Generative Training and Smoothing of Hierarchical Phrase-Based Translation Models

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker
Stephan Peitz
aus Wuppertal

Berichter: Prof. Dr.-Ing. Hermann Ney
Priv.-Doz. Dr. Alexandre Allauzen

Tag der mündlichen Prüfung: 17. März 2017

Für “Oskar”

Acknowledgments

First, I would like to thank Prof. Dr. Hermann Ney for giving me the opportunity to work and to research at the Lehrstuhl für Informatik 6. I have learnt a lot.

I also would like to thank Dr. Alexandre Allauzen for his interest in my work, agreeing to review this thesis and his extensive feedback.

During my studies in computer science at the RWTH Aachen University, I had never expected that machine translation would be the topic I would write a PhD thesis about. With an interest in pattern recognition, I just attended some lectures by Prof. Ney and was later brainwashed by Daviiid Vilar and Daniel Stein to write my diploma thesis about hierarchical phrase-based machine translation (aka "irgendwas mit Syntax"). Thank you guys for giving me that opportunity which has finally resulted in this thesis.

This story is also strongly connected with my friend, flat-mate, room-mate, colleague and co-author Markus Freitag. We both survived our studies in computer science, we both did our diploma theses at i6 and we both managed to write a PhD thesis - well done!

Special cheers go to my long-term office-mate Joern Wuebker (aka Mr. Forced Alignment aka Mr. Peitz) who supported me with his deep knowledge in machine translation and - of course - with his leave-one-out framework.

Even if we had only little research-related interaction, I would like to give a shout-out to Malte Nuhn for having a lot of fun during work and conferences. Special thanks also to Andy Guta: it was not less fun, even if we had research-related interaction. JTP, Tamer Alkhouli, Weiyue Wang and Parnia Bahar: Thanks for being my colleagues and proof-reading my thesis! And cheers to Saab Mansour for knowing where the party is, Matthias Huck for being always well prepared and Arne Mauser for supporting me with useful ideas. Furthermore, I thank my bachelor and master students Leonard Dahlmann and Yifeng Lu.

I also would like to thank my ASR colleagues Markus Nussbaum-Thom, Simon Wiesler and Pavel Golik for their tirelessly support in setting up an ASR system. It was a mess!

Thanks to Stefan Koltermann and Kai Frantzen for maintaining the infrastructure where the experiments for this thesis were conducted. And thanks to the secretaries, in particular Steffi Jansen and Andrea Kierdorf, for interesting conversation and their support in formal matters.

Last but not least, very special thanks go to my family. I have really appreciated their encouraging support during the last six years.

And Doro, my wife, I can not thank you enough for your endless support, your encouraging words, your empathy and your love. Talking with you in hard times was very important to get this done! Thank you!

Abstract

Hierarchical phrase-based translation is a common machine translation approach for translating between languages with significantly different word order. The focus of the first part of this thesis is set on smoothing and training of the translation models used in hierarchical translation. Additionally, we present an improved implementation of the search algorithm and show that our implementation is competitive compared to other state-of-the-art hierarchical phrase-based translation engines. Within the second part of this work, we apply hierarchical phrase-based translation in the context of spoken language translation.

In the state-of-the-art hierarchical translation model extraction process, translation rules and their corresponding translation probabilities are obtained from word-aligned training data by applying simple heuristics. A common issue is that even if a large set of training data is provided, the resulting translation model may suffer from data sparseness. Smoothing is an approach to remedy this problem and is well-known from other natural language processing tasks (e.g. language modeling). The goal of smoothing applied in the scope of machine translation is to model rarely seen translation rules better. In this thesis, we investigate and compare different smoothing techniques for hierarchical phrase-based translation.

Furthermore, the extraction and translation processes are two separated steps. Therefore, the extraction does not take into account whether the obtained translation rules are actually needed in the translation process. To learn whether a translation rule is relevant for the translation process, we pursue the approach of force-decoding the training data. Given a sentence pair of the training data, the translation of the source sentence is constrained to produce the corresponding target sentence. The applied translation rules are then determined and the corresponding translation probabilities re-estimated. In order to be able to translate a large set of training data, an efficient and fast framework is needed. In this work, we introduce such a framework for re-estimating hierarchical translation models. This approach enables us to obtain smaller translation models while simultaneously improving the translation quality. We further compare our proposed scheme with another state-of-the-art translation model training approach, namely discriminative training, on a large-scale Chinese→English translation task.

Spoken language translation is the task of translating automatically transcribed speech. Since most automatic speech recognition systems provide transcriptions without punctuation marks and case information, this information has to be re-introduced before the actual translation takes place. In this work, we show that performing punctuation prediction and re-casing by applying a machine translation system helps to improve the translation quality. In particular, we propose to apply hierarchical translation rather than phrase-based translation for this task. Finally, experiments were conducted on a large-scale English→French spoken language translation task.

All methods described in this thesis have been made freely available to the research community as they were integrated into the open-source translation toolkit *Jane*.

Kurzfassung

Die hierarchische phrasenbasierte Übersetzung ist ein bewährter Ansatz in der maschinellen Übersetzung, um zwischen Sprachen mit unterschiedlichen Wortstellungen zu übersetzen. Der erste Teil dieser Dissertation behandelt das Thema der Glättung und des Trainings von Übersetzungsmodellen, die in der hierarchischen Übersetzung verwendet werden. Zusätzlich präsentieren wir eine verbesserte Implementierung des Suchalgorithmus und zeigen, dass diese konkurrenzfähig im Vergleich mit anderen modernen Implementierungen ist. Im zweiten Teil dieser Arbeit schlagen wir vor, hierarchische phrasenbasierte Übersetzung im Kontext der Übersetzung von gesprochener Sprache zu verwenden.

Im modernen hierarchischen phrasenbasierten Übersetzungsmodell-Extraktionsprozess werden Übersetzungsregeln und die dazugehörigen Übersetzungswahrscheinlichkeiten aus wortalinierten Trainingsdaten basierend auf einfachen Heuristiken extrahiert. Ein bekanntes Problem ist, dass auch wenn eine große Menge an Trainingsdaten zur Verfügung steht, das erzeugte Übersetzungsmodell unter Datenkargheit leiden kann. Glättung ist ein Ansatz, um dieses Problem zu lösen, und findet bereits Anwendung in anderen Sprachverarbeitungsgebieten (wie zum Beispiel in der Sprachmodellierung). In Rahmen der maschinellen Übersetzung sollen selten gesehene Übersetzungsregeln besser modelliert werden. In dieser Dissertation untersuchen und vergleichen wir verschiedene Glättungstechniken für die hierarchische phrasenbasierte Übersetzung.

Ein weitere Problem ist die Separation des Extraktions- und des Übersetzungsprozesses. Die Extraktion beachtet nämlich nicht, ob eine extrahierte Übersetzungsregel wirklich nützlich im Übersetzungsprozess ist. Um zu lernen, ob eine Übersetzungsregel relevant ist, verfolgen wir den Ansatz der Übersetzung der Trainingsdaten. Dabei wird die Übersetzung eines Quellsatzes gelenkt, so dass der zugehörige Zielsatz generiert wird. Die verwendeten Übersetzungsregeln werden dann gespeichert und die dazugehörigen Übersetzungswahrscheinlichkeiten neu berechnet. Um überhaupt große Mengen von Trainingsdaten zu übersetzen, wird ein effiziente und schnelle Implementierung benötigt. In dieser Arbeit stellen wir eine solche Implementierung zur Neuberechnung von hierarchischen Übersetzungsmodellen vor. Dieser Ansatz ermöglicht uns, kleinere Übersetzungsmodelle zu lernen und gleichzeitig die Übersetzungsqualität zu verbessern. Des Weiteren vergleichen wir unseren Ansatz mit einer anderen modernen Übersetzungsmodell-Trainingsmethode, nämlich das diskriminative Training, im Rahmen einer umfangreichen Evaluierung. Das Sprachpaar ist Chinesisch→Englisch.

Die Übersetzung von gesprochener Sprache verbindet automatische Spracherkennung mit maschineller Übersetzung. Da die meisten modernen Spracherkennungssysteme Erkennung ohne Interpunktion und Groß- und Kleinschreibung liefern, müssen diese Information vor dem eigentlichen Übersetzungprozess wieder eingefügt werden. In dieser Arbeit zeigen wir, dass die Modellierung von Interpunktion und Groß- und Kleinschreibung als maschinelle Übersetzung die Übersetzungsqualität verbessern kann. Wir schlagen außerdem vor, dafür ein hierarchisches phrasenbasiertes

Übersetzungssytem zu verwenden, und vergleichen dies mit anderen Ansätzen in einer umfangreichen Evaluierung. Das Sprachpaar ist Englisch→Französisch.

Alle Methoden, die in dieser Dissertation beschrieben wurden, sind der Forschungsgemeinschaft frei zugänglich, da diese in die Open-Source-Software *Jane* integriert worden sind.

Contents

1. Introduction

Around 286 distinct languages are spoken in Europe [Lewis & Simons$^+$ 15] and twenty-four official languages exist in the European Union [European Commission 15]. However, a survey has shown that 46% of the citizens in the European Union are not able to communicate in a language different from their native language [European Commission 12]. *Machine translation* could help to cross this language divide.

Before introducing the research topics of this thesis, we describe the idea of machine translation and its different approaches in Section 1.1. In this work, the focus is set on *hierarchical phrase-based translation*. In Section 1.2, we motivate this approach and give an overview of the related research topics which we will investigate in this thesis. Moreover, we introduce *spoken language translation* in Section 1.3 and describe the application of hierarchical phrase-based translation in this field of research. An outline of this work and an overview of publications published during the course of this thesis are given in Section 1.4 and Section 1.5, respectively.

1.1 Machine Translation

Machine translation (MT) is the task of translating automatically from one (source) language to another (target) language using computers. This idea was already proposed in the 1950's [Bar-Hillel 51, IBM 54]. In these first attempts, handcrafted translation rules (generated by interpreters) were applied by an algorithm to translate from Russian into English. Nowadays, most users apply tools such as Google Translate[1] to translate a text in a foreign language into their native language. However, considering all official languages of the European Union, 264 different translation engines are needed. To provide such a translation service, an approach which learns automatically to translate without any human interaction is required

In statistical machine translation (SMT), the translation rules are learned automatically from data rather than handcrafted by interpreters. Thus, translation engines for many different languages can be provided without human effort. The first attempts to SMT presented in [Brown & Cocke$^+$ 88, Brown & Cocke$^+$ 90, Brown & Della Pietra$^+$ 93] propose to translate a given sentence stepwise word by word. The described models (in literature often referred to as IBM models) have their foundations in statistics. However, in recent years, the concept of *phrase-based translation* as described in [Zens & Och$^+$ 02] has been successfully applied (e.g. Google Translate is based on this approach). Experimental results show improvements in terms of translation quality over the IBM models [Koehn & Och$^+$ 03].

The idea of phrase-based translation is to translate *phrases* (i.e. sequences of words) of the source language into phrases of the target language. Thus, local contextual information is preserved in the translation process. For example, given the German sentence "Ich habe Hunger", a simple

[1] https://translate.google.com/

word-based translation system may translate this sentence into "I have hunger". However, phrase-based translation is able to translate the three German words at once since this sequence of words is stored as a single rule in the translation model. The corresponding English sequence consists of the words "I'm hungry".

In contrast to the IBM models, phrase-based translation is usually divided into two steps, namely *extraction* and *translation*. First, translation rules, which are phrase pairs consisting of a source phrase and a target phrase, are heuristically extracted from bilingual translation examples and assigned a probability based on their frequencies. The set of probabilistic translation rules is called a *translation model*. Note that these rules do not follow any explicit linguistic constraint but are just pairs of consecutive source and target words, respectively. Secondly, given a sentence in the source language, an algorithm searches for the best translation in the target language (i.e. the translation with the highest probability) applying the automatically learned translation rules.

1.2 Hierarchical Phrase-Based Translation

A new type of translation rules is suggested in [Chiang 05]. *Hierarchical phrase-based translation* applies a more abstract type of translation rules in order to model long-range dependencies between words. These *hierarchical rules* are discontinuous phrases with "gaps".

As an example, we extend the German sentence from the previous section: "Ich habe gestern Abend Hunger gehabt .". In German, a verb phrase can be split up into two parts (here: "habe" and "gehabt"). While the first part of the verb phrase follows the subject "Ich", the latter is put at the end of the sentence. In English, the grammatical structure of a sentence is different, e.g. a verb phrase cannot be split up. This specific grammatical construction can only be captured by a very long phrase. However, such a phrase does not capture the grammatical construction in general and has therefore a weak generalization power. Furthermore, this long phrase might not be obtained from the translation examples. In contrast, hierarchical phrase-based translation allows for rules with gaps (denoted by "[...]"). Such a rule translates "Ich habe [...] Hunger gehabt" into "I was hungry [...]". The gaps can be filled with any other rule, e.g. "gestern Abend" and "last night", respectively. An important observation is that this hierarchical rule could be also obtained from the example "Ich habe heute morgen Hunger gehabt" and its translation "I was hungry this morning".

As the example indicates, hierarchical phrase-based translation is usually preferred if the word orders in the source and target language differ significantly, e.g. German and English. However, due to the additional set of translation rules and the higher complexity of the search algorithm, the computational effort is higher in comparison to phrase-based translation. In this thesis, we will present improvements of the implementation resulting in a lower run-time and higher memory efficiency.

1.2.1 Generative Training and Smoothing of Translation Models

As mentioned in the previous section, bilingual translation examples are required for extracting translation rules. However, to obtain these rules, heuristics without any foundation in statistics are applied. Thus, even if millions of sentences are provided, the resulting translation model may suffer from data sparseness. *Smoothing* is a well-known approach to remedy this problem and is also applied in other natural language processing tasks. In language modeling, smoothing allows to assign probabilities to unseen events. However, the goal of smoothing applied in the scope of machine translation is to better model rarely seen rules. In recent years, the focus was set on smoothing of phrase-based translation models. In this thesis, we will investigate and compare different smoothing techniques for hierarchical phrase-based translation.

Furthermore, as the extraction and translation processes are two separated steps, extraction does not take into account whether an obtained translation rule is actually needed in the translation process. This issue is even more crucial for the hierarchical paradigm as a larger set of rules is obtained compared to phrase-based translation. To learn whether a rule is relevant for the translation process, one approach is to run the search algorithm on same set of bilingual translation examples that are used for obtaining the translation rules. This method is called *generative translation model training*. A training framework for hierarchical phrase-based translation will be presented in this work.

1.3 Spoken Language Translation

Another aspect of MT is the type of input which is to be translated. A field of growing interest is the task of *spoken language translation* (SLT) which combines automatic speech recognition with MT (the most popular and related product is the Skype Translator[2]). While most MT engines are designed for translating written text that is grammatically correct, includes proper punctuation and is segmented into sentences, having speech as input has to be handled differently. Spoken language may include speech disfluencies (e.g. fillers and repeated utterances) and punctuation is done implicitly by the speaker. Furthermore, even if automatic speech recognition performs quite well nowadays, recognition errors may still occur. Thus, either the MT system or the input has to be adapted. The advantage of modifying the input by inserting punctuation marks and case information as well as deleting disfluencies is that the actual MT system has not to be changed. In this thesis, we suggest to perform this adaptation of the input by applying a statistical machine translation system. In particular, we propose to apply a hierarchical phrase-based translation system in order to model long-range dependencies between words and punctuation marks using hierarchical rules.

Another important line of research in the context of SLT is sentences segmentation. As most automatic speech recognition systems do not provided proper segmented text, the recognized words have to be divided into sentences. This work does not cover this topic and we assume that the segmentation of the speech recognition output is given and corresponds to at least sentence-like units.

1.4 About this Thesis

This thesis includes two research topics, namely *Generative Training and Smoothing of Hierarchical Phrase-Based Translation Models*, which led to the title of this work, and *Improving Spoken Language Translation*. The corresponding scientific goals are formulated in Chapter 2.

As an introduction to the first topic, the concept of hierarchical phrase-based translation is described in Chapter 4. In addition, improvements of the implementation of the search algorithm in terms of run-time and memory efficiency are presented as well. In Chapter 5, we introduce our framework for generative training of hierarchical translation models. Furthermore, we present several smoothing techniques for hierarchical translation models in Chapter 6. Finally, experimental results of the presented improvements and methods are shown in Chapter 7.

The second topic is covered in Chapter 8 where the focus is set on improving spoken language translation. In particular, we model the adaptation of automatically transcribed speech input as machine translation. The idea is to translate monolingually from automatic transcribed speech without punctuation marks and case information to case-sensitive text with proper punctuation.

[2] http://www.skype.com/en/translator-preview/

Moreover, this approach is able to reduce the amount of disfluencies and recognition errors. As an application of hierarchical phrase-based translation, we propose to employ this approach for the purpose of speech input adaptation.

Finally, this thesis will be concluded in Chapter 9 by showing how the scientific goals were achieved.

1.5 Previously Published

The following scientific publications have been published at peer-reviewed conferences during the course of this thesis:

- Training of Hierarchical Translation Models
  - [Peitz & Mauser$^+$ 12] Forced Derivations for Hierarchical Machine Translation
  - [Peitz & Vilar$^+$ 14] Simple and Effective Approach for Consistent Training of Hierarchical Phrase-based Translation Models
  - [Wuebker & Muehr$^+$ 15] A Comparison of Update Strategies for Large-Scale Maximum Expected BLEU Training
- Spoken Language Translation
  - [Peitz & Freitag$^+$ 11] Modeling Punctuation Prediction as Machine Translation
  - [Peitz & Wiesler$^+$ 12] Spoken Language Translation Using Automatically Transcribed Text in Training
  - [Peitz & Freitag$^+$ 14] Better Punctuation Prediction with Hierarchical Phrase-Based Translation
- RWTH Aachen University's Translation Toolkit *Jane*
  - [Stein & Vilar$^+$ 11] A Guide to Jane, an Open Source Hierarchical Translation Toolkit
  - [Wuebker & Huck$^+$ 12] Jane 2: Open Source Phrase-based and Hierarchical Statistical Machine Translation
  - [Huck & Peter$^+$ 12] Hierarchical Phrase-Based Translation with Jane 2
- Syntactical Enhancements
  - [Stein & Peitz$^+$ 10] A Cocktail of Deep Syntactic Features for Hierarchical Machine Translation
  - [Vilar & Stein$^+$ 10b] If I Only Had a Parser: Poor Man's Syntax for Hierarchical Machine Translation
- Modeling
  - [Huck & Peitz$^+$ 12a] Discriminative Reordering Extensions for Hierarchical Phrase-Based Machine Translation
  - [Wuebker & Peitz$^+$ 13b] Improving Statistical Machine Translation with Word Class Model
  - [Freitag & Feng$^+$ 13] Reverse Word Order Models
- International Evaluation Campaigns

  - [Mansour & Peitz$^+$ 10] The RWTH Aachen Machine Translation System for IWSLT 2010
  - [Huck & Wuebker$^+$ 11] The RWTH Aachen Machine Translation System for WMT 2011
  - [Freitag & Leusch$^+$ 11] Joint WMT Submission of the QUAERO Project
  - [Feng & Schmidt$^+$ 11] The RWTH Aachen System for NTCIR-9 PatentMT
  - [Boudahmane & Buschbeck$^+$ 11] Advances on Spoken Language Translation in the Quaero Program
  - [Wuebker & Huck$^+$ 11] The RWTH Aachen Machine Translation System for IWSLT 2011
  - [Huck & Peitz$^+$ 12b] The RWTH Aachen Machine Translation System for WMT 2012
  - [Freitag & Peitz$^+$ 12] Joint WMT 2012 Submission of the QUAERO Project
  - [Peitz & Mansour$^+$ 12] The RWTH Aachen Speech Recognition and Machine Translation System for IWSLT 2012
  - [Peitz & Mansour$^+$ 13a] Joint WMT 2013 Submission of the QUAERO Project
  - [Peitz & Mansour$^+$ 13b] The RWTH Aachen Machine Translation System for WMT 2013
  - [Freitag & Peitz$^+$ 13] EU-BRIDGE MT: Text Translation of Talks in the EU-BRIDGE Project
  - [Shaik & Tüske$^+$ 13] The RWTH Aachen German and English LVCSR Systems for IWSLT-2013
  - [Wuebker & Peitz$^+$ 13a] The RWTH Aachen Machine Translation Systems for IWSLT 2013
  - [Peitz & Wuebker$^+$ 14] The RWTH Aachen German-English Machine Translation System for WMT 2014
  - [Freitag & Peitz$^+$ 14] EU-BRIDGE MT: Combined Machine Translation
  - [Wuebker & Peitz$^+$ 14] The RWTH Aachen Machine Translation Systems for IWSLT 2014
  - [Freitag & Wuebker$^+$ 14] Combined Spoken Language Translation
  - [Peter & Toutounchi$^+$ 15] The RWTH Aachen German to English MT System for IWSLT 2015

- Other Contributions
  - [Lehnen & Peter$^+$ 13] (Hidden) Conditional Random Fields Using Intermediate Classes for Statistical Machine Translation
  - [Parra Escartín & Peitz$^+$ 14] German Compounds and Statistical Machine Translation. Can they get along?
  - [Freitag & Peter$^+$ 15] Local System Voting Feature for Machine Translation System Combination

2. Scientific Goals

Hierarchical phrase-based translation has become a state-of-the-art approach for translating between languages with significantly different word order, e.g. Chinese and English. Most methods applied in phrase-based MT can also be applied in hierarchical translation, e.g. smoothing [Foster & Kuhn$^+$ 06] and generative training of translation models [Wuebker & Mauser$^+$ 10]. However, a study of several smoothing methods for the hierarchical phrase-based translation approach is still missing and the generative training of hierarchical translation models could not yet be scaled to larger translation tasks. Moreover, different re-estimation schemes for the generative training have not been investigated and a comparison to the discriminative training approach has not been performed so far. Furthermore, hierarchical translation models are usually larger and the complexity of search algorithm is higher compared to phrase-based translation. Thus, the hierarchical approach comes with higher computational costs and besides an efficient search algorithm, an efficient implementation in terms of actual decoding time and memory consumption is needed.

Given that we pursue the following scientific goals in this thesis:

- Developing further RWTH's hierarchical phrase-based translation engine towards a fast and memory-efficient decoder which is competitive with other external state-of-the-art engines
- Systematic study of several smoothing methods for the hierarchical phrase-based translation approach
- Implementation of a generative training framework for hierarchical translation which can be applied on large-scale translation tasks
- Analysis of the effectiveness of generative training of hierarchical translation models in terms of translation quality and size of the translation model
- A comparison of different approaches for re-estimating hierarchical translation models
- A comparison to discriminative training on a large-scale translation task

As an additional application of hierarchical translation, we want to investigate the potential of hierarchical translation rules in the scope of spoken language translation. In recent years, phrase-based translation was successfully applied to perform punctuation prediction [Hassan & Ma$^+$ 07, Ma & Tinsley$^+$ 08, Peitz & Freitag$^+$ 11, Cho & Niehues$^+$ 12]. By employing hierarchical translation rules to model long-range dependencies between words and punctuation marks, punctuation prediction can be further improved. Furthermore, re-casing and disfluency removal can be also done is this step. We therefore pursue the following additional goals:

- Modeling punctuation prediction, re-casing and disfluency removal as hierarchical phrase-based translation

- An extensive comparison with other state-of-the-art approaches on a large-scale spoken language translation task

3. Preliminaries

In this chapter the terminology and notation as well as some theoretical background for a better understanding of the following work are introduced. We will only give a short overview and refer to the corresponding bibliography for more details. The concepts described in this chapter are not part of our contributions.

3.1 Terminology and Notation

Since the process of translation can be considered as a deciphering problem, the search algorithm for finding the best translation is called *decoder*. While we call a human-translated document *reference*, a translation generated by the decoder is denoted as a *hypothesis translation* (or *hypothesis*). The first n translations are stored in an n-best list according to the model score. A collection of documents and their corresponding references will be referred to as a *bilingual corpus*.

For experimental evaluation, we split some portions from our corpus. The larger part is used as *training data* (`train`) for the models. One or several smaller documents, called *development sets* (`dev`) and *test sets* (`test`), are used for parameter tuning and final evaluation, respectively. Note that training, development and test data are disjoint to obtain meaningful results.

Throughout this work, we will denote a source sentence of length J (i.e. the sentence contains J words) as $F = f_1^J = f_1, f_2, \ldots, f_J$ and a target sentence consisting of I words as $E = e_1^I = e_1, e_2, \ldots, e_I$. The n-th sentence pair of the training data is denoted as (F_n, E_n). A sequence of n words is called *n-gram*.

3.2 Statistical Machine Translation

The goal of machine translation is to produce an automatically generated translation of a given source sentence. The main idea of the statistical machine translation (SMT) is to learn translation rules from bilingual training data, to compute the associated probabilities and to build up a translation model without any human interaction. With this kind of translation models, many possible translations are generated for a given sentence. The choice of the best translation is based on statistical decision theory. The standard method to choose a translation is the Bayes decision rule.

The Bayes decision rule tells us to choose the target sentence with the highest probability to minimize the sentence error rate. More formally, for a given source sentence f_1^J which is to be translated into a target sentence e_1^I, we choose the translation $\hat{e}_1^{\hat{I}}$ which maximizes the posterior probability $Pr(e_1^I|f_1^J)$:

$$f_1^J \mapsto \hat{e}_1^{\hat{I}}(f_1^J) = \underset{I, e_1^I}{\operatorname{argmax}} \left\{ Pr(e_1^I|f_1^J) \right\} \tag{3.1}$$

One of the main challenges in statistical machine translation is to properly define a probability distribution $Pr(e_1^I|f_1^J)$ which is estimated from bilingual training data.

In order to allow for independent modeling, the probability distribution $Pr(e_1^I|f_1^J)$ is decomposed by applying the Bayes theorem:

$$\begin{aligned} f_1^J \mapsto \hat{e}_1^{\hat{I}}(f_1^J) &= \underset{I,e_1^I}{\operatorname{argmax}} \left\{ Pr(e_1^I|f_1^J) \right\} && (3.2) \\ &= \underset{I,e_1^I}{\operatorname{argmax}} \left\{ \frac{Pr(e_1^I)Pr(f_1^J|e_1^I)}{Pr(f_1^J)} \right\} && (3.3) \\ &= \underset{I,e_1^I}{\operatorname{argmax}} \left\{ Pr(e_1^I)Pr(f_1^J|e_1^I) \right\} && (3.4) \end{aligned}$$

The probability distribution is decomposed into a translation model $Pr(f_1^J|e_1^I)$, which models the correspondence between source and target sentence, and a language model $Pr(e_1^I)$, which scores the well-formedness of the target sentence. This source-channel approach was introduced by [Brown & Cocke$^+$ 90].

3.3 Log-Linear Modeling

In state-of-the-art machine translation, the posterior probability $Pr(e_1^I|f_1^J)$ is directly modeled [Och & Ney 02]. The *log-linear model combination* considers different models $h_m(f_1^J, e_1^I)$ called *feature functions* (or *features*). Each feature is multiplied with a corresponding scaling factor λ_m.

$$Pr(e_1^I|f_1^J) = \frac{\exp\left(\sum_{m=1}^M \lambda_m h_m(f_1^J, e_1^I)\right)}{\sum_{\tilde{e}_1^I} \exp\left(\sum_{m=1}^M \lambda_m h_m(f_1^J, \tilde{e}_1^I)\right)} \quad (3.5)$$

In addition to the translation and language model, an arbitrary number of features can be added to the log-linear model combination. These features can be probability distributions or just simple heuristics and are mainly estimated from a Viterbi word aligment (Section 3.4) . With the log-linear model combination, the Bayes decision rule is rewritten as following:

$$\begin{aligned} f_1^J \mapsto \hat{e}_1^{\hat{I}}(f_1^J) &= \underset{I,e_1^I}{\operatorname{argmax}} \left\{ Pr(e_1^I|f_1^J) \right\} && (3.6) \\ &= \underset{I,e_1^I}{\operatorname{argmax}} \left\{ \frac{\exp\left(\sum_{m=1}^M \lambda_m h_m(f_1^J, e_1^I)\right)}{\sum_{\tilde{e}_1^I} \exp\left(\sum_{m=1}^M \lambda_m h_m(f_1^J, \tilde{e}_1^I)\right)} \right\} && (3.7) \\ &= \underset{I,e_1^I}{\operatorname{argmax}} \left\{ \exp\left(\sum_{m=1}^M \lambda_m h_m(f_1^J, e_1^I)\right) \right\} && (3.8) \\ &= \underset{I,e_1^I}{\operatorname{argmax}} \left\{ \sum_{m=1}^M \lambda_m h_m(f_1^J, e_1^I) \right\} && (3.9) \end{aligned}$$

The architecture of an SMT system following this approach is illustrated in Figure 3.1.

To learn the scaling factors λ_m of the log-linear model combination, we need a gradient-free optimization strategy since Equation 3.9 includes an argmax operation. In this work, we apply the

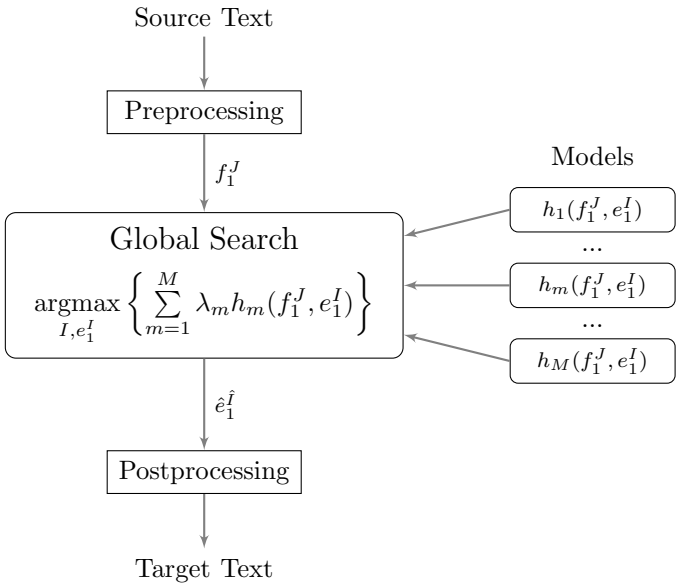


Figure 3.1: Illustration of an SMT system architecture following the log-linear modeling approach.

minimum-error-rate-training (MERT) method [Och 03] on n-best lists to find parameters which minimize an error rate. This method is derived from Powell's algorithm [Fletcher & Powell 63]. The basic idea of MERT is to optimize one scaling factor λ_k at a time while the others are fixed. Thus, we are able to compute the score of a given hypothesis e_1^I as a linear function

$$f(\lambda_k) = \lambda_k h_k(f_1^J, e_1^I) + \sum_{m=1, m \neq k}^{M} \lambda_m h_m(f_1^J, e_1^I). \tag{3.10}$$

Applied on each hypotheses of a given n-best list, the algorithm computes the intersection points of the upper envelope within the linear function. Then, at each intersection point the error measure is computed and the scaling factor producing the lowest error is selected. Other state-of-the-art optimization strategies for MT are the margin-infused relaxed algorithm [Chiang & Knight$^+$ 09] and PRO [Hopkins & May 11] which are usually applied on a larger set of parameters. Since in this work we have a small set of scaling factors, we apply MERT in our experiments.

3.4 Word Alignment

The training of state-of-the-art SMT systems usually starts with word-aligning the bilingual training corpus. Given a sentence and its corresponding translation, the *word alignment* is a mapping from several words of a source sentence to several words of the target sentence. More formally, for a given sentence pair (f_1^J, e_1^I) and an alignment $\mathcal{A} \subseteq \{1 \dots J\} \times \{1 \dots I\}$, the word f_j is aligned to e_i iff $(j, i) \in \mathcal{A}$. An example of an alignment is given in Figure 3.2.

An alignment with reasonable quality can be automatically computed with GIZA++ [Och & Ney 03] from the training data. This toolkit applies the EM-algorithm to train word-based translation models IBM-1 to IBM-5 [Brown & Della Pietra$^+$ 93] and the hidden Markov alignment model [Vogel & Ney$^+$ 96]. For training an SMT system, the *Viterbi* alignment $\hat{a}_1^J$ for a given

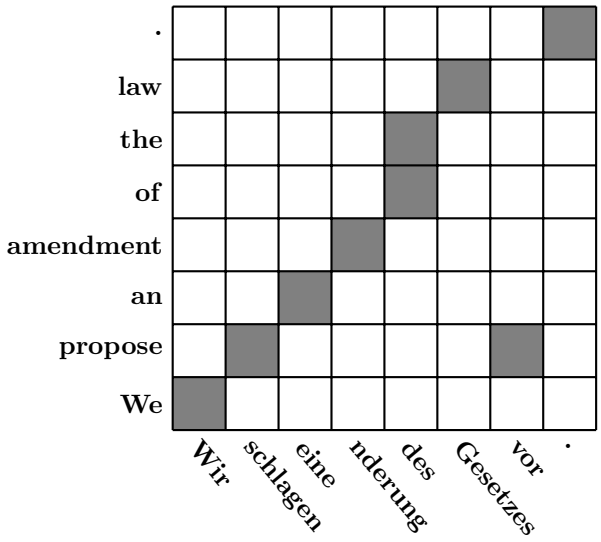


Figure 3.2: Example of a word alignment for the source sentence "Wir schlagen eine Änderung des Gesetzes vor ." and its translation "We propose an amendment of the law .".

sentence pair (f_1^J, e_1^I) is extracted with

$$(f_1^J, e_1^I) \mapsto \hat{a}_1^J(f_1^J, e_1^I) = \underset{a_1^J}{\operatorname{argmax}}\, Pr(a_1^J | f_1^J, e_1^I) \tag{3.11}$$

$$= \underset{a_1^J}{\operatorname{argmax}}\, Pr(a_1^J, f_1^J | e_1^I). \tag{3.12}$$

In practice, alignments are created in both translation directions and combined with a heuristic [Och & Ney 03]. Most of the models used in a state-of-the-art SMT system are estimated using the Viterbi alignments.

