro due . volevo sapere quanto costa il servizio in camera per una bottiglia di vino per due persone . grazie .
H	good morning , I am calling from room six fourth two . I would like to know how much it costs the room service for a bottle of wine for two people . thank you .
R	good morning , I am calling from room six four two . I would like to know how much it costs the room service for a bottle of wine for two persons . thank you .

9.3 Verbmobil Corpus

9.3.1 The Corpus

Experiments were performed on the VERBMOBIL corpus. This corpus consists of spontaneously spoken dialogs in the appointment scheduling domain [Wahlster 2000]. A summary of the corpus used in the experiments is given in Table 9.17. In Table 9.18 the sizes of the special purpose transducers are given.

Table 9.17: Training and test conditions for the VERBMOBIL task. The trigram perplexity (PP) is given.

		German	English
Train	Sentences	34 465	
	Words	363 514	383 509
	Voc.	6 381	3 766
Test	Sentences	147	
	Words	1 968	2 173
	PP	–	19.7

The sentences from the training corpus were segmented into shorter segments using sentence marks as breakpoints. This resulted in 43 609 bilingual phrases running form 1 word up to 82 words in length. The longest phrases were discarded as it is very unlikely that they will match other sentences. So, for the construction of the translation patterns only 40 000 sentence pairs were used, the longest sentences containing sixteen source words. Starting from those simple phrases, successively more transducers were applied up to the full cascade. A total of 15 682 translation patterns containing one or more labels resulted and nearly 4 500 sentence pairs became identical when words or word sequences were replaced by labels.

9.3.2 The Transducers

9.3.3 Effect of Grammar

A simple translation memory without any categorization gives insufficient coverage on unseen test data. With the part-of-speech transducer we get one or more translations for each word in the vocabulary. But only by applying transducers which handle longer translation patterns is word reordering possible.

In Table 9.20 the results are given for different combinations of transducers. The baseline (T) is the combination of all special purpose transducers (name, spell, number, date, word tags) plus the simple translations patterns. Then the grammar was added and finally the compound translation patterns. The trigram language model for the target language was applied in selecting the best translation, but no error tolerant matching was allowed.

Table 9.18: Size of the transducers.

Transducer	Patterns
Names	442
Numbers	342
Spell	60
SimpleDate	161
CompoundDate	173
WordTags	6 714
Grammar	124

Table 9.19: Example for the application of the bilingual grammar.

```
VP # PPER VMFIN PP VVINF # PPER VMFIN VVINF PP
VP { PPER   { ich # I # -0.1 }
     VMFIN  { möchte # want # -0.1 }
     PP     { APPR { mit # with # -0.1 }
              { PPER { Ihnen # you # -0.1 }
                NP   { ART { einen # a # 0.01 }
                       { NN { Termin # date # -0.1 }
                # a date # -2.09 }
     # a date with you # -6.29 }
     VVINF  { vereinbaren # to arrange # -0.1 }
# I want to arrange a date with you # -12.59 }
```

We observe a clear effect in word error rate and subjective sentence error rate. The use of the bilingual grammar, also very restricted, improves translation quality. Applying the compound translation patterns gives an additional small improvement.

In Table 9.19 a simple and a more involved example for the reordering effect of the bilingual grammar are given. The first translation pattern operates solely on the level of POS tags whereas the second example generates a hierarchical structure. We are not concerned whether the source sentence parses are correct, good translations is what we are looking for.

9.3.4 Effect of Language Model

The next experiment shows the effect of applying a language model for the target language. A word-based trigram language model was interpolated with the scores from the transducers. In Table 9.21 the effect of the scaling between the two models is shown.

There is a clear drop in the WER when switching on the language model. This is due to the fact, that several translation hypotheses have the same score from the transducers.

Table 9.20: Effect of bilingual grammar on translation quality: T = POS-tagging, G = grammar, C = compound translation patterns.

Transducer	mWER[%]	SSER[%]
T	41.2	25.8
TG	39.7	22.5
TGC	38.8	22.1

So, it is rather by chance if the best translation for a given word is chosen. The language model for the target language helps in doing this.

Table 9.21: Effect of language model on word error rate and subjective sentence error rate.

LM Scale	mWER[%]	SSER[%]
0.0	49.3	31.8
0.2	38.8	23.5
0.5	38.8	22.1
1.0	39.4	23.8
5.0	42.6	27.4

There is a second benefit gained from the language model: sometimes the source sentence can be covered with only very short source patterns. That is to say, word context is hardly taken into account. With a language model context is brought into play again. If the language model scaling factor is increased too much translation quality deteriorates again. So, a good balance between both knowledge sources is necessary.

In Table 9.22 some examples which show the effect of the language model are given. The first translation is without language model, the second is the translation obtained when the language model score is added using a scaling factor of 0.5.

9.3.5 Effect of Error Tolerant Matching

Finally, the effect of error tolerant matching has been investigated. Only for the simple and compound translation patterns errors have been allowed in matching parts of the input sentences to stored translation patterns. The effect of increasing the error threshold is given in Table 9.23.

We see a considerable improvement when allowing for a small number of errors in matching the translation patterns to the input sentence. However, if the match gets too sloppy serious errors occur which alter the meaning of the sentence. For longer sequences of words the number of errors allowed becomes higher than the default score for substitutions. In such a case content words can be substituted.

Table 9.22: Examples for the effect of the language model.

	erst wieder ab dem sechzehnten.
no LM	starting from the sixteenth only again.
with LM	only starting from the sixteenth.

	ja, wunderbar. machen wir das so, und dann treffen wir uns dann in Hamburg.
no LM	yes, nice. will we do which right, after all we meet us after all in Hamburg.
with LM	fine. let us do it like that, and then we will meet then in Hamburg.

Table 9.23: Effect of error tolerant matching.

Errors per word	mWER[%]	SSER[%]
0.0	38.8	22.1
0.2	38.3	20.3
0.4	37.0	21.0
0.6	39.6	24.2

An example of how the same source sentence gets different translations when more matching errors are allowed is given in Table 9.24.

