Rule Extraction for Multi Bottom-up Tree Transducers

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Philosophie (Dr. phil.) genehmigte Abhandlung

Vorgelegt von

Nina Seemann

aus Stuttgart

Hauptberichter:	Dr. Andreas Maletti
1. Mitberichter:	Prof. Dr. Jonas Kuhn
2. Mitberichter:	Prof. Dr. Kevin Knight

Tag der mündlichen Prüfung: 24. Oktober 2016

Institut für Maschinelle Sprachverarbeitung
der Universität Stuttgart

2016

Abstract

Following the invention of computers, it was always a dream to obtain translations automatically. If we give a machine a sentence it should return a sentence in another language expressing the same meaning. In the subfield of statistical machine translation (SMT), this translation is achieved with the help of statistical models. Those models use large text collections to automatically learn basic translation units that model the translation from a source sentence into a target sentence. The basic translation units can be single words or phrases consisting of multiple words. Other approaches, called syntax-based SMT, use rules of some formal grammar as their basic translation units.

Syntax-based SMT systems easily allow the use of linguistic annotations. Rules can contain nonterminal symbols which can encode linguistic annotations. Furthermore, one can decide whether such annotations are used for both the source and target language, for one language only, or if those annotations are excluded altogether. The integration of linguistic annotations yielded mixed results. In some cases translation quality significantly improves whereas in others it seems to hurt coverage and thus overall translation quality. While the use of annotations for both languages generally did not result in good translation quality, the use for one language only showed improvements. However, the best results are often obtained by a syntax-based SMT system that excludes all linguistic annotations.

The underlying formal grammars vastly vary with respect to their expressive power. Synchronous context-free grammars are widely used but more powerful for-

malisms like synchronous tree substitution grammars or local multi bottom-up tree transducers (ℓMBOT) have also been proposed for translation. In this thesis, we introduce a translation model that is based on ℓMBOTs. Our work focuses on automatically learning the basic translations units for ℓMBOTs with varying linguistic annotations. We implemented an already existing algorithm that extracts the minimally required basic translation units for an ℓMBOT model with linguistic annotations for both the source and the target language. Furthermore, we present three implementations of algorithms which extract more than the minimally required basic translation units. These algorithms are designed for ℓMBOT models with (1) linguistic annotations on both sides, (2) linguistic annotations for the target language, and (3) no linguistic annotations at all.

For all ℓMBOT models, we present an evaluation in terms of translation quality. In addition, we conduct various analyses that illuminate certain positive aspects of an ℓMBOT system and we explain the impact of these aspects to SMT.

Deutsche Zusammenfassung

Auf die Erfindung des Computers folgte der Traum, Übersetzungen automatisch zu erhalten. Wenn man einer Maschine einen Satz gibt, soll sie einen Satz in einer anderen Sprache ausgeben, der weiterhin die ursprüngliche Bedeutung hat. In der Teildisziplin der statistischen maschinellen Übersetzung (SMÜ) wird die Übersetzung mit Hilfe statistischer Modelle erreicht. Diese Modelle benutzen große Textsammlungen, um automatisch die Übersetzungseinheiten zu lernen, die die Übersetzung eines Quellsatzes in einen Zielsatz modellieren. Die Übersetzungseinheiten können einzelne Wörter sein oder Phrasen, die aus mehreren Wörtern bestehen. Andere Ansätze, syntax-basierte SMÜ genannt, benutzen Regeln einer formalen Grammatik als ihre Übersetzungseinheiten.

Syntax-basierte SMÜ-Systeme erlauben auf einfache Weise das Benutzen von linguistischen Annotationen. Regeln können Nichtterminalsymbole enthalten, welche linguistische Annotationen kodieren können. Linguistische Annotationen können wahlweise für die Quell- und Zielsprache verwendet werden oder nur für eine der beiden Sprachen. Es ist auch möglich, gar keine linguistischen Annotationen zu benutzen. Das Einbinden linguistischer Annotationen führte zu gemischten Ergebnissen. In manchen Fällen wurde die Übersetzungsqualität erheblich verbessert, während in anderen Fällen eine Verschlechterung beobachtet wurde. Das Einbinden von linguistischen Annotationen für Quell- und Zielsprache zeigt generell keine gute Übersetzungsqualität, für das Einbinden für nur eine Sprache hingegen sind Verbesserungen bestätigt worden. Die besten Ergebnisse liefern jedoch meistens syntax-basierte SMÜ-Systeme ohne linguistische Annotationen.

Die zugrundeliegenden formalen Grammatiken variieren stark im Hinblick auf ihre Mächtigkeit. Weitestgehend werden synchrone kontext-freie Grammatiken verwendet, aber auch mächtigere Formalismen wie Baumsubstitutionsgrammatiken oder lokale, aufsteigende Mehrfachbaumübersetzer (ℓMBOT) wurden für die Übersetzung vorgeschlagen. In dieser Arbeit stellen wir ein Übersetzungsmodell basierend auf ℓMBOTs vor. Unsere Arbeit konzentriert sich auf das automatische Lernen der Übersetzungseinheiten für ℓMBOTs mit variierenden linguistischen Annotationen. Wir haben einen zuvor vorgeschlagenen Algorithmus implementiert, der das Minimum an benötigten Übersetzungseinheiten für ein ℓMBOT-Modell mit linguistischen Annotationen für die Quell- und Zielsprache extrahiert. Desweiteren präsentieren wir drei Algorithmen, welche in der Lage sind, sinnvolle zusätzliche Übersetzungseinheiten zu extrahieren. Diese Algorithmen sind auf ℓMBOT-Modelle mit (1) linguistischen Annotationen für Quell- und Zielsprache, (2) linguistischen Annotationen auf der Zielsprache und (3) keinerlei linguistischen Annotationen zugeschnitten.

Für jedes unserer ℓMBOT-Systeme präsentieren wir eine Auswertung bezüglich der Übersetzungsqualität. Zusätzlich stellen wir verschiedene Analysen vor, die bestimmte positive Aspekte eines ℓMBOT-Modells beleuchten und aufzeigen, wie SMÜ von diesen Aspekten profitieren kann.

Acknowledgments

First of all, I would like to thank Andreas Maletti for being the best supervisor a PhD student can ask for. Whenever I got lost on my journey to this thesis, he was there like a beacon showing me the way back home. I also want to thank the members of the doctoral committee – Jonas Kuhn, Kevin Knight, and Ulrich Hertrampf as well as Thang Vu for agreeing to substitute for Kevin.

Next, I want to thank my fantastic colleagues, Fabienne Braune and Daniel Quernheim. Fabienne patiently explained to me the basics of Statistical Machine Translation and how to handle Moses while Daniel likewise patiently explained semirings and weighted automata. I really loved working with both of you!

When working on SMT, it is always nice to talk to people who understand your problems and troubles. So I want to thank the rest of the IMS-SMT group for their support: Fabienne Cap (neé Fritzinger), Marion Di Marco (neé Weller), Anita Ramm (neé Gojun), and Alex Fraser.

Of course, I want to thank each member of the IMS. There are some who need to be named — Kati Schweizer, Arndt Riester, and Torgrim Solstad for allowing me a first glance at scientific work, Andre Blessing for solving server issues and recovering data, Anders Björkelund and Wolfgang Seeker for their insightful comments on dependency parsing, and Sybille Laderer for her support on bureaucratic stuff.

Outside the IMS, I am grateful to have the best friends ever. And a big *Thank you* to the crew and members of the East Side Gym for making me train hard and laugh loud.

Den größten Dank schulde ich meinen Großeltern: Ohne Euch hätte ich niemals diesen Punkt in meinem Leben erreicht! Und natürlich bin ich sehr dankbar für meine Nichten, die mir immer wieder deutlich machen, dass es sehr viel wichtigere Sachen gibt als eine Dissertation. Und Christian danke ich dafür, dass er immer an meiner Seite steht.

The work in this thesis was supported by the DFG grant Tree Transducers in Machine Translation.

Contents

Chapter 1

Introduction

1.1 Statistical Machine Translation

Following the invention of computers, it was always a dream to obtain translations automatically. If we give a machine a sentence it should return a sentence in another language expressing the same meaning. Traditionally, translations are performed by professionals who need to be well-trained for this task. But with the advent of computers, people started to focus on machine translation (MT), i.e. automatically obtained translations. Initially, the computer was only supposed to execute programmed rules to perform translation, which required deep knowledge of both languages and the crafting of hand-written rules.

Over the years, the processing power of computers significantly increased and large text collections in multiple languages became available. Furthermore, some of those texts were available with the same content for different language pairs, nowadays called a *parallel corpus*. It contains two texts where text E consists of the sentences from a source language and the text F consists of the sentences of a target language. A parallel corpus is usually sentence-aligned, i.e. the i-th sentence of E is a translation of the i-th sentence of F and vice versa. The sentences shown in Figure 1.1 (ignoring the splines connecting the words for the moment) form an

example for such a sentence pair $\langle e, f \rangle$ from a parallel corpus. This gave rise to the subfield of statistical MT (SMT) with the aim of achieving machine translation via the application of statistical models. Usually, the process is broken down to the sentence level, so we aim for a sentence-by-sentence translation. We start with the acquisition of *basic translation units* which breaks down the process even further. Over time, various models have been proposed that differ with respect to their *basic translation units*. The first SMT systems were the word-based models of Brown et al. (1990, 1993). Their basic translation units are *words* and the outputs are word-by-word translations. The underlying algorithm learns from the sentence pairs $\langle e, f \rangle$ of the parallel corpus which words in e and f are most likely translations of each other. We display an example translation in Figure 1.1. The links between the words indicate that, for example, 'concludes' generated 'sind' and 'beendet', and 'explanations' translates into 'Erklärungen'.

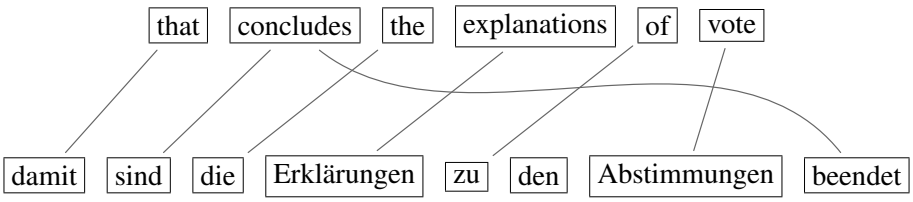

Figure 1.1: An example for an English-to-German word-based translation.

From another point of view, we can say that a word-based model generates through its translation step an alignment between words. Thus, if we translate a sentence pair $\langle e, f \rangle$ of a parallel corpus, we obtain an additional *word-alignment A*. Hence we obtain a *word-aligned parallel corpus* with sentence pairs $\langle e, A, f \rangle$. In fact, the translation shown in Figure 1.1 constitutes a word-aligned sentence pair. The phrase-based translation systems (Koehn et al., 2003) are based on word-aligned sentence pairs. In these systems, the basic translation units are *phrases*. Those phrases can consist of a (not necessarily linguistic) sequence of words. Furthermore, those systems are enhanced with distortion and reordering models which are useful for local reorderings. Phrase-based systems often yield translations that are more fluent when compared to those of word-based systems.

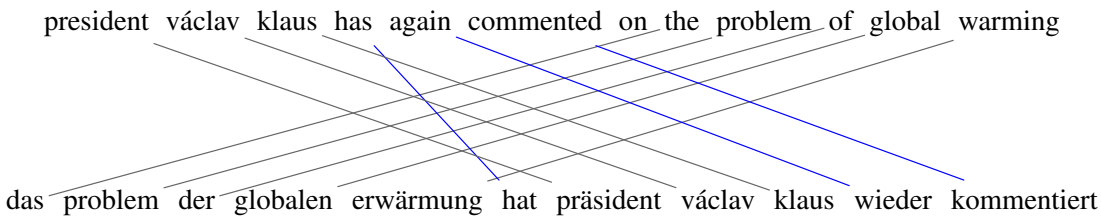

Figure 1.2: An example for a long distance reordering.

Certain language pairs mandate rather complicated restructuring of the sentence during translation. Consider the sentences shown in Figure 1.2 where large parts of the English sentence need to be reordered in the German sentence. For example, the German counterpart of 'on the problem of global warming' is realized at the beginning of the German sentence. Another interesting point is the part 'has again commented'. The German translation is 'hat wieder kommentiert' but the German grammar requires to first realize 'hat', then the translation of 'president václav klaus', and finally, 'wieder kommentiert'. But reorderings over a long distance are hard to obtain for phrase-based systems. Thus, researchers started focusing on formalisms that can model these kinds of reorderings in a more natural way. This resulted in *syntax-based models*, where the translation is modeled recursively. Here, the basic translation units are called *rules*. Rules are based on some grammar formalism, and the formalism allows to encode long distance reorderings inside these rules. Quite a lot of grammar formalisms with varying expressive power have been proposed. One of the most successful approaches is the *hierarchical phrase-based system* (Chiang, 2007), which is formally based on synchronous context-free grammars (SCFG) (Aho and Ullman, 1969) using the simple nonterminal X to mark places, where rules can be applied recursively. We illustrate some SCFG rules in Figure 1.3. Rule r_1 models the translation of 'concludes' into 'sind [...] beendet' by letting the sequence [...] be filled with some other material. A translation given these rules can be obtained by starting with r_1 in which we synchronously replace X_1 with r_2. Next, we can synchronously substitute X_2 with r_3.

Other authors suggest to include linguistic annotations for the source and the

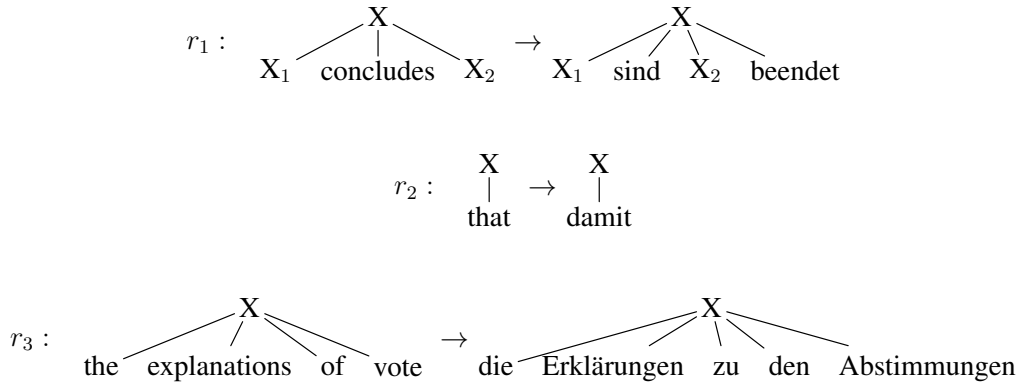

Figure 1.3: Some example SCFG rules.

target sentences which results in *tree-to-tree models*. These models require a constituent parser being applied to both languages. A *constituency parser* generates a syntax tree for a sentence. It starts with the identification of the part-of-speech tag for each word and then recursively builds syntactic categories over certain words until the whole sentence is annotated. We show an example of a word-aligned sentence pair with English and German parse trees in Figure 1.4. Each word is annotated by its part-of-speech tag ('concludes' is an 'VBZ', 'vote' is an 'NN') and over those the syntactic categories are spanned, like 'DT NNS' is a nominal phrase 'NP' or 'APPR ART NN' constitutes a prepositional phrase 'PP'. The complete sentence is assigned the category 'S'. The syntactic categories (also called *labels*) are then used as the nonterminals inside the SCFG rules instead of the simple nonterminal X. But this did not lead to any improvements, contrary it showed that those systems are too restrictive to obtain translations of good quality (Lavie et al., 2008; Ambati and Lavie, 2008).

The restrictions of SCFG-based tree-to-tree models can be weakened by using syntactic annotations only on one side of the rules (*string-to-tree models* and *tree-to-string models*). Another possibility is to use a more powerful formalism than SCFG. Most formalisms (including SCFGs) allow only for continuous translation. A formalism that models discontinuous translation is the synchronous tree sequence substitution grammar (STSSG) of Zhang et al. (2008) and Sun et al. (2009) that allows

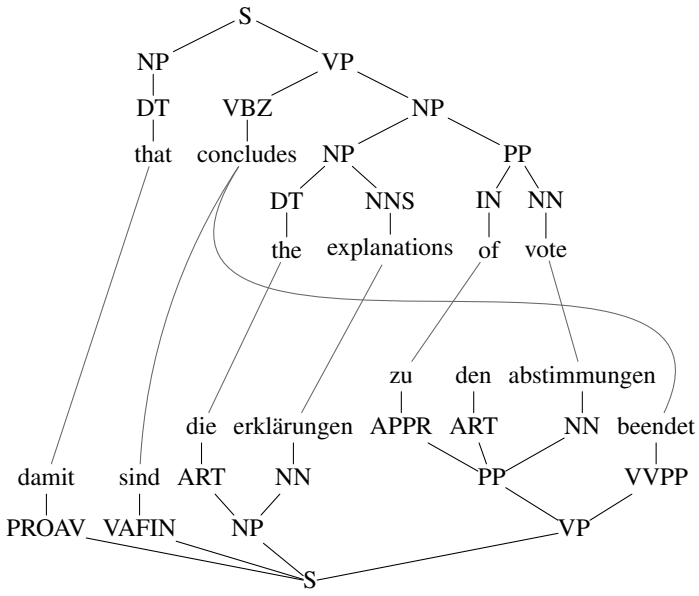

Figure 1.4: Sentence pair with both-sided parses and word-alignment.

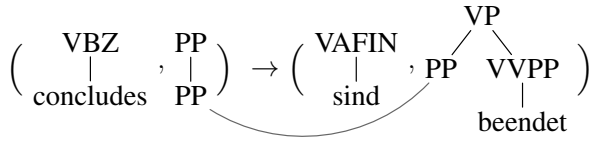

Figure 1.5: An example STSSG rule.

a *sequence of trees*. We show an example rule in Figure 1.5, where we have a sequence of two trees for both the English side and the German side of the rule. The most important observation is that this model allows for discontinuities by having a gap between the trees. Comparing the English tree sequence with the English parse of Figure 1.4 it is obvious that one part of the tree – (NP (DT the) (NNS explanations)) – is not given in this rule. Accordingly, the German counterpart – (PP (APPR zu) (ART den) (NN Abstimmungen)) – is also not given in this rule. By allowing discontinuities, the rules become more flexible and can capture certain linguistic phenomena more elegantly. For a continuous translation system, a similar translation can only be obtained with a large and specific rule. Contrary, the STSSG rule of Figure 1.5 can achieve this translation with a smaller rule which furthermore allows for a flexible application as it is not in a specific context.

This thesis contributes to the field of syntax-based statistical machine translation. The underlying formalism is the *shallow local multi bottom-up tree transducer* (ℓMBOT) of Braune et al. (2013) which in turn is a variant of the *local multi bottom-up tree transducer* of Maletti (2011). An ℓMBOT offers a middle ground between SCFGs and STSSGs. On the one hand, it allows only *shallow rules* similar to SCFG rules and in contrast to STSSG rules. But on the other hand, it offers *discontinuities* by allowing a sequence of trees inside its rules for the target language side, which can be seen as a restricted version of STSSG rules and an extension of SCFG rules. This work mainly focuses on the acquisition of the basic translation units for ℓMBOTs in different settings.

Tree-to-Tree ℓMBOT models We implemented the rule extraction algorithm of Maletti (2011) that obtains a set of *minimal tree-to-tree ℓMBOT rules*. We evaluated our translation system on a English-to-German translation task gaining significant improvements over a tree-to-tree SCFG baseline.

It is known that minimal rules are in general not well suited for machine translation. Hence, we presented and implemented the first parameterized *non-minimal tree-to-tree ℓMBOT rule extraction algorithm*. An evaluation on three different translation tasks shows that the non-minimal tree-to-tree ℓMBOT systems outperform the minimal tree-to-tree ℓMBOT system in terms of translation quality, but they do not improve over a tree-to-tree SCFG system.

String-to-Tree ℓMBOT models Our evaluation of the tree-to-tree ℓMBOT systems confirms that tree-to-tree models are too restrictive in comparison to models including no linguistic annotation. As string-to-tree models yield better translation quality in general, we presented and implemented a *non-minimal string-to-tree ℓMBOT rule extraction algorithm*. Our evaluation on three translation tasks shows significant improvements over the respective string-to-tree SCFG baselines.

Having obtained such significant improvements, we wanted to evaluate another

two language tasks. But for those target languages (Russian and Polish), only dependency parsers are available which might return non-projective dependency parse trees. Consequently, we apply a (non-projective) dependency parser on our target languages and applied a lifting technique that not only projectivizes the non-projective parses but also documents the lifts in the labels. Next, we transformed the dependency tree structures into constituent-like tree structures. Then we were able to obtain our string-to-tree ℓMBOT systems. Our evaluation confirms that translation systems based on ℓMBOTs are quite suitable for such translation tasks.

String-to-String ℓMBOT models The last logical step is to move to a setting similar to the hierarchical phrase-based system. We also presented and implemented a parameterized rule extraction algorithm to obtain a set of (non-minimal) string-to-string ℓMBOT rules. We evaluated our translation system on two translation tasks and showed that our ℓMBOT achieves comparable results to the hierarchical SCFG baseline system.

Analysis of Discontinuity For all three settings and all translations tasks, we will present an analysis of the rules used in the evaluations. We will explain which kinds of discontinuous rules exist and how the use of those rules changes across the different settings. Furthermore, we try to estimate their impact on the translation quality.

Large-Scale Experiments without Restrictions All our performed experiments can be considered as large-scale. The sizes of our used parallel corpora are suitable for well-known shared tasks like WMT. We do not heavily restrict the number of words per sentence of our parallel corpora, and we do not restrict the test sets at all. Furthermore, to the best of our knowledge, this is the first discontinuous translation system which does not restrict the number of discontinuities per rule.

Kaeshammer (2015) limits the parallel corpus and the test set to 30 words per sentence. Zhang et al. (2008) and Sun et al. (2009) use a very small parallel corpus and additionally limit the test set to 50 characters per sentence. We will explain their work more thoroughly in Section 2.2.

1.2 Outline of the Thesis

In Chapter 2 we start with the introduction of the underlying mechanisms of statistical machine translation. We will explain which data we need and we will give a proper introduction to synchronous context-free grammars. Furthermore, we present the three major processes of statistical machine translation — *training, tuning,* and *evaluation*. In addition, we will introduce necessary components like the language model and the decoder, and explain the traditional syntax-based translation model. We conclude this chapter by giving an overview of related work.

We will present the theoretical background of this thesis, namely shallow local multi bottom-up transducers (ℓMBOTs) in Chapter 3. As these kinds of transducers are in general build on trees, we will give a formal definition for such structures. Next, we will introduce the already existing decoder for ℓMBOTs and present the translation model for this system. Furthermore, we will repeat the formal rule extraction algorithm for such transducers and show which kinds of rules are extracted. This work presents the first implementation of the algorithm and, hence, its first evaluation. Our experiment shows a significant improvement over a baseline system. Additionally, we will analyze the translations to prove that ℓMBOT rules are useful.

In Chapter 4 we will first motivate our new approach on ℓMBOTs. We will still be exploring the tree-to-tree setting but with non-minimal rules. We will present the first parameterized rule extraction for non-minimal tree-to-tree rules and depict a pseudo-algorithm. Naturally, we also evaluate this setting on three different translation tasks. More precisely, we evaluate against the minimal ℓMBOT of Chap-

ter 3 as well as against a baseline system. As before, we will give an analysis of the translations to provide insight of the usefulness of such non-minimal rules.

Next, we will explore the ℓMBOT in the string-to-tree setting in Chapter 5. This new setting makes it necessary to adapt our theoretical model, which we will illustrate first. We will thoroughly explain the rule extraction algorithm. Of course, we evaluate this setting on three translation tasks and will present our findings. Again, we analyze the translations to show how much the system relies on our ℓMBOT rules. Furthermore, we will try our string-to-tree ℓMBOT on two additional translation tasks. Our additional target languages are Eastern European languages (Russian and Polish). For these languages, only dependency parsers are available. But the parse trees of dependency parsers are in general not suited for our translation system that requires constituency parse trees. Further complications can occur from non-projective parse trees which do not allow for a hierarchical tree representation. Hence, we present our approach on how to obtain projective dependency parse trees and then further convert them into constituent-like tree representations. The evaluation shows that the string-to-tree ℓMBOT is much more suited for these translation tasks than a continuous translation system.

In Chapter 6 we will present our translation system based on string-to-string ℓMBOTs. We start by presenting the required adaption of our theoretical model for this setting. Next, we show how to obtain a set of string-to-string ℓMBOT rules by explaining the underlying rule extraction algorithm. We will evaluate our system against a baseline system on two translation tasks and analyze the translations with respect to the usefulness of discontinuous rules.

We will further analyze our ℓMBOT systems in the various settings in Chapter 7. First, we show how many sentential forms were obtained, and we try to estimate the use of discontinuous rules as truly discontinuous. We conclude this chapter with an qualitative analysis of the translations for which we gained significant improvements.

Finally, in Chapter 8 we will conclude this thesis by summarizing the contribu-

tions as well as repeating our conclusions from the analysis of the use of discontinuous rules. Furthermore, we give an overview of different approaches to explore and possibly improve the ℓMBOT translation system in future work.

Chapter 2

Background

In this chapter, we present the three major steps required to obtain statistical translation systems — *training, tuning,* and *evaluation*. We start with the formal introduction to synchronous grammars and then show how to obtain a set of rules for the different syntax-based models. Furthermore, we introduce important components like the language model and the decoder. Finally, we show the standard features for syntax-based translation systems and describe the tuning and evaluation process. We conclude this chapter with an overview of related work.

2.1 Syntax-based Statistical Machine Translation

Synchronous Context-free Grammars Aho and Ullman (1969) present the earliest synchronous grammars nowadays called *synchronous context-free grammars* (SCFG). We slightly adjust the original definition. Recall from the introduction that we use a parallel corpus with sentences from two different languages. Furthermore, we want to use the syntactic annotations from a parser. The names (labels) of the syntactic categories differ from language to language, so we distinguish between source nonterminals (the source language labels) and target nonterminals (the target language labels). Thus, our SCFG have the form

$G = (V_s, V_t, \Sigma, \Delta, R, S_s, S_t)$, where

- V_s is a finite set of source nonterminals,
- V_t is a finite set of target nonterminals,
- Σ is a finite set of source symbols (lexical elements of the source language),
- Δ is a finite set of target symbols (lexical elements of the target language),
- R is a finite set of rules,
- $S_s \in V_s$ is the source start nonterminal, and
- $S_t \in V_t$ is the target start nonterminal.

A sentential form of G is a triple (α, β, Π) where $\alpha \in (V_s \cup \Sigma)^*$, $\beta \in (V_t \cup \Delta)^*$, and Π is a permutation on nonterminal occurrences in α and β such that the i-th nonterminal occurrence in α is linked to the $\Pi(i)$-th nonterminal occurrence in β. A rule is an object $(A_s, A_t) \rightarrow (\alpha, \beta, \Pi)$ where A_s and A_t are source and target nonterminals, respectively, and (α, β, Π) is a sentential form.

Given a set of rules R, the rewrite relation $\Rightarrow$ is defined on sentential forms. Suppose that $(\alpha_1, \beta_1, \Pi_1)$ is a sentential form of G, in which A_s is the i-th nonterminal occurrence in α_1 and A_t is the $\Pi_1(i)$-th nonterminal occurrence in β_1, and $r = (A_s, A_t) \rightarrow (\gamma, \delta, \Pi)$ is a rule of R, then a combined sentential form $(\alpha_2, \beta_2, \Pi_2)$ can be constructed by

- replacing the i-th nonterminal occurrence in α_1 by γ to obtain α_2,
- replacing the $\Pi(i)$-th nonterminal occurrence in β_1 by δ to obtain β_2, and
- defining Π_2 as follows.

If γ has m nonterminal occurrences with $m \geq 0$, then Π_2 is defined by:

$$\forall j < i : \Pi_2(j) = \begin{cases} \Pi_1(j) & \text{if } \Pi_1(j) < \Pi_1(i) \\ \Pi_1(j) + m - 1 & \text{otherwise} \end{cases} \tag{2.1}$$

$$\forall j > i : \Pi_2(j + m - 1) = \begin{cases} \Pi_1(j) & \text{if } \Pi_1(j) < \Pi_1(i) \\ \Pi_1(j) + m - 1 & \text{otherwise} \end{cases} \tag{2.2}$$

$$\forall d \text{ with } 1 \leq d \leq m : \Pi_2(i+d-1) = \Pi_1(i) + \Pi(d) - 1 \tag{2.3}$$

We also write $(\alpha_1, \beta_1, \Pi_1) \underset{r}{\Rightarrow} (\alpha_2, \beta_2, \Pi_2)$. Equations 2.1 and 2.2 define the new permutation for nonterminals that were present in the sentential form and Equation 2.3 defines the permutation of the nonterminals present in the rule. A derivation is obtained by recursively rewriting the initial sentential form $(S_s, S_t, [1])$ with rules of G. When all nonterminals are replaced, a final sentential form $(\sigma, \delta, \emptyset)$ is obtained, which contains a *string pair* σ, δ where $\sigma \in \Sigma^*$ and $\delta \in \Delta^*$.