3.5 Phrase-Based Machine Translation

In the first approaches to SMT, translation proceeds word-by-word. This has the disadvantage that local context is not considered. The translation of a word however can depend on the surrounding words. Furthermore, even when the word-by-word model is extended with a fertility model to map a word in the source language to multiple words in target language, the other way around is not handled by this approach.

The phrase-based translation approach however is able to model the local context and many-to-one mappings using the concept of *bilingual phrases*. A bilingual phrase is a phrase pair formed by a sequence of words of a source sentence and its translation and are extracted from a word-aligned corpus.

Figure 3.2 shows an alignment for the sentence pair:

German: Wir schlagen eine Änderung des Gesetzes vor .

English: We propose an amendment of the law .

Given an alignment $\mathcal{A}$ of a sentence pair (f_1^J, e_1^I), we can extract a set of phrase pairs $\mathcal{P}$. A subsequence $(f_{j_1}^{j_2}, e_{i_1}^{i_2})$ is a phrase pair, if the following conditions are true:

source phrases	target phrases
Wir	We
schlagen eine Änderung des Gesetzes vor	propose an amendment of the law
eine	an
eine Änderung	an amendment
eine Änderung des	an amendment of the
eine Änderung des Gesetzes	an amendment of the law
Änderung	amendment
Änderung des	amendment of the
Änderung des Gesetzes	amendment of the law
des	of
des Gesetzes	of the law
Gesetzes	law
.	.

Figure 3.3: All possible bilingual phrases extracted from the alignment shown in Figure 3.2. The maximum phrase length is set to 6 words in this example.

1. The words in $f_{j_1}^{j_2}$ and $e_{i_1}^{i_2}$ are aligned only to words within the pair.
2. There is at least one alignment point between the pair.

More formally, the set of phrase pairs $\mathcal{P}$ is defined as:

$$\begin{aligned}\mathcal{P}(f_1^J, e_1^I, \mathcal{A}) = \big\{(f_{j_1}^{j_2}, e_{i_1}^{i_2}) \,|j_1, j_2, i_1, i_2 \text{ s.t.} \\ \forall (j,i) \in \mathcal{A} : j_1 \leq j \leq j_2 \Leftrightarrow i_1 \leq i \leq i_2 \\ \wedge \exists (j,i) \in \mathcal{A} : (j_1 \leq j \leq j_2 \wedge i_1 \leq i \leq i_2)\big\}.\end{aligned} \tag{3.13}$$

In practice phrases shorter than a maximum length of words are extracted. An example set of phrases which are extracted from the alignment in Figure 3.2 is shown in Figure 3.3.

Before the actual translation, the given source sentence f_1^J has to be segmented first in order to use the extracted set of phrase pairs. In the following we take the notation from [Zens 08]. A segmentation of a sentence pair (f_1^J, e_1^I) into K phrase pairs is defined as a sequence s_1^K with

$$s_k := (i_k; b_k, j_k), \text{ for } k = 1, \ldots, K. \tag{3.14}$$

i_k is the end position of the k^{th} phrase in the target sentence. The pair (b_k, j_k) contains the start and end positions of the source phrase. Thus, each s_k describes the alignment of a source phrase $\tilde{f}_k := f_{b_k} \ldots f_{j_k}$ to the target phrase $\tilde{e}_k := e_{i_{k-1}+1} \ldots e_{i_k}$. We only allow for segmentations without overlapping of phrases. In other words, each source position is covered by one source phrase. More formally:

$$\bigcup_{k=1}^{K} \{b_k, \ldots, j_k\} = \{1, \ldots, J\}, \tag{3.15}$$

$$\{b_k, \ldots, j_k\} \cap \{b_{k'}, \ldots, j_{k'}\} = \emptyset \text{ for all } k \neq k'. \tag{3.16}$$

However, this definition of segmentations does not require the source positions of the phrases to be monotonous, i.e. jumps on the source side are allowed.

The features of the log-linear model combination depend on the chosen segmentation. Thus, s_1^K is a hidden variable in the log-linear model combination:

$$f_1^J \mapsto \hat{e}_1^{\hat{I}}(f_1^J) = \underset{I,e_1^I}{\operatorname{argmax}} \sum_{K,s_1^K} Pr(e_1^I, s_1^K | f_1^J) \tag{3.17}$$

$$\approx \underset{e_1^I}{\operatorname{argmax}} \left\{ \max_{s_1^K} Pr(e_1^I, s_1^K | f_1^J) \right\} \tag{3.18}$$

$$= \underset{e_1^I}{\operatorname{argmax}} \left\{ \max_{s_1^K} \sum_{m=1}^{M} \lambda_m h_m(f_1^J, e_1^I; s_1^K) \right\} \tag{3.19}$$

In Equation 3.17, the maximum approximation is applied to make the search feasible.

In the following, we describe standard features that are used for statistical phrase-based translation.

3.5.1 Standard Models

Phrase Translation Model

The phrase translation probabilities $p(\tilde{f}|\tilde{e})$ and $p(\tilde{e}|\tilde{f})$ are estimated using relative frequencies. This is done during training by simply counting the number of occurrences of the source phrase $\tilde{f}$ being aligned to the target phrase $\tilde{e}$ in a sentence of the bilingual corpus. Thus, the probability $p(\tilde{f}|\tilde{e})$ is estimated as

$$p(\tilde{f}|\tilde{e}) = \frac{N(\tilde{f}, \tilde{e})}{\sum_{\tilde{f}'} N(\tilde{f}', \tilde{e})}, \tag{3.20}$$

where $N(\tilde{f}, \tilde{e})$ is number of occurrences that $\tilde{f}$ and $\tilde{e}$ are aligned in a sentence. The probability $p(\tilde{e}|\tilde{f})$ is defined analogously.

In the log-linear framework the phrase-based translation model for a given sentence pair (f_1^J, e_1^I), which is segmented into K phrase pairs $(\tilde{f}_1^K, \tilde{e}_1^K)$, is defined as follows:

$$h_{\text{Phrase,T2S}}(f_1^J, e_1^I, s_1^K) = \log \prod_{k=1}^{K} p(\tilde{f}_k|\tilde{e}_k) \tag{3.21}$$

$$h_{\text{Phrase,S2T}}(f_1^J, e_1^I, s_1^K) = \log \prod_{k=1}^{K} p(\tilde{e}_k|\tilde{f}_k) \tag{3.22}$$

Word-Based Lexicon Model

Some phrases occur rarely in the training corpus. As a result, the probability of these phrases is overestimated by the relative frequencies in Equation 3.20. The *word-based lexicon feature* uses word translation probabilities $p(f|e)$ to smooth the phrase translation probabilities. In particular, the feature takes all combinations of one source and one target word into account. By including the empty word e_0, unaligned source words are modeled as well. The feature function is defined as

$$h_{\text{Lex,T2S}}(f_1^J, e_1^I, s_1^K) = \log \prod_{k=1}^{K} \prod_{j=b_k}^{j_k} \left(p(f_j|e_0) + \sum_{i=i_{k-1}+1}^{i_k} p(f_j|e_i) \right). \tag{3.23}$$

The inverse feature is defined analogously using the word probabilities $p(e|f)$. The word probabilities can be either taken from the IBM-1 model or computed as relative frequencies by counting occurrences in the training corpus given a Viterbi alignment. In this work, the latter approach is applied unless otherwise stated.

Additional smoothing features will be presented in Chapter 6.

Language Model

The language model estimates the probability of the translated sentence in the target language. For that purpose, a general language model considers the history e_1^{i-1} of a word e_i.

$$Pr(e_1^I) = \prod_{i=1}^{I} p(e_i|e_1^{i-1}) \tag{3.24}$$

However, in practice, a standard n-gram language model is used, which does not consider the complete history.

$$h_{LM}(f_1^J, e_1^I, s_1^K) = \log \prod_{i=1}^{I} p(e_i|e_{i-n+1}^{i-1}) \tag{3.25}$$

In state-of-the-art machine translation systems, a language model smoothing with modified Kneser-Ney discounting [Chen & Goodman 96] is employed.

Word and Phrase Penalty

To control the number of words in the translation, a *word penalty* is applied:

$$h_{\text{WordPenalty}}(f_1^J, e_1^I, s_1^K) = I. \tag{3.26}$$

A *phrase penalty* counts the number of phrases in the segmentation:

$$h_{\text{PhrasePenalty}}(f_1^J, e_1^I, s_1^K) = K. \tag{3.27}$$

When using a positive scaling factor $\lambda_{\text{PhrasePenalty}}$, longer (and thus fewer) phrases will be preferred in the segmentation.

Distortion Penalty Model

As arbitrary jumps are allowed in the segmentation, a simple *distortion penalty* assigns scores to jumps based on their distance. Thus, jumps are disfavored unless the other features assign a higher score to the reordered sentence.

$$h_{\text{DM}}(f_1^J, e_1^I, s_1^K) = -\sum_{k=1}^{K+1} |j_{k-1} - b_k + 1| \tag{3.28}$$

It is possible for a segmentation s_1^K to end in the middle of the source sentence. For this reason, the summation in Equation 3.28 further includes the case $k = K + 1$. We define $b_{K+1} := J + 1$. This means that the last summand includes scores for a jump from j_K past the sentence end.

3.5.2 Search

The goal of the search algorithm is to find the target sentence e_1^I maximizing the log-linear score:

$$f_1^J \mapsto \hat{e}_1^{\hat{I}}(f_1^J) = \underset{I,e_1^I}{\operatorname{argmax}} \left\{ \sum_{m=1}^{M} \lambda_m h_m(f_1^J, e_1^I) \right\} \tag{3.29}$$

$$= \underset{e_1^I}{\operatorname{argmax}} \left\{ \max_{s_1^K} \sum_{m=1}^{M} \lambda_m h_m(f_1^J, e_1^I; s_1^K) \right\}. \tag{3.30}$$

The maximization has to be carried out over all possible segmentations of the source sentence including all possible reorderings of the source sentence. Thus, the search space can be very large and it has been shown that the problem of finding the exact best phrase-based translation is NP-hard [Knight 99]. In practice the search algorithm has to use approximations.

In the following, the *source cardinality synchronous search* (SCSS) algorithm is applied for phrase-based translation. An extensive description of this decoder is given in [Zens 08].

The basic idea of this algorithm is to extend *partial translation hypotheses* phrase by phrase. At each step, a partial hypothesis is extended by selecting a part of the source sentence and choosing the corresponding translation from the phrase set. In other words the algorithm jumps through the source sentence while the target sentence is generated monotonically.

During search, the algorithm has to ensure that a source word is not translated twice. Hence, each partial hypothesis is associated with the set of position that have been already translated. This set is named *coverage set* and the *source cardinality* of a hypothesis is the number of covered words.

In Section 8.1, the concept of coverage sets is extended to handle lattice input rather than sentence input in the scope of spoken language translation.

3.5.3 Hierarchical Reordering Model

Phrase-based translation systems with a set of features as described in Section 3.5 do not model reordering well. The distortion model only takes the distance at which the words are reordered into account. If a reordered sentence is to be chosen by the decoder, the distortion model scores need to be compensated by having higher language model scores. Thus, the quality of the reordering highly depends on the language model. However, the reordering information from the bilingual data are not used with this approach.

An approach to make use of the bilingual data is described in [Tillmann 04]. A lexicalized reordering model is introduced in order to model that some words are more likely to be reordered than others. A phrase pair can be in one of three *orientations* with respect to the previous phrase pair: *monotone* (i.e. no reordering), *swap* or *discontinuous*. During phrase extraction, the probabilities $p(o|\tilde{f}, \tilde{e})$ of the orientations o are estimated using relative frequencies for each bilingual phrase $(\tilde{f}, \tilde{e})$. These probabilities are integrated into the log-linear model combination and used during decoding. With this model, local reorderings can be captured quite well (e.g. swapping two phrases). However, the reordering of phrases that are not adjacent is not modeled.

The hierarchical reordering model (HRM) as introduced by [Galley & Manning 08] allows multiple phrases to be merged into *blocks*. Each block can then be treated as a single phrase when orientations are determined. With this solution, even long-distance reorderings are modeled. This model can be efficiently integrated into the SCSS decoding algorithm.

Furthermore, this approach is extended for hierarchical phrase-based translation in [Huck & Wuebker$^+$ 13].

3.6 Evaluation Measures

In machine translation, the automatic evaluation of the generated translations is an important challenge. Using human evaluation is extensive but time-consuming. To evaluate a huge amount of translations produced by translation systems, we need a fast, automatic and language independent evaluation method. This method should also correlate highly with human evaluation. To evaluate the quality of a translation, the translation of each source sentence is compared with a human generated reference translation. In the following section, three metrics are presented where the first one is mainly used for automatic speech recognition and the latter ones are state-of-the-art metrics in machine translation.

3.6.1 WER

The *Word Error Rate* (WER) is an error metric which is defined as the Levenshtein distance divided by the number of words in the reference. The Levenshtein distance for a sentence e_1^I and a reference sentence $e'^{I'}_1$ is the minimum number of edit operations that need to be performed to turn e_1^I into $e'^{I'}_1$ on a word level.

$$\text{WER} = \frac{\text{number of edit operations}}{\text{number of reference words}} \tag{3.31}$$

The allowed edit operations are insertion, deletion and substitution of words. On a document level, the error rate can be calculated by counting the number of edit operations for each sentence, then summing them and normalizing by the number of reference words in total. This metric is often used in automatic speech recognition to determine the quality of recognized sentences compared to a human reference transcription.

3.6.2 TER

The *Translation Edit Rate* (TER) is an error metric and is defined as the minimum number of edit operations needed to change a translation so that it equals to the closest the reference translation [Snover & Dorr$^+$ 06]. Similar to the word error rate, the edit operations include insertion, deletion and substitution of single words. In addition, a shift moves a contiguous sequence of words within the translation. Each edit operation has equal cost. This measure is normalized by the average length of the references. More formally:

$$\text{TER} = \frac{\text{number of edit operations}}{\text{average number of reference words}} \tag{3.32}$$

In practice, the TER is computed on document level by collecting the edit counts over the whole data set.

3.6.3 BLEU

The *Bilingual Evaluation Understudy* (BLEU) is an evaluation metric introduced by [Papineni & Roukos$^+$ 02]. BLEU is a precision measure and uses a *modified n-gram precision* combined with a *brevity penalty*. To compute the modified n-gram precision $p_n(e_1^I, e'^{I'}_1)$ for a given translation e_1^I and reference translation $e'^{I'}_1$, the number of occurrences of an n-gram w_1^n in the translation present in the reference translation is counted. The minimum of both counts is divided by the total number of n-grams in the translation. The counts are denoted by $C(w_1^n|e_1^I)$. If no n-gram is found in the translation, the n-gram precision equals zero. Furthermore, p_n for 1 to N n-grams are added and divide the sum by N to get a geometric mean.

The brevity penalty BP is necessary to penalize too short translations.

$$\textsc{Bleu}(e_1^I, e'^{I'}_1) = BP(I, I') \cdot \exp\left(\frac{1}{N}\sum_{n=1}^{N} \log p_n(e_1^I, e'^{I'}_1)\right) \tag{3.33}$$

with

$$BP(I, I') = \begin{cases} 1 & \text{if } I > I' \\ \exp(1 - I'/I) & \text{if } I \leq I' \end{cases} \tag{3.34}$$

$$p_n(e_1^I, e'^{I'}_1) = \frac{\sum_{w_1^n} \min\left\{C(w_1^n|e_1^I), C(w_1^n|e'^{I'}_1)\right\}}{\sum_{w_1^n} C(w_1^n|e_1^I)} \tag{3.35}$$

Bleu is defined to take more than one reference translation into account and is usually computed on document level by collecting the n-gram counts over all translations. This avoids an n-gram precision equals zero. N is usually set to 4.

3.7 Discriminative Training

Discriminative training is a powerful method to learn a large number of features. In contrast to the generative training scheme, this approach directly optimize the translation model with respect to a given error metric. Following [He & Deng 12], the goal is to optimize a maximum expected Bleu objective under a parameter set Θ:

$$\langle \textsc{Bleu} \rangle_\Theta = \sum_{F \in \mathbb{F}} \sum_{E \in \mathbb{E}} p_\Theta(E, F) \textsc{Bleu}(E), \tag{3.36}$$

where $\mathbb{F}, \mathbb{E}$ is the entire space of correct sentences in the source and target language. To make the training procedure feasible, the search space is approximated with n-best lists:

$$\langle \textsc{Bleu} \rangle_\Theta = \frac{1}{N}\sum_{n=1}^{N} \sum_{E \in \mathbb{E}_\Theta(F_n)} p_\Theta(E|F_n) \textsc{Bleu}(E), \tag{3.37}$$

where N is the size of the training corpus and $\mathbb{E}_\Theta(F)$ the n-best list generated from F under parameters Θ.

While previous work presented in [Tromble & Kumar$^+$ 08] and [Rosti & Zhang$^+$ 11] used the expected Bleu score as objective to learn the scaling factor of an MT system and a system combination, respectively, the approach as described above updates both the lexicon model and the translation model.

[Wuebker & Muehr$^+$ 15] propose to apply RPROP as update method and a leave-one-out heuristic [Wuebker & Mauser$^+$ 10] to circumvent over-fitting. Furthermore, following an approach similar to [Auli & Galley$^+$ 14], each feature type is first discriminatively trained, then condensed into a single feature for the log-linear model combination and finally optimized with MERT.

In Chapter 7, we compare our generative approach introduced in Chapter 5 with discriminative training.

4. Hierarchical Phrase-Based Translation

In this chapter we describe the approach of hierarchical phrase-based translation [Chiang 05] which can be considered as an extension of the phrase-based approach (cf. Section 3.5). After giving a short introduction (Section 4.1), we describe the extraction of the hierarchical translation model in Section 4.2. In Section 4.3, the decoding process which can be considered as probabilistic parsing procedure is depicted. We further present details of some improvements done during the course of this thesis for the hierarchical decoder in Section 4.3.4 and describe the models which are used in our hierarchical baseline systems (Section 4.4). Finally, the contributions of this work (Section 4.5) are distinguished from previous work in Section 4.6.

4.1 Introduction

The hierarchical phrase-based translation approach can be motivated by the fact that in phrase-based translation long-range reorderings have to be estimated by additional models such as the hierarchical reordering model (cf. Section 3.5.3). Furthermore, the hierarchical phrase-based translation approach allows for more abstract phrases which generalize better. This new and more general type of phrases is generated by replacing parts of a larger phrase with a "gap". The resulting *hierarchical phrases* represent the dependency between the words before the gap and the words after the gap and further allow long-range reorderings by having more than one gap. Thus, reordering information is directly encoded in the translation model.

Based on the example introduced in Section 3.5 (Figure 3.2 and Figure 3.3) we further motivate the advantage of hierarchical phrases. The phrase-based extraction process only generates a very specific phrase including the German verb "vorschlagen" which is separated into two parts ("schlagen" and "vor"). The generated phrase "(schlagen eine Änderung des Gesetzes vor, propose an amendment of the law)" does not generalize very well. In contrast, the hierarchical approach allows for discontinuous phrases which are more abstract and generalize better, e.g. "wir schlagen [...] vor" which is translated to "we propose [...]".

An example for the incorporation of reordering information is provided by the following sentence pair:

German: Das Gesetz sollte so geändert werden *wie* wir es gestern *vorgeschlagen haben* .

English: The law should be changed *as* we *proposed* it yesterday .

The German particle "wie" induces a subordinate clause with the verb phrase "vorgeschlagen haben" at the very end. However, in the English sentence, the corresponding verb "proposed" follows the subject "we" right after the particle "as". This difference in the grammatical structure can be captured by hierarchical phrases with two gaps, e.g. "(wie wir [...]$^{\sim 0}$ gestern [...]$^{\sim 1}$ haben ., as we [...]$^{\sim 1}$ [...]$^{\sim 0}$ yesterday .)"

4.2 Hierarchical Translation Model

The translation model in hierarchical phrase-based translation is formulated as synchronous context-free grammar (SCFG) [Lewis & Stearns 68]. Similar to phrase-based translation, the model is extracted from word-aligned bilingual sentences. Since a grammar consists of rules, we will use the term *rules* rather than *phrases* in the following. Before we explain the actual extraction process in detail, we give a formal definition of SCFGs.

4.2.1 Synchronous Context-free Grammar

SCFGs are derived from context-free grammars (CFGs) [Chomsky 56]. A CFG noted by G is formally defined as a 4-tuple $(\mathcal{N}, \mathcal{T}, \mathcal{R}, S)$, where

- $\mathcal{N}$ is a finite set of non-terminals,
- $\mathcal{T}$ is a finite set of terminals with $\mathcal{N} \cap \mathcal{T} = \emptyset$,
- $\mathcal{R} \subset \mathcal{N} \times (\mathcal{T} \cup \mathcal{N})^*$ is a set of production rules,
- and $S \in \mathcal{N}$ is the start symbol of G.

The rule (X, α) with $X \in \mathcal{N}$ and $\alpha \in (\mathcal{T} \cup \mathcal{N})^*$ is written as $X \to \alpha$. Given three strings $\alpha, \beta, \gamma \in (\mathcal{T} \cup \mathcal{N})^*$ and two rules $r_1 = X \to \beta$ and $r_2 = X \to \alpha X \gamma$, we can generate the string $\alpha\beta\gamma$ by applying rule r_1 on r_2. This is denoted by $\alpha X \gamma \Rightarrow \alpha\beta\gamma$. Given a sequence of strings $\alpha_1, \alpha_2, \ldots, \alpha_n$ such that α_1 is transformed into α_n by applying several rules of G, we write $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n$ or $\alpha_1 \overset{*}{\Rightarrow} \alpha_n$. This application of rules is named *derivation*. The language $L(G)$ defined by the grammar G is the set of all strings that can be derived from start symbol S. The decision problem if $x \in L(G)$ for a given string $x \in \mathcal{T}^*$ can be efficiently solved with the Cocke-Younger-Kasami (CYK) algorithm [Kasami 65, Younger 67, Cocke 69]. This parsing algorithm will be later extended in Section 4.3.2 for the purpose of hierarchical phrase-based translation.

However, in translation we need a grammar which generates a pair of strings. This property is given by the definition of synchronous context-free grammars which consists of bilingual rules. Similar to CFGs, rules of a SCFG have a non-terminal on the left-hand side. On the right-hand side of the rules, a pair of strings of non-terminals and terminals is defined. Thus, a SCFG consists of two sets of terminals:

- $\mathcal{T}_f$ is the finite set of source terminals
- and $\mathcal{T}_e$ is the finite set of target terminals.

The right-hand side of a rule is a three-tuple $(\alpha, \beta, \sim)$ with $\alpha \in (\mathcal{T}_f \cup \mathcal{N})^+$, $\beta \in (\mathcal{T}_e \cup \mathcal{N})^+$ and a one-to-one correspondence $\sim$ between the non-terminals of α and β. In the following, we name the string α source language part (or source side) and the string β target language part (or target side) of a rule, respectively. Furthermore, in the scope of machine translation, we usually speak of (source and target) words rather than (source and target) terminals. A derivation is now defined over pairs of strings and the corresponding non-terminals on the right-hand side are rewritten synchronously on the source and target side.

4.2.2 Rule Extraction

In this section, we describe the extraction of rules for hierarchical translation. The extraction process is designed in a way that a grammar is generated which is able to parse every given source sentence. Following the original paradigm of hierarchical translation [Chiang 05], the set of non-terminals consists of a generic non-terminal X and a start symbol S only, i.e. $\mathcal{N} = \{S, X\}$. There are several attempts to extend the set of non-terminals towards syntactical labels, but this topic is beyond the scope of this work and we would like to refer the reader to [Zollmann & Venugopal 06, Peitz 10]. Furthermore, the number of non-terminals on the right-hand side of the rules (i.e. the number of gaps) is limited to a maximum of two as a trade-off between the complexity while decoding and the ability to perform long-range reorderings. This results in three types of rules:

- *lexical rules* without non-terminal symbols $X \rightarrow \langle \alpha, \beta \rangle$,
- *hierarchical rules* with one non-terminal $X \rightarrow \langle \alpha_1 X^{\sim 0} \alpha_2, \beta_1 X^{\sim 0} \beta_2 \rangle$ and
- *hierarchical rules* with two non-terminals $X \rightarrow \langle \alpha_1 X^{\sim 0} \alpha_2 X^{\sim 1} \alpha_3, \beta_1 X^{\sim 0} \beta_2 X^{\sim 1} \beta_3 \rangle$ or $X \rightarrow \langle \alpha_1 X^{\sim 0} \alpha_2 X^{\sim 1} \alpha_3, \beta_1 X^{\sim 1} \beta_2 X^{\sim 0} \beta_3 \rangle$

where $\alpha, \alpha_i \in \mathcal{T}_f^*$ and $\beta, \beta_i \in \mathcal{T}_e^*$ with $1 \leq i \leq 3$. Note that the one-to-one correspondence $\sim$ between the non-terminals on source side and the non-terminals on the target side is denoted implicitly with $^{\sim 0}$ and $^{\sim 1}$. We will stick to this notation in the following.

Starting point of the extraction process is a set of lexical rules $\mathcal{R}_0$ similarly defined as the set of phrase pairs $\mathcal{P}$ (cf. Equation 3.13). The only difference is that lexical rules have a non-terminal on the left-hand side. In practice, lexical rules with a maximum length of ten words are extracted while the maximum length of a phrase pair is usually set to 6 words. Longer lexical rules are needed to extract hierarchical rules covering long-range dependencies. However, such long lexical rules usullay do not improve the translation quality. Given a sentence pair (f_1^J, e_1^I) and an alignment $\mathcal{A}$, we define the set of lexical rules as

$$\begin{aligned} \mathcal{R}_0(f_1^J, e_1^I, \mathcal{A}) = \big\{ X \rightarrow \langle f_{j_1}^{j_2}, e_{i_1}^{i_2} \rangle \mid & j_1, j_2, i_1, i_2 \text{ s.t.} \\ & \forall (j,i) \in \mathcal{A} : j_1 \leq j \leq j_2 \Leftrightarrow i_1 \leq i \leq i_2 \\ & \wedge \exists (j,i) \in \mathcal{A} : (j_1 \leq j \leq j_2 \wedge i_1 \leq i \leq i_2) \big\} . \end{aligned} \tag{4.1}$$

The set of hierarchical rules is defined recursively depending on the number of non-terminals n with $1 \leq n \leq N$ where N is the maximum of non-terminals (i.e. maximum number of gaps)

$$\begin{aligned} \mathcal{R}_n(f_1^J, e_1^I, \mathcal{A}) = \Big\{ X \rightarrow \langle \alpha_1 X^{\sim n} \alpha_2, \beta_1 X^{\sim n} \beta_2 \rangle \mid & \alpha_1, \alpha_2 \in (\mathcal{T}_f \cup \mathcal{N})^*, \beta_1, \beta_2 \in (\mathcal{T}_e \cup \mathcal{N})^* \\ & \wedge \exists j_1, j_2, i_1, i_2 : j_1 \leq j_2, i_1 \leq i_2 : \\ & \Big(X \rightarrow \langle \alpha_1 f_{j_1}^{j_2} \alpha_2, \beta_1 e_{i_1}^{i_2} \beta_2 \rangle \in \mathcal{R}_{n-1}(f_1^J, e_1^I, \mathcal{A}) \\ & \wedge X \rightarrow \langle f_{j_1}^{j_2}, e_{i_1}^{i_2} \rangle \in \mathcal{R}_0(f_1^J, e_1^I, \mathcal{A}) \Big) \Big\} . \end{aligned} \tag{4.2}$$

In other words, whenever a phrase contains a sub-phrase, this sub-phrase is replaced by a non-terminal and stored as hierarchical rule. The fully defined rule set $\mathcal{R}$ is the union of the lexical and hierarchical rules

$$\mathcal{R}(f_1^J, e_1^I, \mathcal{A}) = \bigcup_{n=0}^{N} \mathcal{R}_n(f_1^J, e_1^I, \mathcal{A}) \tag{4.3}$$

with $N = 2$ and completes the definition of the grammar G which formalizes the hierarchical translation model:

$$G = (\mathcal{T}_f, \mathcal{T}_e, \mathcal{N}, \mathcal{R}, S). \tag{4.4}$$

Figure 4.1 and Figure 4.2 show subsets of lexical rules and hierarchical rules, respectively, which are extracted from the word alignment in Figure 3.2. The extraction process of hierarchical rules

$X \rightarrow \langle$schlagen eine Änderung des Gesetzes vor, propose an amendment of the law$\rangle$
$X \rightarrow \langle$eine Änderung des Gesetzes, an amendment of the law$\rangle$
$X \rightarrow \langle$des Gesetzes, of the law$\rangle$
$X \rightarrow \langle$Gesetzes, law$\rangle$

Figure 4.1: Subset of lexical rules extracted from the word alignment in Figure 3.2.

$X \rightarrow \langle$schlagen $X^{\sim 0}$ vor, propose $X^{\sim 0}\rangle$
$X \rightarrow \langle X^{\sim 0}$ schlagen eine $X^{\sim 1}$ vor, $X^{\sim 0}$ propose an $X^{\sim 1}\rangle$

Figure 4.2: Subset of hierarchical rules extracted from the word alignment in Figure 3.2.

as formulated in Equation 4.2 is depicted in Figure 4.3.

The grammar is extended by three additional rules in order to map the start symbol to other non-terminals (i.e. to the generic non-terminal X) and to introduce the sentence-start symbol $<s>$ and sentence-end symbol $</s>$ (Equation 4.5 and 4.6). The sentence-start and sentence-end symbols are important for a proper incorporation of the language model in the decoding process. Note that this additional rule set is different to the one as presented in [Chiang 05] and implemented in [Stein 12]. The glue rule allows for a simple concatenation of rules for monotone translation (Equation 4.7).

$$S \rightarrow \langle \text{<s>}, \text{<s>} \rangle \tag{4.5}$$
$$S \rightarrow \langle S^{\sim 0} \text{ </s>}, S^{\sim 0} \text{ </s>} \rangle \tag{4.6}$$
$$S \rightarrow \langle S^{\sim 0} X^{\sim 1}, S^{\sim 0} X^{\sim 1} \rangle \tag{4.7}$$

Figure 4.4 shows a possible derivation using rules from Figure 4.1 and Figure 4.2.

Finally, the rules of the SCFG are extended with weights. This allows us to assign scores as in the phrase-based approach to each rule (cf 3.5.1). We will give further details about the used feature functions and their definitions in the Section 4.4.

4.3 Hierarchical Phrase-Based Decoding

The translation process in state-of-the-art hierarchical phrase-based translation can be considered as probabilistic parsing problem. Given an input sentence in the source language, this sentence is parsed using the source language part of the SCFG. In this work, we perform this step

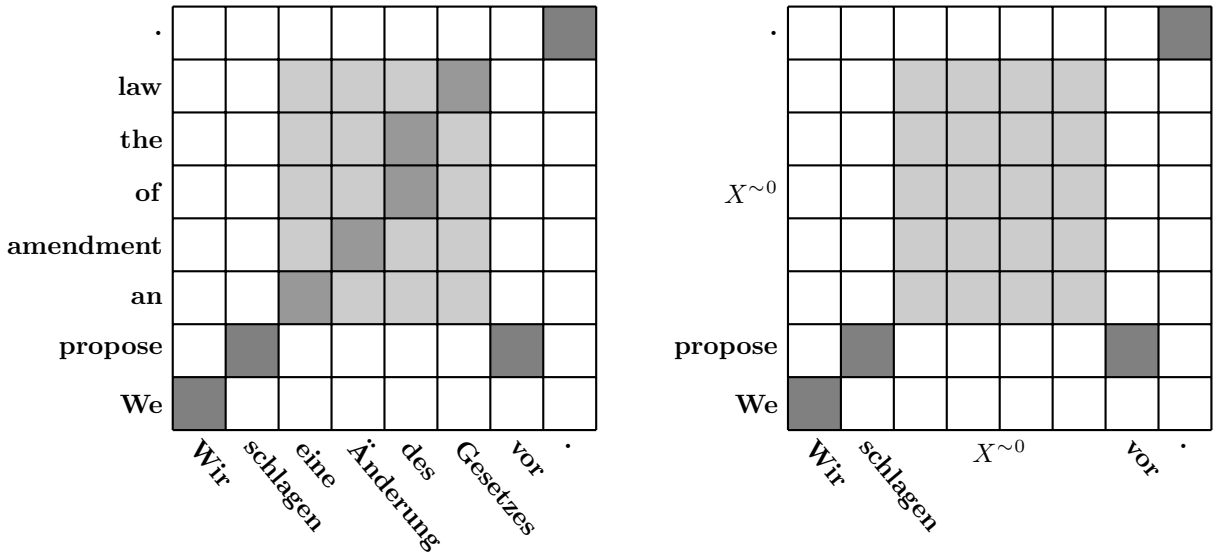


Figure 4.3: Extraction process of a hierarchical rule (cf. Figure 4.2).