Table 9.24: Examples for the effect of error tolerant matching.

	ja , wunderbar . machen wir das so , und dann treffen wir uns dann in Hamburg .
0.0	fine . let us do it like that , and then we will meet then in Hamburg .
0.2	fine . let us do that , and then we will meet in Hamburg .
0.4	fine . let us do it like that , and then we will meet in Hamburg .
0.6	fine . let us do it like that , and then we will meet in your office .

9.4 Experiments on Nespole Corpus

9.4.1 The Corpus

The training data for the SMT system was originally collected in the Nespole! speech-to-speech MT project [Lavie et al. 2001]. Several dialogs were recorded from telephone conversations between an Italian tourist office and native English- and German-speaking clients. The agents, native speakers of Italian, spoke English or German for the data collection.

Table 9.25 shows that the corpus is very small. Nearly 50% of the German vocabulary and about 40% of the English Vocabulary occurs only once in the corpus.

Table 9.25: Training corpus statistics.

	German	English
Sentences	3182	3182
Words	14992	15572
Vocabulary	1367	1041
Singletons	645	410

For testing the translation systems, a number of the dialogs were held out. The results reported here are for three of the held-out dialogs originally recorded in German. One dialog (70 sentences) was used as cross-evaluation data to run our optimization experiments on the SMT system. Two dialogs (82 sentences) were then used as test data in a comparative evaluation between the SMT system and the Nespole! IL-MT system. The training data fails to cover 29% of the types in this test set, giving a token OOV rate of 11%.

9.4.2 Evaluation Methods

In our experiments we applied both automatic and manual evaluation. To evaluate our SMT optimization efforts, we used the automatic evaluation metric Bleu score as proposed in [Papinini et al. 2001]. The Bleu score is based on n-gram precisions when comparing the system translation with several human reference translations. As precision without recall favors short translations, a length penalty is combined with the weighted average of those precisions for the final result.

Human evaluation was carried out for the comparative evaluation of the IL-MT and the SMT systems. The evaluators were presented with the German turn and the two translations. Grading was done on a 3-point scale:

- Good: for translations which give the required information and which are easy to understand, i.e. no critical syntactic errors.

- Okay: for translations which give useful information, even if they are syntactically not correct.
- Bad: for missing translations or for translations which give no useful or even misleading information.

For long turns, information units were identified beforehand and the turns segmented accordingly. Human graders then assigned quality scores on a per-segment basis.

9.4.3 SMT Optimization Experiments

9.4.3.1 Transducer Configurations

We experimented with five transducers, $\{L, P, P_2, R, M, I\}$. L is the statistical lexicon as it is produced by the HMM alignment program. It contains only word-to-word translations. P represents phrase-level alignments. P_2 is the phrase-level product of bidirectional HMM alignment. R is a transducer for some fixed number and date expressions that was hand-coded for German-English translation. It is domain-independent and reusable. M is constructed from an online German-English lexicon. I is the transducer extracted from the interlingual analysis grammars.

Table 9.26: Evaluation results for cross-evaluation set: text input.

Configuration	Bleu Score	C-Cov	V-Cov
L	0.1893	89.18	70.90
LR	0.1903	89.83	72.12
LM	0.1926	93.27	81.21
LRP	0.2350	90.32	72.72
$LRPI$	0.2434	90.49	73.33
$LRMPI$	0.2432	95.08	85.45
LRP_2	0.2654	90.81	73.93
$LRMP_2$	0.2522	94.91	84.24
LRP_2I	0.2714	90.98	74.54
$LRMP_2I$	0.2613	95.24	85.45

Table 9.26 shows the effect of combining these transducers on system performance. For each configuration of the translation system the Bleu score is given. The last two columns in the table give corpus coverage, i.e. how many words from the test corpus were translated, and the vocabulary coverage, i.e. how many word types from the test corpus were translated.

The baseline result of 0.1893 comes from translating with transducer L alone. Adding transducer R gave a small improvement. Transducers P and P_2 gave more significant improvements of 23% and 40%, respectively, over L and R alone. Adding transducer I gave no improvement when added to the baseline system, but accounted for small improvements when used in conjunction with phrase transducers $\{P, P_2\}$.

Transducer M, the background lexicon, gave a large boost in type and token coverage, but translation quality as measured by the Bleu score went down. This points to a problem with adding a general-purpose lexicon: all translation probabilities in the lexicon are equal, and do not reflect the distribution of the training data.

9.4.3.2 Effect of the Large Language Model

Improvements to the language model were made by retraining it on a larger monolingual corpus. First, the English side of the background lexicon was added. In addition we used data from in the Verbmobil project. The Verbmobil corpus is about 500,000 words in size.

Table 9.27: Effect of large language model.

Configuration	Small LM	Large LM
L	0.1893	0.1782
LM	0.1926	0.2298
$LRMP$	0.2334	0.2703
$LRMP_2$	0.2522	0.3141
$LRMP_2I$	0.2613	0.3172

The results of using this larger language model can be seen in Table 9.27. For convenience, the results from using the small language model are repeated in this table. The larger language model almost always helped to improve translation quality. The effect is most pronounced in those configurations which use the background lexicon transducer as well.

9.4.3.3 Background Lexicon as Training Data

In the final experiment the large background lexicon was added to the training corpus for the alignment model. In this way the vocabulary covered in the general-purpose lexicon becomes part of the statistical lexicon transducer L, and the separate background lexicon transducer M is left out.

Results for some transducer configurations are represented in Table 9.28 and show a clear improvement. Again, the results when translating with the background lexicon as a separate transducer are repeated for comparison.

9.4.4 Comparing SMT and IL-MT

To put the performance of the SMT system into perspective we compared it to an existing IL-MT system [Lavie et al. 2001] which was developed as part of the Nespole! project. The Bleu scores and the results from human evaluation are given in Table 9.29 for text (human transcribed) and speech (speech recognizer transcribed) input. The numbers for

Table 9.28: Effect of adding background lexicon to training corpus.

Configuration	Separate	Integrated
LM	0.2298	0.2050
$LRMP$	0.2703	0.2813
$LRMP_2$	0.3141	0.3275
$LRMP_2I$	0.3172	0.3300

'Good', 'Okay' and 'Bad' translations are the sum of two evaluators. To condense those numbers an average score for the human evaluation was calculated by giving each good translation a score of 1, each okay translation a score of 0.5 and each bad translation a score of 0.0.