It is obvious that a SCFG can model translation. Let us illustrate a SCFG by an example with the following rules:

r_1 : (S, TOP) $\rightarrow$ (DT VP, PDS VP, [1,2])
r_2 : (VP, VP) $\rightarrow$ (VBZ AP, VAFIN ADJP, [1,2])
r_3 : (AP, ADJP) $\rightarrow$ (JJ RB, ADV ADJA, [2,1])
r_4 : (DT, PDS) $\rightarrow$ (That, Das)
r_5 : (VBZ, VAFIN) $\rightarrow$ (is, ist)
r_6 : (JJ, ADJA) $\rightarrow$ (true, wahr)
r_7 : (RB, ADV) $\rightarrow$ (actually, tatsächlich)

The grammar shows that the same nonterminal 'VP' is used for verbal phrases in both languages but, for example, an adjective phrase is called 'AP' in English and 'ADJP' in German. We start a derivation for the grammar given above with (S, TOP, [1]) and obtain the following derivation:

(S, TOP, [1])

$D_1 : \underset{r_1}{\Rightarrow}$ (DT VP, PDS VP, [1, 2])

$D_2 : \underset{r_4}{\Rightarrow}$ (That VP, Das VP, [1])

$D_3 : \underset{r_2}{\Rightarrow}$ (That VBZ AP, Das VAFIN ADJP, [1, 2])

$D_4 : \underset{r_3}{\Rightarrow}$ (That VBZ JJ RB, Das VAFIN ADV ADJA, [1, 3, 2])

$D_5 : \underset{r_5}{\Rightarrow}$ (That is JJ RB, Das ist ADV ADJA, [2, 1])

$D_6 : \underset{r_6}{\Rightarrow}$ (That is true RB, Das ist ADV wahr, [1])

$D_7 : \underset{r_7}{\Rightarrow}$ (That is true actually, Das ist tatsächlich wahr)

As illustration, we show how to compute the new permutation Π_4 for the combined sentential form obtained in derivation step D_4. The nonterminal 'VBZ' in the sentential form D_3 with permutation Π_3 is not replaced and we compute its new permutation by Equation (2.1) because it occurs before the replaced nonterminal. Hence, $\Pi_4(1) = \Pi_3(1) = 1$ because $\Pi_3(1) = 1 < \Pi_3(2) = 2$. To compute the permutation for the nonterminals present in rule r_3 with permutation Π, we use Equation (2.3). For the nonterminal 'JJ' we obtain $\Pi_4(2) = \Pi_4(2 + 1 - 1) = \Pi_3(2)+\Pi(1)-1 = 2+2-1 = 3$ and the permutation for 'RB' is $\Pi_4(3) = \Pi_4(2+2-1) = \Pi_3(2) + \Pi(2) - 1 = 2 + 1 - 1 = 2$. This results in the new permutation [1, 3, 2].

Besides SCFG, there are many more types of synchronous grammars like Synchronous Tree Adjoining Grammars (Shieber and Schabes, 1990) or Synchronous Tree Substitution Grammars (Eisner, 2003). A detailed introduction to synchronous grammars is given by Chiang (2006).

2.1.1 Training

Training is the process of deriving a set of rules from a given word-aligned parallel corpus and assigning those rules a probability. In the following, we explain how to obtain rules for the hierarchical phrase-based model, the string-to-tree model and the tree-to-tree model. Finally, we show how to compute the translation probabili-

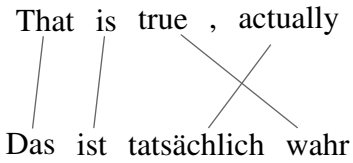

Figure 2.1: A word-aligned sentence pair.

ties and lexical probabilities.

Hierarchical Phrase-based Model

Chiang (2005, 2007) presents the hierarchical phrase-based model. It aims to combine the power of the phrase-based approach with a syntactic formalism. However, it is only formally syntax-based but not linguistically motivated. It is based on a SCFG but there is only one nonterminal X. The approach requires a simple (i.e. non-parsed) word-aligned parallel corpus C which consists of word-aligned sentence pairs $\langle e, A, f \rangle$. An example of such a pair is shown in Figure 2.1. Rules are extracted for each pair of C, so in the following let $\langle e, A, f \rangle$ be a word-aligned sentence pair which contains a source language sentence e, a target language sentence f, and an alignment $A \subseteq [1, \ell_e] \times [1, \ell_f]$, where ℓ_e and ℓ_f are the lengths of the sentences e and f, respectively, and $[i, i'] = \{j \in \mathbb{Z} \mid i \leq j \leq i'\}$ is the span (closed interval of integers) from i to i' for all positive integers $i \leq i'$. A *source phrase e* is simply a span $[i, i'] \subseteq [1, \ell_e]$ and correspondingly, a *target phrase f* is a span $[j, j'] \subseteq [1, \ell_f]$.

Rule extraction is based on *initial phrase pairs* that are consistent with the alignment (see Definition 1). Given this notion, we present a formal definition of the extracted hierarchical SCFG rules in Definition 2 based on Chiang (2007) and Hoang (2011).

Definition 1 *Given a word-aligned sentence pair $\langle e, A, f \rangle$, a pair $\langle [i, i'], [j, j'] \rangle$ is an initial phrase pair of $\langle e, A, f \rangle$ if and only if for all $(k, k') \in A$: $k \in [i, i']$ if and only if $k' \in [j, j']$.*

For convenience reasons, the permutation of Aho and Ullman (1969) is replaced by indices, which define the alignment between the nonterminal symbols in the sense that equally indexed nonterminals are related by the permutation. Thus, the subscripts indicate which nonterminals in α and γ have to develop synchronously. Following, we display rule r_3 of our example grammar in both variants.

Permutation:	(AP, ADJP) $\rightarrow$	(JJ RB, ADV ADJA, [2,1])
Indices:	(AP, ADJP) $\rightarrow$	(JJ_1 RB_2, ADV_2 ADJA_1)

Definition 2 *The set G of rules extracted from C is the smallest set satisfying the following conditions:*

1. *Initial rules:*
   If $\langle [i, i'], [j, j'] \rangle$ is an initial phrase pair for some $\langle e, A, f \rangle \in C$, then $(X, X) \rightarrow (e[i, i'], f[j, j'])$ is an initial rule, and in particular a rule.
2. *Subtraction step:*
   If $r = (X, X) \rightarrow (\alpha, \gamma)$ is a rule and $(X, X) \rightarrow (\alpha', \gamma')$ is an initial rule such that $\alpha = \alpha_1 \alpha' \alpha_2$ and $\gamma = \gamma_1 \gamma' \gamma_2$, then $(X, X) \rightarrow (\alpha_1 X_k \alpha_2,\ \gamma_1 X_k \gamma_2)$ is a rule, where k is an index (nonterminal alignment) not used in r.

Extracting all rules leads to an unmanageable number of rules as already pointed out by Chiang (2005, 2007) and he suggests to apply the following constraints:

(1) Initial rules $(X, X) \rightarrow (\alpha, \gamma)$ are limited to a length of 10 terminals in α and γ.
(2) A rule $(X, X) \rightarrow (\alpha, \gamma)$ is limited to five symbols (nonterminals and terminals) in α.
(3) In the subtraction step, α' must have length greater than one.
(4) A rule $(X, X) \rightarrow (\alpha, \gamma)$ can have at most two nonterminal occurrences in α.
(5) It is prohibited for a rule $(X, X) \rightarrow (\alpha, \gamma)$ to have adjacent nonterminal occurrences in α.
(6) A rule must have at least one pair of words that was aligned in some aligned sentence pair.

We display some hierarchical SCFG rules in Figure 2.2 that can be extracted

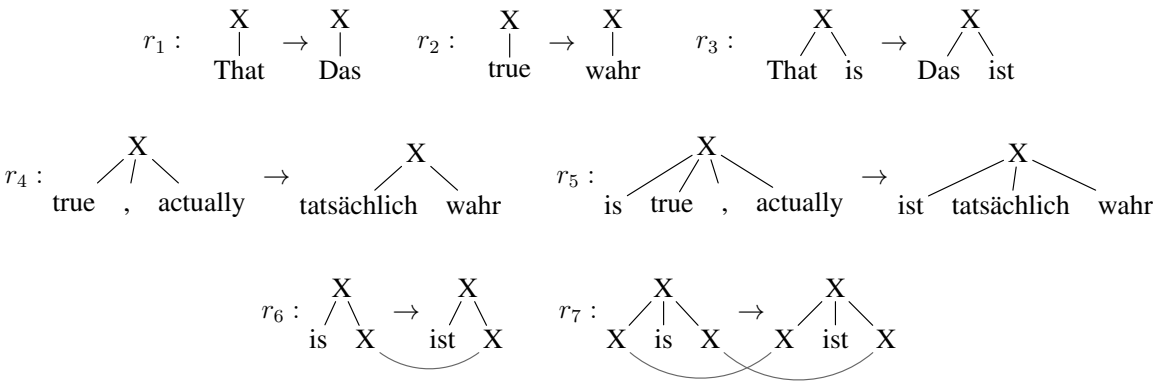

Figure 2.2: Some extractable hierarchical SCFG rules.

from the sentence pair shown in Figure 2.1. Throughout this thesis, we keep on illustrating rules in this tree-like representation (where appropriate). Furthermore, we refer to the tree left of the arrow as the *left-hand side* of a rule and we call the tree to the right of the arrow the *right-hand side* of a rule. Let us return to the rules in Figure 2.2. We show some initial rules in the first and second row and the rules in the bottom row are obtained by the subtraction step. For example, r_6 is obtained by subtracting r_4 from r_5. We indicate the alignment between nonterminals (indices k) by links.

Hoang et al. (2009) implemented the rule extraction algorithm from Definition 2 in MOSES (Koehn et al., 2007). The constraints stated above are also implemented and define the baseline system in our experiments of Section 6.5.

String-to-Tree Models

These kinds of models require a word-aligned parallel corpus C with a constituency parse tree for the target language side. An example entry of a word-aligned sentence pair with target tree parse is depicted in Figure 2.3. The rule extraction algorithm for this setting is based on the one for hierarchical SCFG with some modifications (see Definition 3).

Definition 3 *The set G of rules extracted from C is the smallest set satisfying the*

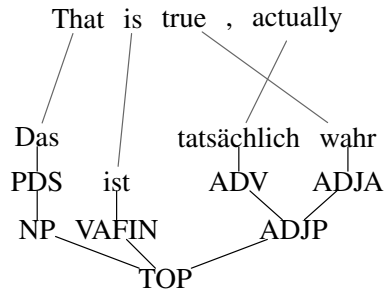

Figure 2.3: A word-aligned sentence pair with target parse tree.

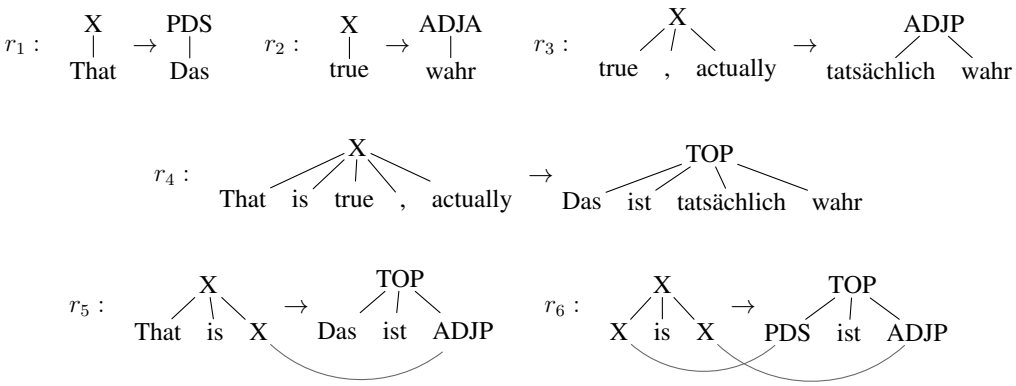

Figure 2.4: Some extractable string-to-tree SCFG rules.

following conditions:

1. *If* $\langle [i, i'], [j, j'] \rangle$ *is an initial phrase pair for some* $\langle e, A, f \rangle \in C$ *and the span* $[j, j']$ *is governed by a syntactic category* B *in the target-side parse for* f*, then* $(X, B) \rightarrow (e[i, i'], f[j, j'])$ *is a rule.*
2. *If* $r = (X, B) \rightarrow (\alpha, \gamma)$ *is a rule and* $\langle [i, i'], [j, j'] \rangle$ *is an initial phrase pair for some* $\langle e, A, f \rangle \in C$*, where the span* $[j, j']$ *is governed by a syntactic category* B'*, such that* $\alpha = \alpha_1\ e[i, i']\ \alpha_2$ *and* $\gamma = \gamma_1\ f[j, j']\ \gamma_2$*, then* $(X, B) \rightarrow (\alpha_1 X_k \alpha_2,\ \gamma_1 B'_k \gamma_2)$ *is a rule, where* k *is an index not used in* r*.*

We display some initial rules for the word-aligned sentence pair in Figure 2.3 in the first and second row of Figure 2.4. Rule r_5 is obtained by subtracting r_3 from r_4. By subtracting both r_1 and r_3 from r_4 we obtain r_6.

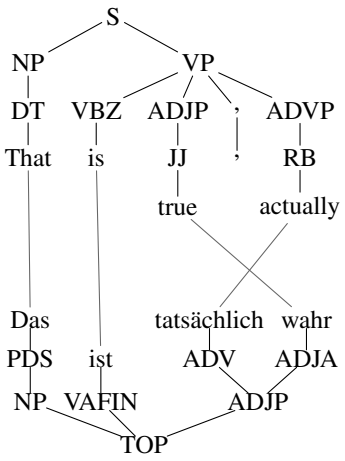

Figure 2.5: A word-aligned sentence pair with source and target parse tree.

This model is also implemented in MOSES (Koehn et al., 2007; Hoang et al., 2009) and the same constraints as for hierarchical SCFG rule extraction can be applied. We use a system with such rules as a baseline system in our experiments of Section 5.5.

Tree-to-Tree Models

For tree-to-tree models, both sides of the word-aligned parallel corpus C require syntactic parse trees. In Figure 2.5 we display our running example for the task at hand. We formally define rules in Definition 4, which is again a modification of the hierarchical (and string-to-tree) SCFG rules.

Definition 4 *The set G of rules for C is the smallest set satisfying the following conditions:*

1. *If $\langle [i, i'], [j, j'] \rangle$ is an initial phrase pair for some $\langle e, A, f \rangle \in C$ and the span $[i, i']$ is governed by a syntactic category B in the source-side parse for e and the span $[j, j']$ is governed by a syntactic category B' in the target-side parse for f, then $(B, B') \rightarrow (e[i, i'], f[j, j'])$ is a rule.*
2. *If $r = (B, B') \rightarrow (\alpha, \gamma)$ is a rule and $\langle [i, i'], [j, j'] \rangle$ is an initial phrase pair for some $\langle e, A, f \rangle \in C$, where the span $[i, i']$ is governed by a syntactic cate-*

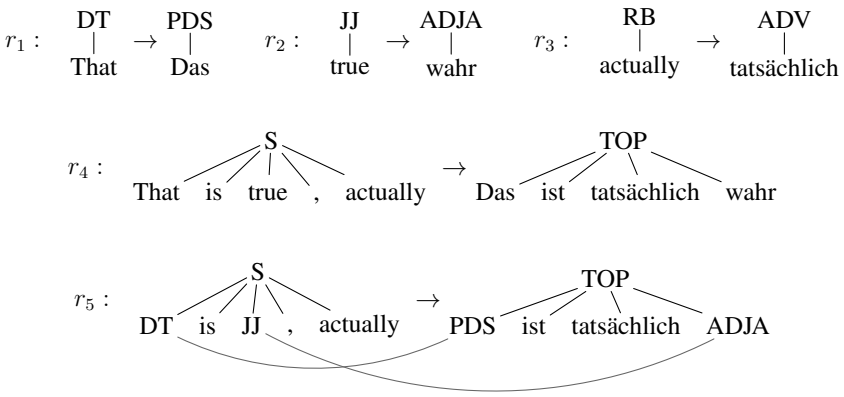

Figure 2.6: Some extractable tree-to-tree SCFG rules.

gory D and the span $[j, j']$ is governed by a syntactic category D', such that $\alpha = \alpha_1\ e[i, i']\ \alpha_2$ and $\gamma = \gamma_1\ f[j, j']\ \gamma_2$, then $(B, B') \rightarrow (\alpha_1 D_k \alpha_2,\ \gamma_1 D'_k \gamma_2)$ is a rule, where k is an index not used in r.

Rules r_1–r_4 shown in Figure 2.6 are examples for initial rules for the word-aligned and bi-parsed sentence pair of Figure 2.5. Rule r_5 is obtained in the subtraction step by subtracting rules r_1 and r_2 from rule r_4.

MOSES (Koehn et al., 2007; Hoang et al., 2009) also provides an implementation of this rule extraction and the same constraints as for hierarchical SCFG rule extraction can be applied. But the parse trees for both the source and the target language side act as natural constraints and typically, constraints (4)–(6) are not enforced. In our experiments in Section 4.4 we use a baseline system based on these tree-to-tree SCFG rules.

Glue Grammar All syntax-based translation systems require a so-called *glue grammar* to ensure that a translation can always be generated. The glue grammar allows to concatenate partial translations without any reordering, i.e. the rules are combined sequentially whenever a recursive application is not possible. To ensure a smooth glue rule application, the input sentences are wrapped into tags, <s> and </s>. This wrapping makes it necessary to have standard glue rules which allow for sequential rule combination and top glue rules which are used

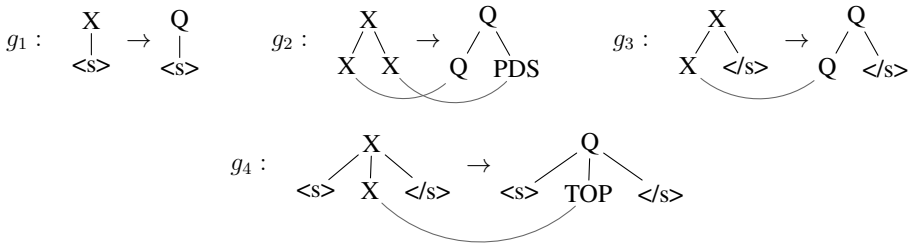

Figure 2.7: Top row: standard glue rules. Bottom row: top glue rule.

when the translation constitutes a full derivation. We display some glue rules for tree-to-tree and string-to-tree SCFG models in Figure 2.7. We will explain the application of such rules in Section 2.1.3. The glue rules are generated while extracting SCFG rules from a parallel corpus.

Bidirectional Translation Probabilities

After all rules are extracted from each sentence pair of a parallel corpus, we need to estimate the impact for each rule. To this end, we count how often a rule $\alpha \rightarrow \gamma$ (in our tree representation) with left-hand side α and right-hand side γ has been extracted. This number is stored in $count(\alpha, \gamma)$. Given this count, we can compute

$$\varphi(\gamma|\alpha) = \frac{count(\alpha, \gamma)}{\sum_{\gamma'} count(\alpha, \gamma')} \ ,$$

i.e. to estimate the probability of γ given α we compute the relative frequency by normalizing over all rules with the same left-hand side. Let us illustrate this in an example. Consider Table 2.1, where we collected some counts and omitted the root labels, which are always X. For example, we have the counts for the English 'the X' being translated into the three German variants 'der X', 'die X', and 'das X', respectively. In total, we have extracted 100 rules that translate 'the X' into some γ. We compute $\varphi(\text{der X}|\text{the X}) = \frac{50}{100} = 0.5$, $\varphi(\text{die X}|\text{the X}) = \frac{20}{100} = 0.2$, and $\varphi(\text{das X}|\text{the X}) = \frac{30}{100} = 0.3$.

Similarly, we estimate $\varphi(\alpha|\gamma)$ by computing the relative frequency by normaliz-

α	γ	$counts(\alpha, \gamma)$
of X	der X	10
of the X	der X	40
the X	der X	50
the X	die X	20
the X	das X	30

Table 2.1: Collected counts for (α, γ).

ing over all rules with the same right-hand side. In this case, we have to look at the γ-column of Table 2.1. Consider that we want to compute how probable it is for 'der X' to be generated by 'of X', 'of the X', and 'the X', respectively. In total, we have again 100 rules and φ(the X|der X) $= \frac{50}{100} = 0.5$, φ(of the X|der X) $= \frac{40}{100} = 0.4$, and φ(of X|der X) $= \frac{10}{100} = 0.1$.

Bidirectional Lexical Translation Probabilities

The extracted rules can consist of terminal and nonterminal symbols. The translation probabilities computed above treat all symbols equally and do not distinguish between terminals and nonterminals. But it is also of interest, how well the terminals on the left- and right-hand side match. Recall that we extract the rules from a parallel corpus of word-aligned sentence pairs $\langle e, A, f \rangle$, so we have the word alignment A. For each input word $\sigma \in \Sigma$ and output word $\delta \in \Delta$ we can thus count how often $e[i, i] = \sigma$ and $f[j, j] = \delta$ for $(i, j) \in A$ in some $\langle e, A, f \rangle \in C$. We can collect the counts for all word pairs (σ, δ) and compute the relative frequencies, which result in the *word translation probabilities* $w(\delta|\sigma)$. The lexical translation probability of a right-hand side γ given a left-hand side α and lexical alignment A is computed by

$$lex(\gamma|\alpha, A) = \prod_{j=1}^{|\gamma|} \frac{1}{|\{i|(i, j) \in A\}|} \sum_{(i,j)\in A} w(\gamma_j|\alpha_i) \ ,$$

i.e. for all lexically aligned word pairs (α_i, γ_j) where α_i is present in α and γ_j is present in γ, we take the product of their respective word translation probabilities.

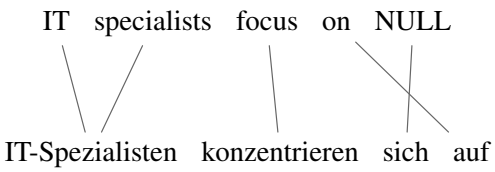

Figure 2.8: Illustration of word-alignments and special NULL token.

If a word δ is aligned to multiple words σ, then the average of the corresponding word translation probabilities is computed. In case of words δ which have no alignment to any word σ, the alignment to a special NULL word is assumed and scored. We illustrate this in an example. Consider the word-alignments given in Figure 2.8, where the word 'IT-Spezialisten' is aligned to two words, 'IT' and 'specialists', and 'sich' is aligned to NULL. The lexical translation probability is computed by

$$\begin{aligned} lex(\gamma|\alpha, A) = & \frac{1}{2}\big(w(\text{IT-Spezialisten}|\text{IT}) + w(\text{IT-Spezialisten}|\text{specialists})\big) \\ & \cdot w(\text{konzentrieren}|\text{focus}) \\ & \cdot w(\text{sich}|\text{NULL}) \\ & \cdot w(\text{auf}|\text{on}) \end{aligned}$$

2.1.2 Language Model

Another important component of a translation system is the *language model*. It supports the modeling of the target language and is therefore obtained from text written in the target language.[1] Its assistance to the translation process is threefold. First, it ensures that the translations are fluent. Furthermore, it helps to decide whether a given word order is grammatically correct or not and in this manner affects word reordering. Finally, it can provide the right word choice. Formally, a

[1] Beside the data from the parallel corpus, one can use additional data taken from monolingual sources.

language model computes the probability $p(W)$ of a string $W = w_1 \dots w_m$. Typically n-gram language models are used, in which the estimation process predicts the probability of one word given a preceding sequence of $(n-1)$ words. The order of the language model is defined by m and the probabilities are computed by a Markov chain:

$$p(W) = \prod_{i=1}^{m} p(w_i | w_{i-(n-1)}, \dots, w_{i-2}, w_{i-1}) \ ,$$

where we assume that $w_j = \square$ for $j \leq 0$.

The estimation of the probability $p(w_i | w_{i-(n-1)}, \dots, w_{i-2}, w_{i-1})$ is straightforward with

$$p(w_i | w_{i-(n-1)}, \dots, w_{i-2}, w_{i-1}) = \frac{count(w_{i-(n-1)}, \dots, w_{i-2}, w_{i-1}, w_i)}{\sum_w count(w_{i-(n-1)}, \dots, w_{i-2}, w_{i-1}, w)} \ ,$$

i.e. we count how often the string $w_{i-(n-1)} \dots w_{i-2} w_{i-1}$ is followed by the word w_i and divide by the sum of all counts, where any word w follows the string. This equation only assigns a non-zero value to an n-gram that occurs in the training data. This is not desired and the empirical counts are adjusted via a smoothing technique. The *Modified Kneser-Ney smoothing* of Chen and Goodman (1998) uses three different discount values that get subtracted from the raw counts of the n-grams. These discounts are then used to assign non-zero probabilities to unseen n-grams.

2.1.3 Decoder

The actual translation of a source sentence into a target sentence is done by the *decoder*. For most syntax-based translation systems, the decoder generates a translation via a chart parser. The parser works in a bottom-up fashion. First, for each source word the rule-table is looked up for a corresponding rule. More precisely, each contiguous span of the source sentence is looked-up by increasing length and (possibly) multiple rules are added to the corresponding chart cell matching this

[1,7] X						
[1,6] X	[2,7] X					
[1,5] X	[2,6] **X** r_4, r_5, r_6	[3,7] X				
[1,4] X	[2,5] X	[3,6] X	[4,7] X			
[1,3] X	[2,4] X	[3,5] X	[4,6] X r_3	[5,7] X		
[1,2] X	[2,3] X	[3,4] X	[4,5] X	[5,6] X	[6,7] X	
[1,1] X <s>	[2,2] X r_1 That	[3,3] X is	[4,4] X r_2 true	[5,5] X ,	[6,6] X actually	[7,7] X </s>

Table 2.2: Chart parse for hierarchical and string-to-tree SCFG systems.

span. In the iteration, a chart entry for a larger span can build on the chart entries for its strict subspans.

For models based on hierarchical SCFGs and string-to-tree SCFGs, each chart cell is created by assuming that each span is covered by 'X'. We illustrate this in Table 2.2, where we want to translate the input sentence 'That is true , actually' (spanning chart cell [2,6]) which is augmented with tags that are introduced to allow for easy glue rule application. We will illustrate an application of glue rules later on and first focus on showing an example of how to use string-to-tree SCFG rules. Given the chart in Table 2.2, the parser starts its iteration on the bottom row. For each cell, the decoder tries to find a rule. Assuming that only the rules from Figure 2.4 are given, the rules r_1 and r_2 can be applied in chart cells [2,2] and [4,4], respectively. For all other words, no matching rule is given. In the next iteration step, the decoder looks for rules matching two consecutive chart entries. This process is repeated until the decoder arrives in chart cell [1,7]. Let us move on with

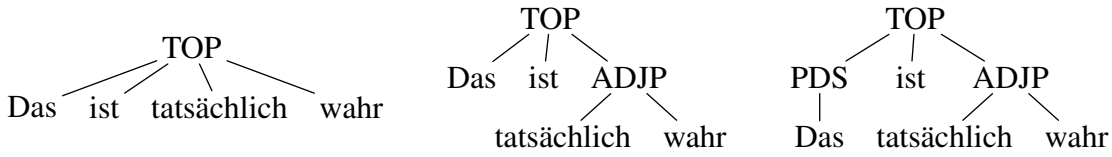

Figure 2.9: The resulting target parse trees in the string-to-tree decoding.

.

our illustration. If we want to find all possible rules that derive the whole sentence spanning [2,6], still assuming that only the rules from Figure 2.4 are given, we have three candidate rules, i.e. r_4, r_5, and r_6. The rule r_4 simply generates the whole target sentence given the source sentence by matching all the terminals. Rule r_5 matches the terminals in chart cell [2,2], chart cell [3,3], and the 'X' in chart cell [4,6], where previously rule r_3 was applied. We underlined the necessary components in Table 2.2. Finally, we can also apply rule r_6 that matches the 'X' in chart cell [2,2] (previously applied r_1), the terminal 'is' in [3,3], and the 'X' in chart cell [4,6] (previously applied r_3). We boxed the necessary components in Table 2.2. The rules that generate the best translation depend on their scores. Note that the use of different rules leads to quite different parse trees for the target sentence. We display the generated parse trees for the target sentence in Figure 2.9. It is noteworthy, that the generated parse trees for the translations are (typically) not part of the output of the translation system. The decoder returns the sentences only. The target parse nonterminals are only used by the decoder as constraints on which rules can be applied. Hence, the labels of the target parse tree have the same role as in constituent parsing and help to ensure the correct syntactic structure of the target sentence.