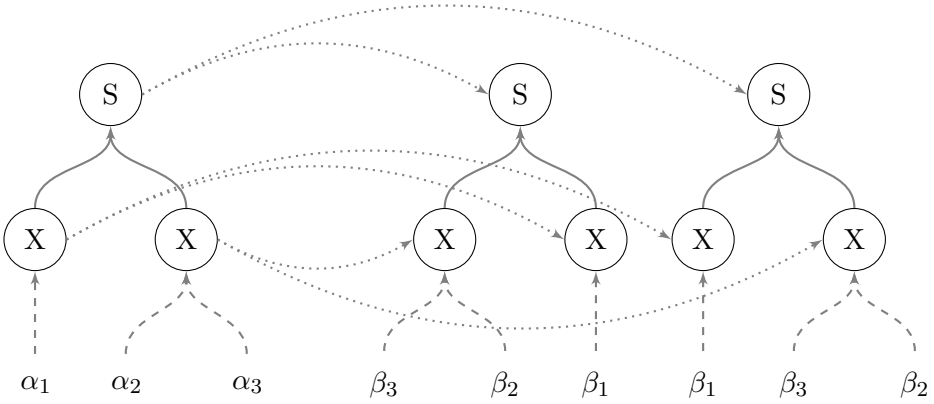
$$
\begin{aligned}
& S \\
\Rightarrow\ & \langle S^{\sim 0}\ \text{</s>},\ S^{\sim 0}\ \text{</s>} \rangle \\
\Rightarrow\ & \langle S^{\sim 0}\ X^{\sim 1}\ \text{</s>},\ S^{\sim 0}\ X^{\sim 1}\ \text{</s>} \rangle \\
\Rightarrow\ & \langle S^{\sim 0}\ X^{\sim 1}\ X^{\sim 1}\ \text{</s>},\ S^{\sim 0}\ X^{\sim 1}\ X^{\sim 1}\ \text{</s>} \rangle \\
\overset{*}{\Rightarrow}\ & \langle \text{<s> wir schlagen } X^{\sim 0} \text{ vor </s>},\ \text{<s> we propose } X^{\sim 0}\ \text{</s>} \rangle \\
\Rightarrow\ & \langle \text{<s> wir schlagen eine Änderung vor </s>},\ \text{<s> we propose a change </s>} \rangle
\end{aligned}
$$

Figure 4.4: Example of a possible derivation applying rules shown in Figure 4.1 and 4.2.

with a modified version of the *CYK+* algorithm [Chappelier & Rajman 98] (Section 4.3.2). The output of this algorithm is a *hypergraph*, which represents all possible derivations of the input sentence. A formal description is given in Section 4.3.1. By using the associated target part of the applied rules, for each derivation a translation is generated. The actual search is carried out applying the *cube pruning* algorithm (Section 4.3.3). A detailed description of the implementation is given in Section 4.3.4.

4.3.1 Hypergraphs

A *hypergraph* is a more general definition of a graph where a *hyperedge* connects one *hypernode* with several hypernodes. In the context of hierarchical translation, hypergraphs are used to represent derivations for a given SCFG in a compact way. Given a derivation d, each non-terminal appearing in any string in d is represented by a hypernode. Furthermore, each incoming hyperedge represents the rule with which the corresponding non-terminal was substituted. The hypergraph is created by parsing a given source string with the source side of the rules of a SCFG. Furthermore, a target parse forest can be induced with the target parts of the applied rule. An example is given in Figure 4.5.



$$S \rightarrow \langle X^{\sim 0} X^{\sim 1}, X^{\sim 1} X^{\sim 0} \rangle$$
$$X \rightarrow \langle \alpha_1, \beta_1 \rangle$$
$$X \rightarrow \langle \alpha_2 \alpha_3, \beta_3 \beta_2 \rangle$$

Figure 4.5: Example of a hypergraph generated with a simple SCFG. The hypergraph is the result of parsing the source string $\alpha_1\alpha_2\alpha_3$ with the source side of the rules. Two possible target parse tree can be induced with the target parts of the applied rules. The yields of these parse trees are the target strings $\beta_3\beta_2\beta_1$ and $\beta_1\beta_3\beta_2$.

How a hypergraph is created for a given source sentence and a SCFG is described in the following section. After parsing the source sentence, the decoder extracts and scores then different target strings (i.e. translation hypotheses) to find the best translation. This process is explained in Section 4.3.3.

4.3.2 Parsing with CYK+

The CYK+ algorithm [Chappelier & Rajman 98] is based on the well-known CYK algorithm which decides if an input string is generated by a given CFG. For the purpose of hierarchical translation, we describe a slightly modified version of the CYK+ algorithm which works with SCFGs rather than CFGs. Furthermore, the modified algorithm generates a hypergraph as described in the previous section.

However, before the CYK+ algorithm can be applied, hierarchical rules have to be transformed to *non-partially lexicalized rules*. While the original CYK algorithm only works with grammars in Chomsky normal form, the requirement for the CYK+ algorithm is to have only non-terminals or terminals on the right-hand side of the rules. This is achieved by replacing each terminal by a new *lexical non-terminal*. In practice, this is done efficiently when reading the rule set.

Similar to the definition of a standard CYK chart, a CYK+ chart is defined as following. For a given input sentence f_1^J, the CYK+ chart is a table with $\frac{J(J+1)}{2}$ cells. Each cell (l, j) spans the substring f_j^{j+l-1} where j is the starting position and $j + l - 1$ the length with $1 \leq l \leq J$ and $1 \leq j \leq J - l + 1$. A cell (l, j) contains two lists of items:

- the type-1 list contains non-terminals Y that parse a substring f_j^{j+l-1}.
- the type-2 list represents partial parses of f_j^{j+l-1} for which there is a rule $Y \rightarrow \alpha_1\alpha_2$ with

$\alpha_1 \overset{*}{\Rightarrow} f_j^{j+l-1}$ and a non-empty string of non-terminals α_2. This is denoted by $\alpha_1\bullet$.

The parsing step of CYK+ is divided into two procedures: the standard *cell filling procedure* which is similar to the parsing step of the CYK algorithm and the *self-filling procedure*. The input sentence is generated by the grammar if the start symbol of the grammar is found in the type-1 list of cell $(J, 1)$. The complete modified CYK+ algorithm is provided in Algorithm 4.1. In this algorithm, lists are denoted with square brackets [] and the concatenation of two lists with ++.

Algorithm 4.1 Modified CYK+ algorithm.

Input: source sentence f_1^J, rule set $\mathcal{R}$
Output: hypergraph representing all valid parses of f_1^J

```
1:  create an initial hypernode h
2:  for j = 1, ..., J do
3:      if ∃ a set of rules R of the form Y → ⟨f_j, ·⟩ ∈ R then
4:          create hypernode n associated with Y to type-1 list of cell (1, j)
5:          add hyperedge from [h] to n, associated with R
6:  for l = 1, ..., J do
7:      for j = 1, ..., J − l + 1 do
8:          for k = 1, ..., l − 1 do
9:              for each (α•, L) in the type-2 list (k, j) do
10:                 for each (Z, n) in type-1 list of (l − k, j + k) do
11:                     if ∃ a set of rules R of the form Y → ⟨αZ, ·⟩ ∈ R then
12:                         create hypernode n′ associated with Y in type-1 list of cell (l, j)
13:                         add a hyperedge from L ++ [n] to n′, associated with R
14:                     if ∃ a set of rules R of the form Y → ⟨α1Zα2, ·⟩ ∈ R then
15:                         add (α1Z•, L ++ [n]) to type-2 list of cell (l, j)
16:         for each (Z, n) in the type-1 list (l, j) do
17:             if ∃ a set of rules R of the form Y → ⟨Z, ·⟩ ∈ R then
18:                 create hypernode n′ associated with Y in type-1 list of cell (l, j)
19:                 add hyperedge from L ++ [n] to n′, associated with R
20:             if ∃ a set of rules R of the form Y → ⟨Zα, ·⟩ ∈ R then
21:                 add (Z•, [n]) to the type-2 list of (l, j)
```

As initialization step, the type-1 lists for length 1 are filled (Line 2-5). For each word of the input sentence, the algorithm tries to find a matching lexical rule in rule set. The left-hand side non-terminal of the corresponding rule is added to the type-1 lists and a hypernode labelled with this non-terminal is created. Furthermore, the new hypernode is connected with the initial hypernode.

The actual parsing is done in the standard cell filling procedure (Line 8-15). A list of non-terminals (L ++ $[n]$) for a given cell (l, j) is generated by combining type-2 items of cell (k, j) with type-1 items of cell $(l-k, j+k)$. A combination is valid if the right-hand side of a rule $Y \to \alpha_1 Z \alpha_2$ matches with a partial parse in the type-2 list of cell (k, j) and a parse of a substring in the type-1 list of cell $(l - k, j + k)$. Then, the non-terminal of the left-hand side of the found rule parses the substring f_j^{j+l-1} and is added to the type-1 list. If α_2 is not empty, the type-2 item $\alpha_1 Z\bullet$ represents the partial parse of f_j^{j+l-1}.

The self-filling procedure is necessary to deal with rules of the form $Y \to Z$ where $Y, Z \in \mathcal{N}$ (Line 16-21). If Z parses f_j^{j+l-1} then Y also parses this substring. This procedure also completes the initialization step as the type-2 lists are updated based on the initialized type-1 lists.

If the start symbol of the given grammar is found in cell $(J, 1)$, the given source sentence can be parsed. The hypergraph generated by the presented algorithm is then used to perform the search as described in the following section.

4.3.3 Search

For the search in hierarchical translation the decision rule shown in Equation 3.9 has to be slightly expanded. The new definition includes the concept of derivations as the goal of the search is to select the derivation d which parses a source sentence $\sigma(d)$ and induces a translation $\tau(d)$ with the highest probability. The decision rules is then rewritten as following:

$$f_1^J \mapsto \hat{e}_1^{\hat{I}}(f_1^J) \quad = \quad \underset{I,e_1^I}{\operatorname{argmax}} \left\{ \max_{\substack{d:\sigma(d)=f_1^J,\\ \tau(d)=e_1^I}} \left\{ \sum_{m=1}^{M} \lambda_m h_m(f_1^J, e_1^J, d) \right\} \right\} \tag{4.8}$$

With a probabilistic version of the parsing algorithm presented in previous section, we could now generate a translation by finding the best scoring parse of the source sentence and extracting it from the yield of parse tree induced with the target side of the applied rules. However, to incorporate a language model which scores the context beyond the rule boundaries, a more sophisticated algorithm is needed. A straightforward solution would be to generate n-best list of possible translations, apply the language model and select the translation with best combined score (i.e. translation and language model score). Since n-best lists are only a crude approximation of the search space, this approach is not used in pratice.

In state-of-the-art hierarchical translation the search is carried out by applying the *cube pruning* algorithm [Chiang 07]. This algorithm efficiently incorporates the language model by scoring partial hypotheses during generation time. Given a hypergraph, the algorithm generates *internal* k-best lists of derivations at each hypernode. The language model is then applied on these internal k-best lists. Thus, the generation of k-best lists is necessary even if only a single-best translation is needed. The first entry in the k-best list of the final hypernode represents the translation with the best combined score.

To describe the cube pruning algorithm as implemented in this work, we extend the definition of derivations with the following attributes:

- A state s represents the language model context of the derivation. To access the state of a derivation we use the notation $d[s]$.
- The function getScore() returns the combined score of the derivation.

The concept of *minimizing state* as described in [Heafield & Hoang$^+$ 11] is quite important to efficiently incorporate the language model score. The cube pruning algorithm (similar to other search algorithm based on dynamic programming) scores many hypotheses by exploiting the fact that an n-gram language model conditions on at most $n-1$ preceding words. The $n-1$ words are named *state*. If two partial hypotheses (i.e. derivations) have equal state, they can be *recombined* and considered as single derivation in the future. The *KenLM* language model toolkit[1] [Heafield 11] provides an efficient data structure for these states. In our preliminary implementation of the cube pruning algorithm, each derivation stores such a state and each hypernode keeps track of all generated states. This is a straightforward extension of the implementation presented in [Vilar 11] which works without this concept.

[1] `https://kheafield.com/code/kenlm/`

Algorithm 4.2 Cube pruning algorithm.

Input: A hypernode and the size k of the k-best list
Output: D, a list with k-best derivations

```
1: D = []
2: C, a heap of derivation candidates
3: S = {} //Set of language model context states
4: c = 0
5: while |C| > 0 and |D| < k do
6:     d = pop(C)
7:     RecombineCandidate(d)
8:     PushSucc(d)
9: sort D

10: function RecombineCandidate(d)
11:     if d[s] ∈ S then
12:         p = D[c]
13:         if d→getScore() > p→getScore() then
14:             p = d
15:     else
16:         D = D ++ [d]
17:         S.insert(d[s])
18:         c++
```

Algorithm 4.2 is applied on each hypernode of the hypergraph in a bottom-up manner. Thus, the algorithm assumes that the derivations of the predecessor hypernodes have already been computed. Each hypernode stores a list D of derivations and a set S of states generated by the derivations. Furthermore, a heap C of possible derivation candidates is provided. How this heap is initialized (Line 2) and kept up-to-date (Line 8) is described in [Vilar 11]. In this work, we focus on efficient storage of derivations and their recombination.

While the heap of candidates is not empty and a certain amount of derivations is not created (Line 5), a new derivation candidate d is dequeued from the heap (Line 6). Before the candidate is added to the list of derivations D, we first check if a derivation with the same state exists (Line 11). When the derivation candidate can be recombined, we further verify whether the score of the candidate is higher than the score of the previously generated derivation (Line 13). If this is true, we update the derivation score (Line 14). If no recombination takes place, the derivation candidate is simply added to the list of derivations and the set of states is updated (Line 16 and 17). Finally, if the amount of derivations is sufficient or the heap of candidates is empty, the list of derivations is sorted (Line 9) and the algorithm is applied to the next hypernode.

4.3.4 Improvements

In this section, we provide details about the improvements which resulted in a competitive state-of-the-art hierarchical decoder. We refer to a *previous implementation* which is described in [Vilar 11] and was part of RWTH Aachen University's open source translation toolkit *Jane*[2] [Vilar & Stein$^+$ 10a, Wuebker & Huck$^+$ 12] until version 2.1. Since version 2.2, the decoder uses states for recombination as described above. Our improvements described in this section are released with version 2.3 [Freitag & Huck$^+$ 14].

[2] `https://www-i6.informatik.rwth-aachen.de/jane/`

The cube pruning algorithm as presented in the previous section uses an additional data structure to keep track of the generated states. In our improved implementation, we drop this additional data structure and use only one single set to keep track of generated derivations. These derivations are then distinguished by their states. Whenever a new derivation candidate is generated, we are now able to check directly whether the derivation has to be recombined or not.

A related but more C++ specific change in the implementation compared to the previous one is that we use pointers of derivations rather than references. This results in a need of a more sophisticated memory management but helps to reduce the memory consumption.

Furthermore, if n-best translations are needed (e.g. for MERT), each derivation keeps track of its n-best recombined hypotheses. In the previous implementation all recombined hypotheses are kept in memory. This structural change also results in a more sophisticated n-best list generation procedure but leads to a lower memory usage during translation.

Another important enhancement in the hierarchical translation framework is not related to the decoder directly but could help to reduce the search error. To make search feasible, the amount of translation options for each rule is limited by *observation histogram pruning*. In the previous implementation, the translation model score is considered for pruning the translation options only. Since version 2.3, the language model score is taken into account as well.

In [Heafield 13] an extensive comparison of different publicly available hierarchical phrase-based translation engines is performed. The authors show that the hierarchical translation framework enhanced during the course of this thesis is competitive in translation speed and memory usage compared with other state-of-the-art hierarchical translation engines such as *Moses* [Hoang & Lopez 09], *Joshua* [Li & Callison-Burch$^+$ 09] and *cdec* [Dyer & Weese$^+$ 10].

In Chapter 7, we compare both implementations (*Jane* version 2.2 and 2.3) in terms of speed and memory consumption.

4.4 Model Definition

In this section, the feature functions of the log-linear model combination for hierarchical phrase-based translation are defined. First, the standard models, which are similar to the models presented for the phrase-based approach, are described. Moreover, we present a word class language model in Section 4.4.2.

4.4.1 Standard Models

The standard models used in hierarchical translation are mainly the same as introduced for phrase-based translation (cf. Section 3.5.1). For the search based on the concept of derivations, the definition of these models has to be slightly adapted.

Let $R(d)$ be a set of rules applied in a derivation d. The direct rule translation model is computed as

$$h_{\text{Rule,T2S}}(f_1^J, e_1^I, d) = \log \prod_{r \in R(d)} p_{\text{Rule,T2S}}(r) \tag{4.9}$$

where $p_{\text{Rule,T2S}}(r)$ is the probability of the source part α being the translation of the target part β for a given rule $r = X \rightarrow \langle \alpha, \beta \rangle$. As for the phrase-based translation model, the probability is estimated as relative frequencies from heuristically extracted counts $N(\alpha, \beta)$:

$$p_{\text{Rule,T2S}}(r) = p(\alpha|\beta) = \frac{N(\alpha, \beta)}{\sum_{\alpha'} N(\alpha', \beta)}. \tag{4.10}$$

The inverse rule translation model is defined analogously and the word-based lexicon models are rewritten in a similar manner for a given rule $r = X \rightarrow \langle \alpha_1^J, \beta_1^I \rangle$

$$p_{\text{Lex,T2S}}(r) = \frac{1}{(I+1)^J} \prod_{j=1}^{J} \left(\mathbb{I}(\alpha_j \notin \mathcal{N}) p(\alpha_j|\beta_0) + \sum_{i=1}^{I} (\mathbb{I}(\alpha_j \notin \mathcal{N}) \mathbb{I}(\beta_i \notin \mathcal{N}) p(\alpha_j|\beta_i)) \right). \quad (4.11)$$

The $\mathbb{I}$ function is 1 if its argument is true and 0 otherwise. In this equation it ensures that non-terminals are not taken into account when computing word lexicon model. The word-lexicon probability $p(\alpha_j|\beta_i)$ is computed as relative frequency given the word-aligned training corpus. Note that β_0 equals the empty word (cf. Equation 3.23).

Besides a word and a rule penalty, the hierarchical baseline includes three simple heuristics which count the number of applied hierarchical rules, the number of applications of the glue rule and the number of applied hierarchical rules with non-terminals on the fringe:

$$h_{\text{WordPenalty}}(r) = |\beta| \quad (4.12)$$

$$h_{\text{RulePenalty}}(r) = 1 \quad (4.13)$$

$$h_{\text{isHierarchical}}(r) = \begin{cases} 1 & \exists X \in \alpha : X \in \mathcal{N} \\ 0 & \text{otherwise} \end{cases} \quad (4.14)$$

$$h_{\text{isGlue}}(r) = \begin{cases} 1 & r = S \rightarrow \langle S^{\sim 0}\ X^{\sim 1},\ S^{\sim 0}\ X^{\sim 1} \rangle \\ 0 & \text{otherwise} \end{cases} \quad (4.15)$$

$$h_{\text{isPaste}}(r) = \begin{cases} 1 & \alpha \in \{X^{\sim 0}\alpha_1, \alpha_1 X^{\sim 0}, X^{\sim 0}\alpha_1 X^{\sim 1}\} \\ 0 & \text{otherwise} \end{cases} \quad (4.16)$$

The aim of such additional heuristics is to guide the hierarchical translation process by preferring either hierarchical rules or lexical rules.

4.4.2 Word Class Language Model

The language model as previously described (cf. Section 3.5.1) suffers from data sparsity issues, i.e. only few training examples are observed resulting in unreliable probability estimates. One approach to reduce the sparsity is to use a smaller vocabulary. This can be achieved by clustering the vocabulary into a fixed number of word classes with an algorithm as described in [Och 99]. In the next step, the words in the training corpus are substituted by the class index and a language model is then trained on this corpus. This results in a smoother distribution due to less sparsity. We denote the class of a given word e as $c(e)$ and define $c(e_i^{i'}) := c(e_i), c(e_{i+1}), \ldots, c(e_{i'})$. Replacing words with classes, the n-gram language model is redefined as

$$h_{wcLM}(f_1^J, e_1^I, d) = \log \prod_{i=1}^{I} p(c(e_i)|c(e_{i-n+1}^{i-1})). \quad (4.17)$$

As an additional advantage higher n-gram orders may be modeled efficiently. This concept is extended for hierarchical translation models in Chapter 6.

4.5 Contributions

The author of this thesis implemented the enhancements of the hierarchical decoding process as described in Section 4.3.3 and 4.3.4 into the RWTH Aachen University's open source translation toolkit *Jane*. Furthermore, the word class language model for hierarchical translation

(Section 4.4.2) was integrated by the author as contribution to [Wuebker & Peitz+ 13b]. As a minor contribution, the author introduced the sentence-start and -end symbols into the extraction process 4.2.2. However, this modification involved some changes in the feature functions and in the decoder.

4.6 Related Work

Hierarchical phrase-based translation was proposed by [Chiang 05] and is part of several open source translation toolkits such as *Moses* [Koehn & Hoang+ 07, Hoang & Lopez 09], *Joshua* [Li & Callison-Burch+ 09], *cdec* [Dyer & Weese+ 10] and *Jane* [Vilar & Stein+ 10a]. All toolkits implement the state-of-the-art cube pruning algorithm [Chiang 07]. Except for *Jane*, they further use the language model interface introduced by [Heafield 11] for a more efficient language model querying. During the course of this thesis, this concept was introduced into *Jane*. In [Heafield 13] an extensive comparison of these hierarchical translation engines including the improved version of RWTH Aachen University's engine is performed. Varieties in translation speed and memory consumption are related to differences in implementation and data structure.

5. Generative Training

In this chapter, we present our framework for generative training of hierarchical translation models. After a motivation given in Section 5.1, the different components of the framework are described in Section 5.2 and Section 5.3. This chapter concludes with a description of the contribution of the author of this thesis (Section 5.4) and an overview of related work (Section 5.5).

5.1 Introduction

The extraction of hierarchical translation models as described in Section 4.2.2 has several shortcomings. The process is based on heuristics without any foundation in statistics and the rules are extracted from Viterbi word alignments. Thus, this approach does not consider whether a rule is extracted from a likely alignment or not. Furthermore, during the extraction process, models employed in decoding are not considered and the translation probabilities are obtained by simple counting.

In this thesis, we investigate a novel approach for hierarchical translation models to go beyond pure counting of heuristically extracted rules. With this approach the translation probabilities are directly estimated by applying an expectation-maximization (EM) inspired algorithm. Similar to the classical EM algorithm [Dempster & Laird$^+$ 77], our algorithm is divided into an expectation step and a maximization step. For the expectation step, the training data are parsed to get all possible derivations between the source and the target sentences. This procedure is called *forced decoding* and is described in detail in the following section. From the resulting parsing trees all applied rules are extracted and the corresponding translation probabilities are re-estimated. We present two different approaches to perform the re-estimation of the translation model in Section 5.3. As maximization step, we propose to re-estimate the translation probabilities either with the inside-outside algorithm (Section 5.3.1) or based on the number of occurrences of the applied rules given the k-best derivations (Section 5.3.2).

5.2 Forced Decoding

Forced decoding consists of several components and makes use of different existing methods which are modified for the purpose of consistent training of hierarchical translation models. The core component is related to the expectation step where the training data are parsed to get all possible derivations. Given a sentence pair of the training data, the translation of the source sentence is constrained to produce the corresponding target sentence. For this constrained decoding process, the language model score is constant as the translation is fixed. Hence, the incorporation of a language model into the hierarchical translation process (cf. Section 4.3.3) is not needed. This results in a simplification of the decoding process as the cube pruning algorithm must not be

employed. Consequently, forced decoding for hierarchical phrase-based translation is equivalent to synchronous parsing of the training data.

5.2.1 Bilingual Parsing

In previous work (c.f. [Blunsom & Cohn$^+$ 08b, Huang & Zhou 09, Čmejrek & Zhou 10]), bilingual parsing of the parallel training data is performed with a bilingual parser which results in a long run-time. To make forced decoding feasible for larger training corpora, we apply a two-parse algorithm instead of full bilingual parsing. The two-parse algorithm performs two monolingual parses, one on the source language sentence f_1^J and one on the target language sentence e_1^I. In our work, each parse is performed by a modified CYK+ variant which will be described in Section 5.2.2. As for bilingual parsing, the time complexity is $\mathcal{O}(J^3 I^3)$. However, the two-parse algorithm improves the average run-time as reported in [Dyer 10].

Given a heuristically obtained rule set $\mathcal{R}$ as described in Section 4.2.2, the parallel training corpus is parsed with the two-parse algorithm.

Source Parse

Given a sentence pair (f_1^J, e_1^I), we first parse the source sentence f_1^J with the source sides of the rules of the given rule set $\mathcal{R}$. Note, we ensure that each source sentence can always be parsed by employing several heuristics during the extraction process to get all necessary rules as described in [Stein 12]. These heuristics allow for extending the phrase blocks at the border if there are unaligned words and for extracting of aligned single words within a phrase pair. In addition, even non-aligned single words are extracted.

From the resulting hypergraph the target sides of the applied rules are extracted. This is done by simply traversing from the final hypernode associated with the start symbol S through the hypergraph. Then, the non-terminals on the left-hand side and the non-terminals of the target side are annotated with the source span of the corresponding non-terminals in the hypergraph. This needs to find only those parses which would be obtained with a bilingual parsing algorithm. Given a hypernode associated with the non-terminal X and covering the source sentence from j to j', rules of the following form are added to a new rule set $\mathcal{R}_t$:

$$X_j^{j'} \rightarrow \langle \alpha, \beta \rangle \tag{5.1}$$

$$X_j^{j'} \rightarrow \langle \alpha_1 X^{\sim 0} \alpha_2, \beta_1 X_k^{k' \sim 0} \beta_2 \rangle \tag{5.2}$$

$$X_j^{j'} \rightarrow \langle \alpha_1 X^{\sim 0} \alpha_2 X^{\sim 1} \alpha_3, \beta_1 X_k^{k' \sim 0} \beta_2 X_l^{l' \sim 1} \beta_3 \rangle \tag{5.3}$$

$$X_j^{j'} \rightarrow \langle \alpha_1 X^{\sim 0} \alpha_2 X^{\sim 1} \alpha_3, \beta_1 X_l^{l' \sim 1} \beta_2 X_k^{k' \sim 0} \beta_3 \rangle \tag{5.4}$$

where $1 \leq j \leq k \leq k' \leq j' \leq J$, $1 \leq j \leq l \leq l' \leq j' \leq J$ and the sub-nodes cover the source sentence from k to k' and l to l', respectively. An example is given in Figure 5.1. Analogously, we annotate applied glue rules

$$S_1^J \rightarrow \langle S^{\sim 0} X^{\sim 1}, S_1^{j \sim 0} X_j^{J \sim 1} \rangle \tag{5.5}$$

where $1 \leq j \leq J$. Hence, the set of non-terminals now consists of several annotated start symbols and generic non-terminals. In our implementation, the annotated target parts are stored in a temporary prefix tree. The leaves of this tree are the annotated left-hand side non-terminals and point to the corresponding original rules. We only store the target sides of the rules which match a sequence in the corresponding target sentence. Thus, the final rule set R_t contains those rules of R which match the source and the target sentence. New rules are not created. To make forced decoding feasible for larger corpora, observation histogram pruning is applied.

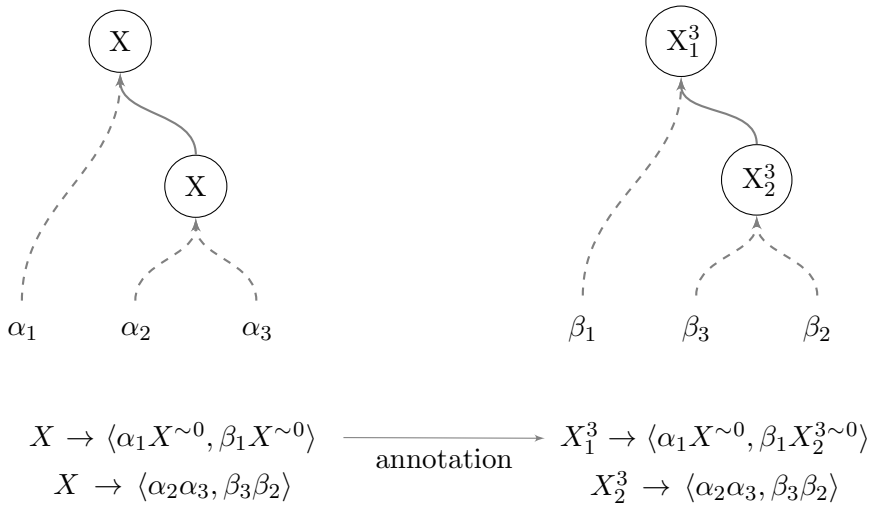


Figure 5.1: The hypergraph on the left is generated by parsing the string $\alpha_1\alpha_2\alpha_3$. By annotating the given rules with the source span, a new grammar is generated which is then applied for parsing $\beta_1\beta_3\beta_2$.

Target Parse

In the second pass, the target sentence e_1^I is parsed using the annotated target sides of the bilingual rules of the new rule set $\mathcal{R}_t$. The target sentence is generated by the new grammar and the forced decoding procedure is successful if the start symbol S_1^J is found in cell $(I, 1)$. In contrast to the source parse, it is possible that a target parse is not found due to the fact that we prune necessary target sides as described before. If a target sentence can not be parsed, we discard the sentence pair. Preliminary experiments have shown that keeping partial parses seems not to improve the translation quality. The generated hypergraph represents all possible synchronous derivations between the source and target sentence. The corresponding example of a hypergraph generated by the annotated target side is also given in Figure 5.1. The search as we will describe in Section 5.3 is performed on this hypergraph in order to extract the applied rules.

5.2.2 Modified CYK+ Variant

Bilingual parsing as described above generates a huge amount of non-terminals for the target parse. For a source sentence of the length J, the maximum number of non-terminals is $\frac{J(J+1)}{2}$. Such a large set of non-terminals increases both memory consumption and decoding time if the modified CYK+ algorithm as introduced in Section 4.3.2 is applied. The main reason for that is the type-2 list which stores each partial parse for each non-terminal. However, most of these parses may not be considered in the later stages of the parse. To avoid these type of lists, we apply a variant of CYK+ introduced by [Sennrich 14] which does not reduce the complexity of the CYK+ algorithm but is more memory efficient and faster. The described algorithm was actually developed for syntax-based machine translation with an extended set of non-terminals [Williams & Koehn 12]. We adapt this approach for the purpose of forced decoding and apply it for parsing the target sentence using the annotated rule set.

The CYK+ variant works recursively from right-to-left rather than iteratively from left-to-right (cf. Section 4.3.2). It starts from the rightmost cell and partial parses are immediately expanded into full ones. Thus, partial parses do not have to be stored in type-2 lists. As a first step,

the chart is initialized as in the original CYK+ algorithm. For each cell (l, j) following step is performed:

- Check all combinations of a partial parse $\alpha\bullet$ and type-1 element Z in a cell $(j, k-j+1)$ with $j \leq k \leq J$ for which a rule of the form $Y \rightarrow \langle \alpha Z \gamma, \cdot \rangle$ in $\mathcal{R}$ exists. If γ is empty, add Y to the type-1 list of cell $(l, j-l+1)$, otherwise recursively repeat this step with $\alpha Z\bullet$ and $j = k+1$.

Similar to the modified CYK+ algorithm described in Section 4.3.2, we modify this algorithm to generate a hypergraph for a given sentence. The full algorithm as implemented in our framework is depicted in Algorithm 5.1.

Algorithm 5.1 Modified CYK+ Variant.