Table 9.29: Evaluation results for IL-MT and SMT.

		Bleu	Good	Okay	Bad	Score
Text	IF	0.068	77	104	227	0.32
	SMT	0.333	124	80	205	0.40
Speech	IF	0.059	64	101	243	0.28
	SMT	0.262	95	83	227	0.34

The Bleu score is much higher for the SMT system than IL-MT system. The human evaluation revealed the same ordering of the systems but with much closer scores. This indicates that the perceptible difference in translation quality is not as large as the Bleu score suggests.

9.5 Experiments on the TIDES Chinese-English translation task

9.5.1 The TIDES Evaluations

TIDES (Translingual Information Detection, Extraction and Summarization) is a DARPA funded research project (http://www.darpa.mil/iao/TIDES.htm). Machine translation as one of the components of this project. Several research groups participate in regular evaluations. These evaluations are organized by NIST (http://www.nist.gov/speech/tests/mt/). Evaluation of translation results is performed in an automatic manner. The generated translation is compared to typically four human reference translations. The current version of the evaluation script, the one used in the experiments reported here, uses conditional probabilities for uni-grams up to five-grams. The probabilities are calculated on the basis of the reference translations.

So far two evaluations have taken place, the so-called Dry-Run evaluation in December 2001 and the evaluation in June 2002. In December 2001 translation systems for translating from Chinese to English were evaluated in three different data tracks:

- Small data track:
  The bilingual corpus to train the translation model on is about one hundred thousand words. In addition a small bilingual Chinese-to-English dictionary can be used.
- Large Data Track:
  A set of large bilingual corpora is provided by the Linguistic Data Consortium which can be used to train the translation model. In addition the full Chinese-to-English dictionary can be used.
- Open Data Track:
  In addition to the large data track resources any other bilingual data can be used. This means that bilingual data can be collected from the internet. As the test data is taken from news published over the internet a deadline for data collection is imposed.

For the evaluation in June Arabic-to-English translation was added. This was the first Arabic-to-English evaluation within the TIDES program and there was only one data track, which was essentially a large data track.

Here, only results for Chinese-to-English translation results are presented, even though the system has been used for Arabic-to-English translation.

9.5.2 The Data

9.5.2.1 The Training Data

To train the Chinese-to-English translation system 4 different corpora were used:

- Chinese tree-bank data (LDC2002E17): this is a small corpus (90K words) for which a tree-bank has been built.
- Chinese news stories, collected and translated by The Foreign Broadcast Information Service (FBIS).
- Hong Kong news corpus distributed through LDC (LDC2000T46).
- Xinhua news: Chinese and English news stories publish by the Xinhua news agency.

The first three corpora are truly bilingual corpora in that the English part is actually a translation of the Chinese. The Xinhua news corpus is not a parallel corpus. The Chinese and English news stories are typically not translations of each other. The Chinese news contains more national news whereas, the English news is more about international events. Only a small percentage of all stories is close enough to be considered as comparable.

Table 9.30: Corpus statistics for the different Chinese-English corpora.

		Chinese		English	
Corpus	Sentences	Size	Vocabulary	Size	Vocabulary
Xinhua Tree	3,540	90,699	8,492	115,531	9,143
FBIS	102,210	3,498,012	30,625	4,030,257	45,121
Hong Kong News	252,593	6,126,808	34,918	6,159,189	55,016
Xinhua News	71,505	2,713,645	31,102	2,680,525	52,369

The FBIS, Hong Kong news and Xinhua news corpora all required sentence alignment. Different sentence alignment methods have been proposed and shown to give reliable results for parallel corpora. For non-parallel but comparable corpora sentence alignment is more challenging as it requires - in addition to finding a good alignment - also a means to distinguish between sentence pairs which are likely to be translations of each other and those which are aligned to each other but can not be considered translations.

The corpus and vocabulary sizes of the different corpora after some pre-processing (see below) are given Table 9.30.

In addition to the bilingual corpora the LDC Chinese-English dictionary (LDC2002E27) was used. This dictionary has about 53,000 Chinese entries with 3 translations each on average. This dictionary was used for the large data track experiments. For the small data track a subset of 10,000 Chinese entries was extracted. A word frequency list was used to decide which of the entries to select into the small dictionary. The two dictionaries will be called LDC-full or just LDC and LDC-10K. The details are summarized in Table 9.31.

9.5.2.2 The Test Data

Two set of test data have been used in the experiments reported in the following sections:

Table 9.31: Corpus statistics for the two Chinese-English dictionaries.

Dictionary	LDC full	LDC 10K
Uniq Chinese entries	54,131	10,000
Translation pairs	81,945	21,486
Chinese Vocabulary	46,304	9,987
English Vocabulary	28,421	9,061

- As development test data the TIDES December 2001 Chinese test data have been used. This is a collection of 105 news articles collected from Xinhua news agency (52 stories), Voice of America 26 stories), and the Zaobao news agency (27 stories). This data set is used to study different aspects of the SMT system, like dependency of translation quality on the size of the language model. It is also used to tune the system by adjusting some parameters to get optimal result.

- The second set used is the TIDES June 2002 Chinese test set. The test set consists of 70 stories taken from Xinhua news and 30 stories taken from Zaobao.

In Table 9.32 the sizes of these two test sets in terms of sentences, words and vocabulary are given. Test set perplexities for the DevTest data will be given in Section 9.5.5.

Table 9.32: Corpus statistics for the two test sets for the Chinese-English translation experiments.

	DevTest	EvalTest
Sentences	993	878
Words	26,168	24,360
Vocabulary	4,651	4,275

9.5.2.3 Preprocessing

A number of preprocessing steps was performed on these corpora. For the Chinese data these included:

- 2 byte character to 1 byte character conversion:
  The Chinese corpora contain names written in Latin alphabet but with 2 byte encoding of these characters. These where converted to 1 byte characters. The advantage is that unknown names can be carried over to the output when translating test sentences. In addition 2 byte encoded punctuation marks as well as digits were replaced by their 1 byte equivalent.

- Word Segmentation:
  The Chinese written text does not use spaces to separate words. Actually, the notion of a word is less precise for Chinese as it is for European languages. Of the

corpora listed above only the tree-bank data is already word segmented. However, the word segmentation used in the tree-bank seems to have been done according to different criteria then the word segmentation underlying the compilation of the LDC Chinese-English dictionary.