For models based on tree-to-tree (and tree-to-string) SCFGs, the parse of the source sentence is taken into account. Beside the nonterminal 'X', each chart cell is additionally created with the corresponding label if that span is covered by a node in the parse tree. We show this in Table 2.3, where the labels of the source parse tree are given. The source labels act as a constraint on which rules can be applied. For example, for 'true' the labels 'JJ' and 'ADJP' are licensed by the parse tree. The

decoder will look up only rules where the root symbol matches either label. Given the rules from Figure 2.6, only rule r_2 matches and this rule is added to the chart cell. If we want to derive the whole source sentence, we have two possible rules, r_4 and r_5. Rule r_4 generates the whole target sentence by matching the source label 'S' in chart cell [2,6] and all terminals present in the input sentence. Rule r_5 uses the previously applied rule r_1 in chart cell [2,2], the terminal 'is' in [3,3], r_2 in chart cell [4,4] as well as the terminals given in chart cells [5,5] and [6,6]. This is made possible by a special feature of the MOSES decoder. As explained above, the label of the source parse tree functions as a constraint. Hence, for the label 'S' only rules with this root symbol are considered. But there is no matching criterion for the nonterminal children. When rule r_3 is applied, the decoder only checks whether the chart cells [2,2] and [4,4] are covered by a rule but completely ignores the labels in these cells.

Glue Rule Application For our decoding example from Table 2.3 we were able to obtain a complete derivation in chart cell [2,6] by recursively applying our rules for the input sentence. Now we need to combine it with a top glue rule to match the augmented sentence spanning [1,7]. To this end, we apply glue rule g_4 from Figure 2.7. It combines the terminal '<s>', the label 'S', and the terminal '</s>' and the chart parse is complete. Since the recursive application can fail at any point in the chart, glue rules are generally applied to ensure the success of the translation process. Hence, glue rule g_1 is applied in chart cell [1,1]. In the same manner, glue rule g_2 is applied in chart cell [1,2] using previously applied glue rule g_1 and the nonterminal 'X' given in chart cell [2,2]. To be precise, for each chart cell [1,n-1] this glue rule is applied (as indicated in Table 2.3). For span [1,n], glue rule g_3 (see Figure 2.7) concatenates the 'X' in chart cell [1,n-1] with the end tag </s>, thus ensuring a successful translation.

In a realistic scenario, for each contiguous span any number of rules may apply. Each applicable rule triggers another chart entry. Hence, it is possible to end up with thousands of chart entries. To keep the size of the chart cells manageable, the

[1,7] X g_4, g_3						
[1,6] X g_2	[2,7] X					
[1,5] X g_2	[2,6] **S**, X	[3,7] X				
[1,4] X g_2	[2,5] VP, X	[3,6] X	[4,7] X			
[1,3] X g_2	[2,4] X	[3,5] X	[4,6] X	[5,7] X		
[1,2] X g_2	[2,3] X	[3,4] X	[4,5] X	[5,6] X	[6,7] X	
[1,1] X g_1 <s>	[2,2] **DT**, NP, X r_1 That	[3,3] VBZ, X is	[4,4] **JJ**, ADJP, X r_2 true	[5,5] , , X ,	[6,6] **RB**, ADVP r_3 actually	[7,7] X </s>

Table 2.3: Chart parse for tree-to-tree and tree-to-string SCFGs.

MOSES decoder uses a CYK+ parser with cube pruning and integrated language model scoring to prune unlikely translations.

An extensive overview of the general decoding process can be found in Koehn (2010) and a detailed description of the MOSES decoder is given in Hoang (2011).

2.1.4 Syntax-based Translation Model

Given a source language sentence e, the task of the translation model is to find the best corresponding target language translation $\hat{f}$; i.e.,

$$\hat{f} = \arg\max_f p(f|e) = \arg\max_f \frac{p(f,e)}{p(e)} = \arg\max_f p(e|f) \cdot p(f) \quad .$$

The best translation $\hat{f}$ is defined by the probability of the translation model $p(f|e)$. Using Bayes' rule we can decompose this probability into $p(e|f)$ and the language model probability $p(f)$ as indicated.

The standard syntax-based translation model computes $p(e|f)$ using the following features.

(1) backward translation probability $\varphi(e|f)$
(2) forward translation probability $\varphi(f|e)$
(3) backward lexical translation probability $lex(e|f)$
(4) forward lexical translation probability $lex(f|e)$
(5) *Word penalty*: models the output length in terms of number of words. It is modelled as a simple counter, i.e. $e^1 \approx 2.718$.
(6) *Rule penalty*: models the usage of rules. Either many (smaller) rules are used to obtain a translation or fewer (bigger) rules are used. Similarly to the word penalty, it is modelled by e^1.
(7) *Glue grammar*: models the usage of glue rules. Either the translations are obtained by recursive application of rules or glue rules are used to concatenate partial translations sequentially. Again, this is modelled as a simple counter.

This way, we can say that the translation process is guided by eight (including the language model) components. In the next step, weights λ_m are introduced that allow one to scale the contribution h_m for each component $h_m(\cdot)$.

$$p(f|e) \approx \arg\max_f \prod_{m=1}^{8} h_m(\cdot)^{\lambda_m} \quad .$$

As this leads to quite low weights, it is rewritten into a log-linear model

$$p(f|e) \approx \arg\max_f \Big(\sum_{m=1}^{8} \lambda_m h_m(\cdot) \Big) \quad .$$

2.1.5 Tuning

Tuning is an optimization process that aims at finding the optimal weights for the features λ_m of the log-linear model. As a prerequisite, we need another parallel corpus called the *tuning set*. This set is typically small (1,000–3,000 sentences) and should be distinct from the parallel corpus used for training. The tuning algorithm starts with a random seed of weights for the features and decodes the complete tuning set. The output is not only one translation but a list with the n-best translations. The translation quality of the n-best list is measured by some metric and the tuning algorithm adjusts the weights based on this metric. Then, the tuning set is decoded again with the adjusted weights, generating another n-best list, which in turn is measured. This process is repeated until some convergence criterion is satisfied. There are multiple tuning algorithms and there are many metrics for translation quality. In our experiments, all systems are tuned with Minimum Error Rate Training (Och, 2003) on the BLEU metric (Papineni et al., 2002).

For a detailed overview of tuning algorithms see Neubig and Watanabe (2016).

2.1.6 Evaluation

After and in the tuning process, we need to measure the quality of a translation system. The final score of a system is reported on another small parallel corpus called the *test set*. This set consists of source sentences and reference sentences, which were translated by human translators. The source sentences are decoded with the optimized weights λ_m for the features of the log-linear model. The Bilingual Evaluation Understudy (BLEU) metric (Papineni et al., 2002) estimates the translation quality by comparing the output of the translation system to the human reference.[2] It computes the *modified n-gram precision* for any n by first counting all n-grams $count_n(o)$ of a system-generated sentence o and its corresponding maximum count in the reference sentence r. If a $count_i(o)$ is higher than the reference count, it gets clipped by the maximum reference value. Next, $match_n(r, o)$ counts how many n-grams in o match r. Furthermore, a *brevity penalty* is applied that penalizes output translations that are shorter than the reference. Typically, the maximum order of n is set to 4 and BLEU-4 can be computed by

$$\text{BLEU-4}(r, o) = min(1, e^{(1-\frac{|r|}{|o|})}) \cdot \prod_{n=1}^{4} \Big(\frac{match_n(r, o)}{count_n(o)}\Big)^{\frac{1}{4}} \quad .$$

Through the usage of n-grams, BLEU captures adequacy and fluency. If a translation uses the same words as the reference (unigrams), the output becomes adequate. If a translation matches longer n-grams with the reference, the output is considered to be fluent.

2.2 Related work

In this section we show work by other authors that present rule extraction algorithms based on different grammar formalisms. We distinguish between hierarchi-

[2]The tuning algorithm also uses this metric to optimize the weights.

cal models, string-to-tree models, and tree-to-tree models. Furthermore, we arrange them according to the underlying formalism.

2.2.1 Hierarchical Models

Kaeshammer (2015) uses a translation system based on *synchronous linear context-free rewriting systems* (SLCFRS) which can be seen as a direct extension of SCFG to discontinuous constituents. The rule extraction algorithm is based on Chiang (2007) but instead of extracting only continuous constituents, also discontinuous constituents are extracted. Beside the constraints for hierarchical rule extraction (see Section 2.1.1), the author also restricts the number of words between discontinuous blocks, does not allow unaligned blocks, and allows only two discontinuous blocks. The training corpus was length-ratio filtered up to 30 words as was the test set. An evaluation for different settings of discontinuities was conducted. It shows that on a German-to-English translation task, a system with discontinuities on the left-hand side of the rules performs best. It performs even better than a hierarchical phrase-based system.

2.2.2 String-to-Tree Models

Synchronous Tree Substitution Grammars

Galley et al. (2004) present an algorithm that extracts minimal string-to-tree STSG rules. For a given source string S, a target tree T, and an alignment A, the authors want to learn the set of rules $\rho_A(S, T)$. To this end, they create an alignment graph G. This graph consists of the target tree T which is augmented with nodes for each element of S. The edges between leaf nodes $t \in T$ and $s \in S$ are defined by the alignment A. To extract rules from G, they propose a two-step algorithm:

1. Compute the frontier set of the alignment graph.
2. For each node of the frontier set, compute the minimal frontier graph frag-

ment rooted at that node.

To compute the set of frontier nodes, every node n in the graph is labeled with its span and complement span. The span is defined by the indices of the first and last word of S that is covered by n. The complement spans of n are the union of the spans in S of all nodes n' in G that are neither descendants nor ancestors of n. If the spans and complement spans of n do not overlap, then n is a frontier node. A frontier graph fragment is defined as a graph fragment where the root and all sinks are in the frontier set. Given this algorithm, the authors are able to extract *minimal string-to-tree STSG rules*. A rule is said to be minimal if the frontier graph fragment is a subgraph of every other frontier graph fragment with the same root. The focus of their contribution is to give a mathematical theory and provide an efficient algorithm. Therefore, the authors provide results on the coverage of their model and they analyze some properties of the transformation rules. But there is no evaluation in terms of translation quality.

Further work on this topic is done by Galley et al. (2006). In this paper, they extend the algorithm of Galley et al. (2004) to extract composed (i.e. non-minimal) string-to-tree STSG rules and propose probability estimates and a training procedure for weighting these rules. Their experiments on two translation tasks (Arabic-to-English and Chinese-to-English) show that their system lags only 6.4 BLEU points behind a then state-of-the-art phrase-based system.

DeNeefe et al. (2007) evaluated a system using minimal STSG rules with a system using composed STSG rules and found that the system using the composed rules improves over the system using minimal rules only.

Marcu et al. (2006) present SPMT which is based on extended tree-to-string transducers (xRS) of Knight and Graehl (2005). In their simplest model (SPMT model 1), given a tuple (Π, F, A), they extract xRS rules that are consistent with the source phrase $F[i, j]$, the target syntactic tree Π, and the alignment A. For each source phrase span, the algorithm traverses the target parse tree until it (a) finds a node that governs that span and creates a fully lexical rule or (b) in case that

the node has children not licensed by that span, those children will be realized as nonterminals in the rule. The algorithm considers all possible source phrases and is therefore adapted from the phrase-based rule extraction (Koehn et al., 2003) and those rules are called *minimally syntactified, lexicalized, phrase-based compatible xRS rules*. To obtain minimal non-lexicalized rules, they follow the approach of Galley et al. (2006). SPMT model 2 extracts additional rules for case (b). The algorithm creates a pseudo-label that governs only the nodes that are consistently aligned. Furthermore, there is a composed variant for both models but composition is allowed only once. They conducted experiments on Chinese-to-English translation tasks. Their experimental results show that SPMT model 2 performs best. Overall, all four variants show improvements over a phrase-based system.

Synchronous Context-free Grammars

Zollmann and Venugopal (2006) present a syntax-augmented machine translation system (SAMT). Their aim is to enrich phrase-based rules with syntactic annotations on the target side. To this end, they extract phrase pairs from a bilingual word-aligned corpus. Then they use a constituent parser to generate a parse tree for each target sentence. To annotate a phrase pair (α, γ), γ is matched with the target parse tree. If the phrase is covered by a syntactic category C, then a rule is generated with this label. Otherwise, the rule is generated with an extended category of the form $C_1 + C_2, C_1/C_2$, or $C_2\backslash C_1$. The extended categories indicate that γ spans two adjacent syntactic categories, a partial syntactic category C_1 missing a C_2 to the right, or a partial syntactic category C_1 missing a C_2 to the left, respectively. As those rules are still completely lexical, the authors adhere to the method of Chiang (2006) to recursively obtain a set of SCFG rules with nonterminals. The experiments show that their system improves over a phrase-based baseline.

Further work on the topic of SAMT was done by Almaghout et al. (2011) who use annotations from Combinatory Categorial Grammar (CCG) of Steedman (2000) and by Hanneman and Lavie (2013) who coarsen the set of CCG labels by clustering

bilingual labels and then removing the source side labels.

Williams and Koehn (2012) extract rules based on Synchronous Tree Substitution Grammars as presented by Galley et al. (2004, 2006) (we explained their algorithm in the paragraph *Synchronous Tree Substitution Grammars* at the beginning of this section). Their work differs with respect to handling composed rules. Rules are only composed if the rule depth does not exceed three, the node count is not higher than 15, and the rule size is not bigger than three. Furthermore, the authors remove unary rules and apply scope pruning as presented by Hopkins and Langmead (2010) to eliminate rules in order to receive a subgrammar that can be parsed in cubic time. The last step includes the removal of all internal nodes inside a rule and consequently, the rules are now in fact SCFG rules.

2.2.3 Tree-to-Tree Models

Synchronous Context-free Grammars

Lavie et al. (2008) extract rules from a word-aligned and bi-parsed parallel corpus. To obtain these rules, they first use a node alignment algorithm. The goal is the identification of aligned node pairs in the source and the target tree which are consistent with the word alignments between the yields of both nodes in the pair. In a first step, they obtain *syntax-based sub-sentential phrases* by extracting all aligned constituent nodes along with their yields from both trees. The second step extracts minimal *synchronous tree fragment pairs*. Each aligned node pair is treated as a tree decomposition point. Their algorithm operates on a top-down traversal of the parse trees. Each aligned node pair triggers a decomposition. The last step converts the rules into synchronous context-free rules by removing the internal tree structure. Hence, only the syntactic label of the roots of the tree fragments and the nodes on the fragment's frontier are kept. The authors do not give any BLEU scores but mention that their approach lags behind state-of-the-art phrase-based systems.

Ambati and Lavie (2008) improved on this work. The authors claim that the lexi-

cal coverage of the model is too weak. They identified the non-isomorphic structure of the parse trees of different languages as the main source for the coverage issue. Hence, they apply a non-isomorphic parse tree restructuring technique. The first step creates additional parse nodes that conform with the word alignment and the source parse tree and project the label from the source parse tree into the target parse tree. The second step merges some of these nodes as the structure now contains two trees – the original one and the projected tree. This way, they end up with one final tree structure. Their experiments show that their restructuring method does improve translation quality significantly compared to the system of Lavie et al. (2008), but a simple phrase-based system is still better.

In Ambati et al. (2009) the non-isomorphic parse tree restructuring technique is applied to both the source and the target parse tree. Their experiments show that this yields the best translation quality compared to the both approaches presented above. Yet, this system does not beat a phrase-based translation system.

Synchronous Tree Substitution Grammars

Zhang et al. (2007) use elementary tree-based structure alignments to model the translation process. They call their rules *PETs* which are pairs of elementary trees with alignment information. A PET is defined as a triple $\langle \xi_s, \xi_t, \tilde{A} \rangle$, where ξ_s is a source elementary tree, ξ_t is a target elementary tree, and $\tilde{A}$ is the alignment between leaf nodes of the two elementary trees. The authors differentiate between initial PETs in which all leaf nodes are terminals and abstract PETs, otherwise. Their rule extraction includes two steps: (1) extraction of initial PETs, and (2) extraction of abstract PETs by removing smaller initial PETs from bigger ones. Rule extraction was done on a parallel corpus containing 9,000 sentence pairs. Their evaluation showed a significant improvement over a phrase-based system (Koehn, 2004a).

Liu et al. (2009) propose a system that uses packed forests on the source and target language side to overcome the parse errors that usually occur in 1-best parses. Their rule extraction algorithm is adapted from Galley et al. (2004) and based on

three steps: (1) identifying correspondences between nodes in forest pairs, (2) identifying minimal rules, and (3) inferring composed rules. For (1) they identify and mark the frontier nodes in both packed forests. To obtain the set of minimal rules, the authors define a frontier tree pair as a triple $\langle t_{\mathrm{s}}, t_{\mathrm{t}}, \sim \rangle$, where t_{s} is a source frontier tree, t_{t} is a target frontier tree, and $\sim$ is a one-to-one correspondence between the frontier leaves of t_{s} and t_{t}. A frontier tree pair is said to be minimal if and only if it is a subgraph of any other frontier tree pair that shares the same root. To obtain composed rules they compose two or more minimal rules. Their experiments show that using packed forests does improve significantly over the same system using only 1-best parse trees. Furthermore, they evaluated how much of an improvement can be gained by using bigger parse forests. The result shows that the BLEU scores do not improve much when having packed forests with over 1M parses per sentence.

Synchronous Tree Sequence Substitution Grammars

Zhang et al. (2008) propose a tree-to-tree model based on tree sequence alignments. A tree sequence is an *ordered sub-tree sequence* that covers a phrase or a consecutive tree fragment in a parse tree. They define a tree sequence translation rule r as a pair of aligned tree sequences $r = \langle TS(f_{j_1}^{j_2}),\ TS(e_{i_1}^{i_2}),\ \tilde{A} \rangle$, where $TS(f_{j_1}^{j_2})$ is a source tree sequence covering span $[j_1, j_2]$ in the source parse tree $TS(f_1^J)$, and $TS(e_{i_1}^{i_2})$ is a target tree sequence covering span $[i_1, i_2]$ in the target parse tree $TS(e_1^I)$, and $\tilde{A}$ are the alignments between leaf nodes of the tree sequences. They extract rules in two steps: (1) extraction of initial rules, i.e. rules where all leaf nodes are terminals, (2) extraction of abstract rules by removing one or more smaller initial rules from a bigger initial rule. To control the number of rules, the authors set three constraints for both initial and abstract rules: (1) depth of a tree is not greater than $h = 4$, (2) number of nonterminals as leaf nodes is not greater than $c = 3$, and (3) number of trees in a sequence is not greater than $d = 4$. Additionally, there is a fourth constraint for initial rules only: a rule has at most seven lexical

elements as leaf nodes. They extracted rules from the FBIS corpus (less than 500K sentence pairs) and limited their tuning and test sets to 50 lexical elements per sentence. Their experiments on a Chinese-to-English translation task show significant improvements over a phrase-based, a SCFG-based, and a STSG-based system.

Sun et al. (2009) extended this work by proposing a translation model based on *non-contiguous tree sequence alignments*. Instead of allowing only continuous tree fragments, they use additionally non-contiguous tree sequence pairs where the sequence consists of subtrees and gaps. A non-contiguous tree sequence translation rule is defined as $r = \langle TS(f(M_{j_1^k}^{j_2^k})),\ TS(e(N_{i_1^l}^{i_2^l})),\ \tilde{A}\rangle$, where $TS(f(M_{j_1^k}^{j_2^k})_{1\leq k\leq m})$ is a non-contiguous source tree sequence covering the span set $M = \{[j_1^k, j_2^k] \mid k = 1,\ldots,m\}$ in the source parse tree $TS(f_1^J)$, $TS(e(N_{i_1^l}^{i_2^l})_{1\leq l\leq n})$ is a non-contiguous target tree sequence covering the span set $N = \{[i_1^l, i_2^l] \mid l = 1,\ldots,n\}$ in the target parse tree $TS(e_1^L)$, and $\tilde{A}$ are the alignments between leaf nodes in the source and target non-contiguous tree sequences. As in Zhang et al. (2008), they use initial and abstract rules. The initial rules are obtained by (a) extracting for contiguous source tree sequences the contiguous and non-contiguous target tree sequences, and (b) extracting for contiguous target tree sequences the non-contiguous source tree sequences. Abstract rules are derived similarly as in Zhang et al. (2008). Additionally, they use the following constraints to limit the number of rules: (1) depth of a tree is not greater than $h = 6$, (2) number of nonterminals as leaf nodes is not greater than $c = 6$, (3) number of trees in a sequence is not greater than $d = 4$, (4) number of lexical elements in an initial rule is not greater than $l = 6$, and (5) maximal number of gaps g in a rule (left-hand side: $g = 1$; right-hand side: $g = 0$). Rules were again obtained from the small FBIS corpus and the tuning and test sets were also restricted to sentences containing up to 50 lexical elements. Their evaluation of a Chinese-to-English translation task shows significant improvements over baselines provided by a phrase-based system and the continuous STSSG-based model of Zhang et al. (2008).

Chapter 3

Minimal Tree-to-Tree ℓMBOT

This chapter starts with the introduction of the theoretical background and the main generative model, namely *shallow local multi bottom-up tree transducers* (ℓMBOT). We present an algorithm that allows us to obtain such transducers from data, introduce the full translation model, and sketch a decoder for ℓMBOT. Finally, an English-to-German translation task is used to evaluate the performance of the ℓMBOT model.

3.1 Shallow Local Multi Bottom-up Tree Transducers

In this section, we present the theoretical generative model used in this approach to syntax-based machine translation. Essentially, it is the *local multi bottom-up tree transducer* of Maletti (2011) but with one additional restriction. Multi Bottom-up Tree Transducers (MBOT) were first introduced by Arnold and Dauchet (1982) and Lilin (1978). The local multi bottom-up tree transducer restricts MBOT to a form that is particularly useful in statistical machine translation by replacing the finite-state behaviour by the common locality tests.

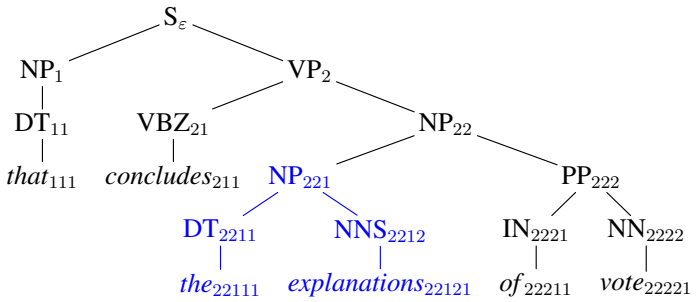

Figure 3.1: Example tree t with various illustrations. Positions are indicated and we have $t(21) = \text{VBZ}$. The subtree marked in blue is $t|_{221}$. The yield is emphasized.

Formal Definition of Trees As syntactic parse trees are utilized in this approach, we first give a formal introduction to trees. Given an alphabet Σ of labels, the set T_Σ of all Σ-*trees* is the smallest set T such that $\sigma(t_1, \ldots, t_k) \in T$ for all $\sigma \in \Sigma$, integers $k \geq 0$, and $t_1, \ldots, t_k \in T$. Intuitively, a tree t consists of a labeled root node σ followed by a sequence $t_1, \ldots, t_k$ of its subtrees. A tree $t \in T_\Sigma$ is *shallow* if $t = \sigma(t_1, \ldots, t_k)$ with $\sigma \in \Sigma$ and $t_1, \ldots, t_k \in \Sigma$, i.e. a tree is shallow if its height is 2.

A node inside a tree is addressed by its position, which is a word consisting of positive integers. Formally, the *positions* $\text{pos}(t) \subseteq \mathbb{N}^*$ of a tree $t = \sigma(t_1, \ldots, t_k)$ are inductively defined by $\text{pos}(t) = \{\varepsilon\} \cup \text{pos}^{(k)}(t_1, \ldots, t_k)$, where

$$\text{pos}^{(k)}(t_1, \ldots, t_k) = \bigcup_{1 \leq i \leq k} \{iw \mid w \in \text{pos}(t_i)\} \ .$$

In plain words, the root of a tree is addressed with the position ε (the empty word). The position iw with $i \in \mathbb{N}$ addresses the position w in the i^{th} direct subtree of the root. In this way, each node in the tree is assigned a unique position. We augment each node of the tree given in Figure 3.1 with its position. Let $t \in T_\Sigma$ and $w \in \text{pos}(t)$. The label of t at position w is $t(w)$, and the subtree rooted at position w is $t|_w$. These notions are also illustrated in Figure 3.1. A position $w \in \text{pos}(t)$ is a *leaf* (in t) if $w1 \notin \text{pos}(t)$. In other words, leaves do not have any subtrees. If all leaf nodes of a tree are concatenated (from left-to-right), then we obtain the *yield* of the tree. The yield of the example tree t in Figure 3.1 is illustrated by emphasizing all leaf nodes.

Given a subset $N \subseteq \Sigma$, we let

$$\mathrm{leaf}_N(t) = \{w \in \mathrm{pos}(t) \mid t(w) \in N,\ w \text{ leaf in } t\}$$

be the set of all leaves labeled by elements of N. When N is the set of nonterminals, we call them *leaf nonterminals*. We extend this notion to sequences $t_1, \ldots, t_k \in T_\Sigma$ by

$$\mathrm{leaf}_N^{(k)}(t_1, \ldots, t_k) = \bigcup_{1 \le i \le k} \{iw \mid w \in \mathrm{leaf}_N(t_i)\}.$$

Let $w_1, \ldots, w_n \in \mathrm{pos}(t)$ be (pairwise prefix-incomparable) positions and $t_1, \ldots, t_n \in T_\Sigma$. Then $t[w_i \leftarrow t_i]_{1 \le i \le n}$ denotes the tree that is obtained from t by replacing (in parallel) the subtrees at w_i by t_i for every $1 \le i \le n$.

Shallow Local Multi Bottom-up Tree Transducers Now we introduce the model, which is a minor variation of the local multi bottom-up tree transducer of Maletti (2011). Let Σ and Δ be the input and output symbols, respectively, and let $N \subseteq \Sigma \cup \Delta$ be the set of nonterminal symbols. Essentially, the model works on pairs $\langle s, (t_1, \ldots, t_k)\rangle$ consisting of a source tree fragment $s \in T_\Sigma$ and a sequence $t_1, \ldots, t_k \in T_\Delta$ of target tree fragments.[1] Such pairs are *sentential forms* of rank k, where k indicates the number of target tree fragments. The sentential form $\langle s, (t_1, \ldots, t_k)\rangle$ is *shallow* if all tree fragments $s, t_1, \ldots, t_k$ in it are shallow.

Together with a sentential form we typically have to store an alignment between nonterminal leaves. Given a sentential form $\langle s, (t_1, \ldots, t_k)\rangle$ of rank k and $1 \le i \le k$, we call t_i the i^{th} translation of s. An *alignment* for this sentential form is an injective mapping $\psi\colon \mathrm{leaf}_N^{(k)}(t_1, \ldots, t_k) \to \mathrm{leaf}_N(s) \times \mathbb{N}$ such that if $(w, i) \in \mathrm{ran}(\psi)$,[2] then also $(w, j) \in \mathrm{ran}(\psi)$ for all $1 \le j \le i$. In other words, if an alignment requests the i^{th} translation, then it should also request all previous translations.

[1] Formally, tree fragments are trees. We use this notion to differentiate between two concepts. Whenever we use the notion of *tree*, we refer to a complete parse tree. On the other hand, a *tree fragment* is only a part of a tree.

[2] $\mathrm{ran}(f)$ for a mapping $f\colon A \to B$ denotes the range of f, which is $\{f(a) \mid a \in A\}$.

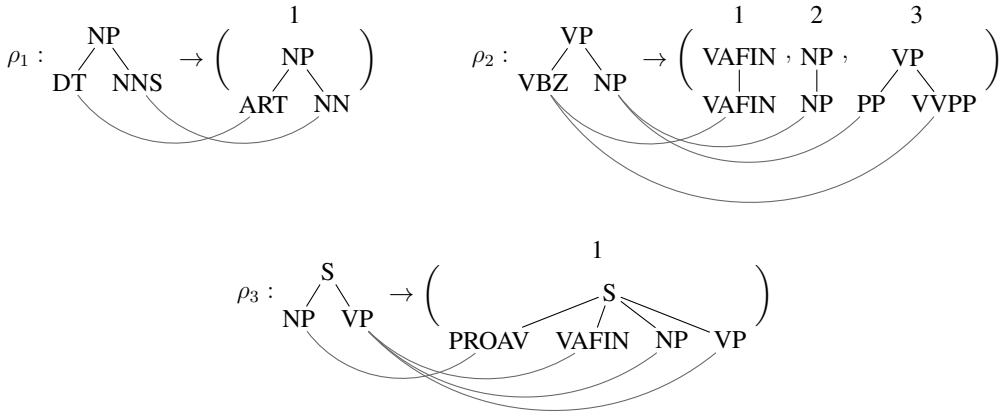

Figure 3.2: Sample ℓMBOT rules.