Input: source sentence f_1^J, rule set $\mathcal{R}$
Output: hypergraph representing all valid parses of f_1^J

```
1: create an initial hypernode h
2: for j = 1, ..., J do
3:     if ∃ a set of rules R of the form Y → ⟨f_j, ·⟩ ∈ R then
4:         create hypernode n associated with Y to type-1 list of cell (1, j)
5:         add hyperedge from [h] to n, associated with R
6: for l = J, ..., 1 do
7:     for j = l + 1, ..., J do
8:         L = []
9:         CONSUME(l, l, j − 1)

10: function CONSUME(int l, int j, int k)
11:     if j = k then
12:         if l ≠ j and ∃ a set of rules R of the form Y → ⟨f_l^{j−l+1}, ·⟩ ∈ R then
13:             create hypernode n associated with Y to type-1 list of cell (l, j)
14:             add hyperedge from [h] to n, associated with R
15:     for each set of rules R of the form Y → ⟨f_l^{j−l+1}•, ·⟩ ∈ R do
16:         for each (Z, n) in type-1 list of (j, k − j + 1) do
17:             L = L ++ [n]
18:             if l ≠ j and ∃ a set of rules R of the form Y → ⟨f_l^{j−l+1} Z, ·⟩ ∈ R then
19:                 create hypernode n′ associated with Y in type-1 list of cell (l, j − l + 1)
20:                 add a hyperedge from L to n′, associated with R
21:             if ∃ a set of rules R of the form Y → ⟨f_l^{j−l+1} Zγ, ·⟩ ∈ R then
22:                 for k = j + 1, ..., J do
23:                     CONSUME(l, k + 1, k)
24:             pop(L)
```

5.2.3 Leave-One-Out

A known issue of generative translation model training is over-fitting [DeNero & Gillick$^+$ 06]. Due to the fact that all rules which are extracted from a sentence pair are used in the forced decoding step, longer rules are often preferred. Even though those long rules only match for a few sentences of the training data and do not generalize very well, they tend to be assigned very high translation probabilities (cf. Section 6.1). In [Wuebker & Mauser$^+$ 10] a leave-one-out method is

described which counteracts the over-fitting. This method modifies the translation probabilities in the forced decoding step for each sentence pair. The occurrences of a given rule in a sentence pair are subtracted from the rule counts obtained from the full training data resulting in the modified translation probability

$$p_{\mathrm{l1o},n}(r) = p_{\mathrm{l1o},n}(\alpha|\beta) = \frac{N(\alpha,\beta) - N_n(\alpha,\beta)}{\sum_{\alpha'} \left(N(\alpha',\beta) - N_n(\alpha',\beta)\right)} \tag{5.6}$$

where $N_n(\alpha,\beta)$ is the count for the rule $X \rightarrow \langle\alpha,\beta\rangle$ that was extracted from n-th sentence pair (F_n, E_n). Singleton rules, which are rules occurring only in one sentence pair, are handled differently. These rules get a low probability which can be computed in two different ways: uniform and length-based. While the uniform leave-one-out method uses a fixed probability for every singleton rule, the length-based leave-one-out method assigns a probability depending on the source and target rule lengths. In this work, we apply the idea of length-based leave-one-out for the hierarchical approach and treat the non-terminals on the right-hand side of the rules in the length-based approach like terminals. Hence, Equation 5.6 is rewritten for length-based leave-one-out

$$p_{\mathrm{lbl1o},n}(\alpha|\beta) = \begin{cases} \frac{N(\alpha,\beta) - N_n(\alpha,\beta)}{\sum_{\alpha'} \left(N(\alpha',\beta) - N_n(\alpha',\beta)\right)} & N(\alpha,\beta) - N_n(\alpha,\beta) > 0 \\ \gamma^{|\alpha|+|\beta|} & \text{otherwise} \end{cases} \tag{5.7}$$

where γ is a fixed penalty and set to e^{-5} in our experiments. In the following example, the longer rule

$$X \rightarrow \langle \text{Und zwar sollen derartige Strafen, It says that this should} \rangle \tag{5.8}$$

is used when leave-one-out is not applied. Such long rules match few sentence pairs only and hardly generalize well to unseen test data. Using leave-one-out, three shorter, more general rules are employed

$$X \rightarrow \langle \text{Und zwar } X^{\sim 0}, \text{ It } X^{\sim 0} \rangle \tag{5.9}$$

$$X \rightarrow \langle \text{sollen } X^{\sim 0}, \ X^{\sim 0} \text{ should} \rangle \tag{5.10}$$

$$X \rightarrow \langle \text{derartige Strafen, says that this} \rangle. \tag{5.11}$$

In practice, leave-one-out is applied after the source parse when the temporary prefix tree of the annotated target parts is created. At the leaves of this prefix tree, the modified translation scores are stored.

Another approach to address the over-fitting problem is to use a Baysian framework as introduced for the phrase-based approach in [DeNero & Bouchard-Côté$^+$ 08]. The basic idea is to apply a prior distribution over the phrase translation probabilities. This idea was later extended for the hierarchical approach [Blunsom & Cohn$^+$ 08a, Blunsom & Cohn$^+$ 09]. However, improvements could only be shown on rather small translation tasks. On larger tasks the gain was smaller and in some cases non-existing. We therefore decided to apply leave-one-out which seems to be a simple but effective method.

5.3 Translation Model Training

In this work, we propose two different approaches to obtain the expected counts of the rules applied during forced decoding. With these counts, the translation probabilities are updated in the maximization step. The first method applies the inside-outside algorithm on the target hypergraph (Section 5.3.1). In Section 5.3.2, we present an algorithm to search for the k-best

derivations between a given sentence pair. Based on these k-best derivations, the occurrences of the applied rules is counted. Both approaches do not apply any pruning but operate on an already pruned hypergraph (cf. Section 5.2.1). Additionally, we describe different methods to merge the generative and the heuristically estimated translation models in Section 5.3.3.

5.3.1 Inside-Outside Algorithm

In [Huang & Zhou 09], the inside-outside algorithm for SCFGs was introduced in order to re-estimate the translation probabilities after performing bilingual parsing. However, due to the fact that we perform two monolingual parses, we apply the inside-outside algorithm, as it was originally defined for PCFGs [Kurihara & Sato 04], on the target hypergraph.

For a given span of words $e_i^{i'}$ and a non-terminal $X_j^{j'}$, the inside probability $\mathcal{I}_{i,i'}(X_j^{j'})$ is the probability that $X_j^{j'}$ covers the sentence from position i to position i'. The outside probability $\mathcal{O}_{i,i'}(X_j^{j'})$ can be considered as the probability of the whole sentence except for the span $e_i^{i'}$. By multiplying both probabilities, we get the overall probability of the sentence. Hence, we define the inside and outside probabilities as follows for a non-terminal $X_j^{j'}$ covering the target sentence from i to i'

$$\mathcal{I}_{i,i'}(X_j^{j'}) = p(X_j^{j'} \overset{*}{\Rightarrow} e_i^{i'}) \tag{5.12}$$

$$\mathcal{O}_{i,i'}(X_j^{j'}) = p(S_1^J \overset{*}{\Rightarrow} e_1^{i-1} X_j^{j'} e_{i'+1}^I). \tag{5.13}$$

Both probabilities can be defined recursively and calculated by traversing the hypergraph in bottom-up order for the inside probabilities and in top-down order for the outside probabilities. Thus, the probability of the whole sentence is

$$\mathcal{I}_{1,I}(S_1^J) = p(S_1^J \overset{*}{\Rightarrow} e_1^I). \tag{5.14}$$

Based on the inside and outside probabilities the expected count $N_{FD}(r)$ is computed for a rule $r = X_j^{j'} \rightarrow \langle \alpha, \beta \rangle$ covering the target sentence from i to i'. In the Equation 5.15, we define the expected count for lexical rules, hierarchical rules with one non-terminal and hierarchical rules with two non-terminals. We further assume that the first non-terminal of the target side $X_k^{k'}$ covers the target sentence from i_1 to i'_1 and the second one $X_l^{l'}$ spans from i_2 to i'_2. Furthermore, $w(r)$ is the score of the log-linear feature combination of the rule r in the initial rule set, i.e. we use the heuristically extracted rule set to initialize these weights. The scaling factors of the log-linear feature combination have been tuned with MERT beforehand.

$$N_{FD}(r = X_j^{j'} \rightarrow \langle \alpha, \beta \rangle) = \begin{cases} \frac{w(r)}{\mathcal{I}_{1,I}(S_1^J)} \mathcal{O}_{i,i'}(X_j^{j'}) & \beta = e_i^{i'} \\ \frac{w(r)}{\mathcal{I}_{1,I}(S_1^J)} \mathcal{O}_{i,i'}(X_j^{j'}) \mathcal{I}_{i_1,i'_1}(X_k^{k'}) & \beta = e_i^{i_1-1} X_k^{k'} e_{i'_1+1}^{i'} \\ \frac{w(r)}{\mathcal{I}_{1,I}(S_1^J)} \mathcal{O}_{i,i'}(X_j^{j'}) \mathcal{I}_{i_1,i'_1}(X_k^{k'}) \mathcal{I}_{i_2,i'_2}(X_l^{l'}) & \beta = e_i^{i_1-1} X_k^{k'} e_{i'_1+1}^{i_2-1} X_l^{l'} e_{i'_2+1}^{i'} \end{cases} \tag{5.15}$$

Note that when the expected counts for a rule are summed up over all hypernodes of each sentence pair and over all sentence pairs of the training data, the annotation is dropped to get the expected count of the original non-annotated rule. For the maximization step, we use the expected counts $N_{FD}(r)$ of a rule $r = X \rightarrow \langle \alpha, \beta \rangle$ to update the translation probability

$$p_{FD}(\alpha|\beta) = \frac{N_{FD}(X \rightarrow \langle \alpha, \beta \rangle)}{\sum_{\alpha'} N_{FD}(X \rightarrow \langle \alpha', \beta \rangle)}. \tag{5.16}$$

This is also done for the inverse translation probability $p_{FD}(\beta|\alpha)$.

5.3.2 k-Best Derivations

Given the target hypergraph from the forced decoding step, we employ a top-down k-best parsing algorithm to find the best k-best derivations between the given source and target sentence. In this step, all models of the translation process are included (except for the language model). The corresponding scaling factors have been optimized with MERT beforehand.

k-Best Parsing

The algorithm used for finding the k-best derivations is similar to [Huang & Chiang 05]. It starts from the final hypernode of the target parse and calls itself recursively only as necessary. In contrast to the cube pruning algorithm presented in Section 4.3.3, we can ignore the language model score which results in a simpler algorithm.

The full algorithm is presented in Algorithm 5.2 and includes a recombination scheme which will be introduced in the following paragraph (Line 15-20). In this algorithm a derivation is represented by a tuple $d = (e, \mathbf{j})$ composed by a hyperedge e and an $|e|$-dimensional vector $\mathbf{j}$ which indexes the k-best derivations in the predecessor hypernodes of hyperedge e. Each hypernode h stores a list of k-best derivations D which can be accessed via $h \rightarrowtriangle D$. Furthermore, the set of rules r applied in a derivation d are stored in a list of sets R. C is a priority queue of candidate derivations. This heap is initialized for each node (Line 7-10) and kept up-to-date in Line 40. The algorithm stops when the desired amount of derivations is found (Line 5 and 11) or the list of candidates is empty.

An illustration of the concept of this algorithm is shown in Figure 5.2. In this example, we want to compute the 5-best list of derivations at the goal hypernode g. This hypernode has two incoming hyperedges e_1 and e_2. The first-best derivation $d_1 = (e_1, \mathbf{j})$ of the goal hypernode g is associated with the hyperedge e_1 which refers to the first-best derivation of corresponding predecessor hypernodes (h_3 and h_4). This concept is applied recursively to these predecessor hypernodes, i.e. the first-best derivation is computed for these hypernodes as well. The second-best derivation d_2 of g is associated to first-best derivation of h_2 and h_3 via e_2 but could also be associated to another derivation following e_1 again.

Recombination

In standard hierarchical phrase-based decoding, partial derivations that are indistinguishable from each other are recombined (cf. Section 4.3.3). Either derivations that produce identical translations or derivations with identical language model context are recombined [Huck & Vilar$^+$ 13]. As in forced decoding the translation is fixed and a language model is missing, both schemes are not suitable.

However, a recombination scheme is necessary to avoid derivations with the same application of rules. Furthermore, recombining such derivations simultaneously increases the amounts of considered derivations during k-best parsing. Given two derivations with the same set of applied rules, the order of application of the rules may be different. Thus, we propose following scheme for recombining derivations in forced decoding: Derivations that produce identical sets of applied rules are recombined. Figure 5.3 shows an example for $k = 3$. Employing the proposed scheme, derivations d_1 and d_2 are recombined since both share the same set of applied rules $\{r_1, r_3, r_2\}$.

5.3.3 Interpolation

In [DeNero & Gillick$^+$ 06] an improvement in translation quality was reported by the interpolation of translation models produced by the generative and heuristic approach. We adopt this

Algorithm 5.2 k-best parsing algorithm.

Input: A hypernode and the size k of the k-best list
Output: D, a list with k-best derivations

```
 1: g = goal hypernode of the hypergraph
 2: g⇾KthBest(k)
 3: return g⇾D

 4: function KthBest(k)
 5:     if |D| ≥ k then
 6:         return
 7:     if C is not initialized then
 8:         C = heap(∅)
 9:         for each incoming hyperedge e of this do
10:             Fire((e, 1))
11:     while |D| < k do
12:         if |D| > 0 then
13:             d = D[|D|]
14:             PushSucc(d)
15:         if |C| > 0 then
16:             d = pop(C)
17:             RecombineCandidate(d)
18:         else
19:             break

20: function RecombineCandidate(d)
21:     r = GetRuleSet(d) //get set of applied rules
22:     if r ∈ R then
23:         PushSucc(d)
24:     else
25:         D = D ++ [d]
26:         R = R ++ [r]

27: function PushSucc(d)
28:     notation: d = (e, j)
29:     for i = 1, ..., |e| do
30:         j' = j
31:         j'[i] += 1
32:         Fire((e, j'))

33: function Fire(d)
34:     notation: d = (e, j)
35:     notation: predecessor hypernodes of e : (h_1, ..., h_|e|)
36:     for i = 1, ..., |e| do
37:         h_i⇾KthBest(j[i])
38:         if |h_i⇾D| < j[i] then
39:             return
40:     push d into C
```

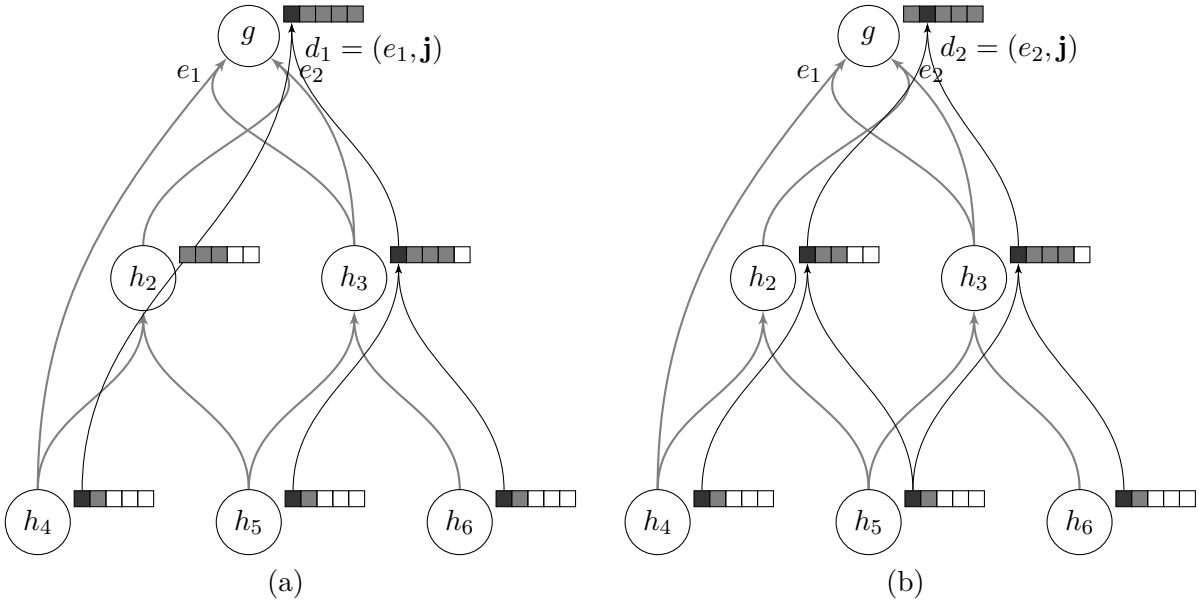


Figure 5.2: Illustration of the k-best parsing algorithm. Each vector associated with a hypernode represents a k-best list (here: 5-best list). The black colored elements are the derivations that are needed for the current derivation d_1 (a) and d_2 (b) at the goal hypernode g. Due to the concept of this algorithm, it can happen that in no other hypernode than the goal hypernode a full 5-best list is computed.

method and perform log-linear interpolation of the learned translation model and the heuristically extracted one. The log-linear interpolated probabilities $p_{\text{logInt}}(r)$ are computed as

$$p_{\text{logInt}}(r) = p(r)^{1-\omega} \cdot p_{FD}(r)^{\omega} \tag{5.17}$$

where ω is the interpolation weight, p is the heuristically estimated translation model and p_{FD} the generative model. The interpolation weight is adjusted on a development set. Note that the resulting translation model is an intersection of both.

When forced decoding is performed on a sub-set of the training data, e.g. on high-quality in-domain data, we adopt the approach as presented in [Bisazza & Ruiz$^+$ 11]. With the *fill-up* method the heuristically estimated translation model $\mathcal{R}$ and the learned translation model $\mathcal{R}_{FD}$ are unified by keeping the translation probabilities from generative model $p_{FD}(r)$ whenever possible. Otherwise, the probabilities from the heuristically estimated translation model $p(r)$ are taken. Given that $\mathcal{R}_{FD} \subseteq \mathcal{R}$, the translation probabilities $p_{\text{fillUp}}(r)$ of the unified translation model are computed as

$$p_{\text{fillUp}}(r) = \begin{cases} p_{FD}(r) & r \in \mathcal{R}_{FD} \\ p(r) & \text{otherwise.} \end{cases} \tag{5.18}$$

In addition, the unified translation model is extended by an additional feature which fires if the rule is obtained in the learned translation model.

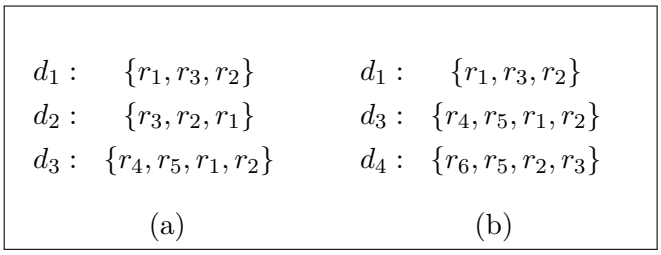


Figure 5.3: Example search space before (a) and after (b) applying recombination.

5.4 Contributions

The generative training approach for hierarchical translation models described in this chapter was joint work of the author of this thesis with Arne Mauser, Joern Wuebker and David Vilar. Arne Mauser suggested using the two-parse and the inside-outside algorithm, Joern Wuebker provided the framework for leave-one-out and David Vilar did the initial implementation of the k-best derivations algorithm. The author implemented the two-parse and the inside-outside algorithm, refined the k-best derivations algorithm and extended it with the presented recombination scheme. Furthermore, the author contributed the idea to adapt the CYK+ variant for forced decoding in order to allow larger training sets. The entire implementation is part of RWTH Aachen University's open source translation toolkit *Jane*. The approach towards generative training of hierarchical translation models and experimental results were published in [Peitz & Mauser$^+$ 12, Peitz & Vilar$^+$ 14].

5.5 Related Work

In recent years, several works have investigated the generative training of the translation model to close the gap between the extraction and the translation process.

In [Marcu & Wong 02], a joint probability model is presented which estimates phrase translation probabilities from a parallel corpus. For aligning the phrases and estimating the probabilities, the EM algorithm is applied. In [Birch & Callison-Burch$^+$ 06] the joint probability model is constrained by a word alignment to limit the complexity.

The problem of over-fitting due to the EM algorithm is analyzed in [DeNero & Gillick$^+$ 06] and a solution is proposed in [Wuebker & Mauser$^+$ 10] by applying leave-one-out. The authors conducted their experiments using a phrase-based translation system on the WMT 2008 German→English task and report an improvement of up to 1.4 points in BLEU and 1.7 points in TER. We will adopt the leave-one-out method in this work and show that its benefits translate to the hierarchical case.

Another approach to learn from decoding on the training data is presented in [Duan & Li$^+$ 12]. In this work, a training method based on forced derivation trees is described. This structure is used to train apart from a translation model, a distortion model, a source language model and a rule sequence model. As first step, they verified their method within a phrase-based system. As future work, the authors suggest to adapt this method for the hierarchical approach.

Besides these publications about phrase training for the phrase-based approach, several works have been presented for hierarchical machine translation during the past years. In most of these papers the idea of bilingual parsing on parallel corpora is described.

In [Blunsom & Cohn$^+$ 08b] a discriminative model using derivations as a hidden variable is presented. In training, they perform a synchronous parsing of the source and target sentences using a modified CYK algorithm over two dimensions with a time complexity of $\mathcal{O}(J^3 I^3)$ where

J is the source sentence length and I the target sentence length. Furthermore, the inside-outside algorithm is employed. The experiments were carried out on a sub-set of the French→English Europarl corpus (170K sentences) and show comparable results. Another observation is that their model improves as they increase the number of parsable training sentences. This observation motivated us to make generative training feasible for larger training corpora.

Bilingual parsing on parallel corpora is also described in [Huang & Zhou 09], [Čmejrek & Zhou$^+$ 09] and [Čmejrek & Zhou 10]. They also use the EM algorithm to re-estimate the translation probabilities. In the maximization step, the expected counts are computed with the inside-outside algorithm to update the translation probabilities. However, this is done for non-lexical rules only. Experiments on the IWSLT 2006 Chinese→English translation task (40K sentences without punctuation and case information) result in a significantly better BLEU score. In [Čmejrek & Zhou$^+$ 09] and [Čmejrek & Zhou 10], this work is extended and they report improvement on a sub-set of the German→English Europarl corpus (300K sentences without punctuation and case information).

In [Heger & Wuebker$^+$ 10], heuristically extracted hierarchical rules are combined with phrases trained as in [Wuebker & Mauser$^+$ 10]. Experimental results on IWSLT Arabic→English and WMT English→German tasks show improvements in translation quality and motivated us to investigate the impact of generative training for the hierarchical approach.

In [Dyer 10] a synchronous parsing algorithm is introduced that is based on two successive monolingual parses. Instead of performing one bilingual parse for a given sentence pair, a two-parse algorithm is applied. This improves the average run-time. The authors report a speed improvement on the same task as in [Blunsom & Cohn$^+$ 08b]. We apply this approach to reduce the average run-time of our forced decoding procedure.

Compared to previously described approaches for generative training of hierarchical translation models, we are able to employ forced decoding on larger tasks. Furthermore, we propose two different schemes to calculate the expected count for all type of rules. Using the inside-outside algorithm we additionally apply threshold pruning on the rule set using the estimated expected counts. This leads to a more consistent pruning and a smaller translation model. Another difference is that we perform leave-one-out to counteract over-fitting. Furthermore, in the forced decoding procedure, we include the log-linear combination of all models which are used in the translation process except for the language model.

6. Smoothing

In this chapter, we describe several smoothing methods starting with a brief introduction in Section 6.1. The smoothing approaches introduced in Section 6.2 and Section 6.3 use count of rules or make use of word classes, respectively and can be applied in both, phrase-based and hierarchical translation. However, syntax-based smoothing as presented in Section 6.4 is explicitly designed for hierarchical translation. This chapter concludes with a description of the contribution of the author of this thesis (Section 6.5) and an overview of related work (Section 6.6).

6.1 Introduction

Smoothing is an important technique in statistical natural language processing to remedy the data sparseness problem. In statistical machine translation, conditional probabilities such as the translation probabilities (cf. Section 4.4.1) are usually overestimated for rare rules. For example, if there is a rule whose source or target side occur only once in the training data, then its translation probability equals 1 which might be higher than those of other bilingual rules with more occurrences. This overestimation can be counteracted by applying smoothing.

In [Foster & Kuhn$^+$ 06], a distinction between two different approaches for smoothing translation models is being made:

- *Black-Box*: Translation units are treated as atomic objects and only the corresponding counts are known.

- *Glass-Box*: Translation units are broken down into their component words.

Word-based lexical smoothing (cf. Section 4.4.1) is an example for the glass-box approach. It makes use of the individual word translation probabilities of the words within a rule. An extensive comparison of different word-based lexicon models for hierarchical phrase-based translation can be found in [Huck & Mansour$^+$ 11]. Smoothing using word classes, as will be presented in Section 6.3, works in a similar fashion. This approach does not assign individual probabilities to words but first maps words of a phrase to word classes and re-estimates the corresponding translation probabilities.

The estimation of translation probabilities for hierarchical rules is even coarser, because they are extracted from lexical rules and their counts are equally distributed from the initial rule count. For a better discrimination of hierarchical rules, an additional feature, which is independent from the rule count, is needed. For this purpose, a glass-box approach designed for hierarchical rules using syntactic information will be presented in Section 6.4.

6.2 Count-Based Smoothing

The count-based smoothing methods presented in this section are related to the black-box approach. They are easy to compute and make use of the count of a rule $N(\alpha, \beta) = N(r)$ with $r = X \rightarrow \langle \alpha, \beta \rangle$. We introduce three different methods.

6.2.1 Frequency Indicator Feature

The frequency indicator feature $h_{\text{FreqInd},\tau}(r)$ [Mauser & Vilar$^+$ 07] penalizes rules that have a count lower than a threshold τ:

$$h_{\text{FreqInd},\tau}(r) = \begin{cases} 1 & N(\alpha, \beta) > \tau \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

In this work, we use four different thresholds ($\tau \in \{1, 2, 3, 5\}$) which results in four additional feature functions in the log-linear model combination.

6.2.2 Enhanced Low-Frequency Feature

As an alternative to the frequency indicator feature, the enhanced low-frequency feature $h_{\text{Elf}}(r)$ [Chen & Kuhn$^+$ 11] introduces a continuously decreasing penalty:

$$h_{\text{Elf}}(r) = \frac{1}{N(\alpha, \beta)} \tag{6.2}$$

The advantage of the enhanced low-frequency feature is that only one additional parameter has to be learned with MERT, which might be more stable under different optimization runs.

6.2.3 Good-Turing Smoothing

In contrast to the previously presented approaches, Good-Turing smoothing [Church & Gale 91], as proposed to apply for phrase-based translation in [Foster & Kuhn$^+$ 06], adjusts the count of a given rule directly. Thus, this approach does not introduce an additional feature function but a re-estimation of the translation probabilities. The idea of Good-Turing smoothing is to reallocate successively count mass from larger to smaller counts. In the scope of language model smoothing, this approach is used to estimate probabilities for unseen n-grams. Here, the aim is to assign rare seen rules an adjusted count, which results in a smoother distribution. Based on the adjusted count $N_{\text{GT}}(\alpha, \beta)$, the translation probability $p_{\text{Rule,T2S}}(r)$ is re-estimated as following:

$$N_{\text{GT}}(\alpha, \beta) = \frac{(N(\alpha, \beta) + 1) \cdot N_{N(\alpha,\beta)+1}}{N_{N(\alpha,\beta)}} \tag{6.3}$$

$$p_{\text{Rule,T2S}}(r) = \frac{N_{\text{GT}}(\alpha, \beta)}{\sum_{\alpha'} N(\alpha', \beta)}, \tag{6.4}$$

where $N_{N(\alpha,\beta)}$ denotes the count of counts, i.e. the number of occurrences of the count $N(\alpha, \beta)$.

In practice, $N_{N(\alpha,\beta)}$ becomes noisy or even 0 for larger counts. In our work, we circumvent this issue by applying Good-Turing smoothing only for counts lower than 10.

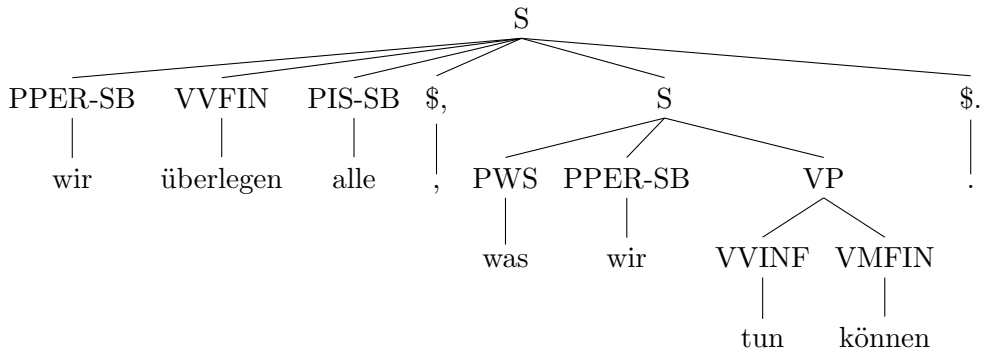


Figure 6.1: Example of a syntactical parse tree (short: syntax tree) for the German sentence "wir überlegen alle, was wir tun können.".

6.3 Class-Based Smoothing

Translation model smoothing as presented in [Wuebker & Peitz$^+$ 13b] is a glass-box technique based on word classes. These word classes can be obtained by the algorithm as described in [Och 99]. Similar to the idea of word class language models introduced in Section 4.4.2, the words in the bilingual training corpus are replaced by their word classes. Thus, the source as well as the target vocabulary are reduced, which leads to a smoother distribution due to less sparsity. We denote the class of a given word w by $c(w)$ and define $c(\alpha) := c(\alpha_i^{i'}) = c(\alpha_i)c(\alpha_{i+1})\dots c(\alpha_{i'})$. Class-based translation model probabilities, word-based lexicon models and additional count-based features are defined analogously to the standard models (c.f. Section 4.4.1) and the previously introduced smoothing feature functions:

$$p_{\text{wcRule,T2S}}(r) = p(c(\alpha)|c(\beta)) = \frac{N(c(\alpha), c(\beta))}{\sum_{\alpha'} N(c(\alpha'), c(\beta))} \tag{6.5}$$

$$p_{\text{wcLex,T2S}}(r) = \frac{1}{(I+1)^J} \prod_{j=1}^{J} \left(\mathbb{I}(\alpha_j \notin \mathcal{N}) p(c(\alpha_j)|\beta_0) + \sum_{i=1}^{I} (\mathbb{I}(\alpha_j \notin \mathcal{N}) \mathbb{I}(\beta_i \notin \mathcal{N}) p(c(\alpha_j)|c(\beta_i))) \right) \tag{6.6}$$

$$h_{\text{wcFreqInd},\tau}(r) = \begin{cases} 1 & N(c(\alpha), c(\beta)) > \tau \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

$$h_{\text{wcElf}}(r) = \frac{1}{N(c(\alpha), c(\beta))}. \tag{6.8}$$

6.4 Syntax-Based Smoothing

In [Zhou & Zhu$^+$ 08], an additional feature based on syntactical information extracted from parse trees (see Figure 6.1) is introduced to discriminate between different hierarchical rules. This approach can be motivated by the following example.

If a hierarchical rule contains the German word "überlegen", which can be translated into

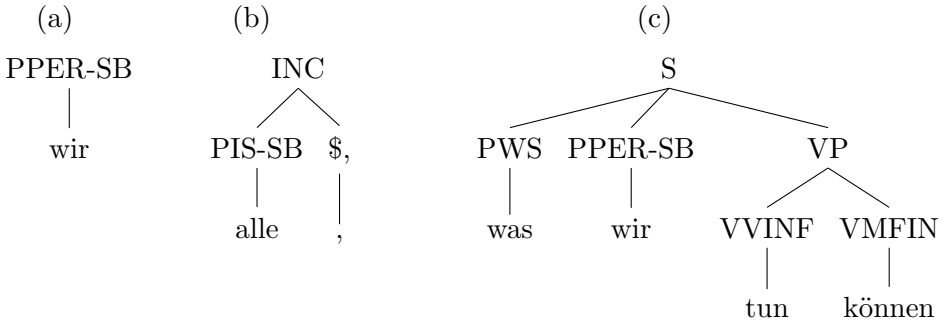


Figure 6.2: Example of tree fragments. Tree fragment (b) is incomplete.

English in several ways depending on the syntactic context, the translation options are ambiguous:

$$X \rightarrow \langle X^{\sim 0} \text{ überlegen } X^{\sim 1},\ X^{\sim 0} \text{ think } X^{\sim 1} \rangle$$
$$X \rightarrow \langle X^{\sim 0} \text{ überlegen } X^{\sim 1},\ X^{\sim 0} \text{ consider } X^{\sim 1} \rangle$$
$$X \rightarrow \langle X^{\sim 0} \text{ überlegen } X^{\sim 1},\ X^{\sim 0} \text{ superior } X^{\sim 1} \rangle$$

These three rules share the same source side but map into different target sides depending on the syntactic role of the non-terminals $X^{\sim 0}$ and $X^{\sim 1}$. This suggests to prefer hierarchical rules that are syntactically more homogeneous, i.e. to measure the variation of syntax structures embodied in the non-terminals of the source side. During decoding, the goal is to choose those derivations which introduce less ambiguity into the translation.

In the context of hierarchical translation model smoothing, this additional feature helps to discount rules where a translation option might be overestimated. For example, the hierarchical rule

$$X \rightarrow \langle X^{\sim 0} \text{ überlegen } X^{\sim 1},\ X^{\sim 0} \text{ think } X^{\sim 1} \rangle$$

is assigned a higher translation probability compared to the rules

$$X \rightarrow \langle \text{wir überlegen } X^{\sim 0},\ \text{we think } X^{\sim 0} \rangle$$
$$X \rightarrow \langle \text{sind überlegen } X^{\sim 0},\ \text{are superior } X^{\sim 0} \rangle.$$

However, the latter ones introduce less ambiguity and are preferred by the additional feature.

6.4.1 Syntactic Variation

The syntactic variation of a non-terminal is measured by computing the similarity of all *tree fragments* of the corresponding sub-phrases. A tree fragment of phrase is a syntactic sub-tree whose leaves span over exactly this phrase. In practice, as the rule extraction process does not follow any syntactic constrains, many phrases are not consistent with such syntactic sub-trees. These *incomplete* tree fragments are then rooted with the label "INC". Examples of tree fragments extracted from the syntax tree shown in Figure 6.1 are given in Figure 6.2. Note that a phrase can have multiple tree fragments.

In this work, syntax trees are obtained by parsing the bilingual corpora with external tools such as the Stanford parser [Klein & Manning 03, Rafferty & Manning 08] or the Berkeley parser [Petrov & Barrett$^+$ 06, Petrov & Klein 07].