The segmentation used in the tree-bank results in a vocabulary of 9,765 words, 3,731 of which are unknown when using the original LDC dictionary. This number is in part so large because the LDC lexicon contains entries which are short phrases, but unsegmented. When re-segmenting and applying the number and date preprocessing to the tree-bank sentences and also segmenting the Chinese entries in the LDC dictionary with the same word segmenter list the number of unknown words reduced to 26 words.

Word segmentation is usually based on a word list. LDC provides a word segmentation toolkit which is essentially a perl script and a word list. Segmentation is based on a longest match criterion, i.e. at each point the longest word from the word list matching the next characters is chosen. This segmenter has been extended in the following way: the word list is augmented with word frequencies. Segmentation is the done running from left to right over the sentence and also from right to left. From these two segmentations, which can be different, the one is selected which gives a higher product of the frequencies of the individual words.

The word list used for segmentation has some influence on the performance of the resulting translation system. A number of experiments have been performed to study this effect. The results indicated that a small word list seems to give better translation results

- Sentence length filtering:
  Sentences longer than a 100 words on either the source or the target side were removed as those long sentences require a long in the alignment process. Also very short sentences of only up to 3 words where removed as automatic sentence alignment did not seem to be very reliable in aligning lists of short items correctly. In addition, sentence pairs where the length of source and target sentence differed more than 50% were deleted as they result most likely from wrong sentence alignment or indicate very free translations.

- Number conversion:
  A small transducer was developed to convert numbers written in Chinese characters to numbers written in digits and range name like 'thousand' and 'million'. The current implementation of the translation program allows to use this kind of transducers as a preprocessing step resulting in a partially translated sentence. These partial translations are treated by the decoder in the same way as partial translations created from loaded transducers.

- Date conversion:
  An additional transducers was build to translate simple Chinese date expressions like days of the week or compound expressions like 'Monday, 11'. Again, this transducer was applied as a preprocessing step.

9.5.3 Analysis: What is in the Data

9.5.3.1 Vocabulary Coverage

To get good translations requires that first of all the vocabulary of the test sentences is well covered by the training data. Coverage can be expressed in terms of tokens, i.e. how many of the tokens in the test sentences are covered by the vocabulary of the training corpus, and in terms of types, i.e. how many of the word types in the test sentences have been seen in the training data.

Let V_{Train} be the source vocabulary of a training corpus and V_{Test} be the source vocabulary of the test corpus. The token or corpus coverage *C-Cov* is then given by:

$$C\text{-}Cov = |w \in Test \wedge w \in V_{Train}|/|Test| * 100$$

And the type or vocabulary coverage *V-Cov* is:

$$V\text{-}Cov = |V_{Test} \cap V_{Train}|/|V_{Test}| * 100$$

A problem with Chinese is of course that the vocabulary depends heavily on the word segmentation. In a way the vocabulary has to be determined first, as a word list is typically used to do the segmentation. There is a certain trade-off: a large word list for segmentation will result in more unseen words in the test sentences with respect to a training corpus. A small word list will lead to more errors in segmentation. For the experiments reported in this paper a word list with 43,959 entries was used for word segmentation.

Table 9.33 gives corpus and vocabulary coverage for each of the Chinese corpora.

Table 9.33: Corpus coverage (C-Voc) and vocabulary coverage of the DevTest test data given different training corpora and dictionaries.

Corpus	Size	Vocabulary	C-Cov	V-Cov
1. LDC dict small	50,219	9,985	75.43	70.29
2. LDC dict large	146,118	54,151	82.95	92.75
3. Xinhua Tree	90,699	8,492	90.75	69.19
4. FBIS	3,498,012	30,625	97.38	94.24
5. Hongkong News	6,126,808	34,918	97.35	91.31
6. Xinhua News	2,713,645	31,102	98.99	95.46
1+3	141,036	13,332	95.44	82.48
2+3+4+5	9,861,637	69,269	99.80	98.88
2+3+4+5+6	12,575,282	74,014	99.84	99.10

The test data used in the following analysis and also in the translation experiments is a set of 993 sentences from different Chinese news wires, which has been used in the TIDES MT evaluation in December 2001.

9.5.3.2 N-gram coverage

The statistical system uses not only word-to-word translations but also phrase translations. The more of the phrases in the test sentences are found in the training data, the better. And longer phrases will generally result in better translations, as they show larger cohesiveness and better word order in the target language.

The n-gram coverage analysis takes all n-grams from the test sentences and finds all occurrences of these n-grams in the different training corpora. The n-grams are only selected within sentences, i.e. they do not cross sentence boundaries. Table 9.34 summarizes the results. The first column gives the length of the n-gram. The other columns give the number of occurrences of these n-grams in the different sub-corpora. Notice that an n-gram contains two (n-1)-grams, three (n-2)-grams, etc. The longest matching n-gram in the Xinhua news corpus was 56 words long, which accounts for 35 of the 138 20-grams in the Xinhua news corpus. These give no additional benefit. Actually, the distinct long n-grams contain 56, 53, 43, 34, 31, 28, 24, and 21 words. Subtracting the number of the embedded shorter n-grams would give a better picture of the distinct long n-grams found in the training corpora. However, two n-grams taken from the test sentences can be overlapping in a training sentence. Therefore, shorter n-grams would be deducted twice, leading to numbers which are too small. As the analysis of the n-gram coverage is intended only to characterize the different corpora, the number of different n-grams can be taken as a reliable indicator.