Definition 5 *A* shallow local multi bottom-up tree transducer *(ℓMBOT) is a finite set R of* rules *together with a mapping $c\colon R \to \mathbb{R}$ such that every rule ρ, written $s \to_\psi (t_1, \ldots, t_k)$, contains a shallow sentential form $\langle s, (t_1, \ldots, t_k)\rangle$ and an alignment ψ for it.*

The components s, $(t_1, \ldots, t_k)$, ψ, and $c(\rho)$ are called the *left-hand side*, the *right-hand side*, the *alignment*, and the *weight* of the rule $\rho = s \to_\psi (t_1, \ldots, t_k)$. Figure 3.2 shows three example ℓMBOT rules with annotated rank k and alignment between nonterminals indicated by links (but without weights). Overall, the rules of an ℓMBOT are similar to the rules of an SCFG (synchronous context-free grammar), but our right-hand sides contain a sequence of tree fragments instead of just a single tree fragment. In addition, the alignments in an SCFG rule are bijective between leaf nonterminals, whereas our model permits multiple alignments to a single leaf nonterminal in the left-hand side (see ρ_2 and ρ_3 of Figure 3.2).

Next, we define the traditional bottom-up semantics[3] to combine rules to form derivations. We need one final notion. Let $\rho = s \to_\psi (t_1, \ldots, t_k)$ be a rule and $w \in \mathrm{leaf}_N(s)$ be a leaf nonterminal (occurrence) in the left-hand side. The w-

[3]The derivations of a (multi) bottom-up tree transducer are typically defined in a bottom-up fashion. This is in opposition to standard synchronous grammars (see Section 2.1) and top-down transducers. An equivalent top-down semantics can be found in Maletti (2011).

rank $\mathrm{rk}(\rho, w)$ of the rule ρ is

$$\mathrm{rk}(\rho, w) = \max \{i \in \mathbb{N} \mid (w, i) \in \mathrm{ran}(\psi)\} \ .$$

For example, for the rules in Figure 3.2 we have

- $\mathrm{rk}(\rho_1, 1) = 1$ and $\mathrm{rk}(\rho_1, 2) = 1$,
- $\mathrm{rk}(\rho_2, 1) = 2$ and $\mathrm{rk}(\rho_2, 2) = 2$, and
- $\mathrm{rk}(\rho_3, 1) = 1$ and $\mathrm{rk}(\rho_3, 2) = 3$.

Definition 6 *The set $\tau(R, c)$ of* weighted sentential forms *of an ℓMBOT (R, c) is the smallest set T subject to the following restriction: If there exist*

- *a rule $\rho = s \to_\psi (t_1, \ldots, t_k) \in R$,*
- *a weighted sentential form $\langle s_w, c_w, (t_1^w, \ldots, t_{k_w}^w)\rangle \in T$ for every $w \in \mathrm{leaf}_N(s)$ with*
  - *$\mathrm{rk}(\rho, w) = k_w$,*[4]
  - *$s(w) = s_w(\varepsilon)$,*[5] *and*
  - *$t_i(w') = t_j^v(\varepsilon)$ with $\psi(iw') = (v, j)$ for every $iw' \in \mathrm{leaf}_N^{(k)}(t_1, \ldots, t_k)$,*[6]

then $\langle s', c', (t_1', \ldots, t_k')\rangle$ is a weighted sentential form, where

- *$s' = s[w \leftarrow s_w \mid w \in \mathrm{leaf}_N(s)]$,*
- *$c' = c(\rho) \cdot \prod_{w \in \mathrm{leaf}_N(s)} c_w$, and*
- *$t_i' = t_i[w' \leftarrow t_j^v \mid \psi(iw') = (v, j)]$ for every $1 \leq i \leq k$.*

Rules that do not contain any nonterminal leaves are automatically weighted sentential forms with their associated rule weight.[7] Otherwise, each nonterminal leaf w in the left-hand side of a rule ρ must be replaced by the source tree fragment s_w of a sentential form $\langle s_w, c_w, (t_1^w, \ldots, t_{k_w}^w)\rangle$, whose root is labeled by the same nonterminal. In addition, the rank $\mathrm{rk}(\rho, w)$ of the replaced nonterminal should match the number k_w of target tree fragments in the selected weighted sentential

[4] If w has n alignments, then the sentential form selected for it has to have suitably many target tree fragments.

[5] The labels have to coincide for the source tree fragment.

[6] Also the labels for the target tree fragments have to coincide.

[7] And consequently, a member of set $\tau(R, c)$.

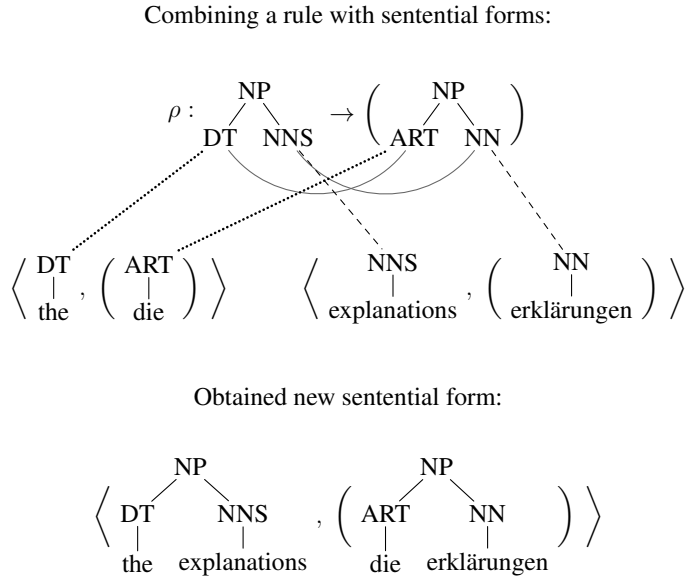

Figure 3.3: Simple rule application.

form. Finally, the nonterminals in the right-hand side that are aligned to w should be replaced by the translation that the alignment requests, provided that the nonterminal matches with the root symbol of the requested translation. The weight of the new sentential form is obtained simply by multiplying the rule weight and the weights of the selected weighted sentential forms. The derivation is finished when all introduced leaf nonterminals are substituted and the complete source sentence is derived. This requires that all discontinuities introduced by the model have to be resolved, hence the last applied rule needs to be continuous (see ρ_3 in Figure 3.2 for a rule that dissolves discontinuities). Therefore, the final weighted sentential form is of the form $\langle s, c, (t)\rangle$. The overall process is illustrated in Figures 3.3 and 3.4.

3.2 Rule Extraction

To obtain the set R of ℓMBOT rules, we implemented the rule extraction algorithm of Maletti (2011) which is repeated in Algorithm 1. The algorithm extracts rules from the sentence pairs of a word-aligned and bi-parsed parallel corpus, which

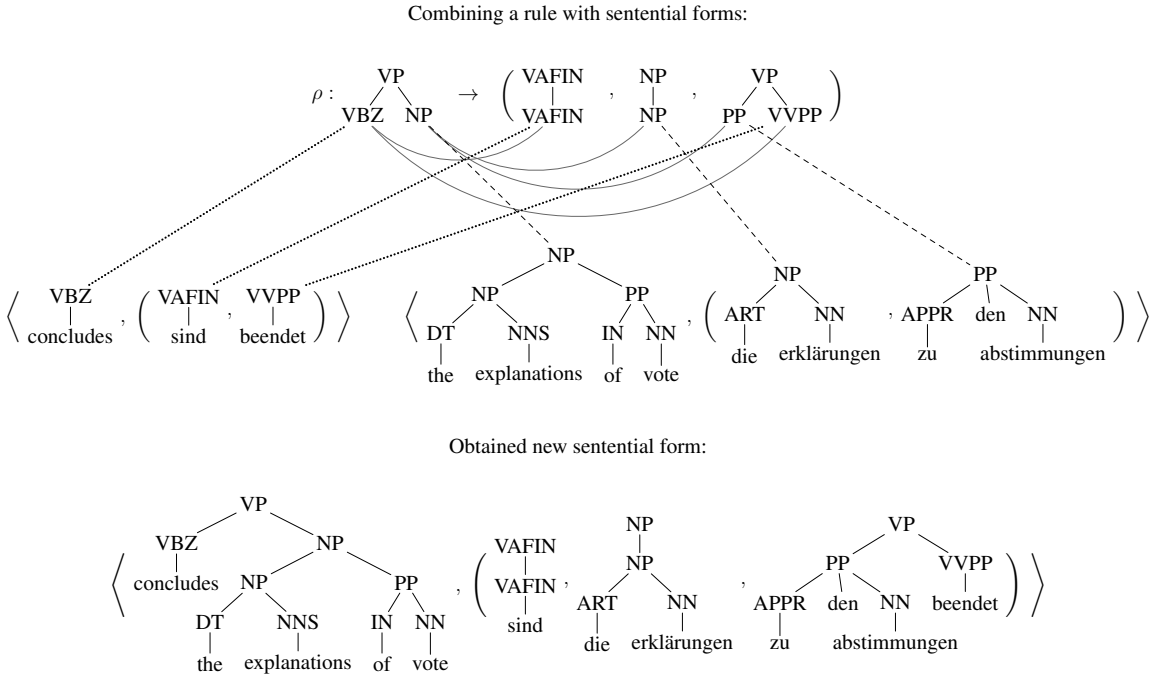

Figure 3.4: Complex rule application.

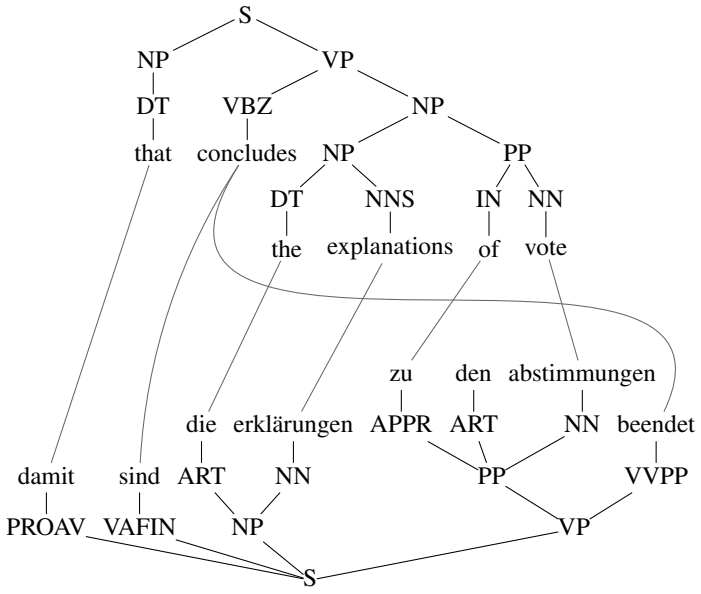

Figure 3.5: Word-aligned bi-parsed sentence pair.

means that parses are available for both the source and the target language sentences. An example of such a sentence pair is presented in Figure 3.5. The algorithm yields a unique decomposition of the sentence pair into tree fragments that do not overlap. To extract a rule for a given node in the source parse tree, the algorithm selects the minimal set of alignments that are necessary to obtain consistently aligned subtrees, which means that all alignments from descendants point into subtrees selected on the other side. For efficiency reasons, it selects the largest subtrees compatible with this selection of alignments. This approach is performed bottom-up and already extracted rules are removed from the parse trees. This trivially yields that each lexical item occurs in exactly one extracted rule. Such rules that only add context when necessary (or unaligned) are generally called *minimal*.

Algorithm 1 Rule extraction for minimal tree-to-tree ℓMBOT rules

Require: word-aligned tree pair (s, t)
Ensure: ℓMBOT rules R
1: **while** there exists a maximal non-leaf node $p \in pos(s)$ and minimal $p_1, ..., p_k \in pos(t)$ such that $s|_p$ and $(t|_{p_1}, ..., t|_{p_k})$ have a consistent alignment **do**
2: add rule $\rho = s|_p \rightarrow_\psi (t_{p_1}, ..., t_{p_k})$ to R with the nonterminal alignments ψ
3: $s \leftarrow s[p \leftarrow s(p)]$ // excise left-hand side of ρ from s
4: $t \leftarrow t[p_i \leftarrow t(p_i)]_{1 \leq i \leq k}$ // excise right-hand side of ρ from t
5: establish alignments according to position
6: **end while**

Let us explain the algorithm step by step. Given a sentence pair, we extract rules by looking for the maximal non-leaf node in the source tree and the minimal nodes in the target tree which have a consistent alignment (line 1). Consistent with the alignment means that no leaf node (terminal or nonterminal) from $s|_p$ is aligned to a leaf node outside of $(t|_{p_1}, ..., t|_{p_k})$ and vice versa. In line 2 we add rules fulfilling this requirement to the rule set R. If the rule contains nonterminal leaf nodes, we add their alignment information to the rule. After obtaining a new rule ρ, we excise the left-hand side of ρ from the source tree (line 3) by replacing the subtree located at $s|_p$ with the nonterminal at $s(p)$. Finally, in line 5 we establish the alignments between the newly introduced leaf nonterminals in the parse trees according to their position. These steps are repeated until all minimal rules are extracted. Figure 3.6

illustrates this procedure for one rule.

All rules that are extractable from the sentence pair of Figure 3.5 are shown in Figure 3.7. Note that these rules are not necessarily shallow. Thus, for each extracted rule $s|_p \rightarrow_\psi (t|_{p_1}, ..., t|_{p_k})$, we obtain its shallow counterpart $\mathrm{flat}(s|_p) \rightarrow_\psi \big(\mathrm{flat}(t|_{p_1}), \ldots, \mathrm{flat}(t|_{p_k})\big)$, where $\mathrm{flat}(u)$ removes all intermediate nodes (all nodes except the root and the leaves). The shallow rules corresponding to their non-shallow variants of Figure 3.7 are shown in Figure 3.8.

As explained in Section 2.1.1, all syntax-based machine translation systems require a *glue grammar*. A translation system based on ℓMBOT requires an additional *ℓMBOT glue grammar*. This grammar ensures that rules with discontinuous target tree fragments can be used in a continuous fashion, i.e. the root nodes of the target tree fragments are concatenated as siblings under a special nonterminal. The ℓMBOT glue rules are collected only from completely lexical rules. As the translation process is performed in a bottom-up chart, it suffices to ensure that those can be used continuously on the first (lexical) level. For larger spans, the standard glue rules (see Figure 2.7 in Section 2.1.1) can handle the partial translations. An example ℓMBOT glue rule is given in Figure 3.9.[8]

3.3 MBOTMOSES Decoder

The model presented above is implemented in the syntax-based component of the MOSES open-source toolkit (Koehn et al., 2007; Hoang et al., 2009). The standard MOSES syntax-based decoder handles SCFG rules only; i.e, rules with one single tree fragment on the source and the target language side. Roughly speaking, SCFG rules are ℓMBOT rules with exactly one target tree fragment. The MBOTMOSES branch supports ℓMBOT rules, in which arbitrarily many target tree fragments are allowed.

[8] In a MOSES system, the special nonterminal is 'X' whereas our special nonterminal is 'ROOT'. The standard and top glue rules for our system are also generated with 'ROOT'.

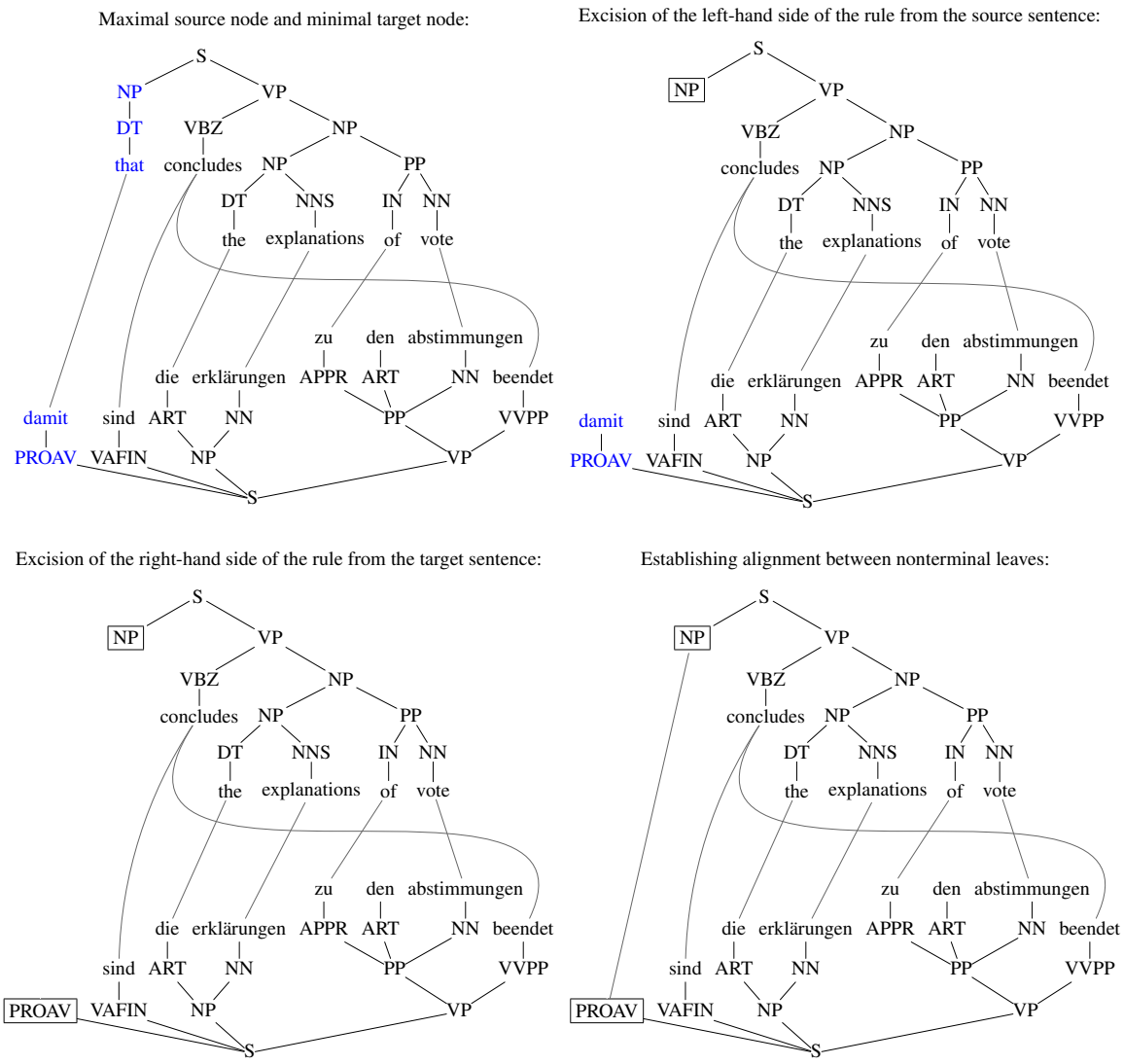

Figure 3.6: Illustration of the rule extraction algorithm.

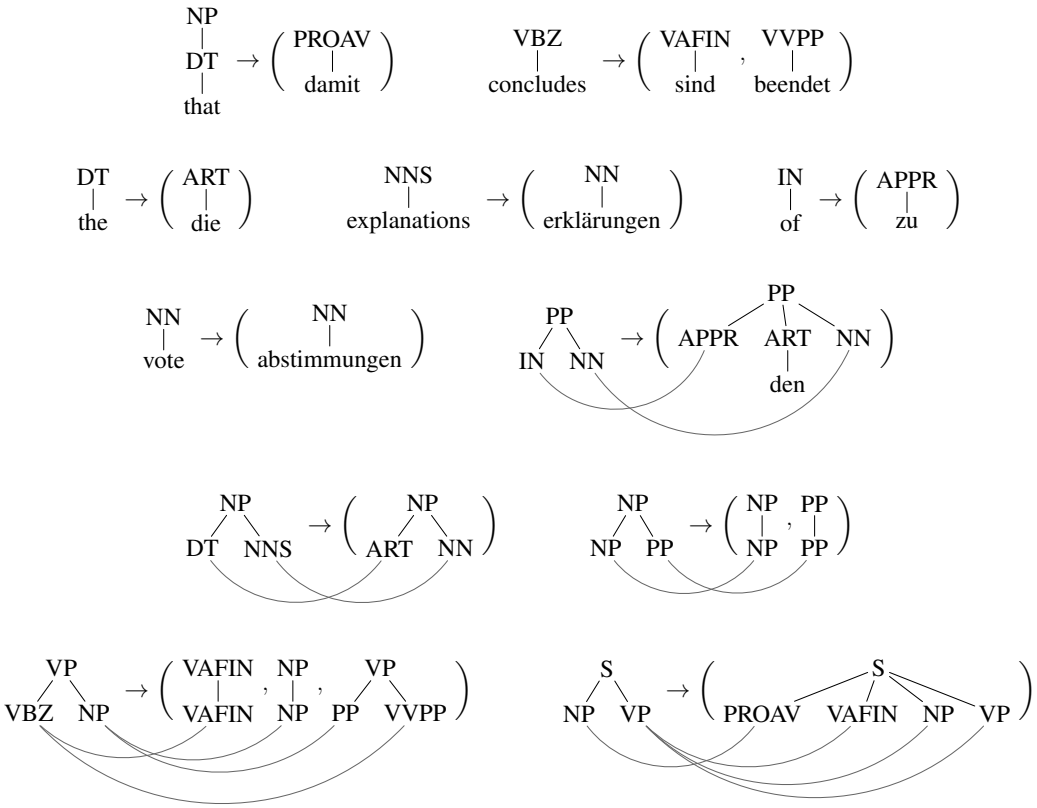

Figure 3.7: Extracted (even non-shallow) rules. We obtain our rules by making those rules shallow.

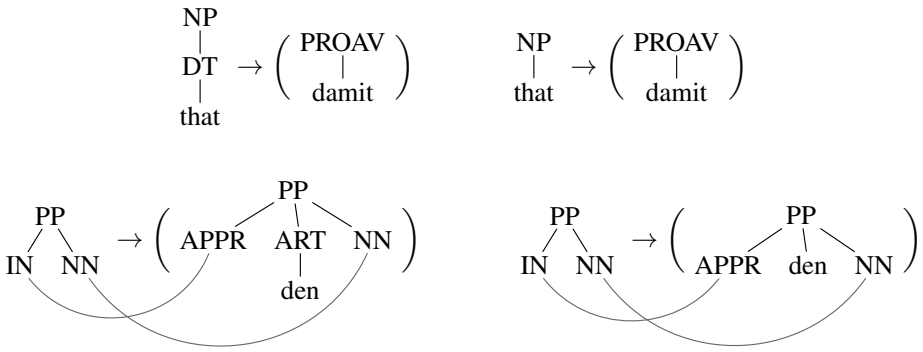

Figure 3.8: Non-shallow rules and their shallow counterparts.

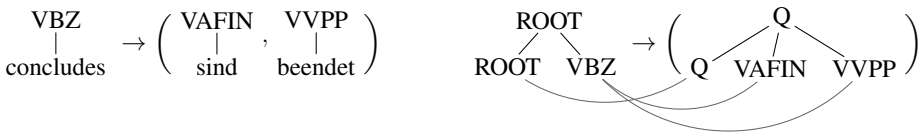

Figure 3.9: ℓMBOT glue rule (right) for the discontinuous rule displayed on the left.

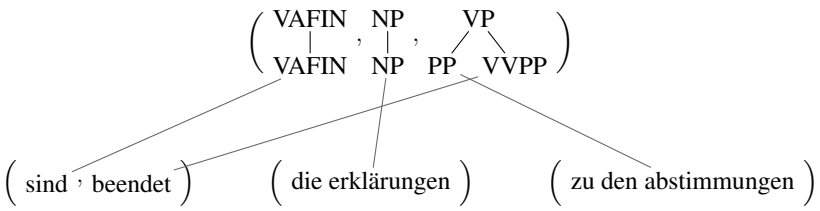

Figure 3.10: Illustration of LM scoring.

The MBOTMOSES syntax-based decoder uses a CYK+ chart parsing algorithm in which each source sentence is parsed and contiguous spans are processed in a bottom-up fashion. A rule is applicable[9] if the left-hand side of it matches the nonterminal assigned to the full span by the parser and the (non-)terminal assigned to each subspan.[10] In order to speed up the decoding process, cube pruning (Chiang, 2007) is applied to each chart cell in order to select the most likely hypotheses for subspans. The language model (LM) scoring is directly integrated into the cube pruning algorithm. Thus, LM estimates are available for all considered hypotheses. However, language model scoring inside the cube pruning algorithm was adapted to accommodate ℓMBOT rules. Because the target tree fragments can remain discontinuous after hypothesis creation, LM scoring has to be done individually for each target tree fragment. If a rule has rank k, then k strings $w_1, \ldots, w_k$ are used to collect the lexical information. These strings can later be used discontinuous, so all of them are LM-scored separately.

Figure 3.10 illustrates the LM scoring of a sentential form where a rule of rank 3 is applied (the construction of the sentential form is illustrated in Figure 3.4). When processing the rule, an LM estimate is computed by expanding all nonterminal leaves of the three target tree fragments rooted at (1) VAFIN, (2) NP, and (3) VP. This results in collecting in w_i the following lexical material:

- w_1: sind
- w_2: die erklärungen

[9]Note that the notion of applicable rules differs from the default in MOSES.

[10]Theoretically, this allows that the decoder ignores unary parser nonterminals, which could also disappear when we make our rules shallow; e.g., the left-hand side of ρ in Figure 3.3 can be matched by a rule with left-hand side NP(the, explanations).

- w_3: zu den abstimmungen beendet

Note how the nodes for VAFIN and VVPP are assembled from a (discontinuous) sentential form. This means that the terminals have been considered as independent until now. So far, the LM scorer could only score their associated unigrams. After application, the terminal leaf of VAFIN is still scored as a unigram but at least the terminal leaf of VVPP becomes part of a bigger target tree fragment. This results in a LM estimate computed by $score = LM(w_1) \cdot LM(w_2) \cdot LM(w_3)$. Clearly, the LM scores get more accurate for larger spans and when the last rule is applied[11], the complete output tree is generated and the LM scores are computed over the yield of said output tree.

A slightly more detailed overview can be found in Braune et al. (2013) and a detailed description is available in Braune (2015).

3.4 Translation Model

As usual, the task of the decoder is to find the best corresponding target language translation $\hat{f}$ of the source language sentence e licensed by the translation model and the language model, i.e.,

$$\hat{f} = \arg\max_f p(f \mid e) = \arg\max_f p(e \mid f) \cdot p(f)$$

The probability $p(f)$ is provided by the language model for the target side and the probability $p(f|e)$ by the ℓMBOT model. We estimate $p(e \mid f) \cdot p(f)$ by a log-linear model (Och, 2003) of features $h_m(\cdot)$ with weights λ_m scored on sentential forms $\langle s, (t) \rangle$ of our extracted ℓMBOT M such that the yield of s reads e[12] and the yield of t reads f.

[11] This is either a rule where the root nodes for the left- and right-hand side are start symbols of the source and target parse tree or a top glue rule. Both rules share the characteristic of having rank 1, hence there are no more discontinuities at this point and the complete output is scored.

[12] Actually, s must embed in the parse tree of e; see Section 3.3.

$$p(f|e) \propto \max_{\langle s,(t)\rangle} \prod_{m=1}^{10} h_m\big(\langle s,(t)\rangle\big)^{\lambda_m}$$

The features are scored on a sentential form $\langle s,(t)\rangle$ by scoring them on the rules used to build it and taking the product of the such obtained values. Our model uses the following features $h_m(s \rightarrow_\psi (t_1,\ldots,t_k))$ for a general rule $\rho = (s \rightarrow_\psi (t_1,\ldots,t_k)$[13]:

(1) the backward translation probability $\varphi\big(s|(t_1,\ldots,t_k)\big)$,
(2) the forward translation probability $\varphi\big((t_1,\ldots,t_k)|s\big)$,
(3) the backward lexical translation probability $lex\big(s|(t_1,\ldots,t_k)\big)$,
(4) the forward lexical translation probability $lex\big((t_1,\ldots,t_k)|s\big)$,
(5) the target side *language model*, scored directly on $(t_1,\ldots,t_k)$,
(6) the *word penalty*, i.e. a constant for each word in $(t_1,\ldots,t_k)$,
(7) the *rule penalty*, i.e. a constant for each applied rule,
(8) the *gap penalty*, i.e. a constant for each target tree fragment,
(9) a constant for each *glue grammar rule*, and
(10) a constant for each *ℓMBOT glue grammar rule*.

The rule weights required for (1) are relative frequencies normalized over all rules with the same right-hand side. In the same fashion, the rule weights required for (2) are relative frequencies normalized over all rules with the same left-hand side. Additionally, rules that were extracted at most 10 times are discounted by multiplying the rule weight by 10^{-2}.

The lexical weights for (3) and (4) are obtained by multiplying the word translations $w(f_i|e_j)$ [respectively, $w(e_j|f_i)$] of lexically aligned words (f_i, e_j) across (possibly discontinuous) target tree fragments.[14] Whenever a source word e_j is aligned to multiple target words, we average over the word translations.[15]

[13]Note that every rule ρ contains a shallow sentential form by definition (see Definition 5).
[14]The lexical alignments are different from the alignments used with a sentential form.
[15]If the word e_j has no alignment to a target word, then it is assumed to be aligned to a special NULL word and this alignment is scored.

$$h_3(\langle s, (t_1, \ldots, t_k)\rangle) = \prod_{\substack{\text{lexical item } e \\ \text{occurs in } s}} \text{average } \{w(f|e) \mid f \text{ aligned to } e\}$$

The computation of the language model estimates for (5) is adapted to score partial translations consisting of discontinuous units. This is explained in more detail in Section 3.3.