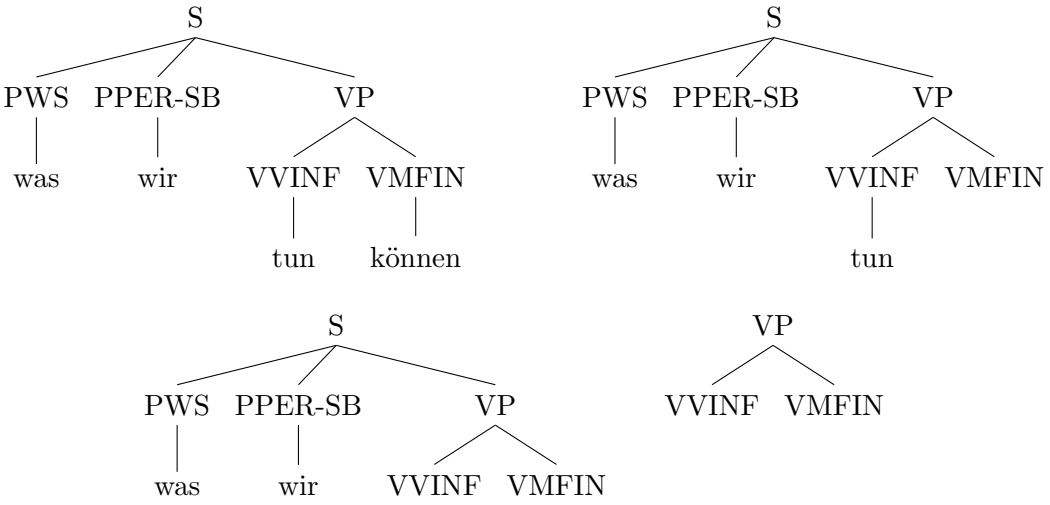


Figure 6.3: Example of sub-set trees extracted tree fragment (c) of Figure 6.2.

6.4.2 Tree Fragment Similarity

The similarity between two different tree fragments T_1 and T_2 is denoted by $K(T_1, T_2)$ and defined as the number of common *sub-set trees* [Collins & Duffy 01]. A sub-set tree is a sub-graph with more than one node of the corresponding tree fragment and includes entire (except for leaves) rule productions. Figure 6.3 enumerates a list of sub-set trees for tree fragment (c) in Figure 6.2.

Let N be the number of all possible sub-set trees. Then, a tree fragment T can be represented as an N-dimensional vector $\mathbf{h}(T) = (h_1, h_2, \ldots, h_N)$ where the function $h_i(T)$ returns the count of occurrences of the i-th sub-set tree in T. Given two tree fragments T_1 and T_2 , the similarity $K(T_1, T_2)$ can be computed by the inner product of $\mathbf{h}(T_1)$ and $\mathbf{h}(T_2)$.

To compute the similarity between two tree fragments for a large N efficiently, we employ convolution kernels [Collins & Duffy 01]. Let N_1 and N_2 be sets of nodes in the tree fragments T_1 and T_2, respectively. The indicator function $I_i(n)$ equals 1 iff a i-th sub-set tree is rooted at node n. The similarity between two tree fragments $K(T_1, T_2)$ is estimated as

$$K(T_1, T_2) = \mathbf{h}(T_1) \cdot \mathbf{h}(T_1) \tag{6.9}$$

$$= \sum_{i=1}^{N} h_i(T_1) h_i(T_2) \tag{6.10}$$

$$= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_{i=1}^{N} \mathbb{I}_i(n_1) \mathbb{I}_i(n_2) \tag{6.11}$$

$$= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} C(n_1, n_2) \tag{6.12}$$

where $C(n_1, n_2)$ can be computed in polynomial time due to the following recursive definition:

- If the rule productions at n_1 and n_2 are different, then $C(n_1, n_2) = 0$.
- If the rule productions at n_1 and n_2 are the same:
  - If n_1 and n_2 are leaves, then $C(n_1, n_2) = 1$.

– Otherwise, $C(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} 1 + C(ch(n_1, j), ch(n_2, j))$, where $nc(n)$ is the number of children of n and $ch(n, j)$ returns the j-th child of node n and λ is a decay factor to discount the effects of deeper tree structures (with $0 < \lambda \leq 1$).

The complexity of this algorithm is $\mathcal{O}(|N_1||N_2|)$. In our implementation, we follow the refinements introduced by [Moschitti 06] which result in linear run-time on average. We further compute the normalized similarity

$$K'(T_1, T_2) = \frac{K(T_1, T_2)}{\sqrt{K(T_1, T_1)} \cdot \sqrt{K(T_2, T_2)}}. \tag{6.13}$$

6.4.3 Purity Feature

Based on the similarity between two tree fragments, the *purity* of a parse tree forest embodied in a non-terminal X of a hierarchical rule is estimated as the average similarity of all tree fragments:

$$Pur(X) = \frac{2}{N(N-1)} \sum_{j=2}^{N} \sum_{i=1}^{j-1} K'(T_i, T_j) \tag{6.14}$$

where N is the number of tree fragments in the parse tree forest of X. The *purity feature* is then defined for a rule $r = X \rightarrow \langle \alpha, \beta \rangle$ as following:

$$h_{\text{purity}}(r) = \begin{cases} \log \min \left(Pur(X^{\sim 0}), Pur(X^{\sim 1}) \right) & \exists X^{\sim 0}, X^{\sim 1} \in \alpha \\ \log Pur(X^{\sim 0}) & \exists X^{\sim 0} \in \alpha \\ 0 & \text{otherwise} \end{cases} \tag{6.15}$$

Note that for lexical rules the feature equals 0.

As in the discussion of [Zhou & Zhu$^+$ 08] as future work suggested, the purity feature could be computed for the target language, too. In this work, we follow this suggestion and differentiate between both approaches using the *source purity* $h_{\text{srcPurity}}(r)$ and *target purity* $h_{\text{tgtPurity}}(r)$ in the following. Furthermore, both features can be combined either by being added as separate feature functions to the log-linear model combination or by being linearly interpolated:

$$h_{purity}(r) = \frac{h_{\text{srcPurity}}(r) + h_{\text{tgtPurity}}(r)}{2}. \tag{6.16}$$

The combined feature $h_{purity}(r)$ is added as additional feature to the log-linear model combination.

6.5 Contributions

The author of this thesis implemented the Good-Turing smoothing approach into RWTH's open source translation toolkit *Jane* and supervised the implementation of the purity feature in the scope of [Lu 14]. Both, the count frequency indicator feature and enhanced low-frequency feature were already part of *Jane*. The word class translation models for hierarchical translation were the authors' contribution to [Wuebker & Peitz$^+$ 13b].

6.6 Related Work

Smoothing is an important method for statistical natural language processing and widely applied in language modeling [Kneser & Ney 95, Chen & Goodman 98]. However, the research on smoothing of translation models in statistical machine translation is relatively rare.

A systematic study of different smoothing methods for phrase-based translation, e.g. Good-Turing smoothing, is provided in [Foster & Kuhn$^+$ 06]. These methods could also be applied as standard smoothing techniques for hierarchical translation. In this chapter, we have presented some of the most effective ones.

The comparison presented in [Chen & Kuhn$^+$ 11] can be seen as an extension to the work described above introducing new features, e.g. the enhanced low-frequency feature. However, a comparison of smoothing methods in a hierarchical translation framework is still missing.

A smoothing feature designed for hierarchical translation was firstly presented in [Zhou & Zhu$^+$ 08] by using syntactic information from parse trees of the source language. On a small English→Chinese translation task from the travel domain, the authors reported an improvement of 0.7 BLEU over a baseline which includes lexical smoothing but no other additional smoothing method. In this work, we adopt this approach and further compute the feature using syntactic information from the target language. In addition, we present two ways of combining both feature functions and later compare them on an already smoothed baseline in Section 7.2.3.

Experimental results in [Durrani & Haddow$^+$ 13] suggest to use Good-Turing smoothing combined with a simple count-based feature for phrase-based translation. We follow this suggestion and compare Good-Turing discounting in combination with different features.

Another line of research is smoothing with word-based lexicon models. An extensive comparison of different approaches for hierarchical phrase-based translation was done in [Huck & Mansour$^+$ 11].

7. Experimental Evaluation

In this chapter, we present the experimental evaluation of the previously introduced techniques and improvements. All experiments are carried out with RWTH Aachen University's open-source translation toolkit *Jane* [Vilar & Stein$^+$ 10a, Wuebker & Huck$^+$ 12, Freitag & Huck$^+$ 14] on different translation tasks. A comprehensive comparison of all methods on a large-scale translation task is provided in Section 7.4. This chapter will be concluded with a discussion of the experimental results in Section 7.5.

7.1 Hierarchical Phrase-Based Translation with Jane

In this section, we evaluate the improved implementation of the hierarchical translation decoder described in Section 4.3.4. Since the release of *Jane* version 2.3, these enhancements are part of the translation toolkit. Table 7.1 shows the results of this comparison on the WMT 2011 German→English task. Note that for this task an estimated 5-gram language model and hierarchical translation model were provided in scope of the comprehensive comparison of different hierarchical translation engine done in [Heafield 13]. `newstest2011` was used as test set. For this comparison, both *Jane* versions apply observation histogram pruning including the language model score to generate the same translations.

Table 7.1: Speed and memory comparison of different hierarchical phrase-based decoders on the WMT 2011 German→English translation task.

	`newstest2011`	
decoder	**speed [words/second]**	**memory usage [gigabyte]**
Jane 2.2	7.0	14.2
Jane 2.3	8.9	**10.3**
Moses 3.0	**11.1**	16.7

The improved implementation seems to have an impact on both translation speed and memory consumption. The memory usage was reduced by up to 4 gigabyte and the translation speed was increased by up to 1.9 words per second. Such improvements are also important for techniques such as the discriminative training (cf. Section 3.7) where hierarchical decoding of larger training corpora is needed. Discriminative training using the improved implementation was performed for RWTH Aachen University's official submission to the BOLT 2014 Chinese→English evaluation campaign (cf. Section 7.4.4). The comparison with the current version of *Moses* [Hoang 15] confirms the results of [Heafield 13]: While *Jane* consumes less memory, *Moses* is faster.

Table 7.2: Comparison of different count-based smoothing techniques on the WMT 2014 German→English task. The baseline includes lexical smoothing only. The enhanced low-frequency introduces less additional feature functions into the log-linear model combination compared with the frequency indicator. The Good-Turing smoothed baseline has the same amount of log-linear models since the translation probabilities are replaced.

	# log-linear models	newstest2012		newstest2013	
		BLEU [%]	TER [%]	BLEU [%]	TER [%]
baseline	10	23.4	60.3	26.1	56.9
+ frequency indicator	13	24.1	59.2	26.5	56.0
+ enhanced low-frequency	11	24.4	59.0	26.5	56.0
baseline with Good-Turing	10	24.1	59.3	26.6	56.0
+ frequency indicator	13	24.3	59.1	26.7	55.9
+ enhanced low-frequency	11	24.1	59.3	26.7	55.9

7.2 Smoothing of Hierarchical Translation Models

In this section, the different smoothing techniques presented in Chapter 6 are evaluated. Each type of smoothing is individually compared on different translation tasks. An overall comparison of all methods exclusively performed for this thesis on a state-of-the-art translation task is provided in Section 7.4.

7.2.1 Count-Based Smoothing

Setup

The experiments were conducted on the WMT 2014 German→English translation task (Section A.1.2). The translation model was trained on all available bilingual training data and the baseline includes the standard models as described in Section 4.4.1. The 4-gram language model was trained on the respective target side of the bilingual data, $\frac{1}{2}$ of the Shuffled News Crawl corpus, $\frac{1}{4}$ of the 10^9 French-English corpus and $\frac{1}{2}$ of the LDC Gigaword Fifth Edition corpus. The monolingual data selection is based on cross-entropy difference as described in [Moore & Lewis 10]. For the baseline language model, we trained separate models for each corpus, which were then interpolated. The parameters of the feature functions were tuned on `newstest2012` with MERT using BLEU as optimization criterion. More details about the setup and its preprocessing can be found in [Peitz & Wuebker$^+$ 14].

Results

The results of this experiments are shown in Table 7.2. On the blind test set (`newstest2013`), both the frequency indicator feature and the enhanced low-frequency feature leads to the same improvements over the standard baseline as well as the baseline using Good-Turing estimated translation probabilities. It seems there is no clear advantage of one feature over the other. However, smoothing with Good-Turing provides the better baseline and can be slightly improved by 0.1 in BLEU and TER with the frequency indicator feature or the enhanced low-frequency feature.

Table 7.3: Impact of word class translation and word class language model on the IWSLT 2012 German→English task. The baseline is smoothed with the frequency indicator approach.

	dev2010		tst2010	
	BLEU [%]	TER [%]	BLEU [%]	TER [%]
baseline (+ frequency indicator)	29.6	50.3	27.9	52.5
+ word class translation model	29.8	50.3	28.1	52.3
+ word class language model	30.0	49.8	28.2	51.7

7.2.2 Class-Based Smoothing

Setup

To evaluate the class-based smoothing models introduced in Section 6.3, we performed experiments on the IWSLT 2012 German→English task. Data statistics and descriptions are given in Sections A.3.1. The translation models were trained on the in-domain portion of the training data. The 4-gram baseline language model is trained on all data. The number of word class is 100 and the word class language model has a 7-gram context size. More details can be found in [Wuebker & Peitz+ 13b].

Results

The improvements over the already smoothed baseline are small but consistent (Table 7.3). While the word class translation models improves over the baseline by 0.2 in BLEU and TER on the blind test set `tst2010`, applying a word class language model on tops leads to an improvement of 0.1 in BLEU and 0.7 in TER.

7.2.3 Syntax-Based Smoothing

Setup

The syntax-based smoothing approach for hierarchical translation models (Section 6.4) was evaluated on the IWSLT 2013 German→English task. Data statistics and descriptions are given in Sections A.3.2. For rapid experiments, the translation model was trained on the in-domain portion of the training data which are parsed with the Stanford parser [Klein & Manning 03, Rafferty & Manning 08]. The language model is a large 4-gram LM with modified Kneser-Ney smoothing [Kneser & Ney 95, Chen & Goodman 98], trained with the SRILM toolkit [Stolcke 02]. The complete News Commentary, Europarl v7 and Common Crawl corpora as well as selected parts of the Shuffled News and LDC English Gigaword corpora are used as additional monolingual data. The data selection is based on cross-entropy difference [Moore & Lewis 10]. In total, the LM is trained on 1.7 billion running words. The baseline was augmented with the frequency indicator feature function.

Results

The results of this experiments are shown in Table 7.4. The prior features were applied on an already smoothed baseline. On `tst2010`, improvements of 0.9 in BLEU were achieved with

Table 7.4: Impact of syntax-based smoothing on the IWSLT 2013 German→English task. The baseline is smoothed with the frequency indicator approach.

	dev2010		tst2010		tst2011	
	BLEU [%]	TER [%]	BLEU [%]	TER [%]	BLEU [%]	TER [%]
baseline	31.3	48.5	29.7	50.3	33.8	45.9
+ frequency indicator	31.5	48.2	29.9	50.0	33.9	45.6
+ source purity	32.3	47.4	30.8	49.5	34.6	45.1
+ target purity	32.0	47.2	30.8	48.9	35.3	44.1
+ source and target purity	32.0	47.7	30.5	49.5	34.3	45.4
+ purity linear interpolation	32.2	47.4	30.8	49.1	35.0	44.6

both source and target prior. However, adding both features to the log-linear model combination (source and target prior) leads to mixed results.

7.3 Generative Training of Hierarchical Translation Models

In this section, the framework for generative training of hierarchical translation models presented in Chapter 6 is evaluated. Each approach of re-estimating of the translation model is individually compared on different translation tasks. A comprehensive comparison of all methods exclusively performed for this thesis on a large-scale translation task is provided in Section 7.4.

7.3.1 Inside-Outside Algorithm

Setup

Our experiments were carried out on the WMT 2012 German→English and French→English tasks. For both tasks, we selected parallel sentences according to two criteria: Only sentences of maximum 100 tokens are considered and the ratio of the vocabulary size of a sentence and the number of its tokens is minimum 80% i.e. we remove sentences that have too many repeated words. Data statistics are given in Section A.1.1. Given the training data, a word alignment was learned with GIZA++. The resulting alignment was used to extract the initial rule set. The initial rule set is extracted heuristically and named *heuristic rule set* in the following. The generatively learned rule set is called *learned rule set*. Note that the heuristic rule set is used to initialize the forced decoding procedure. As baseline system a standard hierarchical phrase-based system with the standard set of models was employed.

We applied length-based leave-one-out as described in Section 5.2.3 and compared to a setup using a rule table learned without leave-one-out. For all experiments, we used a 4-gram language model with modified Kneser-Ney smoothing which was trained with the SRILM toolkit [Stolcke 02]. The scaling factors of the features were optimized for BLEU on the development set with MERT on 100-best lists. The performance of the different setup was evaluated on the development (`newstest2010`) and the test set (`newstest2011`).

After the forced decoding step, the expected counts for re-estimating the translation probabilities were obtained with the inside-outside algorithm as introduced in Section 5.3.1.

Table 7.5: Preliminary experiments on the development and test set of the WMT 2012 German→English task. The cutoff threshold was selected based on the BLEU score of the development set.

	cutoff threshold	newstest2010 BLEU [%]	newstest2010 TER [%]	newstest2011 BLEU [%]	newstest2011 TER [%]	**% of full rule set**
baseline	-	20.8	62.5	19.1	63.4	100
no leave-one-out	0	20.3	63.4	18.5	64.6	92.0
length-based leave-one-out	0	21.0	61.8	19.7	63.1	92.0
	0.0001	21.0	62.0	19.4	63.3	32.2
	0.001	21.1	61.8	19.3	63.3	23.4
	0.01	21.2	62.0	19.3	63.1	13.2
	0.1	21.4	61.7	19.5	63.1	4.9
	0.15	21.4	62.0	19.6	63.0	3.9
	0.2	21.0	62.7	19.2	62.5	3.2

Results

As preliminary experiments on the German→English task, we compared forced decoding with and without leave-one-out (Table 7.5). The length-based leave-one-out method outperforms forced decoding without leave-one-out in terms of BLEU and is also slightly better than the baseline. Furthermore, we pruned the final learned rule set by dropping all rules which got a summed up expected count lower than a given threshold. Discarding such rules seems to improve the translation quality and in addition reduces the size of the rule set.

During the forced decoding procedure, around 93% of the sentences of the German→English corpus and around 97% of French→English corpus were parsed with the two-parse algorithm. The non-parsable sentences were skipped. In general, those are longer sentences which are misaligned usually caused by liberal or wrong translation. Thus, even the full learned rule set does not contain all rules of the initial rule set.

Finally, we tested the best setup (length-based leave-one-out with cutoff threshold of 0.1) on the test translation set and achieved an improvement of 0.4 points in BLEU and 0.3 points in TER for German→English task and an improvement of 0.4 points in BLEU and 0.3 points in TER for French→English task (Table 7.6). The results of the experiment using the heuristic rule set filtered to contain the same rules as the pruned learned rule set (baseline (filtered)) are similar to the setup using the translation probabilities learned with the EM-inspired algorithm. This observation shows that using filtered rules performs as least as good as using the full rule set. However, it also means that reestimating the translation probabilities does not seem to bring further improvements in terms of BLEU.

We achieved further improvement applying a log-linear interpolation of the learned rule set with the heuristic one as described in Section 5.3.3. The interpolation weight was adjusted on the development set and set to $\omega = 0.2$. Note that the resulting translation model is an intersection of both. Our final result shows an improvement of 0.7 BLEU points and 0.8 TER points over the baseline on the German→English task and an improvement of 1.0 points in BLEU and 0.9 points in TER on the French→English task.

Table 7.6: Final results for the WMT 2012 German→English and French→English tasks.

	newstest2011			
	German→English		French→English	
	BLEU [%]	TER [%]	BLEU [%]	TER [%]
baseline	19.1	63.4	24.6	57.2
baseline (filtered)	19.5	63.3	-	-
forced decoding (inside-outside)	19.5	63.1	25.0	57.2
log-linear interpolation ($\omega = 0.2$)	19.8	62.6	25.6	56.3

7.3.2 k-Best Derivations

Setup

The experiments were carried out on the IWSLT 2013 German→English translation task. Data statistics and descriptions are given in Sections A.3.2. The baseline system was trained on all available bilingual data and used a 4-gram LM with modified Kneser-Ney smoothing [Kneser & Ney 95, Chen & Goodman 98], trained with the SRILM toolkit [Stolcke 02]. As additional data sources for the LM we selected parts of the Shuffled News and LDC English Gigaword corpora based on cross-entropy difference [Moore & Lewis 10]. This makes a total of 1.7 billion running words for LM training. Tuning of the log-linear parameter weights was done with MERT on a provided development set. As optimization criterion we used BLEU.

Forced decoding was performed on the TED talks portion of the training data (∼140K sentences). In both tasks, around 5% of the sentences could not be parsed. In this work, we just skipped those sentences.

Results

Figure 7.1 shows the performance of setups using translation models with re-estimated translation probabilities. The setups vary in the k-best derivation size extracted in the forced decoding step and in the application of leave-one-out. Based on the performance on the development set, we selected the setup with $k = 500$ using leave-one-out. Table 7.7 shows the final results for the German→English task. In order to provide a comparable system, we augmented the baseline with an in-domain translation model as both generative and discriminative training were performed on the in-domain data only. This results in an improvement of up to 0.6 points in BLEU and 0.9 points in TER. The generative training approach improves over the augmented baseline by up to 0.7 in BLEU and 0.4 in TER and is slightly better than the discriminative training scheme.

7.4 Comprehensive Comparison

In this section, we present a comparison of several methods introduced in the previous chapters. While in the previous sections of this chapters most methods were evaluated separately on different translation tasks, the following experiments, which were exclusively performed for this thesis, were conducted on a large-scale translation task in order to compare all introduced methods. After a description of the translation task, we compare all smoothing techniques in Section 7.4.2 and the different translation model training schemes in Section 7.4.3. In addition, we pick the best setup obtained in this work, augment it with some additional models and compare our final system with

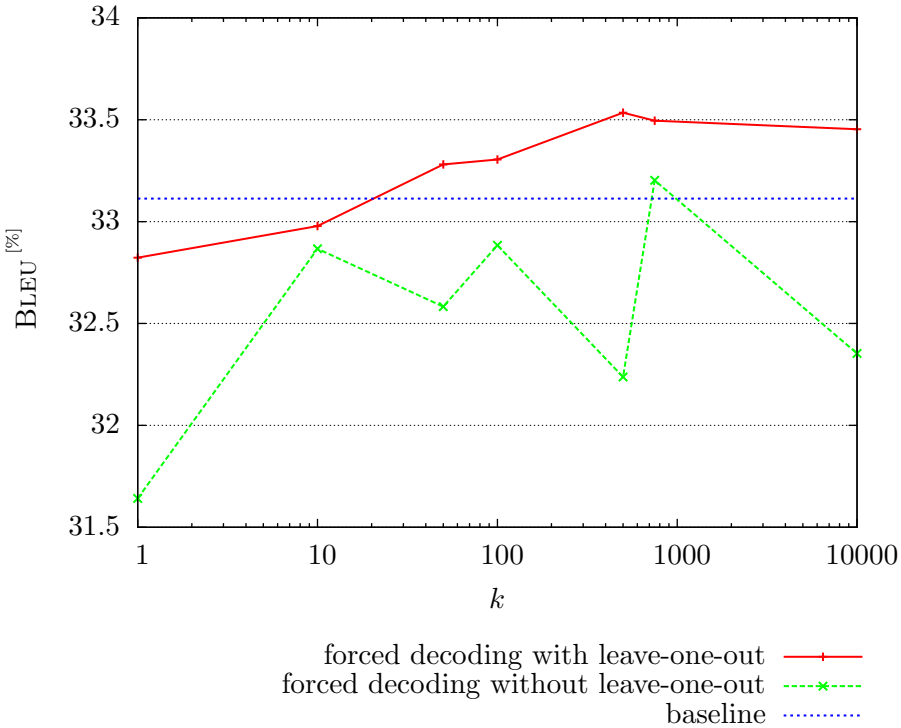


Figure 7.1: BLEU scores on the IWSLT German→English task of setups using translation models trained with different k-best derivation sizes. Results are reported on `dev2010` with and without leave-one-out.

RWTH Aachen University's final submission (including discriminative training) and submission by other teams (e.g. IBM and Stanford University) for this task (Section 7.4.4).

7.4.1 Setup

The experiments were conducted on the BOLT 2014 Chinese→English translation task (discussion forum). The 5-gram language model was trained on ∼2.9 billion running words. The evaluation measure is $\frac{\text{TER}-\text{BLEU}}{2}$, denoted as (T-B), where smaller numbers are better. Data statistics and descriptions are given in Sections A.2.1. We used `DEV12-tune` as a development set to tune the parameters of the log-linear model combination with MERT and some hyper-parameters manually (e.g. an interpolation weight). The optimization criterion was TER-BLEU. The different systems are then compared on two blind test sets, namely `DEV12-dev` and `P1R6-dev`. For the syntax-based smoothing approach, we parsed the Chinese source data of the bilingual training corpus with the Berkeley parser[1] and the corresponding English target data with the Stanford parser[2]. For word-based lexical smoothing we took the word probabilities from the IBM-1 model as suggested in [Huck & Mansour+ 11]. The authors investigated different lexicon models for hierarchical phrase-based machine translation and reported improvements of up to 1.5 points in BLEU compared with a setup using word probabilities from lexicon models estimated from word-aligned training data.

[1] `https://github.com/slavpetrov/berkeleyparser`
[2] `http://nlp.stanford.edu/software/lex-parser.shtml`

Table 7.7: Results for the IWSLT 2013 German→English task comparing our generative training approach (forced decoding, k-best) with discriminative training. For a fair comparison, we augmented the baseline with an in-domain translation model as both generative and discriminative training were perform on the in-domain data.

	tst2010		tst2011	
	BLEU [%]	TER [%]	BLEU [%]	TER [%]
baseline	30.5	49.7	35.7	44.1
+ in-domain translation model	31.1	48.8	35.9	43.8
+ discriminative training	31.8	48.4	36.4	43.2
+ forced decoding (k-best)	31.8	48.3	36.6	43.0

Table 7.8: Extensive comparison of several smoothing methods on the BOLT 2014 Chinese→English translation task. The baseline includes lexical smoothing only.

	DEV12-dev (T-B) [%]	P1R6-dev (T-B) [%]
baseline	24.2	24.7
+ frequency indicator	23.9	24.3
+ enhanced low-frequency	23.7	24.3
+ word class language model	23.7	24.2
+ word class translation model	23.7	24.1
+ source purity	23.8	24.2
+ target purity	23.7	24.0
+ source and target purity	24.0	24.4
+ purity linear combination	23.8	24.4
baseline with Good-Turing	23.8	24.3
+ frequency indicator	23.9	24.2
+ enhanced low-frequency	23.8	24.1

7.4.2 Smoothing Results

The result of the comparison of different smoothing approaches is shown in Table 7.8. The count-based features (frequency indicator, enhanced low-frequency and Good-Turing) seems to improve the translation quality in all cases. Comparing the frequency indicator feature with the enhanced low-frequency feature, the latter one performs slightly better. Smoothing with Good-Turing also improves over the baseline. However, adding additional count-based feature functions on top of the Good-Turing smoothed baseline does not seem to help much. Furthermore, the impact of the class-based and syntax-based smoothing approaches is small on this translation task compared to previous results (cf. Table 7.3 and Table 7.4). In these experiments, applying the target purity feature works best. The combination of both purity feature functions leads again to mixed results. This confirms the results from previous experiments (cf. Table 7.4).

Table 7.9: Preliminary experiments on the development set of the BOLT 2014 Chinese→English task. Translation model training (k-best and inside-outside) was performed on all training data. The baseline includes the enhanced low-frequency feature.

	k-best size	DEV12-tune (T-B)[%]	**% of full rule set**	**speed [words/second]**
forced decoding (k-best)	1	19.3	2.5	10.0
	100	**18.8**	6.8	10.0
	200	19.1	7.7	9.5
	500	18.9	9.1	9.5
	1000	19.0	10.3	9.3
	2000	18.9	11.4	9.3
	10000	19.4	14.1	8.7
forced decoding (inside-outside)		19.1	50.7	7.5
baseline		19.2	100.0	5.4

Table 7.10: Comparison of both translation model training methods (inside-outside versus k-best) on the BOLT 2014 Chinese→English translation task. Translation model training was performed on all training data. The baseline includes the enhanced low-frequency feature.

	DEV12-tune (T-B)[%]	DEV12-dev (T-B)[%]	P1R6-dev (T-B)[%]
baseline	19.2	23.7	24.3
forced decoding (inside-outside)	19.1	23.6	24.4
log-linear interpolation ($\omega = 0.8$)	18.8	23.3	24.1
forced decoding (100-best)	18.8	23.7	24.3
log-linear interpolation ($\omega = 0.5$)	**18.7**	**23.4**	**24.1**

7.4.3 Hierarchical Translation Model Training Results

When running forced decoding on all training data, 76% of the training sentences could be parsed. To determine the optimal k for the k-best parsing algorithm, we compare in Table 7.9 several k-best derivation sizes on the development set. We further compare these results with the translation model training method using the inside-outside algorithm. The optimal k-best size was set to 100. This results in a translation model size reduction of around 93%. With a smaller translation model, the translation speed could be increased from 4.1 words per second to 10 words per second. We further obtained a small improvement in terms of $\frac{\text{TER}-\text{BLEU}}{2}$.

Table 7.10 shows the results on two blind test sets. It seems that the improvement as observed on the development set could not be carried over to the test sets. However, by log-linear interpolation with the heuristically extracted translation model (cf. Section 5.3.3), a slight improvement of up to 0.4 points in $\frac{\text{TER}-\text{BLEU}}{2}$ was observed. The interpolation weights were adjusted on the development set. In particular, the interpolation weight ω was set to 0.8 for the setup using the inside-outside algorithm and to 0.5 for the k-best parsing algorithm.

Table 7.11: Final translation results on BOLT Chinese→English Discussion Forum task. The baseline includes lexical smoothing only. For RWTH Aachen University's final submission, the baseline was augmented with the enhanced low-frequency feature, the word class language model and the hierarchical reordering model. It further included the discriminative training approach, a recurrent neural network language model (RNNLM) and a recurrent neural network translation model (RNNTM).

	DEV12-dev (T-B)[%]	P1R6-dev (T-B)[%]
baseline	24.2	24.7
+ enhanced low-frequency	23.7	24.3
+ word class language model	23.7	24.2
+ hierarchical reordering model	23.4	23.7
+ forced decoding (100-best, interpolation $\omega = 0.5$)	22.8	23.3
+ discriminative training[3]	22.5	22.7
+ RNNLM + RNNTM[4]	21.8	21.8
University of Maryland	23.2	22.9
Universit du Maine, Le Mans	22.4	22.6
Stanford University	22.5	23.0
IBM T.J. Watson Research Center	20.9	21.1

7.4.4 Final Comparison

For the final comparison, we augmented the baseline with the enhanced low-frequency feature, the word class language model and the hierarchical reordering model as it was done for RWTH Aachen University's official submission. The corresponding setups were provided by Joern Wuebker in order to compare our work with the discriminative training. The results are depicted in Table 7.11. The word class language model seems to have only a small impact on the translation quality. The $\frac{\text{TER}-\text{BLEU}}{2}$ score was slightly increased by 0.1 on P1R6-dev. However, with the hierarchical reordering model, an improvement of up to 0.5 points can be observed. Applying our generative training approach, we further see an improvement of up to 0.6 points in $\frac{\text{TER}-\text{BLEU}}{2}$. The discriminative training however outperforms our proposed method by up to 0.6 points. For the final submission, recurrent neural network language [Sundermeyer & Ney$^+$ 15] and translation models [Sundermeyer & Alkhouli$^+$ 14] were added which increased the translation quality by up 0.9 points in $\frac{\text{TER}-\text{BLEU}}{2}$ on top of the discriminative training.

7.5 Conclusion

In this chapter, we experimentally evaluated the methods and improved implementation introduced in the Chapters 4, 5 and 6. First, each method was tested separately. We later compared all smoothing approaches and the generative training procedure on a large-scale Chinese→English translation task.

In Section 7.1, we showed that our improved implementation of the cube pruning algorithm results in a speed up of 1.9 words per second while the memory consumption was reduced by 4

[3] Provided by Joern Wuebker
[4] Official RWTH Aachen University submission

gigabyte. A comparison with *Moses* further showed that our implementation is competitive with other state-of-the-art implementations. We provided a fast and efficient decoder which is needed to perform discriminative training as it was done for RWTH Aachen University's final submission in the BOLT 2014 Chinese→English evaluation campaign (cf. Section 7.4.4).

In Section 7.2.1, we evaluated different count-based smoothing approaches. It seems that there is no huge difference in applying the frequency indicator, the enhanced low-frequency feature or Good-Turing smoothing. Even the combination of different approaches resulted in slight improvements only. However, experiments presented in Section 7.4.2 indicated that the enhanced low-frequency feature can lead to better results compared with the frequency indicator and the Good-Turing smoothing approach. Another advantage of enhanced low-frequency feature is that it is easy to compute and introduces only one additional weight in the log-linear model combination. During the course of this thesis, the enhanced low-frequency feature became a standard model of the RWTH Aachen University's hierarchical phrase-based baseline and replaced the frequency indicator.

In Section 7.2.2, we presented evaluation results of the class-based translation and language models. The improvements were small but consistent. Similar results were observed in the comprehensive comparisons in Section 7.4.2. Since the word class language model is easy to compute and to plug-in into the log-linear model combination, it became one of the standard model in RWTH Aachen University's hierarchical phrase-based baseline during the course of this thesis.