Table 9.34: N-gram coverage in different sub-corpora.

n	n-grams	TB	FB	HK	XS	TB.FB	TB.FB.HK	TB.FB.HK.XS
2	17793	4935	10665	9334	11503	11225	12621	13683
3	22018	1857	5180	3818	6525	5740	6990	8663
4	22386	650	1637	913	2735	1966	2396	3628
5	21851	230	548	212	1283	698	810	1611
6	21059	95	210	54	745	280	314	884
7	20162	41	83	7	486	117	123	545
8	19226	21	34	1	368	52	53	395
9	18279	15	15		310	29	29	321
10	17344	11	7		275	18	18	281
11	16434	8	4		249	12	12	253
12	15541	6	2		228	8	8	230
13	14668	5	1		213	6	6	214
14	13821	4			200	4	4	200
15	12998	3			187	3	3	187
16	12207	2			176	2	2	176
17	11439	1			166	1	1	166
18	10704				156			156
19	9994				147			147
20	9311				138			138

We see that especially the Xinhua news corpus contains a large number of word sequences which also occur in the test data. This is no surprise, as part of the test sentences come from Xinhua news, even though they date from a year not included in the training data. Adding this corpus to the other training data therefore gives the potential to extract more and longer phrase to phrase translations which should result in better translations. However, this corpus is not a strictly parallel corpus, i.e. it is not guaranteed that a sentence pair is actually a translation pair. The effect of adding such noisy data needs therefore to be studied.

9.5.4 Training the Translation Models

IBM1 and HMM alignment models were trained in the forward direction, i.e. Chinese-to-English and in the reverse direction, i.e. English-to-Chinese. The IBM1 models were trained for 5 iterations, the HMM models for 2 iterations. Lexicon, position and total alignment perplexities are shown in Table 9.35 for training the alignment models on the small and on the large data. For the large data track to sets of alignment models were trained, one using clean parallel data only, and one using the comparable Xinhua news corpus in addition. The larger vocabularies for the large data track corpora lead to higher alignment perplexities. Notice also that adding the noisy corpus leads to a significant increase in alignment perplexity.

Table 9.35: Training perplexities.

Corpus	Model	Lex-PP	Pos-PP	Total-PP
Small	IBM1	1.34	40.38	53.97
	HMM	3.59	7.02	25.24
	IBM1-re	1.55	32.07	49.63
	HMM-rev	4.92	5.34	26.29
Large clean	IBM1	3.53	34.99	123.44
	HMM	7.61	13.31	101.34
	IBM1-rev	3.16	33.42	105.72
	HMM-rev	11.16	7.03	78.61
Large noisy	IBM1	3.88	36.81	142.85
	HMM	8.77	12.82	121.34
	IBM1-rev	3.39	35.52	120.48
	HMM-rev	13.10	7.08	92.79

9.5.5 Language Models

To evaluate the effect of the language model in the SMT system several LMs were build and used.

- Language models of different sizes trained on the large Xinhua News corpus. This corpus consists of 10 years of news published by the Xinhua news agency in English.

After some cleaning a corpus of 160 million words resulted. Smaller corpora were created by using only a part of the sentences selected evenly from the entire corpus. The size of the LMs ranged from 100 thousand words to 160 million words. These LMs are named Xinhua_K100, ..., Xinhua_M160.

- A language model build on the English part of the bilingual corpus used for training the translation model. Actually, two versions were build, one using the tree-bank, the FBIS corpus, and the Hong Kong News corpus, a second one adding the 2.7 million word Xinhua News corpus.
- A language model using all available data, i.e the English part from all bilingual corpora, including the full LDC Chinese-English dictionary, and the Xinhua news corpus.

The corpora and vocabulary sizes for the different LMs are shown in Table 9.36. The suffix array implementation generates a vari-gram language model. A cross-evaluation corpus is used to calculate the backing-off parameters. The perplexities for this held-out data are also given in the table.

Table 9.36: Corpus size, vocabulary size and tri-gram training set perplexities.

LM	Size	Vocabulary	Perplexity
Xinhua_K100	100,002	12,155	280.62
Xinhua_K200	200,022	17,917	254.39
Xinhua_K500	500,003	29,176	211.10
Xinhua_M001	1,000,028	42,256	185.42
Xinhua_M002	2,000,010	60,784	156.98
Xinhua_M005	5,000,006	99,738	130.12
Xinhua_M010	10,000,010	145,809	112.72
Xinhua_M020	20,000,011	213,292	97.21
Xinhua_M050	50,000,010	346,127	77.76
Xinhua_M100	100,000,017	494,697	62.46
Xinhua_M160	163,750,139	644,311	53.89
LM_TB+FB+HK	10,304,977	79,673	52.38
LM_TB+FB+HK+XS	12,985,502	104,351	56.50
LM_All	177,000,678	669,134	53.70

A number of observations can be made from these numbers:

- The vocabularies become very large. The suffix array based language model implementation allows to use the entire vocabulary without applying any threshold as to the size of the vocabulary or a minimal count a tri-gram has to be seen.

  By tagging the data, i.e. replacing numbers and the simple date expressions by tags, the vocabulary sizes and training set perplexities could be reduced. But as during decoding the LM is applied to the words training on tagged data would actually be a disadvantage.

- A second observation is that despite the growing vocabularies the training perplexities are reduced significantly. This goes parallel with the token/type ratio, i.e. the average number a word is seen in the training corpus. This ratio increases from 8.2 for the Xinhua_K100 data to 254.15 for the Xinhua_M160 data.
- The two language models trained on the English parts of the bilingual corpora have a training perplexity which is about the same as the largest Xinhua language model, even though they are trained on much smaller corpora. The vocabularies of these two corpora are significantly smaller than the vocabulary of the 10 million word Xinhua corpus. This leads to larger token to type ratios of 129.34 and 124.44.
- Adding the English part from the bilingual corpora to the large Xinhua news corpus results in a language model which has nearly the same training set perplexity as the language model build on the Xinhua data alone.

Of course, these training set perplexities have to be used with some caution, as they depend on the cross-evaluation corpus, which was different for the different LMs. The entire training data was in each case split into the proper training data and the held-out data. These training set perplexities therefore reflect the homogeneity of the corpus not necessarily how good the LM generalizes to new data.

To evaluate the different language models the test set perplexities for the Dec-2001 test set was calculated. The tri-gram perplexities for each of the 4 reference translations and for the combined set are given in Table 9.37.