The features (8) and (10) are specific to the ℓMBOT model. The *gap penalty* is defined as 100^{1-k}, where k is equal to the rank of the rule. This feature is intended to allow the model to tune the amount of discontinuity to the specific target language, i.e. use rules with a higher or lower rank k. Finally, the ℓMBOT glue grammar is similar to the standard glue grammar (see Section 2.1.1). Either the sentential forms are built by substituting (discontinuous) rules or ℓMBOT glue rules are used to concatenate them in a continuous way.

Features (6), (7), and (9) coincide with the corresponding features explained in Section 2.1.4.

3.5 Experimental Results

In this section we report the setup and results as presented by Braune et al. (2013).

3.5.1 Setup

The baseline system for the experiments is the syntax-based component of the MOSES open-source toolkit (Koehn et al., 2007; Hoang et al., 2009). Linguistic syntactic annotation is used on both the source and the target language side (*tree-to-tree*). The contrastive system is the ℓMBOT-based translation system presented here. We provide this system with a set of minimal ℓMBOT rules (as presented in Section 3.2) as well as a set of SCFG rules. We do not impose any maximal span

English to German	
training data	4th Europarl corpus (Koehn, 2005); News commentary
training data size	≈ 1.5M sentence pairs
source-side parser	Charniak and Johnson (2005)
target-side parser	BitPar (Schmid, 2004)
language model	4-gram SRILM (Stolcke, 2002)
add. LM data	SdeWaC (Web-as-Corpus Consortium, 2008)
LM data size	≈ 44M sentences
tuning data	part of WMT 2009 (Callison-Burch et al., 2009)
tuning size	1,025 sentences
test data	part of WMT 2009
test size	1,026 sentences

Table 3.1: Summary of the experimental setup.

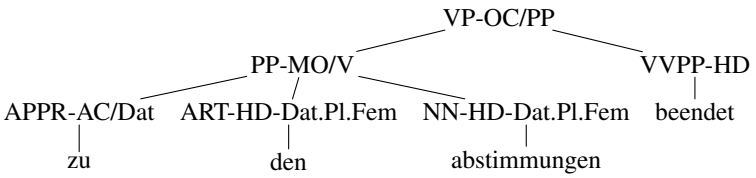

Figure 3.11: Labels with functional and morphological annotations.

restriction on either system.

For both systems, the used training data is from the 4th version of the *Europarl Corpus* (Koehn, 2005) and the *News Commentary* corpus. The corpora were length-ratio filtered up to 50 tokens per sentence. Both translation models were trained with approximately 1.5 million bilingual sentences. The word alignments were generated by GIZA++ (Och and Ney, 2003) with the *grow-diag-final-and* heuristic (Koehn et al., 2005). The English side of the bilingual data was parsed using the Charniak parser (Charniak and Johnson, 2005), and the German side was parsed using BitPar (Schmid, 2004). The German labels provided by BitPar are typically enriched with functional and morphological annotations. To avoid possible sparseness issues, those additional annotations were removed. Figure 3.11 shows the full labels assigned by BitPar for an excerpt of the German parse tree for our running example from Figure 3.5. We obtain our labels by removing everything after the first dash and the dash.

System	BLEU
tree-to-tree SCFG baseline	12.60
ℓMBOT	**13.06**

Table 3.2: Evaluation results. The bold result is a statistically significant improvement over the baseline (at confidence $p < 0.05$).

The 4-gram language model was trained on the German sentences in the training data augmented by the SdeWaC corpus (Web-as-Corpus Consortium, 2008), whose generation is detailed in Baroni et al. (2009). The weights λ_m in the log-linear model were trained using minimum error rate training (Och, 2003). We used the News 2009 test set to generate the tuning and the test set. The even numbered sentences were used for tuning and the odd numbered sentences for test. Table 3.1 gives additional information about the setup for our experiments.

Both systems use glue-rules as introduced before, which allow them to concatenate partial translations without performing any reordering.

3.5.2 Evaluation

We measured the overall translation quality with the help of 4-gram BLEU (Papineni et al., 2002), which was computed on tokenized and lowercased data for both systems. The results of our evaluation are reported in Table 3.2.

We can observe from Table 3.2 that our ℓMBOT-based system outperforms the baseline. We obtain a BLEU score of 13.06, which is a gain of 0.46 BLEU points over the baseline. This improvement is statistically significant at confidence $p < 0.05$, which we computed using the pairwise bootstrap resampling technique of Koehn (2004b).

Additionally, we report the number of ℓMBOT rules used by our system when decoding the test set. We try to estimate their impact on the translation quality by inspecting the statistics on the rules used in the sentential forms. Consequently, only rules that produce part of the final output counts. The current tools of MBOT-

	Lex	Struct	Total	Target tree fragments 2	3	4
continuous	23,175	18,355	41,530			
discontinuous	315	2,516	2,831	2,480	323	28

Table 3.3: Number of rules per type used when decoding test (Lex = lexical rules; Struct = structural rules).

MOSES (Braune et al., 2013) only allow the counting of rules used during decoding. At present, it is infeasible to track discontinuous fragments through the sentential forms to decide whether they are actually assembled continuously or discontinuously. Thus, discontinuous rules only indicate a potential discontinuity. We show the statistics in Table 3.3. By *Lex* we denote rules containing only lexical items. The label *Struct* stands for rules containing at least one leaf nonterminal. The results show that approx. 6% of all rules used by our ℓMBOT-system have discontinuous target tree fragments. Furthermore, the reported numbers show that the system also uses rules in which lexical items are combined with nonterminals. Moreover, Table 3.3 presents the number of rules with n target tree fragments used during decoding.

3.6 Summary

We presented a tree-to-tree ℓMBOT-based machine translation system which allows for discontinuous target tree fragments. A rule extraction algorithm as well as a decoder were already available but not yet evaluated. To this end, we implemented the algorithm to obtain a set of minimal rules. Furthermore, we evaluated our obtained translation system on an English-to-German translation task and obtained significant improvements over a standard (continuous) MOSES tree-to-tree translation system based on SCFG rules. Additionally, we inspected the statistics on the rules used in the sentential forms and can confirm that indeed discontinuous rules were used to decode the test set.

Chapter 4

Non-Minimal Tree-to-Tree ℓMBOT

We start this chapter with the motivation for non-minimal tree-to-tree ℓMBOT. Then we will introduce a new notion that enables us to sketch an efficient algorithm and explain the underlying idea of our rule extraction before presenting it in pseudo-code. Finally, we evaluate the obtained non-minimal ℓMBOT in three large-scale translation tasks. In addition, we will compare it to its minimal variant and a SCFG baseline as well as to a (hierarchical) phrase-based system.

4.1 Motivation

The ℓMBOT rule extraction (Maletti, 2011) presented in Chapter 3 extracts minimal rules only. It yields a unique decomposition of the sentence pair into tree fragments that do not overlap. Particularly, each lexical item of a sentence pair occurs in exactly one rule extracted from that sentence pair (as already explained in Section 3.2). However, minimal rules are unsuitable for fixed phrases consisting of rare words because minimal rules encourage small fragments and thus word-by-word translations. Consequently, such fixed phrases will often be assembled inconsistently by substitution from small fragments. Non-minimal rules encourage a consistent translation by covering larger parts of the sentence. This is shown by

DeNeefe et al. (2007), who used composed (i.e., non-minimal) rules to improve translation quality in the string-to-tree setting of Galley et al. (2004). With the hope of achieving similar improvements for ℓMBOT, we develop a rule extraction for non-minimal rules.

4.2 Rule Extraction

For the task at hand, the development of an efficient rule extraction algorithm for non-minimal tree-to-tree ℓMBOT rules, we essentially follow the approaches of Koehn et al. (2003), Och and Ney (2004), and Chiang (2007) which are all based on consistently aligned phrase pairs. We show how to extend this notion to consistently aligned rule spans. Since the algorithm works on the input and output strings, we first adjust our view on the training data. Instead of a word-aligned tree pair (s, t) as in the minimal case, we primarily use a word-aligned sentence pair $\langle e, A, f\rangle$ for which parses are available. Our training corpus consists of *word-aligned sentence pairs* $\langle e, A, f\rangle$ as introduced in *Hierarchical Phrase-based Model* of Section 2.1.1. Recall that a *source phrase* is a span $[i, i'] \subseteq [1, \ell_e]$ and correspondingly, a *target phrase* is a span $[j, j'] \subseteq [1, \ell_f]$. A *rule span* is a pair $\langle p, \varphi\rangle$ consisting of a source phrase p and a sequence $\varphi = p_1 \cdots p_n$ of (non-overlapping) target phrases $p_1, \ldots, p_n$.[1] If $n = 1$ (i.e., there is exactly one target phrase in φ) then $\langle p, \varphi\rangle$ is also a phrase pair (Koehn et al., 2003).

Next, we lift the notion of consistently aligned phrase pairs to our rule spans. Simply put, for a consistently aligned rule span $\langle p, p_1 \cdots p_n\rangle$ we require that it respects the alignment A in the sense that the origin i of an alignment $(i, j) \in A$ is covered by p if and only if the destination j is covered by $p_1, \ldots, p_n$. Formally, the rule span $\langle p, p_1 \cdots p_n\rangle$ is *consistently aligned* if for every $(i, j) \in A$ we have $i \in p$ if and only if $j \in \bigcup_{k=1}^{n} p_k$. For example, given the word-aligned sentence pair in Figure 4.1, the rule span $\langle [2, 4],\ [2, 4]\,[8, 8]\rangle$ is consistently aligned, whereas the phrase

[1] Spans overlap if their intersection is non-empty.

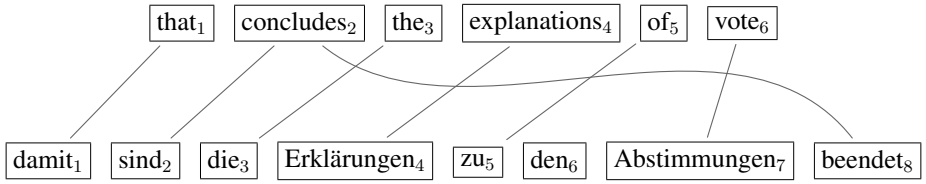

Figure 4.1: Word-aligned sentence pair.

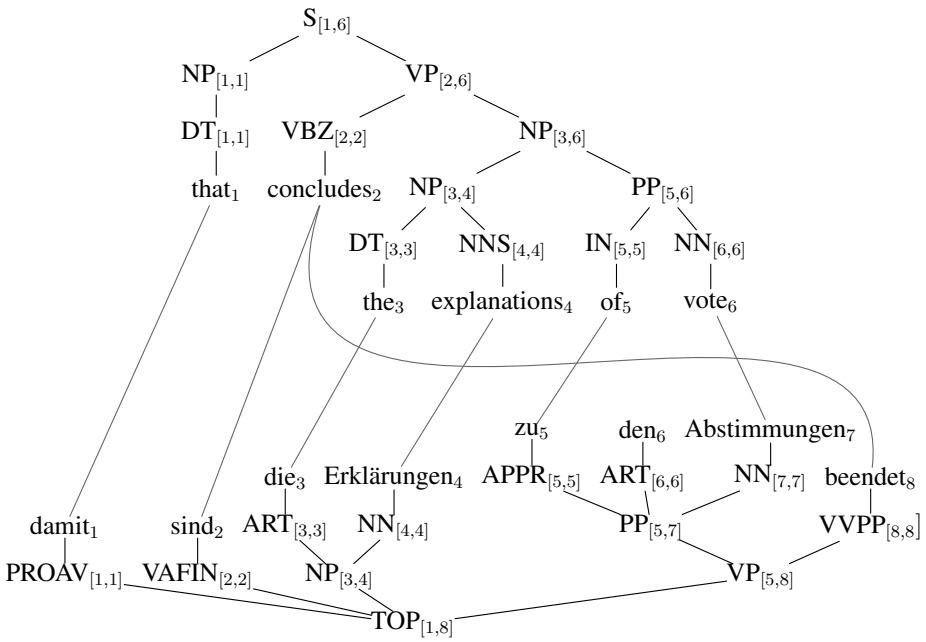

Figure 4.2: Bi-parsed word-aligned sentence pair.

pair $\langle [2,4],\ [2,8] \rangle$ is not.

We still incorporate parse trees for both our source and target language. Thus, for each word-aligned sentence pair $\langle e, A, f \rangle$ we have a parse tree s for e and a parse tree t for f. This is illustrated in Figure 4.2. In the minimal tree-to-tree ℓMBOT, we addressed a tree fragment u by its position w. Each such node w of a parse tree *governs* a unique phrase. We have indicated those phrases as subscript to the non-lexical node labels in the source and the target parse trees of Figure 4.2 for our running example. A consistently aligned rule span $\langle p, p_1 \cdots p_n \rangle$ of $\langle e, A, f \rangle$ is *compatible with s and t* if (i) there exists a node η of s such that η governs p and (ii) there exist nodes $\eta_1, \ldots, \eta_n$ of t such that η_k governs p_k for all $1 \leq k \leq n$. For

example, given the word-aligned sentence pair and parse trees s and t in Figure 4.2, the consistently aligned rule span $\langle [2,4],\ [2,4]\,[8,8]\rangle$ is (i) not compatible with s because there is no node in s that governs $[2,4]$ and (ii) not compatible with t because there is no node in t that governs $[2,4]$. However, we can have two separate rule spans that cover this data. The rule spans $\langle [2,2],\ [2,2]\,[8,8]\rangle$ and $\langle [3,4],\ [3,4]\rangle$ are both consistent with the alignment and compatible with s and t. Note that not every consistently aligned rule span compatible with s is necessarily compatible with t. For example $\langle [3,6],\ [3,7]\rangle$ is consistently aligned and compatible with s but not compatible with t. For the same data, rule span $\langle [3,6],\ [3,4]\,[5,7]\rangle$ is consistently aligned and compatible with both s and t.

Now we can sketch the algorithm. At first, we extract initial rules and from those we derive additional rules. We continue to work with the word-aligned sentence pair $\langle e, A, f\rangle$ with parse trees s for e and t for f. For each consistently aligned rule span $\langle p, p_1 \cdots p_n\rangle$ that is compatible with both s and t we extract the rule $\mathrm{flat}(s_\eta) \to \big(\mathrm{flat}(t_{\eta_1}), \ldots, \mathrm{flat}(t_{\eta_n})\big)$, where

- $\mathrm{flat}(u)$ removes all internal nodes from u,
- s_η is the tree fragment rooted in a node η of s that governs p, and
- t_{η_k} is the tree fragment rooted in η_k of t for each selection of nodes $\eta_1, \ldots, \eta_n$ such that n_k governs p_k for each $1 \leq k \leq n$.

The rules obtained in this manner are called *initial rules for* $\langle e, A, f\rangle$*,* s*, and* t. For example, for the rule span $\langle [2,2],\ [2,2]\,[8,8]\rangle$ we extract only one initial rule. More precisely, we have

- $s_\eta =$ (VBZ concludes),
- $t_{\eta_1} =$ (VAFIN sind),
- and $t_{\eta_2} =$ (VVPP beendet).

The function flat leaves s_η, t_{η_1}, and t_{η_2} unchanged. But for the rule span $\langle [3,4],\ [3,4]\rangle$ we extract

- $s_\eta =$ (NP (DT the) (NNS explanations)) and

- $t_{\eta_1} = \big(\text{NP } (\text{ART die}) \; (\text{NN Erklärungen})\big)$

for which the function returns $\text{flat}(s_\eta) = (\text{NP the explanations})$ and $\text{flat}(t_{\eta_1}) = (\text{NP die Erklärungen})$. We display all initial rules extractable for our running example in Figure 4.3. In principle there are even further nonsense initial rules like those that introduce unnecessary discontinuity which we want to prevent. The boxed rules in Figure 4.3 are such nonsense rules. The rule on the left introduces a unnecessary discontinuity because the target parse tree allows for a consecutive span and hence a continuous rule which is displayed on top. For the rule on the right a continuous span is not given by the target parse tree but we still consider it an unnecessary discontinuity because the token 'den' at [6,6] is an unaligned word and we do not wish to induce more discontinuity by such tokens.

All initial rules are completely lexical in the sense that they never contain a non-lexical leaf in the source nor in any target tree fragment. We introduce non-lexical rules using the same approach as for the hierarchical rules of Chiang (2007). Roughly speaking, we obtain a new rule r'' by "excising" an initial rule r from another rule r' and replacing the removed part by

- the root label of the left-hand side of r in the source tree fragment of r'
- the root label(s) of the right-hand side of r in the target tree fragment(s) of r', and
- linking the removed parts appropriately,

so that the flatted substitution of r into r'' can yield r'. This "excision" process is illustrated in Figure 4.4, where we remove the middle initial rule from the topmost initial rule. The result is displayed at the bottom in Figure 4.4. Formally, the set of *extractable rules* R for a given word-aligned sentence pair $\langle e, A, f \rangle$ with parse trees s for e and t for f is the smallest set subject to the following two conditions:

- Each initial rule is in R and thus extractable.
- For every initial rule r and extractable rule $r' \in R$, any flat rule r'', into which we can substitute r to obtain ρ with $\text{flat}(\rho) = r'$, is in R and thus extractable.[2]

[2]A rule $\rho = s \to (t_1, \ldots, t_n)$ is *flat* if $\text{flat}(\rho) = \rho$, where $\text{flat}(\rho) = \text{flat}(s) \to (\text{flat}(t_1), \ldots, \text{flat}(t_n))$.

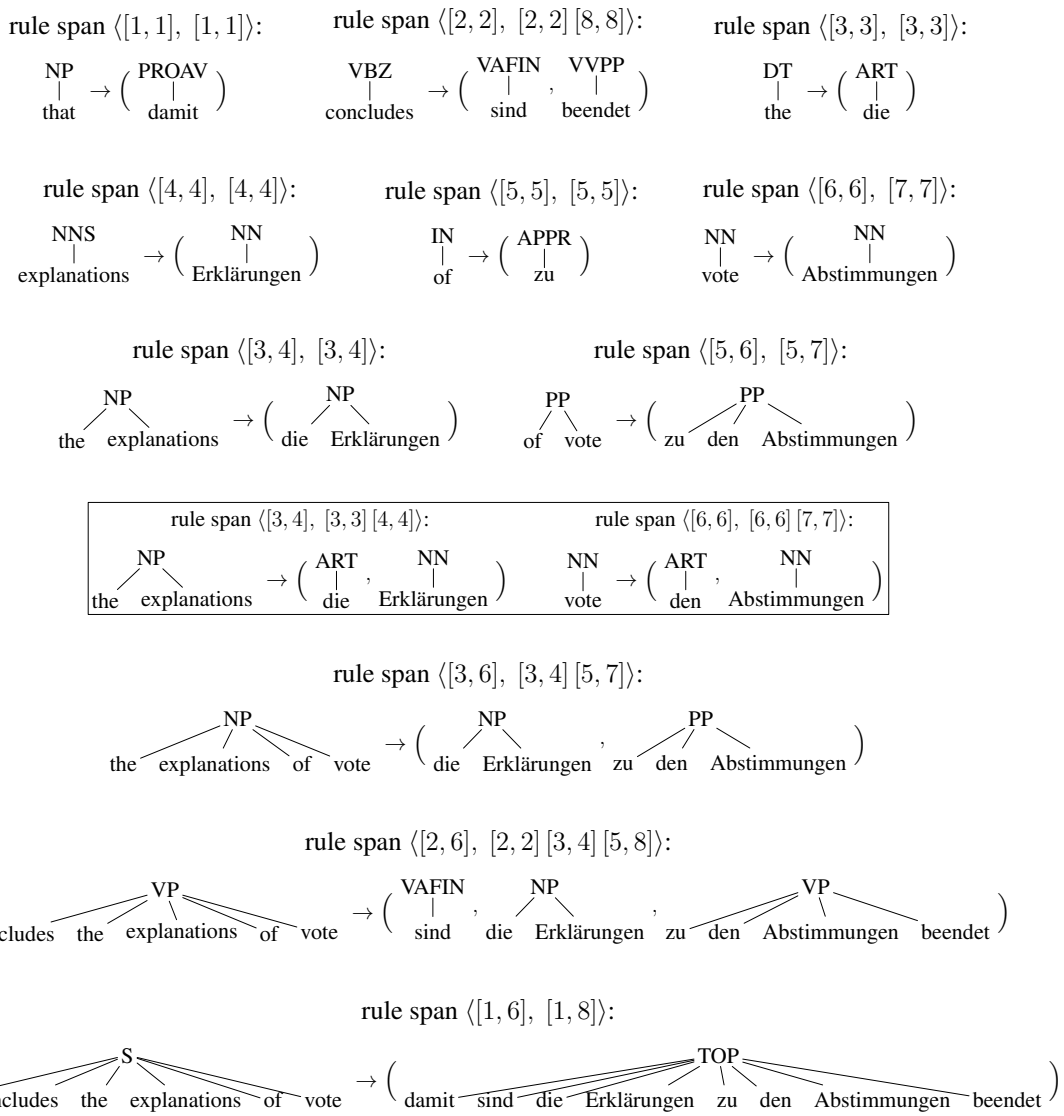

Figure 4.3: Some initial rules extractable from the bi-parsed and word-aligned sentence pair of Figure 4.2.

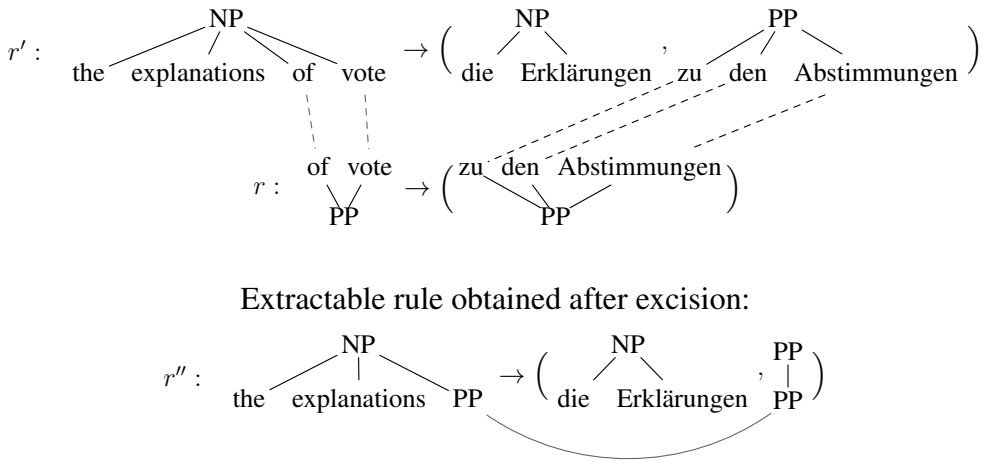

Figure 4.4: Excision of the middle initial rule from the topmost initial rule.

As known from the literature, obtaining the set of all extractable rules is not feasible. This set is generally too large and leads to slow training, slow decoding, and spurious ambiguity. Hence, it is sensible to restrict the number of extracted rules. This typically results in a trade-off between speed and translation quality. Our ℓMBOT rules are restricted by the parse trees for the source and target sentences, but due to the presence of multiple target tree fragments, the ℓMBOT model permits additional flexibility. Consequently, the following additional constraints on rules $s \to (t_1, \ldots, t_n)$ are enforced.[3]

(a) We only consider source phrases p of length at most 10 (i.e., $i' - i < 10$ for $p = [i, i']$). This restricts the set of initial rules. Allowing a greater length results in very slow training. In Figure 4.5 we show the increase of training times for different span sizes and corpus sizes. Interestingly, there is no real difference between a span of 40 and a span of 60 as both settings result in equally slow training.

(b) We only excise initial rules with source phrase p of length at least 2 (i.e. $i' - i \geq 1$ for $p = [i, i']$). This restricts the set of extractable rules. I believe that there is not much to be gained from the excision of a rule containing only a single leaf

[3]The constraints are handled by name-value parameters given to the algorithm. The default values can be easily modified during training.

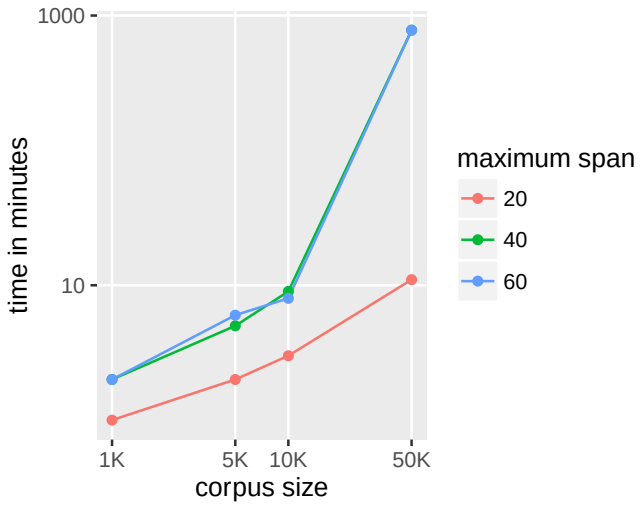

Figure 4.5: Training times with different settings of maximum size.

System	**Number of extracted rules**		
	English-To-German	English-To-Arabic	English-To-Chinese
tree-to-tree SCFG	6,630,590	24,358,001	8,161,362
minimal tree-to-tree ℓMBOT	12,478,160	28,725,229	10,162,325
tree-to-tree ℓMBOT	40,736,687	151,322,970	84,220,528

Table 4.1: Number of extracted rules for the different rule extractions.

terminal.

(c) The leaves of the source tree fragment s consist of at most 5 occurrences of lexical items or nonterminals (i.e. $\ell_s \leq 5$). This has an effect on both initial and extractable rules. While we still consider an initial rule with source phrase p of length 6–10, we do not keep it for decoding. Its only purpose is to allow us to derive extractable rules from it. For example, we do not output the bottom rule in Figure 4.3.

To give a quick overview, we report the number of extracted rules for all translation tasks and rule extractions in Table 4.1. We can immediately confirm that—even in the restricted tree-to-tree setting—the number of extracted ℓMBOT rules for each translation task greatly increases when compared to the SCFG baseline.

The pseudo-code of the extraction algorithm is given in Algorithm 2. We obtain initial rules by iterating over the length of the source sentence. In fact, we start by

looking at source spans of increasing length up to length $maxSpan$[4]. We define the span $[i, i']$ of the source phrase p by the index of the current source word plus the current length (see lines 2–4). This source span has to be compatible with s and we check if a node η spans $[i, i']$. When there is no node that covers this span, we stop immediately and continue with the iteration (lines 5–9). The function FINDCONSISTENTALIGNED is called with the alignment, the current compatible source span, and the parse tree t. Given the span $[i, i']$ and the alignment A, we obtain the alignment points $(k, j) \in A$ for all $k \in [i, i']$. Based on these alignments, we determine the consistently aligned target spans $p_1, \ldots, p_n$. Given these spans, we access the tree fragments rooted at t_{η_k} for each p_k with $1 \leq k \leq n$ and the function returns these (line 10). Finally, in line 11 we add the initial rule to R. We obtain a new rule ρ with unaligned words on the target language side by searching for each target tree fragment t_k whether there are unaligned words to the left or right.[5] We do not want to introduce more discontinuities by unaligned words, hence we ensure in the function FINDUNALIGNEDWORDS that there is a node that covers the new span consisting of the original span p_k plus the span of the unaligned word(s). When this is given, we add the rule to R (lines 13–17). We obtain extractable rules for each initial rule as well as for possible rules with unaligned words by calling EXTRACTABLERULES (line 12 and line 16). In this function (lines 23–34) we are basically using the same approach as for initial rules. We iterate over the source span of r' incrementally increasing the length. In the first iteration, we make sure that the span covers the minimum size (see constraint (b); lines 24–25). Given this new source span, we retrieve all initial rules from R that cover this span (line 26). There can be multiple rules due to unaligned words. In lines 27–31 we excise each rule r from r' as shown in Figure 4.4 and add the result to R_E. Then EXTRACTABLERULES is called again but with r'' and $next\_i$ is defined as the position that comes right after the excised part (line 30). By calling EXTRACTABLERULES recursively, all possible combinations of excisions are considered. The last step is to output all initial and extractable rules

[4]This is the implementation of constraint (a).

[5]Unaligned source words are automatically considered as we iterate over the source sentence.