7.6 Contributions

The comparisons in Section 7.1 were exclusively performed by the author of this work. The translation model, the language model and the test data for the WMT German→English translation task were provided by Kenneth Heafield. Most experiments in Section 7.2 were previously conducted during the course of this thesis. The results shown in Section 7.2.3 were part of Yifeng Lu's master thesis [Lu 14] and the results in Section 7.2.2 were already published in [Wuebker & Peitz$^+$ 13b]. The experiments presented in Section 7.3 were solely performed by the author of this work and were already published in [Peitz & Mauser$^+$ 12, Peitz & Vilar$^+$ 14]. Finally, all experiments in Section 7.4 were exclusively conducted for this thesis. The heuristically extracted translation model and the language models were provided by Joern Wuebker who was also in charge of RWTH Aachen University's official submission.

8. Improving Spoken Language Translation

Spoken language translation (SLT) has become an important application of automatic speech recognition (ASR) and machine translation (MT). The challenge of SLT is to translate automatically transcribed speech rather than written text into another language. In recent years, large research projects have been focussed on speech translation such as the European EU-Bridge project[1], where the focus was set on speech-to-text translation of speeches in the European parliament, and the DARPA-funded GALE project, where TV-shows and broadcast news were translated from Arabic and Chinese to English for intelligence purposes. Nowadays the application of speech translation goes beyond pure research and is employed in products such as the *Skype Translator*[2] by Microsoft.

In most cases the translation of spoken language is divided in two independent parts: First, an ASR system recognizes speech input. Next, the recognized words are translated by an MT system. The interface between both systems can be designed in different ways. The simplest approach is to take the first-best sentence recognized by the ASR system. Since the first-best sentence may contain recognition errors which influence the translation quality, another approach is to consider recognition alternatives which might be correct but were not preferred by the ASR system. A step into this direction is to make use of the first n-best sentences. However, this approach seems to be highly inefficient as each alternative of each sentence has to be translated. A more practical approach takes word lattices as input which represent many recognition alternatives in a compact way. Furthermore, such lattices are naturally provided by an ASR system and are enriched with additional features. Hence, by giving the MT system access to a lattice and including ASR features into its log-linear model combination, the MT decoder is able to find the best fitting sentence representation for the actual translation.

However, neither first-best output nor lattices contain punctuation marks because they are not generated by almost all state-of-the-art ASR systems. The main reason is that in regular spoken text punctuation marks are not made explicitly. Humans can often infer punctuation by prosodic, syntactic or semantic cues, but the tasks is difficult for more casual and conversational speech as grammatic rules are often only loosely observed. This makes the evaluation of speech recognition results with punctuation more ambiguous and therefore recognitions systems tend to be optimized for output without punctuation.

In contrast, most MT systems are trained on text data with proper punctuation marks and expect correctly punctuated texts for an acceptable level of quality in translation. Furthermore, the output of MT is expected to have correct punctuation, even when translating from speech. Therefore, punctuation prediction has to be done at some stage of the speech translation process.

In the following sections we describe and investigate different approaches of punctuation prediction and compare them with direct integration having first-best sentences or lattices as input.

[1] `http://www.eu-bridge.eu`
[2] `http://www.skype.com/en/translator-preview/`

First, we introduce the concept of lattice decoding in our phrase-based translation framework (Section 8.1). In Section 8.3 we present the approach to apply monolingual machine translation systems for punctuation prediction. We further propose to use a hierarchical translation system rather than a phrase-based one (Section 8.3.2). Finally, all introduced schemes of coupling ASR and MT are evaluated and compared on a large-scale spoken language translation task in Section 8.4. The chapter is concluded in Section 8.5. Section 8.6 is dedicated to highlight the contributions of this work in spoken language translation. Related work is described in Section 8.7.

8.1 Lattice Decoding for Spoken Language Translation

In this section the concept of lattice decoding in the scope of spoken language translation is described. After a formal definition of lattices and confusion networks, the modifications of the phrase-based decoder are shortly depicted.

8.1.1 Lattices and Confusion Networks

Lattices (or *word graphs*) are directed graphs with a set of edges E and some nodes (or vertices) V. In a lattice $L = (V, E)$ exactly one start and one end node exist and each edge e is labelled with a word w.

A lattice as an output of an ASR system represents the search space where each path through the lattice is a recognition hypothesis. Following [Hoffmeister 11], each edge e of a given lattice $L = (V, E)$ is labelled with a *posterior probability* $p(e|x_1^T)$ which is defined as the sum of the probabilities of all paths a_1^J going through the edge e:

$$p(e|x_1^T) := \sum_{a_1^J \in L, \exists l: a_l = e} p(a_1^J|x_1^T) \tag{8.1}$$

where x_1^T are the acoustic features used for speech recognition. We will later use this probability as additional feature in the log-linear model combination of the phrase-based translation system.

Another approach to access the search space of an ASR system is to collapse the lattice into a *confusion network* [Bertoldi & Zens$^+$ 07]. A confusion network is a linear lattice where each path from the start to the end visits all nodes. Thus, confusion networks have a simpler topology. However, during the process of collapsing, new paths are introduced which are not present in the original lattice. Given a lattice $L = (V, E)$ with a function $l(e)$ assigning word labels w to each edge $e \in E$, a corresponding confusion network can be described by a function $\delta : E \to \mathbb{N}$ which assigns a *slot* to each edge of the lattice. For two edges $e_1, e_2 \in E$ that are on the same path with e_1 preceding e_2, it holds that $\delta(e_1) < \delta(e_2)$. Thus, the posterior probability for confusion networks has to be redefined as an *slot-wise posterior probability* $p_s(w|x_1^T)$ for a given slot $s \in \mathbb{N}$:

$$p_s(w|x_1^T) := \sum_{\substack{e \in E, \\ l(e)=w, \\ \delta(e)=s}} p(e|x_1^T). \tag{8.2}$$

Constructing a confusion network given a lattice can be done by using the encoded *timing information*. Each node in the lattice includes the time at which they occur in the speech. Using this information, the nodes are clustered. A detail description of this algorithm is given in [Hoffmeister 11]. As an example, the lattice in Figure 8.1 is converted into the confusion network shown in Figure 8.2. To allow for hypotheses of different lengths, the confusions network needs to contain ϵ-edges to skip words.

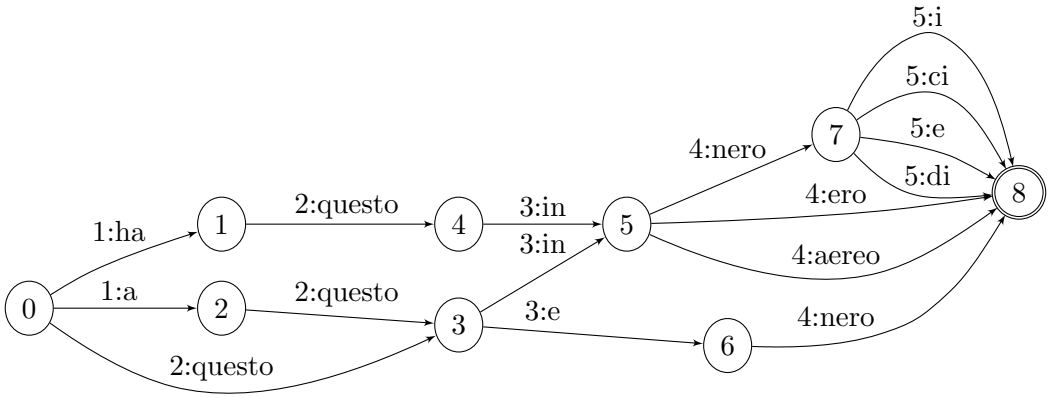


Figure 8.1: ASR lattice that has been labelled with the slots of the collapsed confusion network in Figure 8.2. This example is taken from [Matusov & Ney 11].

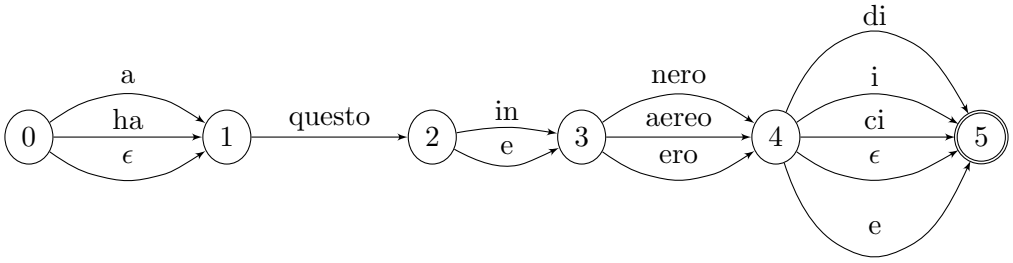


Figure 8.2: Confusion network constructed from the lattice in Figure 8.1. This example is taken from [Matusov & Ney 11].

[Matusov & Hoffmeister$^+$ 08] further proposed to integrate the information from the collapsed confusion network back to the original lattice. The edges of the lattice are labelled by the slot assigned to them in the confusion network and the slot-wise posterior probability. As an example, the edges of the lattice shown in Figure 8.1 are labelled with the slots of the confusion network in Figure 8.2. This representation allows for an efficient decoding of lattices. More details will be given in the following section.

8.1.2 Lattice Decoding

In order to be able to translate word lattices produced by an ASR system rather than single sentences, the concept of phrase-based translation as introduced in Section 3.5 has to be modified. Thus, the decision rule of Equation 3.9 is changed with respect to lattices as input. The key change is that the maximization includes all possible *source hypotheses* contained in the lattice as well. For a given lattice L, let $\mathcal{F}(L)$ be the set of source sentences encoded in the lattice. The decision rule is rewritten as following:

$$\mathcal{F}(L) \mapsto \hat{e}_1^{\hat{I}}(\mathcal{F}(L)) = \underset{I,e_1^I}{\operatorname{argmax}} \left\{ \max_{f'^J_1 \in \mathcal{F}(L)} \left\{ \sum_{m=1}^{M} \lambda_m h_m(e_1^I, f'^J_1) \right\} \right\}. \tag{8.3}$$

Hence, an additional task of the search algorithm is to find the source path f'^J_1 through the lattice that allows for the translation e_1^I with the highest score.

The most sentence-specific concept in the SCSS algorithm is that of the *coverage set*. The coverage set associated with a partial hypothesis stores the *positions* of the source sentence which

have already been translated. Since we want to allow lattice input, the definition of positions and the coverage set needs to be changed.

There are two possible solutions leading to two slightly different decoders. The first solution is to take the set of all nodes in the lattice as the full coverage as described in [Dyer & Muresan$^+$ 08]. This means that we keep track of which nodes have already been translated, i.e. the positions in the search are the nodes in the lattice. With this approach the maximum cardinality in the search is the number of nodes in the lattice.

The second solutions is based on *slots* and requires the edges of the lattice to be labelled as described in the previous section. The advantage of this method is that there are usually fewer slots than nodes, so the run-time of the algorithm is reduced.

In this work, we follow the second approach. Preliminary experiments done in [Dahlmann 15] have shown a lower run-time using the slot-based method while the translation quality is on the same level. Another advantage of the slot-based search is that the slots serve as the positions for reordering models. Hence, both the distortion model (cf. 3.5.1) as well as the hierarchical reordering model (cf. 3.5.3) can be easily adapted for the lattice decoder.

8.2 Punctuation Prediction Strategies

In machine translation an accurate punctuation of the input is crucial as prediction errors affect the translation quality. In [Peitz & Freitag$^+$ 11] a loss of up to 4 BLEU points was obtained if punctuation marks need to be predicted. It was further observed that around 12% of the running words are punctuation marks. However, most ASR systems do not provide an output with punctuation as it is not made explicit in speech. Therefore, punctuation prediction has to be done at some stage in the speech translation pipeline.

There are three stages at which punctuation can be predicted: before, during and after translation [Matusov & Mauser$^+$ 06]. Each of the stages requires a different translation system.

- When the punctuation is predicted before translation, a regular text translation system expecting correctly punctuated input can be used.
- To predict punctuation in the translation process, a translation system without punctuation marks in the source language but with punctuation marks in target language is needed.
- Punctuation prediction in the target language requires a translation system training without any punctuation.

The different approaches are visualized in Figure 8.3.

For all approaches, we assume that the segmentation of the speech recognition output is given and corresponds to at least sentence-like units. The level of annotation can vary from predicting only sentence-end punctuation marks such as full stops and questions marks, or a richer annotation that also contains commas and more challenging types of punctuation such as quotation marks. In this work, all kind of punctuation are considered.

In the following paragraphs, each prediction strategy and the consequences on the translation pipeline are described in more detail.

8.2.1 Prediction in the Source Language

Predicting punctuation in the source language means that the output from the speech recognition system that does not contain any punctuation marks is augmented with punctuation using

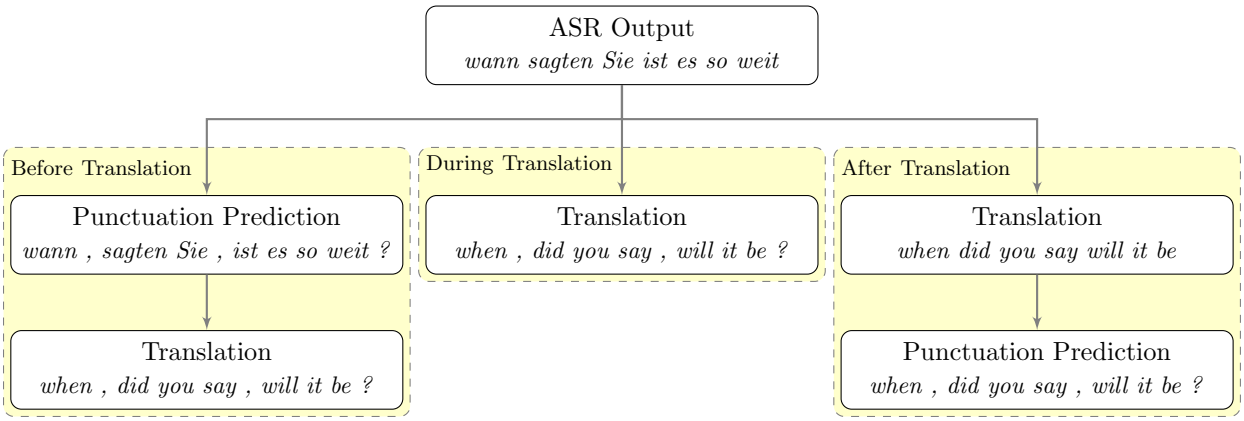


Figure 8.3: Three different stages where punctuation prediction can be done: before, during and after translation.

automatic methods. The main advantage of this methods is that no modification to the training data or the translation system are required and a standard text translation system can be used.

In order to provide a good input for the translation system and to get a good translation, the punctuation prediction has to be as accurate as possible. Errors in the predicted punctuation can affect the translation system output quality. The main reason for this is that longer phrase or rule matches in the translation system are prevented by incorrectly inserted punctuation marks. Therefore, for this approach, the accuracy of the prediction is crucial for the final translation system performance.

The prediction performance itself is influenced by the error rate of the speech recognition system. Recognition errors, that already by themselves are impairing the translation quality, may lead to additional punctuation prediction errors that further increase translation error rate.

Nevertheless, the approach of predicting punctuation marks in the source language remains attractive because of the above stated simplicity in retaining the actual translation system.

8.2.2 Implicit Prediction

Natural languages have a variety of different punctuation systems and rules. Typically, the punctuation in one language cannot always directly be taken over into another language. As translation systems learn from real data, they also implicitly learn to handle the different punctuation styles in source and target language.

The implicit punctuation prediction approach proposed by [Matusov & Mauser$^+$ 06] takes this idea and assume that the source language as produced by the speech recognition system does not contain any punctuation marks and the target language uses regular, full punctuation.

The training data for this machine translation system is preprocessed by removing all punctuation marks from the source language data, while the target language data is kept untouched. The removal is done after the word alignment step which is usually performed with GIZA++. The punctuation marks in the target sentence which are aligned with punctuation marks in the source sentences become non-aligned (e.g. Figure 8.4).

Applying the standard phrase extraction procedure [Och & Tillmann$^+$ 99], phrases with punctuation are not extracted. In order to add phrases with punctuation marks on the target side of a phrase to the translation model, a heuristic is needed which allows for phrase blocks including non-aligned words which are adjacent to phrase boundaries (Figure 8.4(d)).

Employing this heuristic, two different phrase pairs for the same source side are extracted:

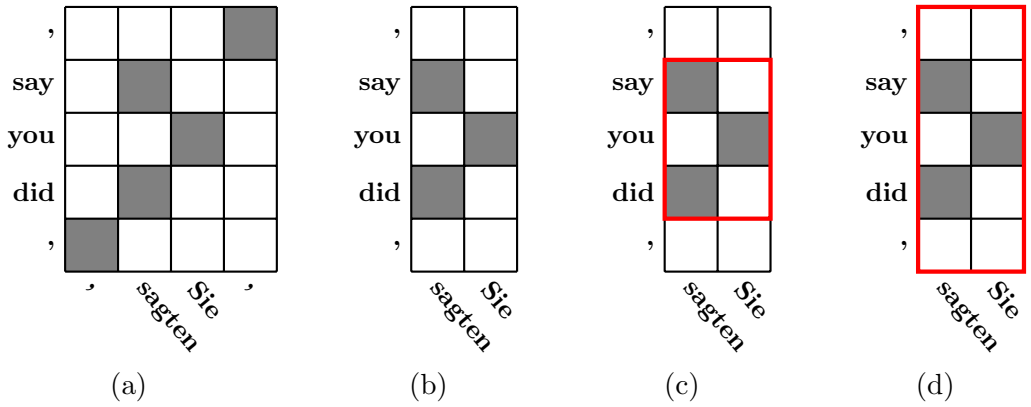


Figure 8.4: For the implicit prediction approach a given alignment (a) is modified by removing punctuation marks in the source sentence (b). (c) shows a standard phrase block. An additional heuristic allows for extended phrase blocks (d).

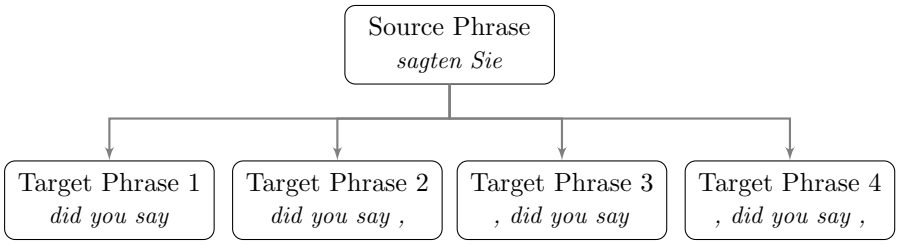


Figure 8.5: In the implicit translation model, given the modified alignment in Figure 8.4, the German source phrase "sagten Sie" has four possible translation options.

One containing the punctuation mark and one without punctuation. Given the modified alignment in Figure 8.4, the German source phrase "sagten Sie" has four possible translation options (Figure 8.5).

For implicit prediction, the same language model as for the regular translation system can be employed. However, the translation model needs to be changed by re-extracting phrase and word lexicon models. Another disadvantage of this method is that prediction and translation are not separate components in the speech translation pipeline. This makes systematic translation errors harder to track down and separate optimization and execution of the components impossible.

8.2.3 Prediction in the Target Language

The prediction in the target language is done after translation. The translation system needed for this method does not have any punctuation in source and target language. All punctuation marks are removed from the training data as well as from the development and test sets. After the translation process, the punctuation marks have to be inserted in the same way as when predicting punctuation marks in the source language.

This method has two major disadvantages. First, in addition to the translation model also the target language model used in translation has to be rebuilt. The second and more severe disadvantage for the final system performance is that the translation produces errors which make the punctuation prediction less accurate. This includes errors resulting from incorrect speech

recognition that are propagated through the translation system as well as errors introduced in the translation process itself. As translation error tends to be higher than speech recognition error, accurate prediction of punctuation marks in the target language is conceptually more error-prone than the other methods presented above.

8.3 Punctuation Prediction with Statistical Machine Translation

In this section we propose to perform punctuation prediction by applying a monolingual translation system. Instead of using a bilingual translation system, where source language and target language are different natural languages, the monolingual system translates from the a natural language without punctuation into the same natural language with proper punctuation.

After an introduction in Section 8.3.1 we explain the advantage of hierarchical phrase-based machine translation for predicting punctuation marks (Section 8.3.2). In Section 8.3.3 we introduce different optimization criteria for tuning monolingual translation systems.

The proposed methods will be extensively compared with a prediction method based on n-gram language models in Section 8.4.

8.3.1 Introduction

Inspired by [Hassan & Ma$^+$ 07], the motivation for this approach is that additional models of the translation system and the possibility to automatically tune the model weights of the system for good performance with respect to an evaluation measure help to predict punctuation correctly. In contrast to language model based prediction, such a system is able to use additional models e.g. the phrase translation probabilities.

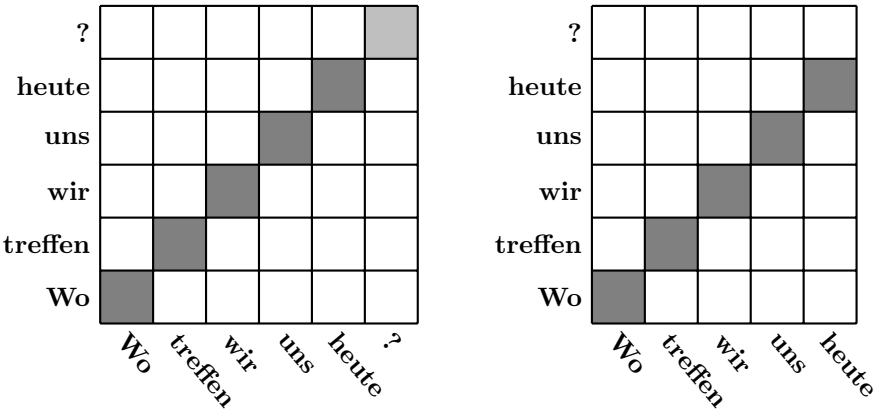
A translation system for punctuation prediction was trained monolingual using the source or target language part of training corpus depending on whether punctuation prediction is performed before or after the actual translation. In order to train the system, two versions of text is created: one without punctuation and one with punctuation. However, an alignment training with GIZA++ is not necessary in this case. The starting point is a training corpus with identical source and target language (with punctuation). Thus, we can assume a monotone alignment where the same words are mapped to each other. Similar to the implicit prediction approach, all punctuation marks from the source language are removed and the punctuation marks in the target sentence which are aligned with punctuation marks in the source sentences become non-aligned. Figure 8.6 shows an example for deleting the punctuation marks in the source sentence.

In order to add phrases such as

$$(\text{Wo treffen wir uns heute}, \text{Wo treffen wir uns heute ?})$$

to the translation model, the same heuristic as applied for the implicit prediction method is needed (Figure 8.7(a)).

The phrase-based translation system used for the punctuation prediction is similar as described in Section 3.5 with the standard set of models. However, due to the fact that monotone translation is performed, a reordering model is not needed. The optimization of the model weights is done with MERT. Usually Bleu is used as optimization criterion. In Section 8.3.3 we motivate a different criterion. The tuning set for the optimization is constructed by removing the punctuation marks. The original text with the punctuation left intact is used as reference.



(a) Monotone alignment

(b) Punctuation marks in the target sentence become unaligned

Figure 8.6: Example of monotone alignments. The second one is without punctuation ("?") in the source sentence. The corresponding punctuation marks in the target sentence become non-aligned.

8.3.2 Monolingual Hierarchical Phrase-Based Translation

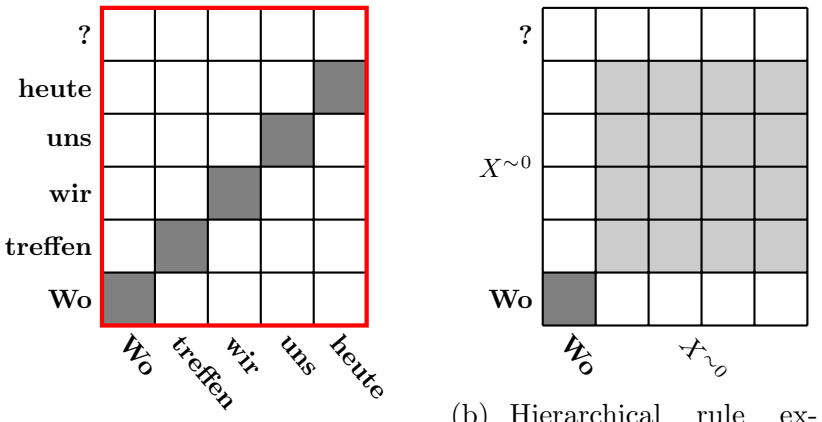
In phrase-based translation, the translation units are bilingual phrases which are pairs formed by a sequence of source language words and its translation. Since a sequence of words can be translated at once, local contextual information is preserved. In the context of punctuation prediction, such information is useful to predict punctuation marks depending of its surrounding words, e.g. commas. However, this approach has its limitation for unseen word sequences and dependencies beyond the local context, e.g. the dependency between a question word and a question mark. If a sequence of words was not seen in the training data, the phrase-based translation system will fall back on shorter phrases with less local contextual information. Thus, more prediction errors can occur. To generalize better and to model dependencies as described above, we need a more abstract form of phrases. In hierarchical translation, such phrases are defined since discontinuous phrases with "gaps" are allowed. Those phrases capture long-range dependencies between words. In terms of punctuation prediction, we want to model dependencies between words and punctuation marks. In addition, by using more abstract phrases, a punctuation prediction system based on hierarchical translation models is more robust for unseen word sequences and generalize better.

As described in Section 4.2.2, hierarchical rules are usually extracted with up to two non-terminal to model reordering implicitly. In case of punctuation prediction, we perform monotone translation. Thus, reordering is not necessary and rules with one non-terminal maximum are sufficient.

For the extraction of the hierarchical translation model, the same heuristic is applied as described in the previous section. By using these additional phrases as initial phrases in the hierarchical rule extraction process (Figure 8.7(b)), hierarchical rules, which model long-range dependencies between words and punctuation marks, can be generated, e.g.

$$X \rightarrow \langle \text{Wo } X^{\sim 0},\ \text{Wo } X^{\sim 0}\ ? \rangle$$
$$X \rightarrow \langle \text{treffen wir } X^{\sim 0},\ \text{treffen wir } X^{\sim 0}\ ? \rangle$$

In the first rule, the question mark on the target side is related to the German question word "wo". In the second rule, the typical German word order for questions (verb "treffen" before



(a) Extended phrase block using additional extraction heuristic

(b) Hierarchical rule extracted from extended phrase block

Figure 8.7: Extraction heuristic applied for initial phrase blocks.

subject "wir") triggers a question mark on the target side. Both rules are more abstract since the gap could be filled with any other phrases during decoding. Even for unseen word sequences, e.g. "Wo treffen wir uns morgen", these rules match. Thus, punctuation prediction based hierarchical translation can generalize better and improve the prediction accuracy.

In the experimental evaluation, we will analyze if such rules influence the decoding process and affect the punctuation prediction accuracy.

8.3.3 Optimization Criteria

In most state-of-the-art SMT systems, MERT is applied to optimize scaling factors of the feature functions using BLEU as optimization criterion. However, the performances of systems predicting punctuation are measured and compared with the F_1-Score which is the harmonic mean of precision and recall. Thus, there is an inconsistency between optimization criterion and evaluation metric. Furthermore, the F_1-Score considers both precision and recall while BLEU is a metric which is based on n-gram precision and does not take recall into account. A criterion including recall is important because it ensures that the punctuation prediction system generates an appropriate amount of punctuation marks. We propose to employ F_α-Score as a more suitable optimization criterion. F_α-Score is a parameterized version of the F_1-Score, where α is a positive real number [Van Rijsbergen 79]:

$$F_\alpha = (1+\alpha) \cdot \frac{(precision \cdot recall)}{\alpha \cdot precision + recall} \tag{8.4}$$

By varying the parameter α, more emphasis can be put on recall or precision. In the experimental evaluation, we will put more weight on recall and tune the systems with $\alpha \in \{1, 2, 3, 4\}$. We might lose precision and overgenerate punctuation marks, but this could be compensable for the actual translation system.

However, tuning a system on F_α-Score directly would not be practical as the positions of the punctuation marks would be ignored. For the optimization, the F_α-Score has to be slightly modified in order to take the predecessor word of each punctuation mark into account.

$$X \rightarrow \langle \text{i'm i'm like, I'm like} \rangle$$
$$X \rightarrow \langle \text{you can you can, you can} \rangle$$

Figure 8.8: Bilingual phrases for eliminating disfluency and introducing case information.

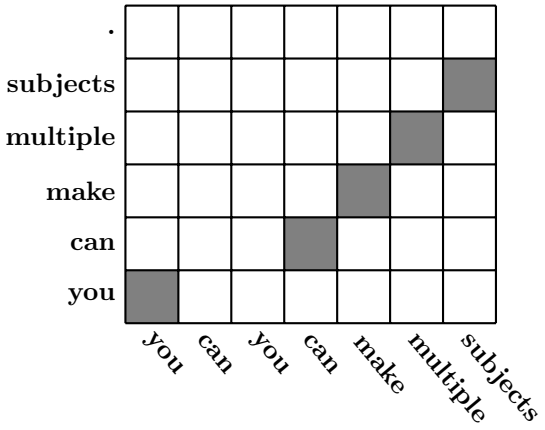


Figure 8.9: Alignment between automatic and manual transcription.

8.3.4 Automatically Transcribed Text as Training Corpus

The monolingual translation system for predicting punctuation marks as described above was trained on written text or manual transcribed data without punctuation in the source language. However, in a spoken language translation scenario, the input is automatic transcription generated by an ASR system. Automatic transcriptions usually contain recognition errors and do not provide case information. To overcome this mismatch, we suggest to train a monolingual system on a bilingual corpus with automatic transcriptions as source language data and manual transcriptions as target language data. The system is then able to translate from transcribed speech with recognition errors to proper output containing punctuation marks and case information. This process can be considered as postprocessing step. With bilingual phrases as shown in Figure 8.8, such a system is further able to eliminate disfluency.

When automatically transcribed text is used as training data, we can not assume a monotone alignment due to the recognition errors. We therefore learn word alignments as for a conventional SMT system (e.g. with GIZA++). Figure 8.9 shows such a word alignment. To obtain a reliable alignment and translation model, both are trained on an unified corpus of automatic and manual transcriptions as source language data.

Parameter tuning for such a translation system can be performed on automatic transcriptions as development set. The references are then the corresponding manual transcriptions. Obviously, the F_α-Score is not a suitable optimization criterion in this case. We suggest to use WER as criterion.

Table 8.1: Punctuation marks in all test sets and in-domain training data with their relative frequencies.

	punctuation mark								
relative frequency	,	.	”	?	-	:	;	!	’
all test sets	46.2	40.2	6.4	3.4	2.5	0.8	0.3	0.2	0.1
training data	46.0	37.7	5.3	3.4	4.4	2.3	0.6	0.3	0.1

8.4 Experimental Evaluation

In this section, we present experimental results of an extensive comparison of all methods introduced in the previous sections. The accuracy of the different punctuation prediction methods is measured in precision, recall and F_1-Score on *pseudo ASR output*. Pseudo ASR output is created by removing punctuation marks from the manual transcriptions. Thus, recognition errors do not occur and case information is preserved. However, for the final comparison, we used automatic transcriptions to investigate the impact of the different schemes in a real spoken language translation scenario. These test sets are marked with “(SLT)”. Automatic transcriptions is created with the RWTH Aachen University state-of-the-art ASR system presented in the following section.

The best performing approach was part of the RWTH Aachen University submission for IWSLT 2014 English→French SLT task. The official results will be presented in Section 8.4.3.

8.4.1 Setup

All experiments are conducted on the IWSLT 2014 English→French corpus using the same translation and speech recognition systems which are described in the following paragraphs. The data is described in Section A.3.3. All models used for punctuation prediction were trained on the in-domain portion of the provided training data. Table 8.1 introduces the punctuation marks which were to be predicted in this task and their relative frequencies for all test sets and in-domain training data. Note that these statistics are given for English (source language). Considering the relative frequencies, training and test data seems to match. Furthermore, in both training and test data 12.5% of the running words are punctuation marks.

English→French Translation System

For the actual translation from English to French a phrase-based translation system was employed. The word alignment is trained with GIZA++ on all available bilingual training data. The baseline includes a hierarchical reordering model and three backoff language model: a large general domain 5-gram language model, an in-domain 5-gram language model and a 7-gram word-class language model. The model weights were learned with MERT using Bleu as optimization criterion. More details can be found in [Wuebker & Peitz$^+$ 14].

English ASR System

Both the single-best outputs as well as the lattices and confusion networks are generated with a state-of-the-art ASR system similar to [Shaik & Tüske$^+$ 13]. The performance of the described ASR system is shown in Table 8.2. The system used in this work is a single ASR system based on 83-dimensional feature vectors obtained by concatenating spectral features with the multi-layer-

Table 8.2: Performance of the applied ASR system in terms of WER obtained after the first pass (vocal tract length normalization (VTLN)) and second pass (constrained maximum likelihood regression (CMLLR)).

pass	tst2010 (SLT)	tst2011 (SLT)
VTLN	17.7	17.6
CMLLR	13.8	14.6

Table 8.3: Comparison of punctuation prediction methods at different stages of the spoken translation pipeline. The oracle score is computed on the translation of the manual transcription with correct punctuation.

		tst2010	
stage of prediction	**prediction method**	BLEU [%]	TER [%]
before translation	language model	29.3	52.0
	phrase-based system	29.3	52.3
	hierarchical system	29.3	51.9
during translation	implicit	29.3	52.9
after translation	language model	28.4	52.6
	phrase-based system	28.8	52.9
	hierarchical system	28.8	52.6
oracle score	-	33.2	48.8

perceptron features. In a second pass, speaker adaptation using constrained maximum likelihood regression (CMLLR) is applied.