Table 9.37: 3-gram test set perplexities for the Dec-2001 test data and for different language models.

	ref12	ref16	ref17	ref21	allRef
Xinhua_K100	299.04	322.38	432.03	376.49	353.37
Xinhua_K200	279.83	300.23	403.69	355.49	330.93
Xinhua_K500	243.38	255.04	374.06	317.81	292.72
Xinhua_M001	221.46	230.27	346.05	291.18	267.41
Xinhua_M002	195.22	206.49	314.83	266.36	240.79
Xinhua_M005	176.13	181.93	292.92	237.61	217.02
Xinhua_M010	157.93	165.30	273.23	213.58	197.24
Xinhua_M020	146.38	151.63	256.36	199.81	183.34
Xinhua_M050	134.11	137.95	243.43	184.54	169.52
Xinhua_M100	125.91	128.50	231.80	173.60	159.49
Xinhua_M160	123.02	125.85	228.98	170.04	156.44
LM_TB+FB+HK	225.66	241.81	374.67	295.87	278.38
LM_TB+FB+HK+XS	185.43	193.93	306.54	238.75	226.13
LM_all	117.45	118.86	217.20	158.42	147.81

There seems to be a marked difference between the four human translations. The first two are obviously nearer in style to the English used in the different news corpora. As expected, there is a strong correlation between the size of the corpus used to train

the language model and the test set perplexity. What is remarkable, however, is the significantly higher test set perplexity of the language models trained on the bilingual data, especially for the LM_TB+FB+HK language model. Now the 10 million word Xinhua language model outperforms the 10.3 million words LM_TB+FB+HK language model by a relative reduction of 29% in test set perplexity. Notice that the LM_all language model gives an significantly lower perplexity then the largest Xinhua LM, even though the differences in the training set perplexity is minimal.

Table 9.38: Test set perplexities for the Dec-2001 test data.

LM	1-gram	2-gram	3-gram
Xinhua_K100	805.74	363.18	353.37
Xinhua_K200	868.02	344.16	330.93
Xinhua_K500	920.95	310.80	292.72
Xinhua_M001	954.55	289.67	267.41
Xinhua_M002	975.88	267.24	240.79
Xinhua_M005	999.40	248.77	217.02
Xinhua_M010	1014.22	234.68	197.24
Xinhua_M020	1021.94	224.88	183.34
Xinhua_M050	1036.50	214.20	169.52
Xinhua_M100	1044.46	208.11	159.49
LM_TB+FB+HK	1128.31	304.48	278.38
LM_TB+FB+HK+XS	1063.59	259.68	226.13
LM_all	1041.13	200.08	147.81

Table 9.38 gives the test set perplexities for uni-gram, bi-gram, and tri-gram language models. As expected, there is a significant reduction in perplexity when going from a uni-gram to a bi-gram and then to a tri-gram language model. The uni-gram perplexities grow with more training data. This is due to the growing vocabulary. For bi- and tri-gram LMs we observe the expected reduction in test set perplexities. And now the two LMs built from the English part of the bilingual corpora has much higher perplexities than the large Xinhua LMs.

9.5.6 The effect of the Language Model

To study the effect of the language model translations on translation quality two sets of experiments were performed. The first one was an ablation study, i.e. using corpora of different sizes to build language models. A second experiment was run to see to what extent different scaling of the language model against the translation model affect the translation scores.

9.5.6.1 Language Model Scaling

Table 9.39 shows the effect of giving different weight to the language model probabilities with respect to the translation probabilities. A 10 million language model and a 100

million language model have been used. The translation model for this experiment has been trained on the small data. The LDC dictionary, the IBM1 lexicon and phrase translations extracted from the reverse HMM alignment were used.

Table 9.39: Effect of language model scaling factor for small data track.

Scaling	Xinhua_M010	Xinhua_M100
0.5	6.6980	6.7597
0.6	6.7250	6.7748
0.7	6.7437	6.7839
0.8	6.7384	6.7723
0.9	6.7256	6.7706
1.0	6.7224	6.7625

As can be seen, there is a small dependency of the translation quality on the language model scaling factor. The optimal value is 0.7 for the small data track for both language models.

9.5.6.2 Language Model Ablation Study

The language models built from different sizes of Xinhua news data were used in the ablation experiment. For the small data track experiment the 10K LDC dictionary, the IBM1 dictionary, and the phrase translations extracted from the HMM alignment were used. Translation model probabilities and language model probabilities were weighted equally. The results are shown in Table 9.40.

Table 9.40: Effect of language model size for large data track.

LM	MTeval	
K100	6.0763	6.1755
K200	6.3349	6.4768
K500	6.4700	6.8847
M001	6.5929	7.0684
M002	6.6769	7.1988
M005	6.6980	7.3913
M010	6.7437	7.5135
M100	6.7839	7.7197

Chapter 10

Conclusion

10.1 Summary

The goal of this work was to combine aspects from different data-driven machine translation approaches like statistical machine translation, example-based machine translation, translation based on finite state transducers or bilingual grammars into a unified approach. The main achievements are:

- A new machine translation approach has been formulated based on cascaded finite state transducers. This formulation is embedded into the Bayesian framework for statistical machine translation by interpreting the emission probabilities as translation probabilities. This approach allows for a flexible combination of fully automatically generated knowledge source with semi-automatically or manually generated knowledge sources.
- The standard HMM-based alignment model has been extended to the alignment of graphs thereby incorporating the hierarchical structure generated by the cascaded transducers into the training of the alignment.
- The statistical machine translation system developed in the work has been tested on a speech translation task where only a very small bilingual training corpus was available. Using the flexibility of the system, which allows to incorporate additional knowledge sources like additional dictionaries, a performance comparable to a state-of-the art knowledge-based system could be achieved. This is the first detailed comparison of statistical and knowledge-based translation in a small data application for which the knowledge-based system has been specifically designed.
- The translation system was applied to a large data task. A detailed analysis showed to what extend test data is covered by the training data in terms of vocabulary and phrases. Phrase-to-phrase translations can be extracted from the Viterbi-path of the word-to-word alignment. To make word translations and phrase translations comparable phrase translation probabilities are calculated on the basis of the word translation probabilities. Using phrase translations extends the statistical machine

translation system into the direction of example based machine translation. However, using probabilities estimated from the bilingual corpus gives significantly better results than example based translation which used heuristics to score translation hypotheses.