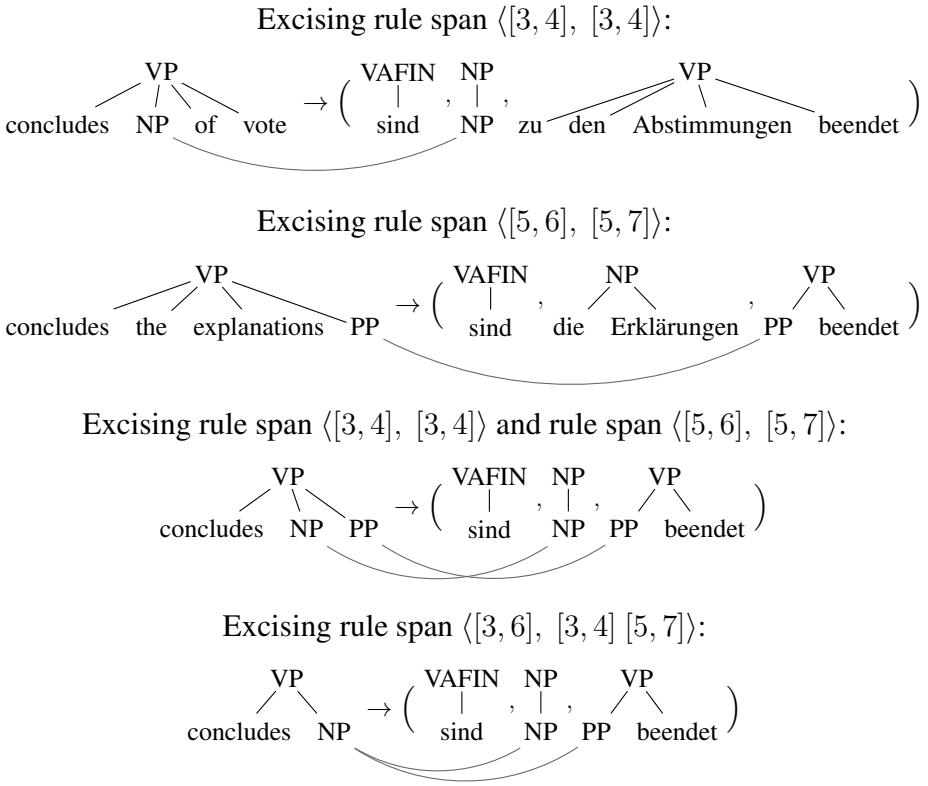

Figure 4.6: Some extractable rules for the rule span $\langle [2,6],\ [2,2]\ [3,4]\ [5,8] \rangle$.

for the given sentence pair that are conform with constraint (c) (line 20).

Figure 4.6 shows some extractable rules for the initial rule $\langle [2,6],\ [2,2][3,4][5,8] \rangle$.

4.3 Translation Model

We use the log-linear model introduced in Section 3.4 to estimate the features $h_m(\cdot)$ with weights λ_m scored on sentential forms $\langle s, (t) \rangle$ of our extracted ℓMBOT such that the yield of s reads e and the yield of t reads f. Our model uses the same ten features and scores them as explained in Section 3.4. There is only one minor difference when computing the translation weights for $\varphi\big(s|(t_1,\ldots,t_n)\big)$ and $\varphi\big((t_1,\ldots,t_n)|s\big)$, respectively: The probabilities of all rules that were extracted at

Algorithm 2 Rule extraction for non-minimal tree-to-tree ℓMBOT rules

Require: word-aligned sentence pair $\langle e, A, f \rangle$ with parse trees s and t
Ensure: ℓMBOT rules R

function INITIALRULES(e, A, f, s, t)
  for $length \leftarrow 1,\ length \leq maxSpan,\ length++$ **do** // ensures constraint (a)
    for $i \leftarrow 0,\ i < |e| - (length - 1),\ i++$ **do**
      $i' \leftarrow i + length - 1$
      if there exists node η of s spanning $[i, i']$ **then** // p compatible with s
        $s \leftarrow$ subtree rooted in η
      else
        continue
      end if
      $(t_1, \ldots, t_n) \leftarrow$ FINDCONSISTENTALIGNED(A, i, i', t)
      $R[i, i'] \leftarrow R[i, i'] \cup \{r' : \langle s \rightarrow (t_1, \ldots, t_n) \rangle\}$ // add initial rule to R
      EXTRACTABLERULES(i, i', r') // find extractable rules
      for all $1 \leq k \leq n$ **do**
        $\rho \leftarrow$ FINDUNALIGNEDWORDS(r', t_k, A)
        $R[i, i'] \leftarrow R[i, i'] \cup \rho$
        EXTRACTABLERULES(i, i', ρ)
      end for
    end for
  end for
  OUTPUTRULES(R, R_E) // ensures constraint (c)
end function

function EXTRACTABLERULES($next\_i, end, r'$)
  for $i \leftarrow next\_i,\ i \leq end,\ i++$ **do**
    for $i' \leftarrow i + (minSpanSource - 1),\ i' \leq end,\ i'++$ **do** // ensures constraint (b)
      $R' \leftarrow R[i, i']$ // retrieve initial rules covering this span
      for all $r \in R'$ **do**
        $r'' \leftarrow$ EXCISE(r, r')
        $R_E[i, i'] \leftarrow R_E[i, i'] \cup \{r''\}$
        EXTRACTABLERULES($i + 1, end, r''$)
      end for
    end for
  end for
end function

most 10 times get discounted by applying Good-Turing smoothing (Good, 1953).[6] This smoothing technique penalizes low frequency rules more than rules with a higher frequency.

In the following experiments, the log-linear model with the adjusted discounting is used to score the minimal and the non-minimal ℓMBOT. The decoder for the ℓMBOT model is provided by the MBOTMOSES branch (Braune et al., 2013) of MOSES as presented in Section 3.3.

4.4 Experimental Results

In this section, we show the setup and results as presented in Seemann et al. (2015b). The main contribution is the experimental evaluation of ℓMBOTs in the tree-to-tree setting. We also compare to standard models in order to evaluate the effect of the discontinuities offered by the ℓMBOT models. We chose to perform experiments for the translation directions English-to-German, English-to-Arabic, and English-to-Chinese. The languages were selected such that constituency parsers and large parallel corpora are readily available. In addition, we selected target languages in which the discontinuity offered by the ℓMBOT models might be useful.

4.4.1 Setup

We summarize the experimental setup in Table 4.2. We applied length-ratio filtering to all training sets by limiting the numbers of token to 80 per sentence. Furthermore, all training sets have been word aligned using GIZA++ (Och and Ney, 2003) using the *grow-diag-final-and* heuristic (Koehn et al., 2005).

The tasks required various forms of preprocessing of the data. The English (source) side of the training data was true-cased and parsed with the provided grammar of the Berkeley parser (Petrov et al., 2006). Next, we comment on the

[6] For the experiments presented in Section 3.5, these rules were discounted by a fixed value.

	English to German	English to Arabic	English to Chinese
training data	7th Europarl (Koehn, 2005)	MultiUN (Eisele and Chen, 2010)	
training data size	≈ 1.8M sentence pairs	≈ 5.7M sentence pairs	≈ 1.9M sentence pairs
source-side parser	Berkeley parser (Petrov et al., 2006)		
target-side parser	BitPar (Schmid, 2004)	Berkeley parser (Petrov et al., 2006)	
language model	5-gram SRILM (Stolcke, 2002)		
add. LM data	WMT 2013 (Bojar et al., 2013)	Arabic in MultiUN	Chinese in MultiUN
LM data size	≈ 57M sentences	≈ 9.7M sentences	≈ 9.5M sentences
tuning data	WMT 2013	cut from MultiUN	NIST 2002, 2003, 2005
tuning size	3,000 sentences	2,000 sentences	2,879 sentences
test data	WMT 2013	cut from MultiUN	NIST 2008 (NIST, 2010)
test size	3,000 sentences	1,000 sentences	1,859 sentences

Table 4.2: Summary of the used resources.

preprocessing tasks that are specific for each translation task.

- *English-to-German:* The German text was also true-cased and parsed with the provided grammar of BitPar (Schmid, 2004). As for the experiments in Section 3.5.1, we removed the functional and morphological annotation from the tags used in the parses.
- *English-to-Arabic:* The Arabic text was tokenized with MADA (Habash et al., 2009) and transliterated according to Buckwalter (2002). Since the Berkeley parser also provides a grammar for Arabic, we parsed the Arabic training data with it.
- *English-to-Chinese:* The Chinese sentences were word-segmented using the Stanford Word Segmenter (Chang et al., 2008). Again, the Berkeley parser-with its provided grammar delivers the parse trees for the Chinese training data.

After the preprocessing steps, we obtained a word-aligned, bi-parsed parallel corpus, to which we applied the rule extractions as described in Sections 3.2 and 4.2 together with the SCFG rule extraction provided by MOSES (Koehn et al., 2007; Hoang et al., 2009), which we use as a baseline.

In all experiments the feature weights λ_m of the log-linear model were trained using minimum error rate training (Och, 2003).

	BLEU		
System	**En-De**	**En-Ar**	**En-Zh**
tree-to-tree SCFG (baseline)	14.50	43.49	17.63
minimal ℓMBOT	14.09	32.88	12.01
non-minimal ℓMBOT	14.41	41.37	16.77
hierarchical phrase-based	17.00	51.71	18.74
phrase-based	16.80	51.90	18.09

Table 4.3: BLEU evaluation results for all 3 translation tasks.

4.4.2 Evaluation

We start by comparing all systems to each other using the BLEU score (Papineni et al., 2002). We also present the results obtained by a phrase-based system (Koehn et al., 2003) and a hierarchical phrase-based system (Chiang, 2005). Both systems were trained with their respective standard settings within the MOSES toolkit. All systems were tuned for BLEU on the tuning data, and we report the BLEU scores obtained by the tuned systems on the true-cased test set for German, on the transliterated test set for Arabic, and on the word-segmented test set for Chinese.

We performed large scale experiments on three major translation tasks, namely English-to-German (En-De), English-to-Arabic (En-Ar), and English-to-Chinese (En-Zh). The goal was to evaluate the following ℓMBOT systems: (i) the minimal tree-to-tree system (Chapter 3), and (ii) the non-minimal tree-to-tree system. The obtained results are reported in Table 4.3.

Let us now discuss the results for the various settings. Overall, we observe that the tree-to-tree systems perform worse than a system incorporating no syntax or a system that is only formally syntax-based. For the baseline system using SCFG rules (i.e., ℓMBOT rules with a single tree fragment on both the left- and right-hand side), this result is not surprising. Already, Lavie et al. (2008) have shown that tree-to-tree rules are too restrictive to achieve good lexical coverage. However, our results show that making rules more flexible by allowing several target tree fragments hurts the performance instead of yielding improvements. This effect is particularly visible

when only using minimal rules. On the English-to-Arabic and English-to-Chinese translation tasks, the minimal ℓMBOT system loses 10.61 and 5.62 BLEU points, respectively, over the baseline. Interestingly, on the English-to-German translation task the loss is only 0.41 BLEU points. Adding non-minimal ℓMBOT rules yields the expected large improvements, but is overall still not good enough to beat the tree-to-tree baseline. This result is interesting insofar as it does not confirm the results of Sun et al. (2009) on large scale experiments with target side discontinuities.[7]

4.4.3 Analysis of Discontinuity

Another goal was to identify whether discontinuous rules are useful and to what extent these are useful. Consequently, we inspected the statistics on the rules used to derive the final sentential forms. Table 4.4 shows the statistics on the rules used during decoding. Continuous rules (i.e., rules with a single tree fragment in the right-hand side) are abbreviated by *cont*, and (potentially)[8] discontinuous rules are abbreviated by *discont*. To provide a deeper analysis, we also distinguish between lexical and structural rules. Lexical rules, abbreviated *Lex*, are rules that contain no leaf nonterminal. Similarly structural rules, abbreviated *Struct*, are rules containing at least one such nonterminal. In this sense, lexical rules are purely lexical and structural rules can contain lexical material.

Minimal ℓMBOT If we only use minimal ℓMBOT rules, then 13.7% on the English-to-German, 12.7% on the English-to-Arabic, and 10.0% on the English-to-Chinese translation task of the rules used during decoding are discontinuous. For all the translation tasks, the majority of the discontinuous rules are structural. This fact is not very surprising since the leaves of the minimal tree-to-tree rules are either lexi-

[7] The experiments of Sun et al. (2009) report scores for the translation task Chinese-to-English for systems trained on less than 500,000 sentences only. Their model allows discontinuities on the source language side, which should be comparable to target-side discontinuities for the opposite translation direction English-to-Chinese.

[8] As explained in Section 3.5.2, we do not know whether a rule was used in a real discontinuous fashion or not.

		English-to-German		**English-to-Arabic**		**English-to-Chinese**	
ℓMBOT	**Type**	*cont.*	*discont.*	*cont.*	*discont.*	*cont.*	*discont.*
minimal	*Lex*	55,910	2,167	18,389	1,138	34,275	516
	Struct	4,492	7,386	2,855	1,920	8,820	4,292
	Total	60,402	9,553	21,244	3,085	43,095	4,808
		69,955		24,329		47,903	
non-minimal	*Lex*	44,951	4,149	9,826	1,605	35,031	771
	Struct	2,850	2,348	1,581	746	2,045	744
	Total	47,801	6,497	11,407	2,315	37,076	1,515
		54,298		13,722		38,591	

Table 4.4: Number of rules per type used when decoding test (Lex = lexical rules; Struct = structural rules; [dis]cont. = [dis]continuous).

cal items or leaf nonterminals. The minimality constraint encourages word-by-word translation, and once the lexical rules are excised, only structural rules remain. Based on the observed high BLEU score losses, it seems that minimal tree-to-tree rules are, at present, unable to correctly assemble discontinuous parts.

Non-minimal ℓMBOT If we additionally use non-minimal tree-to-tree rules, it is observable that we overall need fewer rules to decode the test sets. This is expected as non-minimal rules can cover larger parts of a sentence. Furthermore, the rate of discontinuous rules changes. For English-to-German the rate drops to 12.0%, whereas for the English-to-Arabic translation task it increases to 16.9%. Finally, for English-to-Chinese suddenly only 4% of the rules applied during decoding are discontinuous. The non-minimality encourages large rules, which are more likely to contain only lexical items. As expected, the number of discontinuous lexical rules is always larger than the number of discontinuous structural rules in this setting. This is particularly true for English-to-German and English-to-Arabic, where almost two thirds of the discontinuous rules are lexical, whereas the distribution is almost even for English-to-Chinese. We believe that these lexical discontinuous rules capture relevant idiomatic expressions or encode agreement or correspondences, which yield the large improvements in translation quality over minimal rules only.

In Table 4.5 we distinguish the discontinuous rules further with respect to the

Language Pair	ℓMBOT	Target tree fragments 2	3	4	5	≥ 6
English-to-German	minimal	6,458	2,389	471	34	1
	non-minimal	5,601	821	62	13	–
English-to-Arabic	minimal	2,525	455	67	8	3
	non-minimal	1,577	565	158	38	13
English-to-Chinese	minimal	3,816	900	82	6	4
	non-minimal	1,222	248	36	4	5

Table 4.5: Number of target tree fragments for all three translation tasks and the two different settings.

number of target tree fragments. For all three translation tasks, the majority of discontinuous rules used to decode the test sets have two target tree fragments. On the English-to-German translation task, those rules account to 67.6% for the minimal setting and increase to 86.2% for the non-minimal setting. On the other hand, in the minimal setting for the English-to-Arabic translation task, those rules are accounted for by 81.8% but drop to 68.1% in the non-minimal setting. For the translation from English-to-Chinese, the percentage is almost equal: 79.4% vs 80.6% (minimal vs non-minimal setting).

4.5 Summary

We presented the first parameterized algorithm for the extraction of non-minimal ℓMBOT rules. The original algorithm extracts minimal rules only which are particularly inappropriate for translating fixed phrases. Hence, we hoped to overcome the limitation of the minimal rules. We evaluated the new algorithm together with several other systems in three large scale translation tasks (English-to-German, English-to-Arabic, and English-to-Chinese). As expected, the non-minimal tree-to-tree system performs much better than the corresponding system using only minimal rules, but even the system with non-minimal rules does not beat the SCFG baseline (using non-minimal rules). It seems that discontinuity remains a challenge

for tree-to-tree rules. Furthermore, the analysis of the (possible) discontinuous rules used to decode the test sets shows a strong shift from structural discontinuous rules (used in the minimal system) to lexical discontinuous rules (used more in the non-minimal system).

Chapter 5

Non-Minimal String-to-Tree ℓMBOT

We start this chapter with the motivation of string-to-tree ℓMBOTs. Moving from trees to strings on the source language side makes it necessary to adapt our theoretical model. Furthermore, we will readjust the algorithm from Section 4.2 to extract a set of string-to-tree ℓMBOT rules. Finally, we evaluate our system on three large scale experiments. To put our results in perspective, we compare against a string-to-tree SCFG baseline as well as against systems that are not syntax-based.

5.1 Motivation

The minimal ℓMBOT rule extraction presented in Chapter 3 as well as its non-minimal counterpart presented in the previous chapter extract rules from the sentence pairs of a word-aligned and bi-parsed parallel corpus. In such a corpus, parses are available for the source language sentences and the target language sentences, which might be difficult to obtain. Even if constituent parsers are available, we need to parse the large training corpus, which usually takes significant time and adds a source for errors and specificity, which can lead to lower translation performance and to lower coverage (Wellington et al., 2006). We confirm this in our experiments of Section 4.4.2, where we observe a huge gap between the BLEU scores

of tree-to-tree based systems (SCFG and ℓMBOT) and (hierarchical) phrase-based systems. Furthermore, it is well known (Chiang, 2010) that string-to-tree systems — i.e., parses only on the target-side — generally yield higher translation quality compared to tree-to-tree systems. Hence, we decided to adapt the algorithm presented in Section 4.2 to extract *non-minimal string-to-tree rules* that still offer discontinuities on their right-hand sides. Simply removing the tree structure from the left-hand side of the tree-to-tree ℓMBOT rules is not an option. First, this would preserve parse errors and second, tree-to-tree rules are linguistically motivated as only rules are extracted that match a constituent in the parse tree. But the left-hand sides of string-to-tree rules have no such motivation and thus encourage a consistent translation of larger parts of the source sentence which do not necessarily match syntactic constituents.

5.2 Theoretical Model

As our translation model, we use a string-to-tree variant of the shallow ℓMBOT as introduced in Section 3.1. We keep on calling our variant ℓMBOT for simplicity. The model still works on pairs $\langle s, (t_1, \ldots, t_n) \rangle$ but with a single *source string* s and potentially several *target tree fragments* $t_1, \ldots, t_n$. The source string s is built from the lexical items and the special placeholder X which can occur several times. This pair is a sentential form of rank n and it is shallow if all target tree fragments $t_1, \ldots, t_n$ in it are shallow. We still have to store an alignment between the placeholders X and the nonterminal leaves in the target tree fragments. Just like in the tree-to-tree setting, each placeholder occurrence can link to several leaves in the target tree fragments indicating that these parts are supposed to develop synchronously. However, each non-lexical leaf in the target tree fragments links to exactly one placeholder occurrence. As the left-hand sides of the rules are now based on strings, let us recap strings. A string is of the form $s = w_1 \ldots w_k$ and its length is denoted by $|s| = k$. We can access the x-th letter in w by $w_x = s[x]$ for all $1 \leq x \leq k$. We obtain

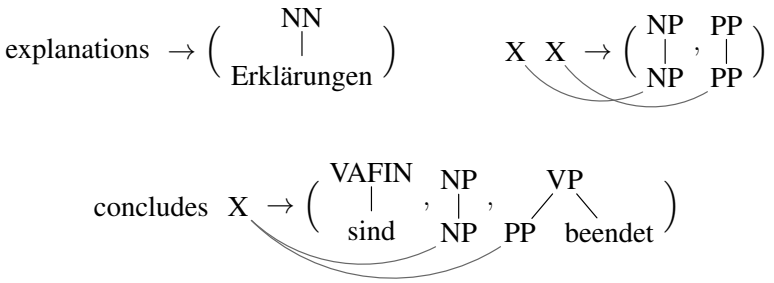

Figure 5.1: Several valid rules for our string-to-tree ℓMBOT.

the set of placeholders in s by $ph(s) = \{x \mid w_x = X\}$. Now we can formally define the alignment as an injective mapping $\psi\colon \mathrm{leaf}_N^{(n)}(t_1, \ldots, t_n) \to ph(s) \times \mathbb{N}$ such that if $(x, i) \in ran(\psi)$, then also $(x, j) \in ran(\psi)$ for all $1 \leq j \leq i$. Furthermore, the x-rank $rk(\rho, x)$ of a rule ρ for each $x \in ph(s)$ is

$$rk(\rho, x) = \max\{i \in \mathbb{N} \mid (x, i) \in ran(\psi)\} \ .$$

Hence, a string-to-tree ℓMBOT is a finite set of rules which have the form $s \to_\psi (t_1, \ldots, t_n)$, containing a shallow sentential form, written $\langle s, (t_1, \ldots, t_n)\rangle$, and an alignment ψ for it (see Definition 5). Figure 5.1 depicts some valid string-to-tree rules.

Before we can formulate how to obtain weighted sentential forms, we need one additional notation. Let $x \in ph(s)$ be the x-th letter and v a string. Then $s[x \leftarrow v]$ denotes the string that is obtained from s by replacing the x-th letter by v. Now we can define how to obtain the set $\tau(R, c)$.

Definition 7 *The set $\tau(R, c)$ of* weighted sentential forms *of the string-to-tree ℓMBOT (R, c) is the smallest set T subject to the following restriction: If there exist*

- *a rule $\rho = s \to_\psi (t_1, \ldots, t_n) \in R$,*
- *a weighted sentential form $\langle s_x, c_x, (t_1^x, \ldots, t_{n_x}^x)\rangle \in T$ for every $x \in ph(s)$ with*
  - $\mathrm{rk}(\rho, x) = n_x$,[1]

[1] If x has n alignments, then the sentential form selected for it has to have suitably many target tree fragments.

– $t_i(x') = t_j^v(\varepsilon)$ *with* $\psi(ix') = (v, j)$ *for every* $ix' \in \mathrm{leaf}_N^{(n)}(t_1, \ldots, t_n)$,[2]

then $\langle s', c', (t'_1, \ldots, t'_n)\rangle$ *is a weighted sentential form, where*

- $s' = s[x \leftarrow s_x \mid x \in ph(s)]$,
- $c' = c(\rho) \cdot \prod_{x \in ph(s)} c_x$, *and*
- $t'_k = t_k[x' \leftarrow t_j^v \mid \psi(ix') = (v, j)]$ *for every* $1 \leq k \leq n$.

Given a rule ρ containing a placeholder X that is linked to nonterminal leaves $N_1, \ldots, N_l$ in the target tree fragments, we select a sentential form whose roots of its target tree fragments read $N_1, \ldots, N_l$. Again, the rank of the placeholder X in ρ should match the number l of target tree fragments in the selected weighted sentential form. When those requirements are fulfilled, the placeholder X is replaced by the source string of the sentential form and the linked leaves of ρ are replaced by the target tree fragments of the selected sentential form. Ideally, the process is repeated until the complete source sentence is derived. As in the tree-to-tree setting, completely lexical rules are automatically weighted sentential forms. Furthermore, the last rule applied must dissolve any earlier introduced discontinuities and the final weighted sentential form is of the form $\langle s, c, (t)\rangle$. We illustrate the process in Figure 5.2, where we replace the placeholder X in the source string, which is linked to the leaves NP and PP in the target tree fragments. The sentential form below matches since its root labels of the target tree fragments read 'NP PP'. Thus, we can substitute the sentential form into ρ and obtain the sentential form shown at the bottom.

5.3 Rule Extraction

In the string-to-tree setting, our training corpus still consists of word-aligned sentence pairs $\langle e, A, f\rangle$, but only for the target side a parse tree is available. We keep on using the notion of a consistently aligned rule span $\langle p, p_1 \cdots p_n\rangle$. We yet have

[2]The labels for the target tree fragments have to coincide.

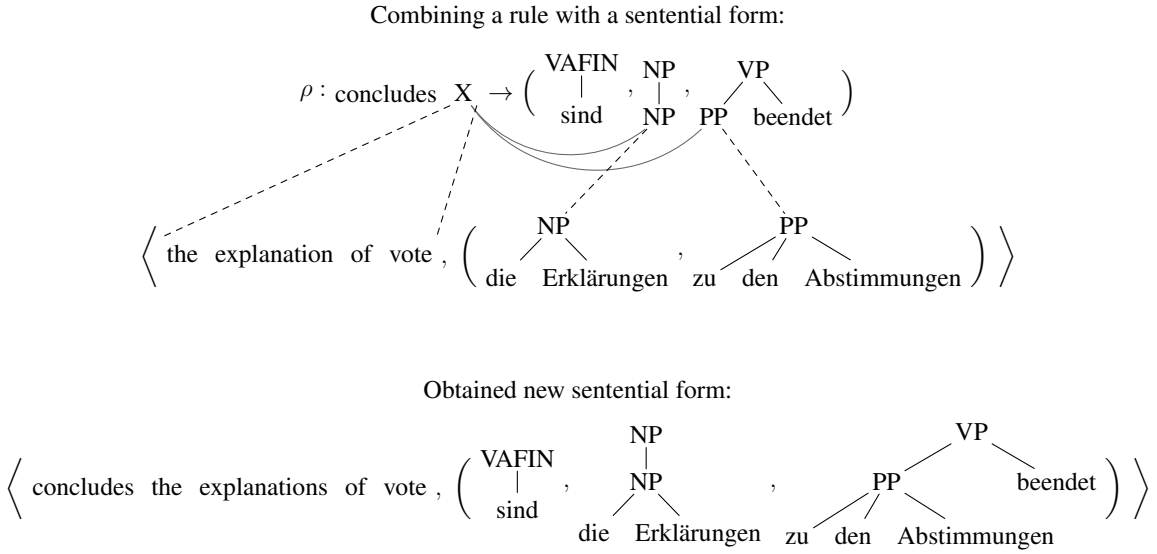

Figure 5.2: Substitution of sentential forms.

available parses for the target language sentences, thus we have a parse tree t for f for each word-aligned sentence pair $\langle e, A, f\rangle$. As before, a consistently aligned rule span $\langle p, p_1 \cdots p_n\rangle$ of $\langle e, A, f\rangle$ is *compatible with* t if there exist nodes $\eta_1, \ldots, \eta_n$ of t such that η_k governs p_k for all $1 \leq k \leq n$.

We depict our running example of a word-aligned sentence pair for the string-to-tree setting in Figure 5.3. If we reconsider the example rule span $\langle [2,4],\ [2,4]\,[8,8]\rangle$ from Section 4.2, then this rule span is still consistently aligned, whereas $\langle [2,4],\ [2,8]\rangle$ is still not. But given the parse tree t in Figure 5.3, the consistently aligned rule span $\langle [2,4],\ [2,4]\,[8,8]\rangle$ is not compatible with t because there is no node in t that governs $[2,4]$. However, for the same data, the rule span $\langle [2,4],\ [2,2]\,[3,4]\,[8,8]\rangle$ is consistently aligned and compatible with t.[3]

As before, we start with the extraction of *initial rules for* $\langle e, A, f\rangle$ *and* t. For each consistently aligned rule span $\langle p, p_1 \cdots p_n\rangle$ that is compatible with t and each selection of nodes $\eta_1, \ldots, \eta_n$ of t such that n_k governs p_k for each $1 \leq k \leq n$, we can extract the rule $e(p) \rightarrow \big(\mathrm{flat}(t_{\eta_1}), \ldots, \mathrm{flat}(t_{\eta_n})\big)$, where

[3]Note that in the non-minimal tree-to-tree setting this rule span was not compatible with the parse tree s for the source sentence because there was no syntactic constituent at [2,4].

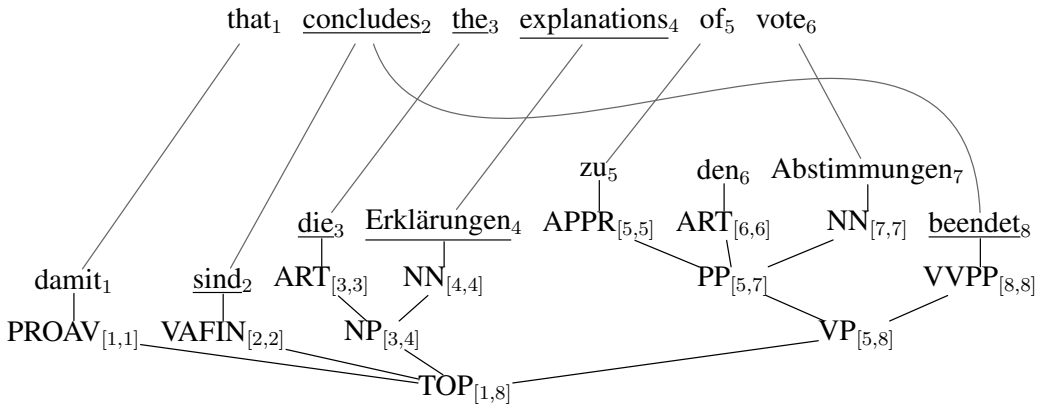

Figure 5.3: Word-aligned sentence pair with target-side parse.

- $e(p)$ is the substring of e at span p,[4]
- $\mathrm{flat}(u)$ removes all internal nodes from u, and
- t_η is the subtree rooted in η for node η of t.