8.4.2 Results

Stage of Prediction

As described in Section 8.2 punctuation prediction can be performed at three different stages in the speech translation pipeline. In a preliminary experiment, prediction at these three stages is compared in terms of translation quality. Table 8.3 shows the results of this comparison. Punctuation prediction after the actual translation is clearly outperformed by both punctuation prediction before translation and during translation. Comparing the two latter ones, prediction before translation is slightly better than implicit prediction in terms of TER. Prediction before translation has however the clear advantage that the actual translation system does not need to be changed. In the following experiments punctuation prediction before translation using different approaches is investigated and compared to implicit prediction. Prediction after translation is not further considered.

Optimization Criterion

Next, we want to study the impact of different optimization criteria as introduced in Section 8.3.3. We tune our monolingual translation systems using the modified F_α-Score as criterion

Table 8.4: Comparison of different optimization criteria for monolingual translation systems (phrase-based and hierarchical). Punctuation prediction is performed before the actual the translation and compare with implicit prediction (implicit) and prediction applying a language model. The oracle score is computed on the translation of the manual transcription with correct punctuation.

	optimization	tst2010				
prediction method	**criterion**	*Precision*	*Recall*	F_1	BLEU [%]	TER [%]
implicit	-	-	-	-	29.3	52.9
language model	-	82.7	65.6	73.2	29.3	52.0
phrase-based system	BLEU	82.1	66.1	73.2	29.3	52.3
	F_1	83.2	66.2	73.7	29.4	52.2
	F_2	72.5	74.7	73.6	29.8	52.7
	F_3	69.3	75.9	72.5	29.8	53.1
	F_4	63.4	78.8	70.2	29.7	53.7
hierarchical system	BLEU	84.9	67.0	74.9	29.3	51.9
	F_1	84.6	66.4	74.4	29.4	51.9
	F_2	72.1	76.8	74.3	29.8	52.4
	F_3	69.0	78.5	73.4	29.8	52.9
	F_4	60.5	81.9	69.6	29.2	54.2
oracle score	-	-	-	-	33.2	48.8

with $\alpha \in \{1, 2, 3, 4\}$ and compare against systems tuned on BLEU. Table 8.4 shows the result of this comparison. Tuning the weights of a monolingual translation system with a more appropriate optimization criterion seems to improve the translation quality and outperforms both the implicit prediction approach and prediction using a language model. In the following all monolingual translation systems use the F_2-Score as optimization criterion.

Word Class Language Model

Another advantage of using a machine translation system for performing punctuation prediction is that an additional model can be added easily to the log-linear model combination. In our experiments both phrase-based and hierarchical monolingual translation system tuned on F_2-Score were extended with a word class language model (wcLM) as introduced in Section 4.4.2. Table 8.5 shows the results of this experiment. By adding a word class language model both punctuation accuracy and final translation quality are increased. The impact is larger for the hierarchical translation system. One reason could be that the hierarchical system allows for scoring longer word sequences by the word class language model. The perplexity computed on the English enriched hypotheses applying the word class language model is slightly lower compared with the phrase-based setup (40.0 vs 39.0).

Phrase-Based versus Hierarchical Translation

Previous results show that both the phrase-based and the hierarchical approach perform on the same level in terms of BLEU. Punctuation prediction with hierarchical machine translation however shows a slight improvement in TER and outperforms the phrase-based approach by adding a word class language model. To further investigate the advantage of applying a hierarchical

Table 8.5: Impact of a word class language model (wcLM) for monolingual translation systems (phrase-based and hierarchical).

prediction method	optimization criterion	tst2010 *Precision*	*Recall*	F_1	BLEU [%]	TER [%]
phrase-based system	F_2	72.5	74.7	73.6	29.8	52.7
+ wcLM		74.0	75.8	74.9	30.1	52.4
hierarchical system	F_2	72.1	76.8	74.3	29.8	52.4
+ wcLM		74.7	78.7	76.6	30.3	52.0

Table 8.6: Comparison of the numbers of applied phrases introducing punctuation marks.

prediction method	tst2010 lexical rules	hierarchical rules
phrase-based system (F_2, wcLM)	4021	-
hierarchical system (F_2, wcLM)	63	4182

translation system, the actual usage of hierarchical rules in the decoding process is analyzed. This is done by simply counting the lexical and hierarchical rules applied during decoding. In particular, we count rules which introduce punctuation marks (Table 8.6). While the phrase-based system naturally uses phrases with a limited context (average target phrase length of 2.9), the hierarchical system mainly employs hierarchical rules to insert punctuation marks, e.g.

$$X \rightarrow \langle \text{what what } X^{\sim 0}, \text{ " what , what } X^{\sim 0} \text{ ? "} \rangle.$$

This rule was extracted from the sentence pair

English without Punctuation: what what do you mean

English with Punctuation: " what , what do you mean ? "

However, the phrase-based extraction process only generates phrases such as

$$(\text{what what , " what , what}).$$

For the test sentence "what what are you feeding" the phrase-based system is not able to produce a question mark at the end of the sentence. The hierarchical system however applies the described hierarchical rule and produces the correct output.

Final Comparison

Finally, all methods for prediction punctuation marks are compared on two different test sets. Table 8.7 shows the results on pseudo ASR output while in Table 8.8 results for real ASR output are provided.

Furthermore, the latter comparison also includes phrase-based translation systems taking confusion networks or lattices rather than sentences as input. Both the use of confusion networks and lattices introduces a new feature namely slot-wise posterior probability into our log-linear model combination.

Table 8.7: Comparison of the best performing systems on two different test sets (pseudo ASR output). Both monolingual translation systems are tuned on F_2-Score and augmented with a word class language model (wcLM). The oracle score is computed on the translation of the manual transcription with correct punctuation.

	tst2010		tst2011	
prediction method	BLEU [%]	TER [%]	BLEU [%]	TER [%]
implicit	29.3	52.9	35.7	45.1
language model	29.3	52.0	35.4	44.8
phrase-based (F_2, wcLM)	30.1	52.4	35.8	45.6
hierarchical (F_2, wcLM)	30.3	52.0	36.1	45.2
oracle score	33.2	48.8	39.5	41.6

Having confusion networks as input results in an improvement of up to 0.6 points in BLEU and 0.8 points in TER while using lattices leads to smaller gains. Enriching first-best output with monolingual translation system performs best in terms of BLEU while having confusion networks as input leads to the best results in terms of TER. Since in confusion networks words can be skipped by choosing the ϵ-edge, disfluency or words with a low score can be ignored during translation. While the selected source paths through the confusion networks do not improve WER, the number of source words is lower (cf. Table 8.9). This leads to less insertion errors and to a better TER score on `tst2010 (SLT)`. For lattices, the WER of the selected source sentence is worse. However, a lower WER does not seem to result in a lower translation quality. A similar effect has been noted by [Patry & Langlais 08].

Furthermore, augmenting the training data of the punctuation prediction systems with automatic transcription as suggested in Section 8.3.4 improves the translation quality by up to 1 point in BLEU and 2.1 points in TER. This is related to disfluency removal as observed when confusion networks are used as input.

8.4.3 Official Results

Table 8.10 shows the official evaluation results of the IWSLT 2014 English→French SLT task. The translation system for the final submission of the RWTH Aachen University includes discriminative training [Wuebker & Muehr$^+$ 15], a recurrent neural network translation [Sundermeyer & Alkhouli$^+$ 14] and recurrent neural network language model [Sundermeyer & Ney$^+$ 15] on top of the baseline as described in the beginning of this section. Further details are given in [Wuebker & Peitz$^+$ 14].

Punctuation prediction was performed using a hierarchical translation system tuned on F_2-Score. The RWTH Aachen University ranked second out of seven in terms of BLEU and TER in this evaluation.

8.5 Conclusion

In this chapter, we have extensively compared different approaches to couple automatic speech recognition and machine translation. In particular, several methods of punctuation prediction at different stages in the spoken language translation pipeline were analyzed. The experiments were conducted on a large-scale task and show that punctuation prediction before the actual translation

Table 8.8: Comparison of the best performing systems on two different test sets (real ASR output). This comparison also includes phrase-based translation systems taking confusion networks or lattices rather than sentences as input. *+ automatic transcription* indicates that the punctuation prediction system was trained on data with automatic transcribed text as source language. The oracle score is computed on the translation of the manual transcription with correct punctuation.

	tst2010 (SLT)		tst2011 (SLT)	
prediction method	BLEU [%]	TER [%]	BLEU [%]	TER [%]
implicit (first-best sentences)	24.2	59.2	28.8	53.2
implicit (lattices)	24.7	58.9	29.0	53.1
implicit (confusion networks)	24.7	58.6	29.4	52.6
phrase-based (F_2, wcLM)	25.2	59.0	29.5	54.0
+ automatic transcription	25.7	57.6	30.5	51.9
hierarchical (F_2, wcLM)	25.5	58.6	29.8	53.1
+ automatic transcription	25.7	57.5	30.5	51.8
oracle score	33.2	48.8	39.5	41.6

Table 8.9: Comparison of different types of input (first-best, lattices, and confusion networks) and the effect on WER and TER.

	tst2010 (SLT)			
input	WER [%]	source words	TER [%]	insertion errors
first-best sentences	13.8	27014	59.2	2206
lattices	14.2	26521	58.9	2184
confusion networks	13.8	26844	58.6	2122

is able to improve the translation quality over a strong baseline. Punctuation prediction using a monolingual hierarchical phrase-based translation system extended with word class language model seems to be the best working option. If appropriate data are available, the monolingual system should be trained on automatic rather than manual transcribed data. These experiments further show that taking confusion networks or word lattices as input is not really necessary to obtain better translation quality.

8.6 Contributions

The different methods for punctuation prediction introduced in this chapter were developed in joint work with Markus Freitag, Arne Mauser, Simon Wiesler and Markus Nußbaum-Thom and published in [Peitz & Freitag$^+$ 11, Peitz & Wiesler$^+$ 12, Peitz & Freitag$^+$ 14]. Arne Mauser suggested to compare the different methods at different stages in the speech translation pipeline and to use automatic transcribed data as training data. Markus Freitag implemented the F_α-Score as optimization criterion into RWTH Aachen University's translation toolkit *Jane*. Simon Wiesler and Markus Nußbaum-Thom helped the author of this thesis to run an ASR system. The author of this thesis came up with the idea to apply hierarchical translation for punctuation prediction,

Table 8.10: Official IWSLT 2014 English→French SLT task evaluation results (case sensitive) [Cettolo & Niehues+ 14]. RWTH Aachen University ranked second out of seven in terms of BLEU and TER.

participant	tst2014 (SLT) BLEU [%]	tst2014 (SLT) TER [%]
Karlsruher Institut für Technologie	*27.5*	57.8
RWTH Aachen University	**27.0**	**57.3**
Universit du Maine, Le Mans	26.8	59.0
University of Edinburgh	25.5	*57.2*
Fondazione Bruno Kessler, Trento	25.4	59.5
LIMSI, Paris	25.2	60.7
University of Sheffield	23.5	59.9

to tune on different optimization criteria and supervised the implementation of lattice input in the scope of [Dahlmann 15]. The experiments described in Section 8.4 were exclusively conducted by the author for this thesis. The MT system used in the official IWSLT 2014 English→French SLT task (cf. Section 8.4.3) was provided by Joern Wuebker and Andreas Guta.

8.7 Related Work

Since almost all state-of-the-art speech recognition systems do not generate punctuation, but most machine translation systems are trained on text data with proper punctuation marks, punctuation prediction has to be done at some stage in the speech translation pipeline. In [Matusov & Mauser+ 06] three different approaches to restore punctuation in already segmented ASR output is presented. In addition to implicit punctuation generation in the translation process, punctuation is predicted as pre- and postprocessing step. For punctuation prediction they apply a tool from the SRI toolkit [Stolcke 02] which uses a language model trained on text with punctuation marks. The implicit punctuation generation seems to work best on IWSLT 2006 corpus, but on TC-STAR 2006 corpus they achieve better results with punctuation prediction on source and target. They point out that on small corpora like IWSLT 2006 falsely inserted punctuation marks in the source side deteriorated the performance of the translation system. However, the IWSLT corpus became larger in the last years and therefore we verify the results in this work on a large-scale spoken language translation task. Furthermore, we use in addition for the punctuation prediction a phrase-based and a hierarchical machine translation system.

Using machine translation for punctuation prediction is firstly described in [Hassan & Ma+ 07]. The authors train a phrase-based statistical machine translation system on a pseudo-bilingual corpus. The case-sensitive target language text with punctuation is considered as the target language and the text without case information and punctuation is used as source language. They applied this approach as postprocessing step in evaluation campaign of IWSLT 2007 and achieved a significant improvement over the baseline.

In [Ma & Tinsley+ 08] the same approach was employed as preprocessing step and compared with language model based prediction within the evaluation campaign of IWSLT 2008. The MT-based punctuation prediction performed worse. In our work, we further investigate this approach and compare it with the language model based prediction at different stages at which the prediction is done. In our analysis we consider translation quality at the end of the translation pipeline as

well as the accuracy of the punctuation prediction.

The approach described in [Lu & Ng 10] is based on conditional random fields (CRF). They extend the linear-chain CRF model to a factorial CRF model using two layers with different sets of tags for punctuation marks respectively sentence types. They compare their novel approach with linear-chain CRF model and language model based prediction on the IWSLT 2009 corpus. Besides the comparison of the translation quality in terms of BLEU, they also compare with respect to precision, recall and F_1-Score. Both in terms of BLEU and in terms of precision, recall and F_1-Score the CRF models outperform the hidden event language model. They claim that using non-independent and overlapping features of the discriminative model leads to these improvements. Similar to this approach, using a statistical machine translation system for punctuation prediction has the advantage of integrating more features beside the language model.

Another approach for coupling ASR and MT is to access word lattices generated by the ASR system. In [Matusov & Hoffmeister$^+$ 08] a phrase-based word lattice decoder for SLT is presented. The experiments were performed on the medium-sized TC-Star Spanish→English SLT task and the result show improvements of up to 0.4 points in BLEU and TER. However, phrase reordering was disabled for the experiments on this task. In our work, we perform lattice and confusion network decoding including a hierarchical reordering model on a large-scale SLT task and compare against sentence-based translation. Confusion network translation is introduced in [Bertoldi & Zens$^+$ 07]. Experimental results on the TC-Star Spanish→English SLT task show improvements of up to 0.7 points in BLEU. The setups applied in these experiments include a simple distance-based reordering model. A hierarchical phrase-based lattices decoder was introduced in [Dyer & Muresan$^+$ 08].

Another line of related research is the postprocessing of ASR output by removing disfluency. In [Cho & Ha$^+$ 13, Cho & Niehues$^+$ 14] disfluency removal is performed by obtaining a disfluency probability for each word applying a CRF. Results on the IWSLT 2013 German→English SLT task show improvements of up to 0.8 in BLEU. In our work, this implicitly done by adding manual transcribed data to the training corpus.

9. Scientific Achievements

In this chapter, we discuss whether we have achieved the scientific goals formulated in Chapter 2.

- In Section 4.3.4, we introduced our improved implementation of the cube pruning decoder which is part of RWTH Aachen University's translation toolkit *Jane* since version 2.3. Experimental results presented in Section 7.1 and [Heafield 13] have shown that this implementation resulted in a competitive hierarchical phrase-based translation engine which is fast and memory-efficient. We improved the translation speed by up to 1.9 words per second (from 7.0 to 8.9) and reduced the memory usage by up to 4 gigabyte compared to the previous implementation.

- We introduced our generative training framework for hierarchical translation models in Chapter 5. The provided framework is both effective and efficient and can be applied on large training data. Experimental results were shown for different translation tasks in Section 7.3.1 and Section 7.3.2. With the generative training approach, we were able to reduce the translation model size by up to 95% while simultaneously improving the translation quality. We further observed improvements of up to 0.6 points in $\frac{\text{Ter}-\text{Bleu}}{2}$ on a large-scale Chinese→English translation task (Section 7.4.3). As shown in the final comparison in Section 7.4.4, discriminative training however outperformed the generative approach.

- In Chapter 6, we described several smoothing techniques. While most of them can be applied in both phrase-based and hierarchical translation, we also described a smoothing scheme designed for the hierarchical approach. First, all methods were evaluated separately on different tasks in Section 7.2.1, 7.2.2 and 7.2.3 and finally compared systematically on a large-scale translation task (Section 7.4.2). The results of this comparison are mixed. While the count-based smoothing approaches led to improvements on all translation tasks, class-based and syntax-based smoothing did not improve the translation quality consistently. As a result of the work presented in this thesis, the enhanced low-frequency feature and the word-class language model became part of RWTH Aachen University's hierarchical phrase-based baseline.

- In Chapter 8, we introduced the idea of modeling punctuation prediction, re-casing and disfluency removal as a hierarchical phrase-based translation task. Experimental results in Section 8.4 show improvements of up to 1.0 Bleu. By applying this approach, we were able to rank second out of seven in the official IWSLT 2014 English→French SLT evaluation campaign in terms Bleu and Ter. Additionally, in order to compare our approach with a strong baseline, we integrated a phrase-based lattice decoder into RWTH Aachen University's translation toolkit *Jane*. We showed that using confusion networks to couple ASR and MT improves the translation quality by up to 0.6 % Bleu over a strong baseline on a large-scale spoken language translation task. However, performing punctuation prediction,

re-casing and disfluency removal by applying a statistical machine translation system before the actual translation seems to perform best. We gained additional improvements by using a more suitable optimization criterion and automatically transcribed speech as training data.

10. Individual Contributions

This chapter summarizes the Section 4.5, 5.4, 6.5, 7.6 and 8.6 to highlight the individual contributions made by the author during the course of this thesis.

The author is responsible for the re-implementation of the hierarchical phrase-based decoder which was released as part of RWTH Aachen University's translation toolkit *Jane* version 2.3 [Freitag & Huck$^+$ 14]. This implementation were described in Section 4.3.4 and evaluated in Section 7.1 and [Heafield 13].

The generative training approach for hierarchical phrase-based translation as described in Chapter 5 was joint work of the author of this thesis with Arne Mauser, Joern Wuebker and David Vilar. Arne Mauser suggested to apply the two-parse algorithm and inside-outside algorithm, Joern Wuebker provided the leave-one-out framework and David Vilar came up with an initial implementation of the k-best parsing algorithm. The author of this thesis implemented both the two-parse algorithm (Section 5.2.1) and the inside-outside algorithm (Section 5.3.1), adapted the leave-one-out framework for hierarchical translation (Section 5.2.3), came up with idea to employ the modified CYK+ variant (Section 5.2.2), refined the k-best parsing algorithm (Section 5.3.2) and augmented it with a new recombination scheme. He is further responsible for all experiments which were previously published in [Peitz & Mauser$^+$ 12, Peitz & Vilar$^+$ 14] (cf. Section 7.3.1 and 7.3.2) or exclusively performed for this work (Section 7.4.3). The entire implementation is part of RWTH Aachen University's translation toolkit *Jane*.

The implementation and experimental evaluation of the word class language model for hierarchical phrase-based translation (cf. Section 4.4.2 and 7.2.2, respectively) were also done by the author of this thesis and previously published in [Wuebker & Peitz$^+$ 13b]. In the scope of this publication, the author further performed the experimental evaluation of the word class translation model for hierarchical phrase-based translation.

Further smoothing approaches as presented in Chapter 6 were either already implemented in *Jane* (Section 6.2.1 and 6.2.2), implemented by the author (Section 6.2.3) or implemented under the author's supervision (Section 6.4) as part of Yifeng Lu's master thesis [Lu 14]. Almost all corresponding experiments were conducted by the author of this thesis (Section 7.2.1, 7.2.2 and 7.4.2). The experimental results shown in Section 7.2.3 were part of Yifeng Lu's master thesis [Lu 14].

Chapter 8 was joint work of the author of this thesis with Markus Freitag, Arne Mauser, Simon Wiesler and Markus Nußbaum-Thom. Arne Mauser suggested to compare the different methods at different stages in the speech translation pipeline and to use automatic transcribed data as training data. Markus Freitag implemented the F_α-Score as optimization criterion into RWTH Aachen University's translation toolkit *Jane*. Simon Wiesler and Markus Nußbaum-Thom helped the author of this thesis to run an ASR system. The author of this thesis came up with the idea to apply hierarchical translation for punctuation prediction and to tune on different optimization criteria. The described methods were previously published in [Peitz & Freitag$^+$ 11, Peitz & Wiesler$^+$ 12, Peitz & Freitag$^+$ 14]. The implementation of lattice input was done in the

scope of [Dahlmann 15] by Leonard Dahlmann under the author's supervision. The experiments described in Section 8.4 were exclusively conducted by the author for this thesis. The MT system used in the official IWSLT 2014 English→French SLT task (cf. Section 8.4.3) was provided by Joern Wuebker and Andreas Guta and is described in [Wuebker & Peitz$^+$ 14].

A. Overview of the Corpora

A.1 Workshop on Statistical Machine Translation

The Workshop on Statistical Machine Translation (WMT) is an annual evaluation campaign that provides training and test data for a number of European language pairs. Its main focus is the translation of news texts.

A.1.1 WMT 2012 German→English and French→English

Statistics for training, development and test data of the German→English and French→English task provided for the *NAACL 2012 Workshop on Statistical Machine Translation*[1] are given in Table A.1. For German, we apply the frequency-based compound splitting method proposed by [Koehn & Knight 03]. We use `newstest2010` for parameter optimization and `newstest2011` as test set.

Table A.1: Corpus statistics for the bilingual training data provided for the WMT 2012 German→English and French→English task. Out-of-vocabulary (OOV) numbers refer to the running words.

		German	English	French	English
`train`:	Sentences	1.8M		1.7M	
	Running Words	44.2M	44.9M	44.2M	40.7M
	Vocabulary	176K	118K	129K	110K
`newstest2010`:	Sentences	2489			
	Running Words	64.7K	62.0K	72.1K	62.0K
	Vocabulary	11K	9K	11K	9K
	OOVs	1248	1068	1439	1452
`newstest2011`:	Sentences	3003			
	RunningWords	76.3K	74.8K	87.8K	74.8K
	Vocabulary	13K	11K	12K	11K
	OOVs	1568	1449	1792	1782

[1] `http://www.statmt.org/wmt12/`

A.1.2 WMT 2014 German→English

Statistics for training, development and test data of the German→English task provided for the *ACL 2014 Workshop on Statistical Machine Translation*[2] are given in Table A.2. For German, we apply the frequency-based compound splitting method proposed by [Koehn & Knight 03] and the part-of-speech-based long-range reordering rules as described in [Popović & Ney 06]. We use `newstest2012` for parameter optimization and `newstest2013` as test set.

Table A.2: Corpus statistics for the bilingual training data provided for the WMT 2014 German→English translation task. `newstest2012` is used as development set and `newstest2013` as blind test set. Out-of-vocabulary (OOV) numbers refer to the running words.

		English	German
`train`:	Sentences	4.1M	
	Running Words	102M	104M
	Vocabulary	748K	648K
`newstest2012`:	Sentences	3003	
	Running Words	75K	73K
	Vocabulary	14K	10K
	OOVs	625	578
`newstest2013`:	Sentences	3000	
	Running Words	65K	65K
	Vocabulary	12K	9K
	OOVs	532	511

A.2 Broad Operational Language Translation Program

The DARPA BOLT (Broad Operational Language Translation) Program aims to develop genre-independent machine translation and information retrieval systems. While earlier DARPA programs made significant strides in improving natural language processing capabilities in structured genres like newswire and broadcasts, BOLT is particularly concerned with improving translation and information retrieval performance for less-formal genres with a special focus on user-contributed content.

A.2.1 BOLT 2014 Chinese→English (Discussion Forum)

Corpus statistics for the BOLT 2014[3] Chinese→English translation task (Discussion Forum) are shown in Table A.3.

A.3 International Workshop on Spoken Language Translation

The *International Workshop on Spoken Language Translation* (IWSLT) is an annual public evaluation campaign focusing on spoken language translation. The domain is lecture-type talks pre-

[2] `http://www.statmt.org/wmt14/`
[3] `https://www.ldc.upenn.edu/collaborations/current-projects/bolt/`

Table A.3: Corpus statistics for the bilingual training data provided for the BOLT 2014 Chinese→English translation task (Discussion Forum). Out-of-vocabulary (OOV) numbers refer to the running words. `DEV12-tune` is used as development set and `DEV12-dev` and `P1R6-dev` are used as blind test sets.

		Chinese	English
`train`:	Sentences	4.1M	
	Running Words	78M	86M
	Vocabulary	384K	817K
`train` (in-domain):	Sentences	65K	
	Running Words	1.4M	1.7M
	Vocabulary	40K	29K
`DEV12-tune`:	Sentences	1845	
	Running Words	38K	47K
	Vocabulary	6K	7K
	OOVs	1018 (2.7%)	78 (0.2%)
`DEV12-dev`:	Sentences	1844	
	Running Words	39K	48K
	Vocabulary	6K	5K
	OOVs	891 (2.3%)	442 (0.9%)
`P1R6-dev`:	Sentences	1124	
	Running Words	24K	25K
	Vocabulary	4K	4K
	OOVs	8 (0.03%)	227 (0.9%)

sented at TED conferences[4]. The translation part of the evaluation campaign is divided into two tracks: translation of automatic transcriptions and translation of manual transcriptions. While the correct manual transcription contains punctuation marks and case information, the automatic transcription does not. Furthermore, recognition errors occur. In all setups, `dev2010` is used as development set for parameter tuning.

A.3.1 IWSLT 2012 German→English

Corpus statistics for the IWSLT 2012[5] German→English task are shown in Table A.4. For German, we apply the frequency-based compound splitting method proposed by [Koehn & Knight 03] and the part-of-speech-based long-range reordering rules as described in [Popović & Ney 06]. `tst2010` is used as blind test set.

A.3.2 IWSLT 2013 German→English

Corpus statistics for the IWSLT 2013[6] German→English task are shown in Table A.5. For German, we apply the frequency-based compound splitting method proposed by [Koehn & Knight 03] and the part-of-speech-based long-range reordering rules as described in [Popović & Ney 06]. `tst2010` and `tst2011` are used as blind test sets.

[4] `https://www.ted.com/`
[5] `http://hltc.cs.ust.hk/iwslt/`
[6] `http://workshop2013.iwslt.org/`

Table A.4: Corpus statistics for the IWSLT 2012 German→English task.

		German	English
`train`:	Sentences	2.2M	
	Running Words	58M	58M
	Vocabulary	212K	144K
`dev2010`:	Sentences	883	
	Running Words	20K	20K
	Vocabulary	4K	3K
	OOVs	214	144
`tst2010`:	Sentences	1565	
	Running Words	31K	32K
	Vocabulary	5K	4K
	OOVs	227	153

A.3.3 IWSLT 2014 English→French

Data statistics for the IWSLT 2014[7] English→French task are shown in Table A.6. `tst2010`, `tst2011` and `tst2015` are used as blind test set.

[7] `http://workshop2014.iwslt.org/`

Table A.5: Corpus statistics for the IWSLT 2013 German→English task.

		German	English
`train`:	Sentences	4.3M	
	Running Words	108M	109M
	Vocabulary	836K	792K
	Singletons	425K	432K
`train` (in-domain):	Sentences	138K	
	Running Words	2.6M	2.7M
	Vocabulary	75K	50K
	Singletons	35K	21K
`dev2010`:	Sentences	887	
	Running Words	20K	20K
	Vocabulary	4K	3K
	OOVs	96	82
`tst2010`:	Sentences	1565	
	Running Words	31K	32K
	Vocabulary	5K	4K
	OOVs	94	68
`tst2011`:	Sentences	1436	
	Running Words	27K	27K
	Vocabulary	5K	4K
	OOVs	91	62

Table A.6: Corpus statistics for the IWSLT 2014 English→French task.

		English	French
`train`:	Sentences	26M	
	Running Words	694M	810M
	Vocabulary	2.2M	2.1M
	Singletons	1.1M	1.1M
`train` (in-domain):	Sentences	185K	
	Running Words	3.6M	3.9M
	Vocabulary	60K	74K
	Singletons	24K	30K
`dev2010`:	Sentences	934	
	Running Words	20K	20K
	Vocabulary	3K	4K
	OOVs	45	63
`tst2010`:	Sentences	1664	
	Running Words	31K	34K
	Vocabulary	4K	5K
	OOVs	44	42
`tst2011`:	Sentences	818	
	Running Words	14K	16K
	Vocabulary	2K	3K
	OOVs	37	44
`tst2014`:	Sentences	1305	
	Running Words	24K	28K
	Vocabulary	4K	4K
	OOVs	83	56

List of Figures

List of Tables

A. List of Algorithms

Bibliography

[Auli & Galley+ 14] M. Auli, M. Galley, J. Gao: Large Scale Expected BLEU Training of Phrase-based Reordering Models. In *Conference on Empirical Methods in Natural Language Processing*, pp. 1250–1260, Doha, Qatar, Oct. 2014.

[Bar-Hillel 51] Y. Bar-Hillel: The Present State of Research on Mechanical Translation. *American Documentation*, Vol. 2, pp. 229–237, 1951.

[Bertoldi & Zens+ 07] N. Bertoldi, R. Zens, M. Federico: Speech Translation by Confusion Networks Decoding. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 1297–1300, Honolulu, HI, USA, April 2007.

[Birch & Callison-Burch+ 06] A. Birch, C. Callison-Burch, M. Osborne, P. Koehn: Constraining the Phrase-Based, Joint Probability Statistical Translation Model. In *Workshop on Statistical Machine Translation*, pp. 154–157, New York City, NY, USA, June 2006.

[Bisazza & Ruiz+ 11] A. Bisazza, N. Ruiz, M. Federico: Fill-up Versus Interpolation Methods for Phrase-Based SMT Adaptation. In *International Workshop on Spoken Language Translation*, pp. 136–143, San Francisco, CA, USA, Dec. 2011.

[Blunsom & Cohn+ 08a] P. Blunsom, T. Cohn, M. Osborne: Bayesian Synchronous Grammar Induction. In *Advances in Neural Information Processing Systems*, pp. 161–168, Vancouver, BC, Canada, Dec. 2008.

[Blunsom & Cohn+ 08b] P. Blunsom, T. Cohn, M. Osborne: A Discriminative Latent Variable Model for Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics*, pp. 200–208, Columbus, OH, USA, June 2008.

[Blunsom & Cohn+ 09] P. Blunsom, T. Cohn, C. Dyer, M. Osborne: A Gibbs Sampler for Phrasal Synchronous Grammar Induction. In *Annual Meeting of the Association for Computational Linguistics*, pp. 782–790, Singapore, Aug. 2009.

[Boudahmane & Buschbeck+ 11] K. Boudahmane, B. Buschbeck, E. Cho, J.M. Crego, M. Freitag, T. Lavergne, H. Ney, J. Niehues, S. Peitz, J. Senellart, A. Sokolov, A. Waibel, T. Wandmacher, J. Wuebker, F. Yvon: Advances on Spoken Language Translation in the Quaero Program. In *International Workshop on Spoken Language Translation*, pp. 114–120, San Francisco, CA, USA, Dec. 2011.

[Brown & Cocke+ 88] P.F. Brown, J. Cocke, S.A. Della Pietra, V.J. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, P.S. Rossin: A Statistical Approach to Language Translation. In

International Conference on Computational Linguistics, pp. 71–76, Buffalo, NY, USA, Aug. 1988.

[Brown & Cocke+ 90] P.F. Brown, J. Cocke, S.A. Della Pietra, V.J. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, P.S. Rossin: A Statistical Approach to Machine Translation. *Computational Linguistics*, Vol. 16, No. 2, pp. 79–85, June 1990.

[Brown & Della Pietra+ 93] P.F. Brown, V.J. Della Pietra, S.A. Della Pietra, R.L. Mercer: The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, Vol. 19, No. 2, pp. 263–311, June 1993.

[Cettolo & Niehues+ 14] M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, M. Federico: Report on the 11th IWSLT Evaluation Campaign, IWSLT 2014. In *International Workshop on Spoken Language Translation*, pp. 2–17, Lake Tahoe, CA, USA, Dec. 2014.

[Chappelier & Rajman 98] J.C. Chappelier, M. Rajman: A Generalized CYK Algorithm for Parsing Stochastic CFG. In *Workshop on Tabulation in Parsing and Deduction*, pp. 133–137, Paris, France, April 1998.

[Chen & Goodman 96] S.F. Chen, J. Goodman: An Empirical Study of Smoothing Techniques for Language Modeling. In *Annual Meeting of the Association for Computational Linguistics*, pp. 310–318, Santa Cruz, CA, USA, June 1996.

[Chen & Goodman 98] S.F. Chen, J. Goodman: An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, Cambridge, MA, USA, Aug. 1998.

[Chen & Kuhn+ 11] B. Chen, R. Kuhn, G. Foster, H. Johnson: Unpacking and Transforming Feature Functions: New Ways to Smooth Phrase Tables. In *Machine Translation Summit*, pp. 269–276, Xiamen, China, Sept. 2011.

[Chiang 05] D. Chiang: A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics*, pp. 263–270, Ann Arbor, MI, USA, June 2005.

[Chiang 07] D. Chiang: Hierarchical Phrase-Based Translation. *Computational Linguistics*, Vol. 33, No. 2, pp. 201–228, June 2007.

[Chiang & Knight+ 09] D. Chiang, K. Knight, W. Wang: 11,001 New Features for Statistical Machine Translation. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 218–226, Boulder, CO, USA, June 2009.