10.2 Outlook

Statistical machine translation has been shown to be widely applicable and to achieve comparable or even better performance than example-based and knowledge-based machine translation even in small data applications for which the applicability of the statistical approach has been denied. On the other side, statistical machine translation has still considerable shortcomings, especially in dealing with structures of languages which can be described in a small set of rule, but which have a large variability in terms of vocabulary. This leads to the desire of incorporating more structure into the statistical approach. The translation approach developed in this work is one way of incorporating more structure. However, the long term goal is to acquire this structural information in an automatic way. Grammar learning is a difficult problem, learning of bi-lingual grammars even more so. Smaller steps, like named entity detection and translation or noun phrase translation are already within reach and can be incorporated into the transducer based translation approach.

The effect of the language model has been shown especially for large vocabulary text translation. Using not only the data from the bilingual corpus to train the language model, but additional data gives a significant reduction in test set perplexity and also a small but significant improvement in translation quality. Language model adaptation has been used successfully in other areas of natural language processing, especially speech recognition. It is to be expected that first translating with a general language model, using this initial translation to select appropriate data for building a specific language model and retranslating with this language model will lead to some improvement in translation quality.

Speech translation is typically realized by translating the first best hypothesis from the speech recognizer. Attempt towards a tighter integration of speech recognition and translation have not been very successful so far. This is still an area which needs further investigations, especially as speech translation becomes more important in dialog systems. The system architecture proposed in this work is particularly suited for studying ways of integrating speech recognition and translation as the word graph generated from the speech recognizer can be used directly to construct the translation graph.

Bibliography

[Alshawi et al. 1998] Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas, "Automatic acquisition of hierarchical transduction models for machine translation," in *COLING-ACL '98: Annual Conf. of the Association for Computational Linguistics and 17th Int. Conf. on Computational Linguistics 1998*, Montreal, Quebec, Canada, August 1998, vol. 1, pp. 41–47.

[Alshawi 1996] Hiyan Alshawi, "Head automata and bilingual tiling: Translation with minimal representations," Preprint-Server für Computational Linguistics, 1996.

[Amengual and Vidal 1998] J. C. Amengual and E. Vidal, "Efficient error-correcting viterbi parsing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20(10), pp. 1109–1116, October 1998.

[Amengual et al. 1997] J. C. Amengual, J. M. Benedi, F. Casacuberta, A. Castano, A. Castellanos, D. Llorens, A. Marzal, F. Prat, E. Vidal, and J. M. Vilar, "Using categories in the Eutrans system," in *Proceedings of the Spoken Language Translation Workshop (EACL97) 1997*, Madrid, Spain, July 1997, pp. 44–53.

[Amengual et al. 2000] J. C. Amengual, J. M. Benedi, F. Casacuberta, A. Castano, A. Castellanos, V. M. Jimenez, D. Llorens, A. Marzal, M. Pastor, F. Prat, E. Vidal, and J. M. Vilar, "The Eutrans-I speech translation system," *Machine Translation, Special Issue, forthcoming*, 2000.

[Berger et al. 1994] Adam L. Berger, Peter F. Brown, Stephen A. Della Pietra, et al., "The candide system for machine translation," in *Proc. , ARPA Workshop on Human Language Technology 1994*, 1994, pp. 157–162.

[Brants 1999a] Thorsten Brants, "Cascaded markov models," in *Proc. of 9th Conference of the European chapter of the Association for Computational Linguistics, EACL-99 1999*, June 1999.

[Brants 1999b] Thorsten Brants, *Tagging and Parsing with Cascaded Markov Models: Automation of Corpus Annotation*, Ph.D. thesis, Universität des Saarlandes, April 1999.

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

[Brown et al. 1993a] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer, "The mathematics of statistical machine translation: Parameter estimation," *Computational Linguistics*, vol. 19, no. 2, pp. 263–311, 1993.

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

[Brown 1996] Ralf D. Brown, "Example-based machine translation in the pangloss system," in *COLING '96: The 16th Int. Conf. on Computational Linguistics 1996*, Copenhagen, Denmark, August 1996, pp. 169–174.

[Brown 1999] Ralf D. Brown, "Adding linguistic knowledge to a lexical example-based translation system," in *Eighth International Conference on Theoretical and Methodological Issues in Machine Translation 1999*, Chester, UK, August 1999, pp. 22–32.

[Dagan et al. 1993] Ido Dagan, Ken Church, and William A. Gale., "Robust bilingual word alignment for machine aided translation," in *Proceedings of the Workshop on Very Large Corpora 1993*, 1993, pp. 1–8.

[Hutchins and Somers 1992] W. John Hutchins and Harold L. Somers, *An Introduction to Machine Translation*, Academic Press, Cambridge, 1992.

[Jekat et al. 1999] Susanne J. Jekat, Lorenzo Tessiore, and Brigitte Lause, "Offline evaluation of verbmobil 0.7," Tech. Rep. Memo 143, DFKI, 1999.

[Kay and Röscheisen 1993] Martin Kay and Martin Röscheisen, "Text-translation alignment," *Computational Linguistics*, vol. 19, no. 1, pp. 121–142, 1993.

[Lavie et al. 2001] A. Lavie, C. Langley, A. Waibel, F. Pianesi, G. Lazzari, P. Coletti, L. Taddei, and F. Balducci, "Architecture and design considerations in nespole!: a speech translation system for e-commerce applications," in *Proceedings of HLT: Human Language Technology 2001*, San Diego, CA, March 2001.

[Nagao 1984] Makoto Nagao, "A framework of a mechanical translation between japanese and english by analogy prinicple," in *Artificial and Human Intelligence*, A. Elithorn and R. Banerji, Eds., chapter 11, pp. 173–180. Elsevier Science Publishers, 1984.

[Ney et al. 2000] Hermann Ney, Sonja Nießen, Franz Josef Och, Hassan Sawaf, Christoph Tillmann, and Stephan Vogel, "Algorithms for statistical translation of spoken language.," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 24–36, January 2000.

[Nießen et al. 1998] Sonja Nießen, Stephan Vogel, Hermann Ney, and Christoph Tillmann, "A DP based Search Algorithm for Statistical Machine Translation," in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics 1998*, Montréal, P.Q., Canada, August 1998, pp. 960–967.

[Nießen et al. 2000] Sonja Nießen, Franz Josef Och, Gregor Leusch, and Hermann Ney, "An evaluation tool for Machine Translation: Fast evaluation for MT research," in *Proceedings of the 2nd International Conference on Language Resources and Evaluation 2000*, Athens, Greece, May 2000, pp. 39–45.