For the rule span $\langle [2,4],\ [2,2]\ [3,4]\ [8,8] \rangle$ we can extract one initial rule with

- $e([2,4]) =$ concludes the explanations,
- $t_{\eta_1} =$ (VAFIN sind),
- $t_{\eta_2} =$ (NP (ART die) (NN Erklärungen)),
- and $t_{\eta_3} =$ (VVPP beendet).

The function flat leaves t_{η_1} and t_{η_3} unchanged, but $\mathrm{flat}(t_{\eta_2}) =$ (NP die Erklärungen). Thus, we obtain the boxed rule of Figure 5.4 where we additionally display some other initial rules. As before, the initial rules are completely lexical and we obtain a new rule r'' by "excising" an initial rule r from another rule r' and replacing the removed part by

- the placeholder X in the source string of r',
- the root label(s) of the right-hand side of r in the target tree fragment(s) of r', and
- linking the removed parts appropriately.

We illustrate this process in Figure 5.5. Formally, the set of *extractable rules* R for a

[4] If $p = [i, i']$, then $e(p) = e[i, i']$ is the substring of e ranging from the i-th token to the i'-th token.

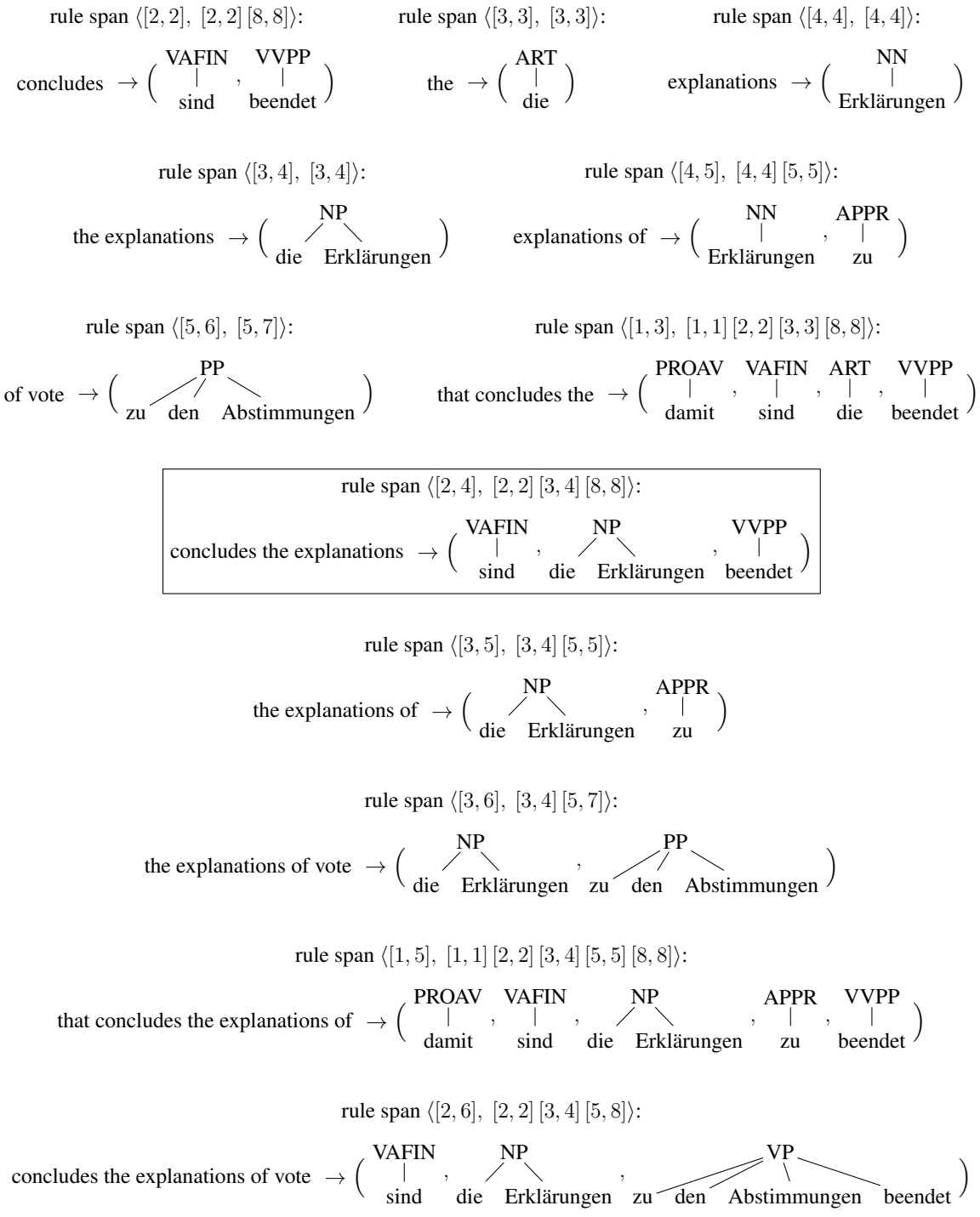

Figure 5.4: Some initial rules extracted from the word-aligned sentence pair and parse of Figure 5.3.

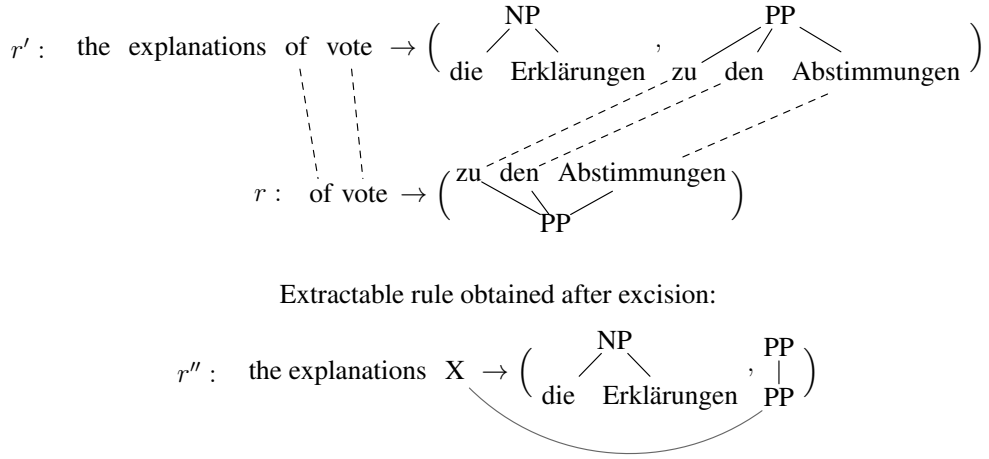

Figure 5.5: Excision of the middle initial rule from the topmost initial rule. Substituting the middle rule into the result yields the topmost rule.

given word-aligned sentence pair $\langle e, A, f \rangle$ with parse tree t for f is the smallest set subject to the following two conditions:

- Each initial rule is in R and thus extractable.
- For every initial rule r and extractable rule $r' \in R$, any flat rule r'', into which we can substitute r to obtain ρ with $\mathrm{flat}(\rho) = r'$, is in R and thus extractable.[5]

As already explained in Section 4.2, it is sensible to restrict the number of extracted rules. While in the tree-to-tree setting the parse trees for both the source and target language side constrained the size, this is not given in the string-to-tree setting. Although our ℓMBOT rules are still restricted by the parses for the target sentences, this is not a real limitation for a model that permits the unlimited presence of multiple target tree fragments. Consequently, we have to impose the existing constraints stated in Section 4.2 and even define additional ones. The following constraints on rules $s \to (t_1, \ldots, t_n)$ are enforced.[6]

(a) We only consider source phrases p of length at most 10 (i.e., $i' - i < 10$ for $p = [i, i']$).

(b) We only excise initial rules with source phrase p of length at least 2 (i.e. $i' - i \geq 1$

[5] A rule $\rho = s \to (t_1, \ldots, t_n)$ is *flat* if $\mathrm{flat}(\rho) = \rho$, where $\mathrm{flat}(\rho) = s \to (\mathrm{flat}(t_1), \ldots, \mathrm{flat}(t_n))$.

[6] Again, the default values can be easily modified during training.

for $p = [i, i']$).

(c) The source string s contains at most 5 occurrences of lexical items or X (i.e. $\ell_s \leq 5$).

(d) We only excise initial rules with target phrases $p_1 \cdots p_n$ of length at least 2 (i.e. $j' - j \geq 1$ for $p_k = [j, j']$ for $1 \leq k \leq n$).

(e) The left-most token of the source string s cannot be X (i.e., $s[1, 1] \neq$ X).

(f) The source string contains at least one lexical item that was aligned in $\langle e, A, f \rangle$.

(g) The source string s cannot have consecutive Xs (i.e., XX is not a substring of s).

For the effects of and the reasoning behind constraints (a)–(c) see Section 4.2. Constraint (d) is an extension of constraint (b). In the string-to-tree setting, also each target tree fragment has to span at least two tokens. Otherwise, training is very slow. In Figure 5.6 we show the steep incline for different corpus sizes when this constraint is not enforced. Constraints (e)–(g) have an effect on extractable rules. Adjacent Xs on the left-hand side are the major cause for spurious ambiguity and are thus prohibited. By requiring that the left-hand side starts with a terminal (constraint (e)), the rules remain lexicalized which allows effective rule filtering before decoding. In addition, without lexical material the chart-based decoder has too many choices for the span of a placeholder X[7] (see Section 2.1.3) which results in very slow decoding. Constraint (f) forces the decoder to take lexical evidence into account. This avoids rules that have lexical material only on one side. Such rules typically either lose lexical information[8] or add (probably) wrong lexical information[9].

In Table 5.1 we report the number of extracted rules for all translation tasks. Relaxing the conditions during rule extraction from non-minimal tree-to-tree to string-to-tree greatly increases the number of extracted rules for each translation task. The string-to-tree rule extraction for ℓMBOTs yields 2–3 times more rules than the non-minimal tree-to-tree rule extraction. In addition, the availability of several

[7] The standard setting in the MBOTMOSES decoder is 20.
[8] If there is a word on the left-hand side but not on the right-hand side.
[9] If there is a word on the right-hand side but not on the left-hand side.

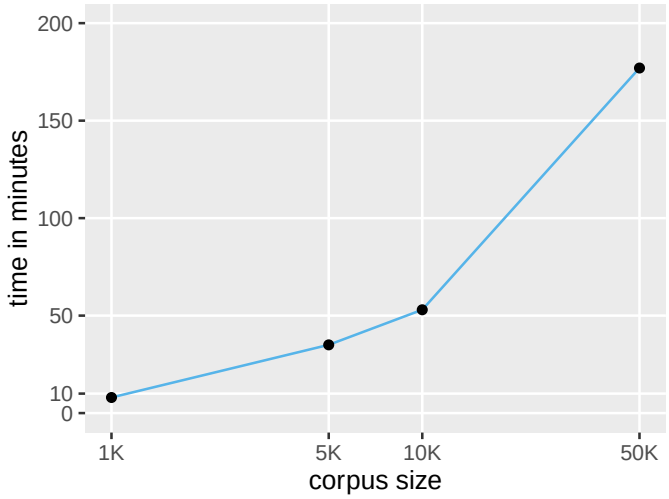

Figure 5.6: Training times for different corpus sizes when not enforcing constraint (d).

System	**Number of extracted rules**		
	English-To-German	English-To-Arabic	English-To-Chinese
tree-to-tree ℓMBOT	40,736,687	151,322,970	84,220,528
string-to-tree SCFG	14,092,729	55,169,043	17,047,570
string-to-tree ℓMBOT	143,661,376	491,307,787	162,240,663

Table 5.1: The number of extracted rules for different extraction algorithms.

target tree fragments (and thus discontinuities) also leads to additional freedom during rule extraction, which is evidenced by the larger numbers of rules extracted for ℓMBOTs compared to those extracted for the SCFG baselines.

As in the tree-to-tree setting, we need a special ℓMBOT glue grammar. We collect glue rules again from completely lexical rules. The grammar allows the decoder to use discontinuous rules in a continuous way by concatenating the root nodes of the target tree fragments as siblings under a special nonterminal.

We present the pseudo-code for extraction of string-to-tree ℓMBOT rules in Algorithm 3. It is an extension of the non-minimal tree-to-tree version of the algorithm (see Section 4.2). In the following, we will point out the important differences. Lines 2 to 4 are exactly as in the tree-to-tree variant as we still iterate by increasing span length over the source sentence. Since the current source span does not

need to be compatible with a parse tree, we simply obtain the source string that is covered by it (line 5). Line 6 is identical to the non-minimal tree-to-tree algorithm and provides us with the consistently aligned target tree fragments for the current source span. In line 7 we add the initial rule to R. We again obtain additional rules with unaligned words on the target language side by using the same approach as in the non-minimal tree-to-tree algorithm (lines 9–13). As before, we obtain extractable rules for each initial rule and possible rules containing unaligned material (line 8 and line 12). To ensure that an extractable rule starts with a terminal (see constraint (e)), we call EXTRACTABLERULES with the index of the next token, i.e. $i + 1$ for source span $[i, i']$. In general, the function EXTRACTABLERULES does the same as in the tree-to-tree version (lines 19 to 36). One difference is that we have to ensure for each rule r in our set that each target tree fragment has the required minimum size (constraint (d); line 24). If that is true, then we excise r from r' as shown in Figure 5.5. The other difference is to make sure that there is (at least) one aligned word pair left in our new rule r'' (constraint (f); line 26). If this is also given, we add r'' to R_E and call EXTRACTABLERULES with r'' and define the index of the next token as $i' + 2$, thus assuring that there will be at least one token between the placeholders in the left-hand side (constraint (g); line 28). The last step is once again to output all rules from R and R_E that conform with constraint (c) (line 16).

As for the non-minimal tree-to-tree algorithm, we display some possible extractable rules for the initial rule $\langle [2, 6],\ [2, 2]\ [3, 4]\ [5, 8] \rangle$ in Figure 5.7.

5.4 Translation Model

Again, the task of the decoder is to find the best corresponding target language translation $\hat{f}$ of the source language sentence e licensed by the translation model and the language model. We use the same log-linear model as introduced in Section 4.3 to tune the features $h_m(\cdot)$ with weights λ_m scored on sentential forms $\langle s, (t) \rangle$ of our extracted ℓMBOT M such that s reads e and the yield of t reads f.

Algorithm 3 Rule extraction for non-minimal string-to-tree ℓMBOT rules

Require: word-aligned sentence pair $\langle e, A, f \rangle$ with parse tree t
Ensure: ℓMBOT rules R

function INITIALRULES(e, A, f, t)
  for $length \leftarrow 1,\ length \leq maxSpan,\ length++$ **do** // ensures constraint (a)
    for $i \leftarrow 0,\ i < |e| - (length - 1),\ i++$ **do**
      $i' \leftarrow i + length - 1$
      $s \leftarrow e([i, i'])$
      $(t_1, \ldots, t_n) \leftarrow$ FINDCONSISTENTALIGNED(A, i, i', t)
      $R[i, i'] \leftarrow R[i, i'] \cup \{r' : \langle s \rightarrow (t_1, \ldots, t_n) \rangle\}$
      EXTRACTABLERULES($i + 1, i', r'$) // ensures constraint (e)
      for all $1 \leq k \leq n$ **do**
        $\rho \leftarrow$ FINDUNALIGNEDWORDS(r', t_k, A)
        $R[i, i'] \leftarrow R[i, i'] \cup \rho$
        EXTRACTABLERULES($i + 1, i', \rho$) // ensures constraint (e)
      end for
    end for
  end for
  OUTPUTRULES(R, R_E) // ensures constraint (c)
end function

function EXTRACTABLERULES($next\_i, end, r'$)
  for $i \leftarrow next\_i,\ i \leq end,\ i++$ **do**
    for $i' \leftarrow i + (minSpanSource - 1),\ i' \leq end,\ i'++$ **do** // ensures constraint (b)
      $R' \leftarrow R[i, i']$ // retrieve initial rules covering this span
      for all $r \in R'$ **do**
        if $minSpanTarget(r) == true$ **then** // ensures constraint (d)
          $r'' \leftarrow$ EXCISE(r, r')
          if $requireAlignedWord(r'', A) == true$ **then** // ensures constraint (f)
            $R_E[i, i'] \leftarrow R_E[i, i'] \cup \{r''\}$
            EXTRACTABLERULES($i + 2, end, r''$) // ensures constraint (g)
          else
            continue
          end if
        end if
      end for
    end for
  end for
end function

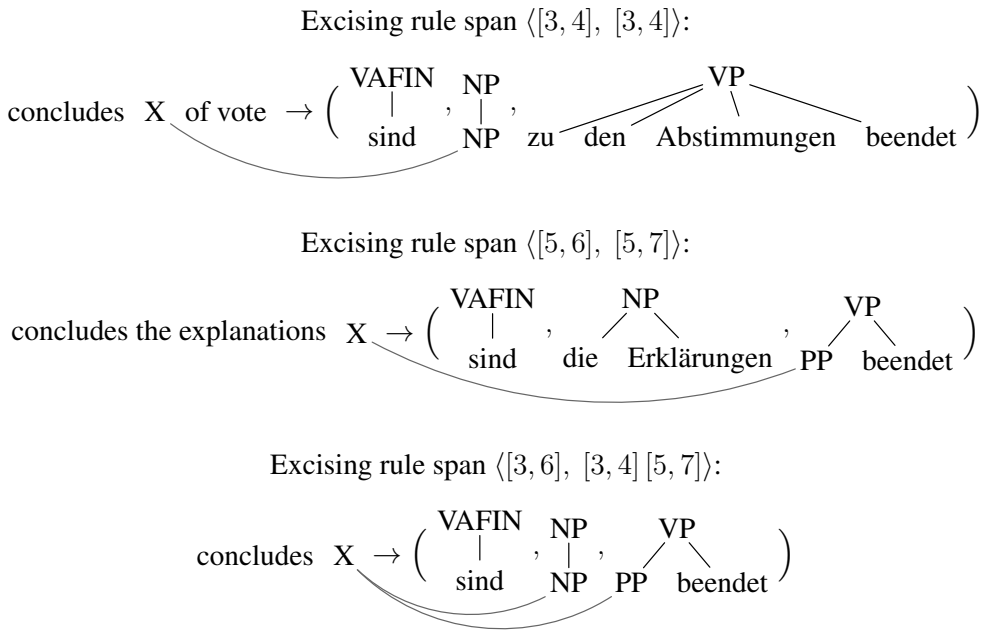

Figure 5.7: Some extractable rules for the rule span $\langle [2, 6], [2, 2]\ [3, 4]\ [5, 8] \rangle$.

We use the MBOTMOSES decoder (Braune et al., 2013) which uses a CYK+ chart parsing algorithm using a standard X-style parse tree which is sped up by cube pruning (Chiang, 2007) with integrated language model scoring. The model uses the same ten features as described in Section 3.4.

5.5 Experimental Results

In this section we report the setup and results of the experiments as presented in Seemann et al. (2015a).

5.5.1 Setup

As a baseline system for our experiments we use the syntax-based component of the MOSES toolkit (Hoang et al., 2009; Koehn et al., 2007). Our system is the translation system based on ℓMBOTs as presented in this chapter. Our and the baseline

	English to German	**English to Arabic**	**English to Chinese**
training data	7th Europarl (Koehn, 2005)	MultiUN (Eisele and Chen, 2010)	
training data size	≈ 1.8M sentence pairs	≈ 5.7M sentence pairs	≈ 1.9M sentence pairs
target-side parser	BitPar (Schmid, 2004)	Berkeley parser (Petrov et al., 2006)	
language model	5-gram SRILM (Stolcke, 2002)		
add. LM data	WMT 2013 (Bojar et al., 2013)	Arabic in MultiUN	Chinese in MultiUN
LM data size	≈ 57M sentences	≈ 9.7M sentences	≈ 9.5M sentences
tuning data	WMT 2013	cut from MultiUN	NIST 2002, 2003, 2005
tuning size	3,000 sentences	2,000 sentences	2,879 sentences
test data	WMT 2013	cut from MultiUN	NIST 2008 (NIST, 2010)
test size	3,000 sentences	1,000 sentences	1,859 sentences

Table 5.2: Summary of the experimental setup.

system use linguistic syntactic annotation (parses) only on the target side (*string-to-tree*). We apply the rule extraction algorithm as proposed in Section 5.3 and impose the restrictions stated there. Additional glue-rules that concatenate partial translations without performing any reordering are used in all systems.

For all experiments (English-to-German, English-to-Arabic, and English-to-Chinese), we use exactly the same data as presented in Section 4.4.1. The only difference is that there are no parses for the source language side anymore, hence the source sentences are simply tokenized and truecased strings. The preprocessing of the target language is the same as in Section 4.4.1. For convenience reasons, we repeat the resources used in Table 5.2.

Again, in all experiments the feature weights λ_m of the log-linear model were trained using minimum error rate training (Och, 2003).

5.5.2 Evaluation

The overall translation quality was measured with 4-gram BLEU (Papineni et al., 2002) on true-cased data for German, on transliterated data for Arabic, and on word-segmented data for Chinese. Significance was computed with Gimpel's implementation (Gimpel, 2011) of pairwise bootstrap resampling with 1,000 samples. Table 5.3 lists the evaluation results. In all three setups the ℓMBOT system significantly outperforms the baseline. For German we obtain a BLEU score of 15.90 which

	BLEU		
System	**En-De**	**En-Ar**	**En-Zh**
string-to-tree SCFG (baseline)	15.22	48.32	17.69
string-to-tree ℓMBOT	**15.90**	**49.10**	**18.35**
MOSESGHKM	17.10	46.66	18.12
hierarchical phrase-based	16.95	51.71	18.49
phrase-based	16.73	50.27	18.09

Table 5.3: Evaluation results. The bold results are statistically significant improvements over the baseline (at confidence $p < 1\%$).

is a gain of 0.68 points. For Arabic we get an increase of 0.78 points which results in 49.10 BLEU. For Chinese we obtain a score of 18.35 BLEU gaining 0.66 points.[10] We also report the BLEU scores for a phrase-based, a hierarchical phrase-based, and a MOSESGHKM system. All three systems were trained with the same data as described in Table 5.2 with their respective standard settings in MOSES. For English-to-German, all three systems are better with differences from 0.83 to 1.20 points. For English-to-Arabic, the hierarchical and the phrase based systems are better (2.61 and 1.17 points) but MOSESGHKM underperforms the ℓMBOT by 2.44 points. On the English-to-Chinese translation task, MOSESGHKM and the phrase-based system perform worse than the ℓMBOT (0.23 and 0.26 points) but the hierarchical phrase-based system outperforms our system by a small margin (0.14 points).

5.5.3 Analysis of Discontinuity

Again, we try to estimate the impact of multiple target tree fragments. Table 5.4 reports how many rules of each type are used during decoding. Out of the rules used for German, 27% were (potentially) discontinuous and 5% were structural. For Arabic, we observe 67% discontinuous rules and 26% structural rules. For translating into Chinese 30% discontinuous rules were used and the structural rules account to 18%. These numbers show that the usage of discontinuous rules tunes to the spe-

[10] NIST-08 also shows BLEU for word-segmented output (`http://www.itl.nist.gov/iad/mig/tests/mt/2008/doc/mt08_official_results_v0.html`). Best constrained system: 17.69 BLEU; best unconstrained system: 19.63 BLEU.

Language pair	Type	Lex	Struct	Total	Target tree fragments 2	3	4	5	≥ 6
English-to-German	cont.	27,351	635	27,986					
	discont.	9,336	1,110	10,446	5,565	3,441	1,076	312	52
English-to-Arabic	cont.	1,839	651	2,490					
	discont.	3,670	1,324	4,994	3,008	1,269	528	153	36
English-to-Chinese	cont.	17,135	1,585	18,720					
	discont.	4,822	3,341	8,163	6,411	1,448	247	55	2

Table 5.4: Number of rules per type used when decoding test (Lex = lexical rules; Struct = structural rules; [dis]cont. = [dis]continuous).

cific language pair. For instance, Arabic utilizes them more compared to German and Chinese. Furthermore, German uses a lot of lexical rules which is probably due to the fact that it is a morphologically rich language. On the other hand, Arabic and Chinese make good use of structural rules.

In addition, Table 5.4 presents a finer-grained analysis based on the number of target tree fragments. Only rules with at most 8 target tree fragments were used. Again, mostly discontinuous rules with two target tree fragments were used. Taking the numbers from the non-minimal tree-to-tree setting into account, we notice for the English-to-German translation task that the rate of 53,2% is much lower (86,2%). For the translation direction English-to-Arabic, there is also a decrease but much smaller: 60.2% vs 68.1%. Interestingly, the percentage for the English-to-Chinese translation task is almost equal with the non-minimal tree-to-tree system: 78.5% and 80.6%. In general, it seems that German and Arabic require some rules with 6 target tree fragments, while Chinese probably does not. We conclude that the number of target tree fragments can be restricted to a language-pair specific number during rule extraction.

5.6 Incorporating Dependency Parses

We obtained significant improvements for our string-to-tree ℓMBOT in terms of translation quality on three different translation tasks. Hence, we want to apply

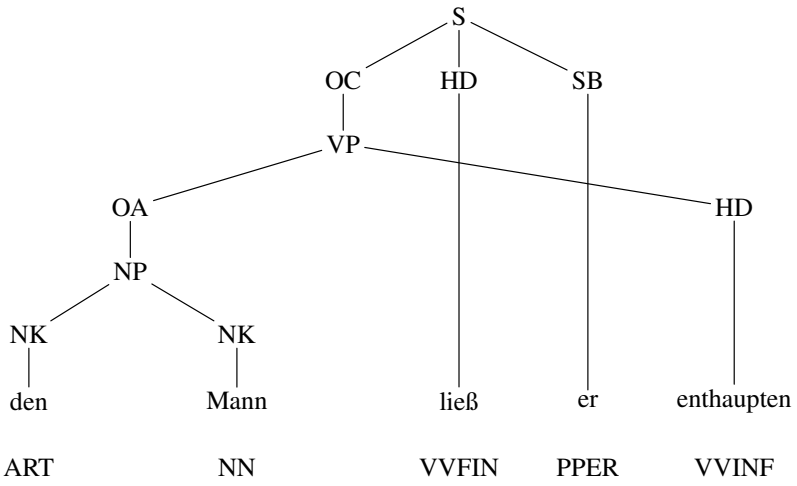

Figure 5.8: German example of a crossing edge taken from TIGER. Gloss: 'the man he let be beheaded'.

our approach on other languages. In fact, we want to adjust our translation system to Eastern European languages. Those languages admit a relatively free word order and we expect that discontinuities occur frequently, making our ℓMBOT an ideally suited syntax-based translation system. But due to the free word order, it is often difficult to express the syntax of such languages in terms of constituency structure (Kallestinova, 2007). Since the parts that need to (grammatically) agree can occur spread out over the whole sentence, constituents cannot be hierarchically organized as in a classical constituency parse tree. In Figure 5.8 we illustrate a German constituency parse tree for the sentence 'den Mann ließ er enthaupten' ('the man he let be beheaded') taken from the TIGER treebank (Brants et al., 2004). The sentence displays two peculiarities. First, there are two main verbs, 'ließ' ('let') and 'enthaupten' ('behead') where 'er' is the subject of 'ließ' and 'der Mann' is the accusative object of 'enthaupten.[11] Second, the sentence exhibits a marked word order as the accusative object 'den Mann' is in subject position. The unmarked word order would be 'er ließ den Mann enthaupten' ('he let the man be beheaded'). The graphical representation shows how the 'VP' governs the 'OA' and the verb 'ent-

[11]The accusative object 'OA' of the verb 'ließ' is not realized. A sentence with an OA could have the structure '[er]$_{SB}$ ließ [den Henker]$_{OA/SB}$ [den Mann]$_{OA}$ enthaupten' in which the OA 'den Henker' ('the hangman') is the logical subject of the embedded verb 'enthaupten'.

haupten'. Due to the marked word order, the OA and the verb are separated by 'ließ er'. The edge from 'VP' to 'HD' is called a *crossing edge* and does not allow for an hierarchical tree. Dependency parses do not pre-suppose such a hierarchical structure and are thus often more suitable for languages with free word order. For the target languages discussed here (we translate into Polish and Russian), only dependency parses are readily available. Those parses relate the lexical items of the sentence via edges that are labeled with the syntactic function between the head and its dependent. Overall, these structures also form trees but they can be non-projective (see Section 5.6.2) for our target languages. Such non-projective dependency trees do not admit a constituent-like tree representation, so the existing string-to-tree ℓMBOT system cannot readily be applied. We first need to convert the dependency trees into projective dependency trees, which can then be converted easily into a constituent-like tree representation. The conversion into projective dependency trees is known to preserve discontinuities, so we expect that our model is an ideally suited syntax-based translation model for those target languages.

In the remainder of this chapter, we first give an overview of dependency parsing and show the structure of the resulting parses. Furthermore, we explain how to convert non-projective parse trees into projective ones and how to transform those into constituent trees. Finally, we evaluate our system on two translation tasks.