[Cho & Ha+ 13] E. Cho, T.L. Ha, A. Waibel: CRF-Based Disfluency Detection using Semantic Features for German to English Spoken Language Translation. In *International Workshop on Spoken Language Translation*, pp. 280–287, Heidelberg, Germany, Dec. 2013.

[Cho & Niehues+ 12] E. Cho, J. Niehues, A. Waibel: Segmentation and punctuation prediction in speech language translation using a monolingual translation system. In *International Workshop on Spoken Language Translation*, pp. 252–259, Hong Kong, Dec. 2012.

[Cho & Niehues+ 14] E. Cho, J. Niehues, A. Waibel: Tight Integration of Speech Disfluency Removal into SMT. In *Conference of the European Chapter of the Association for Computational Linguistics*, pp. 43–47, Gothenburg, Sweden, May 2014.

[Chomsky 56] N. Chomsky: Three Models for the Description of Language. *IRE Transactions on Information Theory*, Vol. 2, No. 3, pp. 113–124, 1956.

[Church & Gale 91] K.W. Church, W.A. Gale: A Comparison of the Enhanced Good-Turing and Deleted Estimation Methods for Estimating Probabilities of English Bigrams. *Computer Speech & Language*, Vol. 5, pp. 19–54, 1991.

[Čmejrek & Zhou$^+$ 09] M. Čmejrek, B. Zhou, B. Xiang: Enriching SCFG Rules Directly from Efficient Bilingual Chart Parsing. In *International Workshop on Spoken Language Translation*, pp. 136–143, Tokyo, Japan, Dec. 2009.

[Čmejrek & Zhou 10] M. Čmejrek, B. Zhou: Two Methods for Extending Hierarchical Rules from the Bilingual Chart Parsing. In *International Conference on Computational Linguistics*, pp. 180–188, Beijing, China, Aug. 2010.

[Cocke 69] J. Cocke: *Programming Languages and Their Compilers: Preliminary Notes*. Courant Institute of Mathematical Sciences, New York University, New York, NY, USA, 1969.

[Collins & Duffy 01] M. Collins, N. Duffy: Convolution Kernels for Natural Language. In *Advances in Neural Information Processing Systems*, pp. 625–632, Vancouver, BC, Canada, Dec. 2001.

[Dahlmann 15] F.L. Dahlmann: Phrase-Based Lattice Decoding and its Applications in Machine Translation. Bachelor's thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, March 2015.

[Dempster & Laird$^+$ 77] A.P. Dempster, N.M. Laird, D.B. Rubin: Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, Vol. 39, No. 1, pp. 1–38, 1977.

[DeNero & Bouchard-Côté$^+$ 08] J. DeNero, A. Bouchard-Côté, D. Klein: Sampling Alignment Structure Under a Bayesian Translation Model. In *Conference on Empirical Methods in Natural Language Processing*, pp. 314–323, Waikiki, HI, USA, Oct. 2008.

[DeNero & Gillick$^+$ 06] J. DeNero, D. Gillick, J. Zhang, D. Klein: Why Generative Phrase Models Underperform Surface Heuristics. In *Workshop on Statistical Machine Translation*, pp. 31–38, New York City, NY, USA, June 2006.

[Duan & Li$^+$ 12] N. Duan, M. Li, M. Zhou: Forced Derivation Tree based Model Training to Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing*, pp. 445–454, Jeju Island, Korea, July 2012.

[Durrani & Haddow$^+$ 13] N. Durrani, B. Haddow, K. Heafield, P. Koehn: Edinburgh's Machine Translation Systems for European Language Pairs. In *Workshop on Statistical Machine Translation*, pp. 114–121, Sofia, Bulgaria, Aug. 2013.

[Dyer 10] C. Dyer: Two Monolingual Parses are Better Than One (Synchronous Parse). In *Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 263–266, Los Angeles, CA, USA, June 2010.

[Dyer & Muresan$^+$ 08] C. Dyer, S. Muresan, P. Resnik: Generalizing Word Lattice Translation. In *Annual Meeting of the Association for Computational Linguistics*, pp. 1012–1020, Columbus, OH, USA, March 2008.

[Dyer & Weese+ 10] C. Dyer, J. Weese, H. Setiawan, A. Lopez, F. Ture, V. Eidelman, J. Ganitkevitch, P. Blunsom, P. Resnik: Cdec: A Decoder, Alignment, and Learning Framework for Finite-state and Context-free Translation Models. In *Annual Meeting of the Association for Computational Linguistics*, pp. 7–12, Uppsala, Sweden, July 2010.

[European Commission 12] European Commission: Eurobarometer: 98% say language learning is good for their children, but tests highlight skills gap. Press release, June 2012. [`http://europa.eu/rapid/press-release_IP-12-679_en.htm`; accessed: 2016-01-19].

[European Commission 15] European Commission: Official languages of the EU. Online, Nov. 2015. [`http://ec.europa.eu/languages/policy/linguistic-diversity/official-languages-eu_en.htm`; accessed: 2016-01-19].

[Feng & Schmidt+ 11] M. Feng, C. Schmidt, J. Wuebker, S. Peitz, M. Freitag, H. Ney: The RWTH Aachen System for NTCIR-9 PatentMT. In *Ninth NTCIR Workshop Meeting*, pp. 600–605, Tokyo, Japan, Dec. 2011.

[Fletcher & Powell 63] R. Fletcher, M.J.D. Powell: A Rapidly Convergent Descent Method for Minimization. *The Computer Journal*, Vol. 6, No. 2, pp. 163–168, 1963.

[Foster & Kuhn+ 06] G. Foster, R. Kuhn, H. Johnson: Phrasetable Smoothing for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing*, pp. 53–61, Sydney, Australia, July 2006.

[Freitag & Feng+ 13] M. Freitag, M. Feng, M. Huck, S. Peitz, H. Ney: Reverse Word Order Models. In *Machine Translation Summit*, pp. 159–166, Nice, France, Sept. 2013.

[Freitag & Huck+ 14] M. Freitag, M. Huck, H. Ney: Jane: Open Source Machine Translation System Combination. In *Conference of the European Chapter of the Association for Computational Linguistics*, pp. 29–32, Gothenburg, Sweden, April 2014.

[Freitag & Leusch+ 11] M. Freitag, G. Leusch, J. Wuebker, S. Peitz, H. Ney, T. Herrmann, J. Niehues, A. Waibel, A. Allauzen, G. Adda, J.M. Crego, B. Buschbeck, T. Wandmacher, J. Senellart: Joint WMT Submission of the QUAERO Project. In *Workshop on Statistical Machine Translation*, pp. 358–364, Edinburgh, UK, July 2011.

[Freitag & Peitz+ 12] M. Freitag, S. Peitz, M. Huck, H. Ney, T. Herrmann, J. Niehues, A. Waibel, A. Allauzen, G. Adda, B. Buschbeck, J.M. Crego, J. Senellart: Joint WMT 2012 Submission of the QUAERO Project. In *Workshop on Statistical Machine Translation*, pp. 322–329, Montreal, Canada, June 2012.

[Freitag & Peitz+ 13] M. Freitag, S. Peitz, J. Wuebker, H. Ney, N. Durrani, M. Huck, P. Koehn, T.L. Ha, J. Niehues, M. Mediani, T. Herrmann, A. Waibel, N. Bertoldi, M. Cettolo, M. Federico: EU-BRIDGE MT: Text Translation of Talks in the EU-BRIDGE Project. In *International Workshop on Spoken Language Translation*, pp. 128–135, Heidelberg, Germany, Dec. 2013.

[Freitag & Peitz+ 14] M. Freitag, S. Peitz, J. Wuebker, H. Ney, M. Huck, R. Sennrich, N. Durrani, M. Nadejde, P. Williams, P. Koehn, T. Herrmann, E. Cho, A. Waibel: EU-BRIDGE MT: Combined Machine Translation. In *Workshop on Statistical Machine Translation*, pp. 105–113, Baltimore, MD, USA, June 2014.

[Freitag & Peter+ 15] M. Freitag, J.T. Peter, S. Peitz, M. Feng, H. Ney: Local System Voting Feature for Machine Translation System Combination. In *Workshop on Statistical Machine Translation*, pp. 467–476, Lisboa, Portugal, Sept. 2015.

[Freitag & Wuebker$^+$ 14] M. Freitag, J. Wuebker, S. Peitz, H. Ney, M. Huck, A. Birch, N. Durrani, P. Koehn, M. Mediani, I. Slawik, J. Niehues, E. Cho, A. Waibel, N. Bertoldi, M. Cettolo, M. Federico: Combined Spoken Language Translation. In *International Workshop on Spoken Language Translation*, pp. 57–64, Lake Tahoe, CA, USA, Dec. 2014.

[Galley & Manning 08] M. Galley, C.D. Manning: A Simple and Effective Hierarchical Phrase Reordering Model. In *Conference on Empirical Methods in Natural Language Processing*, pp. 848–856, Waikiki, HI, USA, Oct. 2008.

[Hassan & Ma$^+$ 07] H. Hassan, Y. Ma, A. Way: MaTrEx: the DCU Machine Translation System for IWSLT 2007. In *International Workshop on Spoken Language Translation*, pp. 69–75, Trento, Italy, Oct. 2007.

[He & Deng 12] X. He, L. Deng: Maximum Expected BLEU Training of Phrase and Lexicon Translation Models. In *Annual Meeting of the Association for Computational Linguistics*, pp. 292–301, Jeju, Republic of Korea, July 2012.

[Heafield 11] K. Heafield: KenLM: Faster and Smaller Language Model Queries. In *Conference on Empirical Methods in Natural Language Processing*, pp. 187–197, Edinburgh, Scotland, United Kingdom, July 2011.

[Heafield 13] K. Heafield: *Efficient Language Modeling Algorithms with Applications to Statistical Machine Translation.* Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA, Sept. 2013.

[Heafield & Hoang$^+$ 11] K. Heafield, H. Hoang, T. Kiso, M. Federico: Left Language Model State for Syntactic Machine Translation. In *International Workshop on Spoken Language Translation*, pp. 183–190, San Francisco, CA, USA, Dec. 2011.

[Heger & Wuebker$^+$ 10] C. Heger, J. Wuebker, D. Vilar, H. Ney: A Combination of Hierarchical Systems with Forced Alignments from Phrase-Based Systems. In *International Workshop on Spoken Language Translation*, pp. 291–297, Paris, France, Dec. 2010.

[Hoang 15] H. Hoang: Moses Version 3.0 Release Notes. Online, Jan. 2015. [`http://www.statmt.org/mosescore/uploads/Internal/D1.4_Moses_v3_Release_Notes.pdf`; accessed: 2016-02-17].

[Hoang & Lopez 09] H. Hoang, A. Lopez: A Unified Framework for Phrase-Based, Hierarchical, and Syntax-Based Statistical Bachine Translation. In *International Workshop on Spoken Language Translation*, pp. 152–159, Tokyo, Japan, Dec. 2009.

[Hoffmeister 11] B. Hoffmeister: *Bayes Risk Decoding and its Application to System Combination.* Ph.D. thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, July 2011.

[Hopkins & May 11] M. Hopkins, J. May: Tuning As Ranking. In *Conference on Empirical Methods in Natural Language Processing*, pp. 1352–1362, Edinburgh, UK, July 2011.

[Huang & Chiang 05] L. Huang, D. Chiang: Better K-best Parsing. In *International Workshop on Parsing Technology*, pp. 53–64, Vancouver, BC, Canada, Oct. 2005.

[Huang & Zhou 09] S. Huang, B. Zhou: An EM Algorithm for SCFG in Formal Syntax-Based Translation. In *International Conference on Acoustics, Speech and Signal Processing*, pp. 4813–4816, Taipei, Taiwan, April 2009.

[Huck & Mansour+ 11] M. Huck, S. Mansour, S. Wiesler, H. Ney: Lexicon Models for Hierarchical Phrase-Based Machine Translation. In *International Workshop on Spoken Language Translation*, pp. 191–198, San Francisco, CA, USA, Dec. 2011.

[Huck & Peitz+ 12a] M. Huck, S. Peitz, M. Freitag, H. Ney: Discriminative Reordering Extensions for Hierarchical Phrase-Based Machine Translation. In *Annual Conference of the European Association for Machine Translation*, pp. 313–320, Trento, Italy, May 2012.

[Huck & Peitz+ 12b] M. Huck, S. Peitz, M. Freitag, M. Nuhn, H. Ney: The RWTH Aachen Machine Translation System for WMT 2012. In *Workshop on Statistical Machine Translation*, pp. 304–311, Montreal, Canada, June 2012.

[Huck & Peter+ 12] M. Huck, J.T. Peter, M. Freitag, S. Peitz, H. Ney: Hierarchical Phrase-Based Translation with Jane 2. *The Prague Bulletin of Mathematical Linguistics*, Vol. 98, pp. 37–50, Oct. 2012.

[Huck & Vilar+ 13] M. Huck, D. Vilar, M. Freitag, H. Ney: A Performance Study of Cube Pruning for Large-Scale Hierarchical Machine Translation. In *Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 29–38, Atlanta, GA, USA, June 2013.

[Huck & Wuebker+ 11] M. Huck, J. Wuebker, C. Schmidt, M. Freitag, S. Peitz, D. Stein, A. Dagnelies, S. Mansour, G. Leusch, H. Ney: The RWTH Aachen Machine Translation System for WMT 2011. In *Workshop on Statistical Machine Translation*, pp. 405–412, Edinburgh, UK, July 2011.

[Huck & Wuebker+ 13] M. Huck, J. Wuebker, F. Rietig, H. Ney: A Phrase Orientation Model for Hierarchical Machine Translation. In *Workshop on Statistical Machine Translation*, pp. 452–463, Sofia, Bulgaria, Aug. 2013.

[IBM 54] IBM: 701 Translator. Press release, Jan. 1954.

[Kasami 65] T. Kasami: An Efficient Recognition and Syntax Algorithm for Context-free Languages. Technical Report AFCLR-65-758, Air Force Cambridge Research Laboratory, Bedford, MA, USA, 1965.

[Klein & Manning 03] D. Klein, C.D. Manning: Accurate Unlexicalized Parsing. In *Annual Meeting of the Association for Computational Linguistics*, pp. 423–430, Sapporo, Japan, July 2003.

[Kneser & Ney 95] R. Kneser, H. Ney: Improved Backing-Off for M-Gram Language Modeling. In *International Conference on Acoustics, Speech, and Signal Processing*, Vol. 1, pp. 181–184, Detroit, MI, USA, May 1995.

[Knight 99] K. Knight: Decoding Complexity in Word-Replacement Translation Models. *Computational Linguistics*, Vol. 25, No. 4, pp. 607–615, Dec. 1999.

[Koehn & Hoang+ 07] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, E. Herbst: Moses: Open Source Toolkit for Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics*, pp. 177–180, Prague, Czech Republic, June 2007.

[Koehn & Knight 03] P. Koehn, K. Knight: Empirical Methods for Compound Splitting. In *Conference of the European Chapter of the Association for Computational Linguistics*, pp. 187–193, Budapest, Hungary, April 2003.

[Koehn & Och+ 03] P. Koehn, F.J. Och, D. Marcu: Statistical Phrase-Based Translation. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 54–60, Edmonton, Canada, May 2003.

[Kurihara & Sato 04] K. Kurihara, T. Sato: An Application of the Variational Bayesian Approach to Probabilistic Contexti-Free Grammars. In *Workshop Beyond Shallow Analyses*, pp. 1–5, Hainan, China, March 2004.

[Lehnen & Peter+ 13] P. Lehnen, J.T. Peter, J. Wuebker, S. Peitz, H. Ney: (Hidden) Conditional Random Fields Using Intermediate Classes for Statistical Machine Translation. In *Machine Translation Summit*, pp. 151–158, Nice, France, Sept. 2013.

[Lewis & Simons+ 15] M.P. Lewis, G.F. Simons, C.D. Fennig: *Ethnologue: Languages of the World*, Vol. 18. SIL International, Dallas, TX, USA, 2015.

[Lewis & Stearns 68] P.M. Lewis, II, R.E. Stearns: Syntax-Directed Transduction. *Journal of the Association for Computing Machinery*, Vol. 15, No. 3, pp. 465–488, July 1968.

[Li & Callison-Burch+ 09] Z. Li, C. Callison-Burch, C. Dyer, J. Ganitkevitch, S. Khudanpur, L. Schwartz, W.N.G. Thornton, J. Weese, O.F. Zaidan: Joshua: An Open Source Toolkit for Parsing-based Machine Translation. In *Workshop on Statistical Machine Translation*, pp. 135–139, Athens, Greece, March 2009.

[Lu 14] Y. Lu: Syntactic Smoothing of Hierarchical Translation Models. Master's thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, Oct. 2014.

[Lu & Ng 10] W. Lu, H.T. Ng: Better Punctuation Prediction with Dynamic Conditional Random Fields. In *Conference on Empirical Methods in Natural Language Processing*, pp. 177–186, Cambridge, MA, USA, Oct. 2010.

[Ma & Tinsley+ 08] Y. Ma, J. Tinsley, H. Hassan, J. Du, A. Way: Exploiting Alignment Techniques in MaTrEx: the DCU Machine Translation System for IWSLT08. In *International Workshop on Spoken Language Translation*, pp. 26–33, Waikiki, HI, USA, Oct. 2008.

[Mansour & Peitz+ 10] S. Mansour, S. Peitz, D. Vilar, J. Wuebker, H. Ney: The RWTH Aachen Machine Translation system for IWSLT 2010. In *International Workshop on Spoken Language Translation*, pp. 163–168, Paris, France, Dec. 2010.

[Marcu & Wong 02] D. Marcu, W. Wong: A Phrase-Based, Joint Probability Model for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing*, pp. 133–139, Philadelphia, PA, USA, July 2002.

[Matusov & Hoffmeister+ 08] E. Matusov, B. Hoffmeister, H. Ney: ASR Word Lattice Translation with Exhaustive Reordering is Possible. In *Interspeech*, pp. 2342–2345, Brisbane, Australia, Sept. 2008.

[Matusov & Mauser+ 06] E. Matusov, A. Mauser, H. Ney: Automatic Sentence Segmentation and Punctuation Prediction for Spoken Language Translation. In *International Workshop on Spoken Language Translation*, pp. 158–165, Kyoto, Japan, Nov. 2006.

[Matusov & Ney 11] E. Matusov, H. Ney: Lattice-Based ASR-MT Interface for Speech Translation. *IEEE Transactions on Audio, Speech & Language Processing*, Vol. 19, No. 4, pp. 721–732, 2011.

[Mauser & Vilar+ 07] A. Mauser, D. Vilar, G. Leusch, Y. Zhang, H. Ney: The RWTH Machine Translation System for IWSLT 2007. In *International Workshop on Spoken Language Translation*, pp. 161–168, Trento, Italy, Oct. 2007.

[Moore & Lewis 10] R.C. Moore, W. Lewis: Intelligent Selection of Language Model Training Data. In *Annual Meeting of the Association for Computational Linguistics*, pp. 220–224, Uppsala, Sweden, July 2010.

[Moschitti 06] A. Moschitti: Making Tree Kernels Practical for Natural Language Learning. In *Conference of the European Chapter of the Association for Computational Linguistics*, pp. 113–120, Trento, Italy, April 2006.

[Och 99] F.J. Och: An Efficient Method for Determining Bilingual Word Classes. In *Conference of the European Chapter of the Association for Computational Linguistics*, pp. 71–76, Bergen, Norway, June 1999.

[Och 03] F.J. Och: Minimum Error Rate Training in Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics*, pp. 160–167, Sapporo, Japan, July 2003.

[Och & Ney 02] F.J. Och, H. Ney: Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics*, pp. 295–302, Philadelphia, PA, USA, July 2002.

[Och & Ney 03] F.J. Och, H. Ney: A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, Vol. 29, No. 1, pp. 19–51, March 2003.

[Och & Tillmann+ 99] F.J. Och, C. Tillmann, H. Ney: Improved Alignment Models for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing*, pp. 20–28, College Park, MD, USA, June 1999.

[Papineni & Roukos+ 02] K. Papineni, S. Roukos, T. Ward, W.J. Zhu: BLEU: A Method for Automatic Evaluation of Machine Translation. In *Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, PA, USA, July 2002.

[Parra Escartín & Peitz+ 14] C. Parra Escartín, S. Peitz, H. Ney: German Compounds and Statistical Machine Translation. Can they get along? In *Workshop on Multiword Expressions*, pp. 48–56, Gothenburg, Sweden, April 2014.

[Patry & Langlais 08] A. Patry, P. Langlais: MISTRAL: A Statistical Machine Translation Decoder for Speech Recognition Lattices. In *International Conference on Language Resources and Evaluation*, pp. 1148–1153, Marrakech, Morocco, May 2008.

[Peitz 10] S. Peitz: Extending Statistical Machine Translation Using Syntax. Diploma thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, March 2010.

[Peitz & Freitag+ 11] S. Peitz, M. Freitag, A. Mauser, H. Ney: Modeling Punctuation Prediction as Machine Translation. In *International Workshop on Spoken Language Translation*, pp. 238–245, San Francisco, CA, USA, Dec. 2011.

[Peitz & Freitag+ 14] S. Peitz, M. Freitag, H. Ney: Better Punctuation Prediction with Hierarchical Phrase-Based Translation. In *International Workshop on Spoken Language Translation*, pp. 271–278, Lake Tahoe, CA, USA, Dec. 2014.

[Peitz & Mansour+ 12] S. Peitz, S. Mansour, M. Freitag, M. Feng, M. Huck, J. Wuebker, M. Nuhn, M. Nußbaum-Thom, H. Ney: The RWTH Aachen Speech Recognition and Machine Translation System for IWSLT 2012. In *International Workshop on Spoken Language Translation*, pp. 69–76, Hong Kong, Dec. 2012.

[Peitz & Mansour+ 13a] S. Peitz, S. Mansour, M. Huck, M. Freitag, H. Ney, E. Cho, T. Herrmann, M. Mediani, J. Niehues, A. Waibel, A. Allauzen, Q.K. Do, B. Buschbeck, T. Wandmacher: Joint WMT 2013 Submission of the QUAERO Project. In *Workshop on Statistical Machine Translation*, pp. 185–192, Sofia, Bulgaria, Aug. 2013.

[Peitz & Mansour+ 13b] S. Peitz, S. Mansour, J.T. Peter, C. Schmidt, J. Wuebker, M. Huck, M. Freitag, H. Ney: The RWTH Aachen Machine Translation System for WMT 2013. In *Workshop on Statistical Machine Translation*, pp. 193–199, Sofia, Bulgaria, Aug. 2013.

[Peitz & Mauser+ 12] S. Peitz, A. Mauser, J. Wuebker, H. Ney: Forced Derivations for Hierarchical Machine Translation. In *International Conference on Computational Linguistics*, pp. 933–942, Mumbai, India, Dec. 2012.

[Peitz & Vilar+ 14] S. Peitz, D. Vilar, H. Ney: Simple and Effective Approach for Consistent Training of Hierarchical Phrase-based Translation Models. In *Conference of the European Chapter of the Association for Computational Linguistics*, pp. 174–179, Gothenburg, Sweden, April 2014.

[Peitz & Wiesler+ 12] S. Peitz, S. Wiesler, M. Nussbaum-Thom, H. Ney: Spoken Language Translation Using Automatically Transcribed Text in Training. In *International Workshop on Spoken Language Translation*, pp. 276–283, Hong Kong, Dec. 2012.

[Peitz & Wuebker+ 14] S. Peitz, J. Wuebker, M. Freitag, H. Ney: The RWTH Aachen German-English Machine Translation System for WMT 2014. In *Workshop on Statistical Machine Translation*, pp. 157–162, Baltimore, MD, USA, June 2014.

[Peter & Toutounchi+ 15] J.T. Peter, F. Toutounchi, S. Peitz, P. Bahar, A. Guta, H. Ney: The RWTH Aachen German to English MT System for IWSLT 2015. In *International Workshop on Spoken Language Translation*, pp. 15–22, Da Nang, Vietnam, Dec. 2015.

[Petrov & Barrett+ 06] S. Petrov, L. Barrett, R. Thibaux, D. Klein: Learning Accurate, Compact, and Interpretable Tree Annotation. In *Annual Meeting of the Association for Computational Linguistics*, pp. 433–440, Sydney, Australia, July 2006.

[Petrov & Klein 07] S. Petrov, D. Klein: Improved Inference for Unlexicalized Parsing. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 404–411, Rochester, NY, USA, April 2007.

[Popović & Ney 06] M. Popović, H. Ney: POS-based Word Reorderings for Statistical Machine Translation. In *International Conference on Language Resources and Evaluation*, pp. 1278–1283, Genoa, Italy, May 2006.

[Rafferty & Manning 08] A.N. Rafferty, C.D. Manning: Parsing Three German Treebanks: Lexicalized and Unlexicalized Baselines. In *Workshop on Parsing German*, pp. 40–46, Columbus, OH, USA, June 2008.

[Rosti & Zhang+ 11] A.V.I. Rosti, B. Zhang, S. Matsoukas, R. Schwartz: Expected BLEU Training for Graphs: BBN System Description for WMT11 System Combination Task. In *Workshop on Statistical Machine Translation*, pp. 159–165, Edinburgh, UK, July 2011.

[Sennrich 14] R. Sennrich: A CYK+ Variant for SCFG Decoding Without a Dot Chart. In *Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 94–102, Doha, Qatar, Oct. 2014.

[Shaik & Tüske+ 13] M.A.B. Shaik, Z. Tüske, S. Wiesler, M. Nussbaum-Thom, S. Peitz, R. Schlüter, H. Ney: The RWTH Aachen German and English LVCSR Systems for IWSLT-2013. In *International Workshop on Spoken Language Translation*, pp. 120–127, Heidelberg, Germany, Dec. 2013.

[Snover & Dorr+ 06] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, J. Makhoul: A Study of Translation Edit Rate with Targeted Human Annotation. In *Association for Machine Translation in the Americas*, pp. 223–231, Cambridge, MA, USA, Aug. 2006.

[Stein 12] D. Stein: *Soft Features for Statistical Machine Translation of Spoken and Signed Languages*. Ph.D. thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, Jan. 2012.

[Stein & Peitz+ 10] D. Stein, S. Peitz, D. Vilar, H. Ney: A Cocktail of Deep Syntactic Features for Hierarchical Machine Translation. In *Association for Machine Translation in the Americas*, Denver, CO, USA, Oct. 2010.

[Stein & Vilar+ 11] D. Stein, D. Vilar, S. Peitz, M. Freitag, M. Huck, H. Ney: A Guide to Jane, an Open Source Hierarchical Translation Toolkit. *The Prague Bulletin of Mathematical Linguistics*, Vol. 95, pp. 5–18, April 2011.

[Stolcke 02] A. Stolcke: SRILM - An Extensible Language Modeling Toolkit. In *International Conference on Spoken Language Processing*, pp. 901–904, Denver, CO, USA, Sept. 2002.

[Sundermeyer & Alkhouli+ 14] M. Sundermeyer, T. Alkhouli, J. Wuebker, H. Ney: Translation Modeling with Bidirectional Recurrent Neural Networks. In *Conference on Empirical Methods in Natural Language Processing*, pp. 14–25, Doha, Qatar, Oct. 2014.

[Sundermeyer & Ney+ 15] M. Sundermeyer, H. Ney, R. Schlüter: From Feedforward to Recurrent LSTM Neural Networks for Language Modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 23, No. 3, pp. 517–529, March 2015.

[Tillmann 04] C. Tillmann: A Unigram Orientation Model for Statistical Machine Translation. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 101–104, Boston, MA, USA, May 2004.

[Tromble & Kumar+ 08] R. Tromble, S. Kumar, F. Och, W. Macherey: Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing*, pp. 620–629, Waikiki, HI, USA, Oct. 2008.

[Van Rijsbergen 79] C.J. Van Rijsbergen: *Information Retrieval*. Butterworths, London, UK, 2nd edition, 1979.

[Vilar 11] D. Vilar: *Investigations on Hierarchical Phrase-Based Machine Translation*. Ph.D. thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, Nov. 2011.

[Vilar & Stein+ 10a] D. Vilar, D. Stein, M. Huck, H. Ney: Jane: Open Source Hierarchical Translation, Extended with Reordering and Lexicon Models. In *Workshop on Statistical Machine Translation*, pp. 262–270, Uppsala, Sweden, July 2010.

[Vilar & Stein+ 10b] D. Vilar, D. Stein, S. Peitz, H. Ney: If I Only Had a Parser: Poor Man's Syntax for Hierarchical Machine Translation. In *International Workshop on Spoken Language Translation*, pp. 345–352, Paris, France, Dec. 2010.

[Vogel & Ney+ 96] S. Vogel, H. Ney, C. Tillmann: HMM-based Word Alignment in Statistical Translation. In *International Conference on Computational Linguistics*, pp. 836–841, Copenhagen, Denmark, Aug. 1996.

[Williams & Koehn 12] P. Williams, P. Koehn: GHKM Rule Extraction and Scope-3 Parsing in Moses. In *Workshop on Statistical Machine Translation*, pp. 388–394, Montreal, Canada, 2012.

[Wuebker & Huck+ 11] J. Wuebker, M. Huck, S. Mansour, M. Freitag, M. Feng, S. Peitz, C. Schmidt, H. Ney: The RWTH Aachen Machine Translation System for IWSLT 2011. In *International Workshop on Spoken Language Translation*, pp. 106–113, San Francisco, CA, USA, Dec. 2011.

[Wuebker & Huck+ 12] J. Wuebker, M. Huck, S. Peitz, M. Nuhn, M. Freitag, J.T. Peter, S. Mansour, H. Ney: Jane 2: Open Source Phrase-based and Hierarchical Statistical Machine Translation. In *International Conference on Computational Linguistics*, pp. 483–491, Mumbai, India, Dec. 2012.

[Wuebker & Mauser+ 10] J. Wuebker, A. Mauser, H. Ney: Training Phrase Translation Models with Leaving-One-Out. In *Annual Meeting of the Association for Computational Linguistics*, pp. 475–484, Uppsala, Sweden, July 2010.

[Wuebker & Muehr+ 15] J. Wuebker, S. Muehr, P. Lehnen, S. Peitz, H. Ney: A Comparison of Update Strategies for Large-Scale Maximum Expected BLEU Training. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 1516–1526, Denver, CO, USA, May 2015.

[Wuebker & Peitz+ 13a] J. Wuebker, S. Peitz, T. Alkhouli, J.T. Peter, M. Feng, M. Freitag, H. Ney: The RWTH Aachen Machine Translation Systems for IWSLT 2013. In *International Workshop on Spoken Language Translation*, pp. 88–93, Heidelberg, Germany, Dec. 2013.

[Wuebker & Peitz+ 13b] J. Wuebker, S. Peitz, F. Rietig, H. Ney: Improving Statistical Machine Translation with Word Class Models. In *Conference on Empirical Methods in Natural Language Processing*, pp. 1377–1381, Seattle, WA, USA, Oct. 2013.

[Wuebker & Peitz+ 14] J. Wuebker, S. Peitz, A. Guta, H. Ney: The RWTH Aachen Machine Translation Systems for IWSLT 2014. In *International Workshop on Spoken Language Translation*, pp. 150–154, Lake Tahoe, CA, USA, Dec. 2014.

[Younger 67] D.H. Younger: Recognition and Parsing of Context-Free Languages in Time n^3. *Information and Control*, Vol. 10, No. 2, pp. 189–208, Feb. 1967.

[Zens 08] R. Zens: *Phrase-based Statistical Machine Translation: Models, Search, Training.* Ph.D. thesis, Computer Science Department, RWTH Aachen University, Aachen, Germany, Feb. 2008.

[Zens & Och+ 02] R. Zens, F.J. Och, H. Ney: Phrase-Based Statistical Machine Translation. In *German Conference on Artificial Intelligence*, pp. 18–32, Aachen, Germany, Sept. 2002.

[Zhou & Zhu+ 08] B. Zhou, X. Zhu, B. Xiang, Y. Gao: Prior Derivation Models for Formally Syntax-Based Translation Using Linguistically Syntactic Parsing and Tree Kernels. In *Workshop*

on Syntax, Semantics and Structure in Statistical Translation, pp. 19–27, Columbus, OH, USA, June 2008.

[Zollmann & Venugopal 06] A. Zollmann, A. Venugopal: Syntax Augmented Machine Translation via Chart Parsing. In *Workshop on Statistical Machine Translation*, pp. 138–141, New York City, NY, USA, June 2006.

B. Curriculum Vitæ

Angaben zur Person
Stephan Peitz
geboren am 7. September 1985 in Wuppertal
Staatsangehörigkeit: deutsch

Schulbildung

1990–1994	GGS Morsbroich
1994–2001	Freiherr-vom-Stein-Gymnasium
2001–2005	Landrat-Lucas-Gymnasium

Studium

2005–2010	Studium der Informatik an der RWTH Aachen University, Nebenfach Medizin Abschluss: Diplom-Informatiker
2010–2016	Promotionsstudium an der RWTH Aachen University

Berufslaufbahn

2009–2010	Studentische Hilfskraft am Lehrstuhl für Mustererkennung und Sprachverarbeitung (Informatik 6) an der RWTH Aachen University
2010–2016	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Mustererkennung und Sprachverarbeitung (Informatik 6) an der RWTH Aachen University
Mai–Juli 2015	Praktikum bei Apple Inc, Cupertino, CA, USA
April 2016–Juni 2017	Software Development Engineer bei Apple GmbH, Deutschland
Seit Juli 2017	Machine Learning Engineer bei Apple Inc, Cupertino, CA, USA