[Nirenburg and Frederking 1994] Sergei Nirenburg and Robert Frederking, "Toward multi-engine machine translation," in *Human Language Technology 1994*, Plainsboro, New Jersey, 1994, pp. 147–151.

[Och and Weber 1998] Franz Josef Och and Hans Weber, "Improving statistical natural language translation with categories and rules.," in *Proc. of the 35th Annual Conf. of the Association for Computational Linguistics and the 17th Int. Conf. on Computational Linguistics 1998*, Montreal, Canada, August 1998, pp. 985–989.

[Och et al. 1999a] Franz Josef Och, Christoph Tillmann, and Hermann Ney, "Improved Alignment Models for Statistical Machine Translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora 1999*, University of Maryland, College Park, Maryland, June 1999, pp. 20–28.

[Och et al. 1999b] Franz Josef Och, Christoph Tillmann, and Hermann Ney, "Improved alignment models for statistical machine translation," in *In Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora 1999*, University of Maryland, College Park, MD, USA, June 1999, pp. 20–28.

[Och 1999] Franz Josef Och, "An efficient method to determine bilingual word classes," in *EACL '99: Ninth Conf. of the Europ. Chapter of the Association for Computational Linguistics 1999*, Bergen, Norway, June 1999.

[Papinini et al. 2001] Kishore Papinini, Salim Roukos, Todd Ward, and Wei-Jing Zhu, "Bleu: a method for automatic evaluation of machine translation," Tech. Rep. RC22176(W0109-022), September 17, 2001, IBM, 2001.

[Sawaf et al. 2000] Hassan Sawaf, Kai Schütz, and Hermann Ney, "On the use of grammar based language models for statistical machine translation," in *6th International Workshop on Parsing Technologies 2000*, February 2000, pp. 231–241.

[Tillmann et al. 1997a] C. Tillmann, S. Vogel, H. Ney, and A. Zubiaga, "A DP-Based Search using Monotone Alignments in Statistical Translation," in *Proceedings of the ACL/EACL '97, Madrid, Spain 1997*, July 1997, pp. 289–296.

[Tillmann et al. 1997b] Christoph Tillmann, Stephan Vogel, Hermann Ney, and Alex Zubiaga, "A DP-based search using monotone alignments in statistical translation.," in *Proc. 35th Annual Conf. of the Association for Computational Linguistics 1997*, Madrid, Spain, July 1997, pp. 289–296.

[Tillmann et al. 1997c] Christoph Tillmann, Stephan Vogel, Hermann Ney, Alex Zubiaga, and Hassan Sawaf, "Accelerated DP based search for statistical translation.," in *Fifth European Conf. on Speech Communication and Technology 1997*, Rhodos, Greece, September 1997, pp. 2667–2670.

[Tillmann 2001] Christoph Tillmann, *Word Re-ordering and Dynamic Programming based Search Algorithm for Statistical Machine Translation.*, Ph.D. thesis, RWTH Aachen, 2001.

[Vidal et al. 2000] E. Vidal, A. di Carlo, and H. Ney, "Example-Based Language Translation Systems," Final report of the EuTrans project (project number 30268), July 2000.

[Vidal 1997] Enrique Vidal, "Finite-state speech-to-speech translation," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing 1997*, 1997, vol. 1, pp. 111–114.

[Vogel et al. 1996] Stephan Vogel, Hermann Ney, and Christoph Tillmann, "HMM-based word alignment in statistical translation," in *COLING '96: The 16th Int. Conf. on Computational Linguistics 1996*, Copenhagen, August 1996, pp. 836–841.

[Vogel et al. 2000a] Stephan Vogel, Sonja Nießen, and Hermann Ney, "Automatic Extrapolation of Human Assessment of Translation Quality," in *2nd International Conference on Language Resources and Evaluation: Proceedings of the Workshop on Evaluation of Machine Translation 2000*, Athens, Greece, May 2000, pp. 35–39.

[Vogel et al. 2000b] Stephan Vogel, Franz Josef Och, Christoph Tillmann, Sonja Nießen, Hassan Sawaf, and Hermann Ney, "Statistical methods for machine translation," in *Verbmobil: Foundations of Speech-to-Speech Translation*, Wolfgang Wahlster, Ed., pp. 377–393. Springer Verlag: Berlin, Heidelberg, New York, 2000.

[Wahlster 2000] Wolfgang Wahlster, Ed., *Verbmobil: Foundations of Speech-to-Speech Translation*, Springer Verlag: Berlin, Heidelberg, New York, 2000.

[Wang and Waibel 1997] Ye-Yi Wang and Alex Waibel, "Decoding algorithm in statistical translation," in *Proc. 35th Annual Conf. of the Association for Computational Linguistics 1997*, Madrid, Spain, July 1997, pp. 366–372.

[Wang 1998] Ye-Yi Wang, *Grammar Inference and Statistical Machine Translation*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, CMU–LTI–98–1, 1998.

[White and O'Connell 1994] John S. White and Theresa A. O'Connell, "Evaluation in the arpa machine translation program: 1993 methodology," in *Human Language Technology 1994*, Plainsboro, New Jersey, März 1994, pp. 135–138.

[Wu 1994] Dekai Wu, "Aligning a parallel english-chinese corpus statistically with lexical criteria," in *Proc. of the 32nd Annual Conf. of the Association for Computational Linguistics 1994*, 1994.

[Wu 1995a] D. Wu, "An Algorithm for Simultaneously Bracketing Parallel Texts by Aligning Words," in *Proceedings of the 33th Annual Conference of the Association for Computational Linguistics, Cambridge, MA 1995*, June 1995, pp. 244–251.

[Wu 1995b] D. Wu, "Grammarless extraction of phrasal translation examples from parallel texts," in *Proc. of the Sixth Int. Conf. on Theoretical and Methodological Issues in Machine Translation 1995*, Leuven, Belgium, 1995.

[Wu 1996] D. Wu, "A Polynomial-Time Algorithm for Statistical Machine Translation," in *Proceedings of the 34th Annual Conference of the Association for Computational Linguistics, Santa Cruz, CA 1996*, June 1996, pp. 152 – 158.