5.6.1 Related Work

The idea of utilizing dependency trees in machine translation is not novel. Bojar and Hajič (2008) built a experimental system for English-to-Czech that models tree-based transfer at a deep syntactic layer. The target sentences are projective dependency trees while the source sentences are constituency parse trees. From those, they extract treelets based based on synchronous tree substitution grammars. Xie et al. (2011) present a dependency-to-string model that extracts head-dependent rules with reordering information. Their model requires a custom decoder to deal with the dependency information in the input. Li et al. (2014) follow up on this

work by transforming these dependency trees into (a kind of) constituency trees. In this approach, they are able to use the conventional syntax-based models of MOSES. In contrast to our work, these two models do not use the syntactic functions provided by the parser but rather extract head-dependent rules based on the lexical items. Sennrich et al. (2015) transformed (non-projective) dependency trees into constituency trees using the syntactic functions provided by the parser. They used the string-to-tree GHKM model (Williams and Koehn, 2012) of MOSES and evaluated their approach on an English-to-German translation task. It shows that the system utilizing the (transformed) dependency parses outperforms competing systems utilizing various variants of constituent parses for the German side. We follow up on their work for translation tasks, where constituent parses are not readily available.

5.6.2 From Dependency Trees to Constituency Trees

We start with a short overview of dependency parsing (Kübler et al., 2009) and introduce the non-projective tree structures that occur as parses. Next, we explain how to convert these structures into projective trees. In the last step, we transform the projective dependency trees into the shape of classical constituency trees.

Dependency Parsing

In a dependency parse, each occurrence of a lexical item (i.e., token) in the input sentence forms a node. The dependency parser constructs a tree structure over those nodes by relating them via edges pointing from a *head* node h to its *dependent* node d. Such an edge is denoted by $h \rightarrow d$. In addition, each edge is assigned a label indicating the type of the syntactic dependence. Often an artificial root node is added for convenience. An example parse for a Polish sentence is depicted in Figure 5.9. The edges can be projective or non-projective. The edge $h \rightarrow d$ is *pro-*

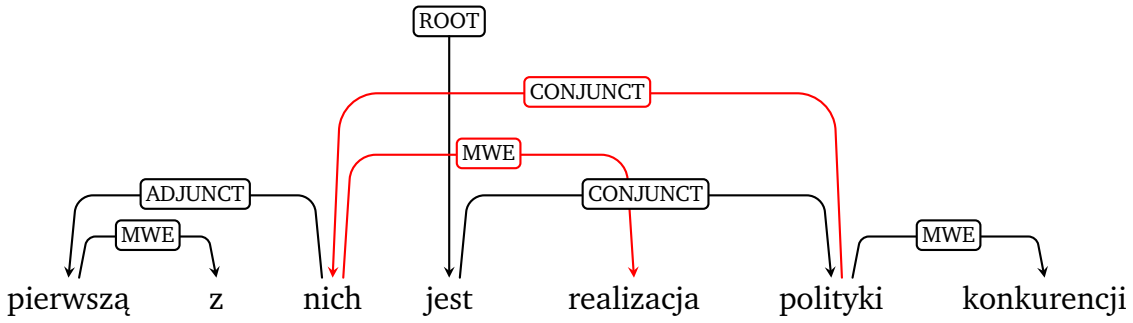

Figure 5.9: Non-projective Polish dependency tree (gloss: *The first is the operation of competition policy*)

jective if and only if its head node h dominates[12] all nodes representing the tokens in the linear span between h and d. A dependency parse is projective if and only if all its edges are projective. A non-projective dependency parse is easily recognized in graphical representations because it has a crossing edge provided that all the edges are drawn on one side of the sentence as in Figure 5.9 (where we marked the non-projective edges in red).

Non-projective dependency structures cannot be directly used in our translation framework, so we first have to turn them into projective trees. Kahane et al. (1998) were the first to introduce *lifting*. Given a non-projective edge $h \rightarrow d$ there exists at least one node n that occurs in the linear span between h and d such that n is not dominated by h. In the lifting process, the edge $h \rightarrow d$ is replaced by an edge $g \rightarrow d$, where g is the lowest node that dominates both h and n (i.e., the least common ancestor of h and n). Repeating this process for all non-projective edges eventually yields a projective tree. Nivre and Nilsson (2005) refined this approach by performing the same replacement but introducing three ways to document the lifting operation in the labels: 'head', 'head+path', and 'path'.[13] The annotation schemes 'head' and 'head+path' might increase the number of labels quadratically, whereas 'path' only introduces a linear number of new labels. Since we deal with millions

[12] A node n *dominates* a node d iff n is an ancestor of d; i.e., there is a path from n to d.

[13] Kahane et al. (1998) use lifting to realize a polynomial-time rule-based parsing algorithm while Nivre and Nilsson (2005) use it to train a parser on lifted edges and recover the former non-projective edges by inverting the annotated lifts.

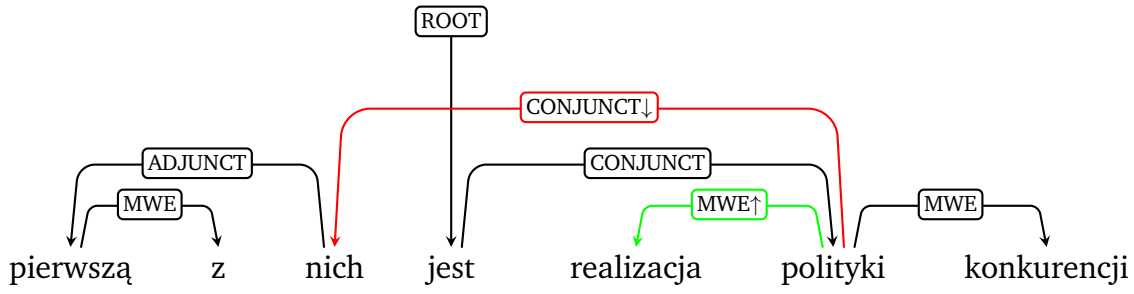

Figure 5.10: Non-projective Polish dependency tree after one 'path'-lifting operation.

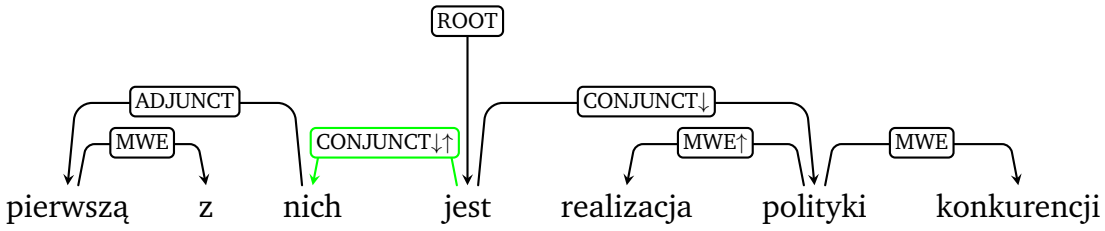

Figure 5.11: Projective dependency parse obtained after second 'path'-lifting operation.

of trees in our syntax-based machine translation experiments, we need to select a compromise between (i) inflating the number of labels and (ii) documenting the lifts. Therefore, we decided to use the 'path' scheme to obtain projective parse trees for our experiments (see Section 5.6.3). Let us explain the 'path' scheme. In the situation described earlier, in which the edge $h \rightarrow d$ was replaced by the edge $g \rightarrow d$, we set the label of $g \rightarrow d$ to the label of the original edge $h \rightarrow d$ annotated by $\uparrow$ to indicate that this edge was lifted. Additionally, all edges connecting the new head g and the syntactical head h are annotated with $\downarrow$ indicating where the syntactic head is found.

We will now illustrate the lifting operation. In Figure 5.9 we have two non-projective edges which are marked in red. The edge 'nich $\rightarrow$ realizacja' is non-projective because 'nich' does not dominate 'jest', which occurs in the relevant linear span. Likewise, the edge 'polityki $\rightarrow$ nich' is non-projective because 'polityki' does not dominate 'jest', which occurs in the relevant linear span. We first apply the lift-

ing operation to the most nested[14] non-projective edge which is 'nich → realizacja' and obtain the (still) non-projective tree in Figure 5.10. In it we have the new edge 'polityki → realizacja' with the label 'MWE↑', indicating that the edge got lifted. Moreover, the edge 'polityki → nich' now has the label CONJUNCT↓ because it is the edge that connects the new head with the syntactical head of 'realizacja'. The edge 'polityki → nich' is still non-projective. Applying lifting once more results in the tree shown in Figure 5.11. Now we have a new edge 'jest → nich' with the label CONJUNCT↓↑ and the edge 'jest → polityki' has the label CONJUNCT↓ because it is the edge that connects the new head with the syntactical head of 'nich'.

In principle, one can imagine other ways to projectivize a tree; e.g., we can just replace the head of a non-projective edge by the root. From a linguistic point of view, it makes more sense to attach it (as described) to the least common ancestor, which in a sense is the minimal required change that leaves the remaining edges in place. Furthermore, the used implementation always lifts the most nested non-projective edge until the tree is projective. In this way, the minimal number of lifts required to projectivize the tree is achieved as demonstrated by Buch-Kromann (2005). We used the lifting implementation by ANDERS BJÖRKELUND in our experiments.

Conversion

Our goal is the investigation of the performance of our string-to-tree ℓMBOT system, so we need syntactic annotations on the target side. First, the target-side sentences (in Polish and Russian) are annotated with part-of-speech tags with the help of TREETAGGER (Schmid, 1994). The TREETAGGER output is then converted into the (comma-separated) CONLL-X format[15], which lists each token of the sentence in one line with 10 attributes like word position, word form, lemma, and part-of-speech tag. A new sentence is started by an empty line. This representation is passed

[14] deepest or most distant from the root
[15] documented on `http://ilk.uvt.nl/conll/`

Corpus	Lang.	Number of labels original	new
EUROPARL	PL	25	67
YANDEX	RU	75	118
Commoncrawl	RU	71	84
News commentary	RU	71	84
Patronymic names	RU	13	0
Names	RU	31	0
WIKI headlines	RU	54	19

Table 5.5: Number of parse labels before and after the 'path'-lifting.

to the MALT parser (Nivre et al., 2006; Sharoff and Nivre, 2011), which fills the remaining attribute fields like position of the head and the label of the dependency edge. The resulting output represents the (potentially) non-projective dependency parses of the target-side sentences.

In the next step, we apply the 'path'-lifting as described in Section 5.6.2. The Polish parses have 2.2% non-projective edges and we performed 500,507 lifts in total to obtain the corresponding projective parses (corpus size: $\approx$ 600K sentences with 14,147,378 tokens). The Russian parses have only 0.4% non-projective edges and we performed 137,893 lifts in total to make the parses projective (corpus size: $\approx$ 1.7M sentences with 30,808,946 tokens). As described in Section 5.6.2 we introduce at most 3 additional labels for each existing label. In Table 5.5 we report for each corpus the exact number of original parse labels and the number of labels newly introduced by the transformation into projective parses.

Finally, we transform the projective dependency parse trees directly into the standard representation of constituent parse trees in MOSES and MBOTMOSES, respectively. We use the part-of-speech tags as pre-terminal nodes. Additionally, we make the labels and part-of-speech tags more uniform as follows:

- All parentheses are labeled 'PAR'.
- All slashes, quotation marks, and dashes are labeled 'PUNCT' and their part-of-speech tag is 'INTJ'.

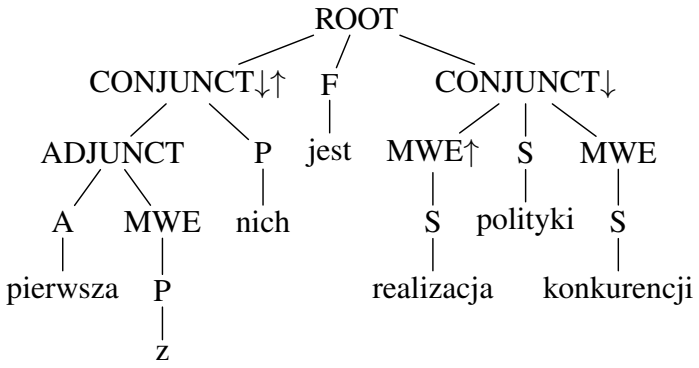

Figure 5.12: Final constituency representation for the parse of Figure 5.11.

- All punctuation marks are labeled 'PUNCT' and their part-of-speech tag is ','.
- If the tagger did not assign a part-of-speech tag, then we label it 'UNK'.

The final constituency tree representation obtained from the projective dependency tree of Figure 5.11 is shown in Figure 5.12.

5.6.3 Experimental Results

In this section we report the setup and results of the experiments as presented in Seemann and Maletti (2015). The ℓMBOT-based system is evaluated on two translation tasks: English-to-Polish and English-to-Russian. For both target languages only (potentially) non-projective dependency parses are easily available. Our goal is to evaluate whether the discontinuity offered by the ℓMBOT model helps in tasks involving such dependency parses. Consequently, the baseline system is again the syntax-based component (Hoang et al., 2009) of the MOSES toolkit (Koehn et al., 2007), which uses a translation model that only permits continuous rules. Both systems are *string-to-tree* in the sense that the projectivized parses are only used on the target side.

	English to Polish	English to Russian
training data	7th Europarl (Koehn, 2005)	WMT 2014 (Bojar et al., 2014)
training data size	≈ 618K sentence pairs	≈ 1.7M sentence pairs
target-side parser	Malt parser (Nivre et al., 2006; Sharoff and Nivre, 2011)	
parser grammar	(Wróblewska and Przepiórkowski, 2012)	(Nivre et al., 2008)
language model (LM)	5-gram SRILM (Stolcke, 2002)	
additional LM data	Polish sentences in Europarl	WMT 2014
LM data size	≈ 626K sentences	≈ 43M sentences
tuning data	cut from Europarl	WMT 2014
tuning size	3,030 sentences	3,000 sentences
test data	cut from Europarl	WMT 2014
test size	3,029 sentences	3,003 sentences

Table 5.6: Summary of the experimental setup.

Setup

We use standard and freely available resources to build our machine translation systems. In summary, for Russian we use the resources provided by the 2014 Workshop on Statistical Machine Translation (Bojar et al., 2014). The Polish data is taken from the EUROPARL corpus (Koehn, 2005).

Next, we describe the preparation and evaluation for both tasks (English-to-Polish and English-to-Russian). First, the training data was tokenized, lowercased and length-ratio filtered up to 80 tokens. We use MGIZA++ (Gao and Vogel, 2008) with the *grow-diag-final-and* heuristic (Koehn et al., 2005) to automatically derive the word alignments. We obtain our ℓMBOT by applying the rule extraction with the restrictions given in Section 5.3. In all our experiments a CYK+ chart parser is used as decoder. The decoder for the baseline is provided by the syntax component (Hoang et al., 2009) of the MOSES framework, and the decoder for the ℓMBOT model is provided by the MBOTMOSES branch (Braune et al., 2013). Glue-rules in both systems ensure that partial translation candidates can always be concatenated without any reordering. The feature weights of the log-linear models were trained with the help of minimum error rate training (Och, 2003) and optimized for 4-gram BLEU (Papineni et al., 2002) on the tuning set (lowercased, tokenized). In the end, the systems were evaluated (also using 4-gram BLEU) on the test set. Significance

judgments of the differences in the reported translation quality (as measured by BLEU) were computed with the pairwise bootstrap resampling technique of Koehn (2004b) on 1,000 samples. Table 5.6 summarizes the setup information.

A particular detail is worth mentioning. We were unable to identify standard tuning and test sets for the English-to-Polish translation task. Consequently, we manually removed one session[16] of the EUROPARL corpus. After removing duplicate sentences, we used the odd numbered sentences as tuning set and the even numbered sentences as test set.

Evaluation

We present the quantitative evaluation for both experiments in Table 5.7. In both cases (English-to-Polish and English-to-Russian) the ℓMBOT system significantly outperforms the baseline, which is the syntax-based component of MOSES. For Polish we obtain a BLEU score of 23.43 resulting in a gain of 2.14 points over the baseline. Similarly, for Russian we achieve a BLEU score of 26.13, which is an increase of 1.47 points over the baseline. To put our results in perspective, we also train a MOSESGHKM system, a phrase-based system, and a hierarchical phrase-based system with standard settings for each translation task on the same resources as described in Table 5.6 and present their evaluation also in Table 5.7. The ℓMBOT system is better than the MOSESGHKM system on both translation tasks (but not statistically significant) which shows that our system performs best among all systems utilizing parses. Overall on the English-to-Polish translation task, the hierarchical system is the best performing system with an increase of 1.13 points over the ℓMBOT system. Similarly, the phrase-based system does outperform all systems on the English-to-Russian translation task with 1.77 points over the ℓMBOT system.

Based on the observed BLEU scores, it seems likely that our ℓMBOT-based approach can almost completely avoid the large quality drop observed between a (hi-

[16]More specifically, we removed sentences 468,743–474,883 from the corpus and only used the remaining sentences for training.

System	BLEU En-Pl	BLEU En-Ru
string-to-tree SCFG (baseline)	21.29	24.66
string-to-tree ℓMBOT	**23.43**	**26.13**
MOSESGHKM	23.31	25.97
hierarchical phrase-based	24.56	27.72
phrase-based	24.35	27.90

Table 5.7: Evaluation results incl. MOSESGHKM-based system, hierarchical system, and phrase-based system for reference. The bold ℓMBOT results are statistically significant improvements over the baseline (at confidence $p < 1\%$).

erarchical) phrase-based system, which does not utilize the syntactic annotation, and a continuous string-to-tree syntax-based model. The availability of discontinuous tree fragments yields significant improvements in translation quality (as measured by BLEU) and an overall performance similar to (hierarchical) phrase-based systems. However, we also observe that outscoring a (hierarchical) phrase-based system remains a challenge, so it remains to be seen whether syntactic information can actually help the translation quality in those translation tasks.

Analysis of Discontinuity

To quantitatively support our claim that the multiple target tree fragments (and the discontinuity) of an ℓMBOT are useful, we once again provide statistics on the ℓMBOT rules that were used to decode the test sets. In Table 5.8 we report how many rules of each type are used during decoding.[17]

For Polish, 41% of all used rules were discontinuous and only 4% were structural. Similarly, 35% of the used Russian rules were discontinuous and again only 4% were structural. The low proportion of structural rules is not very surprising since both target languages are known to be morphologically rich and thus have large lexicons (167,657 lexical items in Polish and 911,397 lexical items in Russian).

[17]The provided analysis tools currently do not support an analysis whether a discontinuous rule was actually used in a discontinuous manner or whether the components were later combined in a continuous manner. The reported numbers thus represent potential discontinuity.

Translation task	Type	Lex	Struct	Total	Target tree fragments 2	3	4	5	≥ 6
English-to-Polish	cont.	25,327	307	25,634					
	discont.	16,312	1,595	17,907	15,805	1,818	254	27	3
English-to-Russian	cont.	24,100	664	24764					
	discont.	12,767	1,108	13,875	11,087	2,308	412	58	10

Table 5.8: Number of rules per type used when decoding test (Lex = lexical rules; Struct = structural rules; [dis]cont. = [dis]continuous).

Another interesting point is the distribution of *discontinuous structural rules*. Polish and Russian use 83% and 62% such rules, respectively, showing that the majority of the used structural rules is discontinuous in both tasks. Additionally using the data of Seemann et al. (2015a) (see Section 5.5.2), we can confirm that morphologically rich languages have a small minority of structural rules (4%, 4%, and 5% for Polish, Russian, and German, respectively), whereas Arabic and Chinese use a much larger proportion of structural rules (26% and 18%, respectively). In addition, we suspect that the additional non-projectivity of Polish makes discontinuous rules more useful (as an indicator for induced discontinuity). Whereas for Russian, German, Arabic, and Chinese approx. 2 out of 3 used structural rules are discontinuous (62%, 64%, 67%, and 68%, respectively), more than 4 out of 5 (83%) used structural rules are discontinuous for Polish.

Finally, we present a fine-grained analysis based on the number of target tree fragments in Table 5.8. Useful Polish rules have at most 6 target tree fragments, whereas Russian rules with up to 9 target tree fragments have been used. Similar numbers have been reported in Seemann et al. (2015a). Using their data, we also note that Polish, Russian, and Chinese seem to use a larger percentage of discontinuous rules with 2 target tree fragments (80%–90%) compared to German and Arabic (50%–60%).

5.7 Summary

We presented an application of a string-to-tree variant of ℓMBOTs to statistical machine translation. Originally, only tree-to-tree rules were extracted, but to overcome the typically lower translation quality of tree-to-tree systems, we abolish the syntactic annotation on the source side and develop a string-to-tree rule extraction algorithm. The obtained system uses rules with a string on the source language side and a sequence of target tree fragments on the target language side. The availability of several target tree fragments in a single rule enables the model to realize discontinuous translations. We expected that particularly translation into languages with discontinuous constituents would benefit from our model. We demonstrate that our string-to-tree ℓMBOT system significantly outperforms the standard MOSES string-to-tree SCFG system on three different large-scale translation tasks (English-to-German, English-to-Arabic, and English-to-Chinese) with a gain between 0.53 and 0.87 BLEU points. In comparison to the best performing system, the ℓMBOT system is only 1.05 BLEU points worse on English-to-German and 0.14 BLEU points worse for English-to-Chinese. On the English-to-Arabic translation task, the best performing system does outscore the ℓMBOT system by 2.61 BLEU points. An analysis of the rules used to decode the test sets suggests that the usage of discontinuous rules is tuned to each language pair. Furthermore, it shows that only discontinuous rules with at most 8 target tree fragments are used. Thus, further research could investigate a hard limit on the number of target tree fragments during rule extraction.

Next, we demonstrate that the discontinuous string-to-tree system significantly outperforms the standard MOSES string-to-tree system on another two translation tasks (English-to-Polish and English-to-Russian) with large gains of 2.14 and 1.47 BLEU points, respectively. Those two target languages have rather free word order, so we expect discontinuities to occur frequently. For both languages, we use a (non-projective) dependency parser to obtain the required target trees, which we projectivize. We then train our translation model on the constituent-like parse trees obtained from the projective dependency trees and evaluate the obtained machine

translation systems. To put our evaluation scores into perspective, we also trained a vanilla phrase-based system, a MOSESGHKM-based system, and a hierarchical system for each translation task. In comparison to the best performing system, the discontinuous string-to-tree system is only 1.13 BLEU points worse on English-to-Polish and 1.77 BLEU points worse for English-to-Russian. It thus remains to be seen whether machine translation systems can benefit from syntactic information in those translation tasks, but the proposed model at least avoids the large quality drop observed for the continuous string-to-tree system. It shows that our system suffers much less from the syntactic discontinuities and is thus much better suited for syntax-based translation systems in such settings.

Chapter 6

Non-Minimal String-to-String ℓMBOT

Finally, we want to completely abandon the syntactic annotations and derive a rule extraction for string-to-string ℓMBOT rules. To this end, we adjust the theoretical model once more and modify the algorithm from Section 5.3 to extract a set of string-to-string ℓMBOT rules. Naturally, we again evaluate our system on two large scale experiments. To put our results in perspective, we compare against a string-to-string SCFG baseline as well as against a phrase-based system.

6.1 Motivation

Our experiments in Chapter 5 showed that in the string-to-tree setting, the ℓMBOT system can beat a SCFG baseline on five different translation tasks. This is a very promising result but for all five language pairs, a hierarchical phrase-based system outperforms the ℓMBOT system. Consequently, we decided to abandon the syntactic annotations and extract *non-minimal string-to-string rules* that still offer discontinuities on their right-hand sides. As before, we do not want to remove the tree structure from the string-to-tree ℓMBOT rules by replacing the syntactic nonterminals by the placeholder X. This would again preserve parse errors and the right-hand sides of the rules would still be linguistically motivated. Hence, we adapt the algorithm

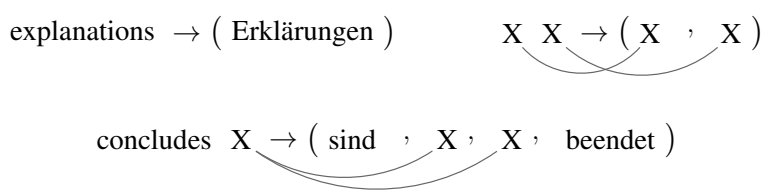

Figure 6.1: Several valid rules for our string-to-string ℓMBOT.

presented in Section 5.3 to extract non-minimal string-to-string rules.

6.2 Theoretical Model

As our translation model, we use a string-to-string variant of the shallow local multi bottom-up tree transducer. We keep on calling our variant ℓMBOT. Our rules are of the form $s \to (t_1, \ldots, t_n)$ with a single *source string* s and a (potential) *sequence of target strings* $(t_1, \ldots, t_n)$, i.e. the source string and the sequence of target strings are built from lexical items and the placeholder X. Both the left- and right-hand side can contain (several occurrences of) the placeholder X. As before, a placeholder in the source string can link to several placeholders in the sequence of target strings but each placeholder occurrence in the right-hand side links to exactly one placeholder on the left-hand side. Again, a string-to-string ℓMBOT is simply a finite collection of such rules. Several valid rules are depicted in Figure 6.1.

Recall from Section 5.2 that a string is of the form $s = w_1 \ldots w_k$ and we access the x-th letter by $w_x = s[x]$ for all $1 \leq x \leq k$. A *sequence of strings* is of the form $(t_1, \ldots, t_n)$ and we access the l-th string by t_l for all $1 \leq l \leq n$. Again, we access the x-th letter in t_l by $t_l[x]$ for all $1 \leq x \leq k$. Now we can obtain the set of placeholders in $(t_1, \ldots, t_n)$ by $ph(t_1, \ldots, t_n) = \bigcup_{i=1}^{n} \{ix \mid t_i[x] = X\}$, i.e. we first access the i-th string inside the sequence and then the x-th letter of the string itself. We obtain the set $ph(s)$ of placeholders in s as explained in Section 5.2. Formally, the alignment between the placeholders is an injective mapping $\psi \colon ph(t_1, \ldots, t_n) \to ph(s) \times \mathbb{N}$ such that if $(x, i) \in ran(\psi)$, then also $(x, j) \in ran(\psi)$ for all $1 \leq j \leq i$. The x-rank $rk(\rho, x)$

of a rule ρ is

$$rk(\rho, x) = \max \{i \in \mathbb{N} \mid (x, i) \in ran(\psi)\} \ .$$

Finally, we extend the substitution for strings as introduced in Section 5.2 to a sequence of strings. Let $x_1, \ldots, x_n \in ph(t_1, \ldots, t_n)$ be positions in a sequence of strings and $v_1, \ldots, v_n$ a sequence of strings. Then $t[x_i \leftarrow v_i]_{1 \leq i \leq n}$ denotes the string that is obtained from $(t_1, \ldots, t_n)$ by replacing in parallel the x_i-th letter by v_i for all $1 \leq i \leq n$.

Definition 8 *The set $\tau(R, c)$ of* weighted sentential forms *of the string-to-string ℓMBOT (R, c) is the smallest set T subject to the following restriction: If there exist*

- *a rule $\rho = s \rightarrow_\psi (t_1, \ldots, t_n) \in R$,*
- *a weighted sentential form $\langle s_x, c_x, (t^x_1, \ldots, t^x_{n_x})\rangle \in T$ for every $x \in ph(s)$ with $\mathrm{rk}(\rho, x) = n_x$,*[1]

then $\langle s', c', (t'_1, \ldots, t'_n)\rangle$ is a weighted sentential form, where

- *$s' = s[x \leftarrow s_x \mid x \in ph(s)]$,*
- *$c' = c(\rho) \cdot \prod_{x \in ph(s)} c_x$, and*
- *$t'_k = t_k[x' \leftarrow t^x_j \mid \psi(ix') = (x, j)]$ for every $1 \leq k \leq n$.*

In a rule ρ, we can replace a placeholder X that is linked to the placeholders $X_1, \ldots, X_l$ in the sequence of target strings by any sentential form whose sequence of target strings match the rank of X. We illustrate substitution in Figure 6.2, where we replace the placeholder X in the source string, which is linked to two placeholders in the right-hand side. The sentential form below matches since its sequence of target strings contains two strings. Thus, we can substitute the sentential form into ρ and obtain the sentential form shown at the bottom of Figure 6.2. Ideally, the substitution process is repeated until the complete source sentence is derived. As before, completely lexical rules are automatically weighted sentential forms and the

[1] If x has n alignments, then the sentential form selected for it has to have suitably many strings inside the sequence of target strings.

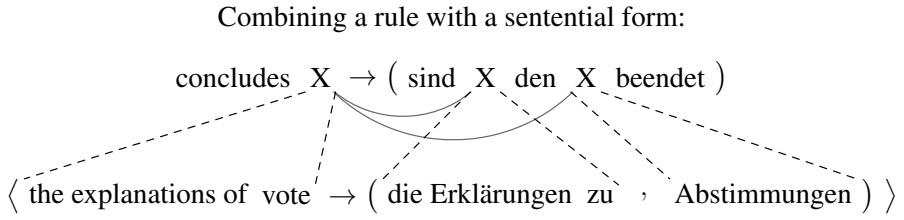

Figure 6.2: Substitution of sentential forms.

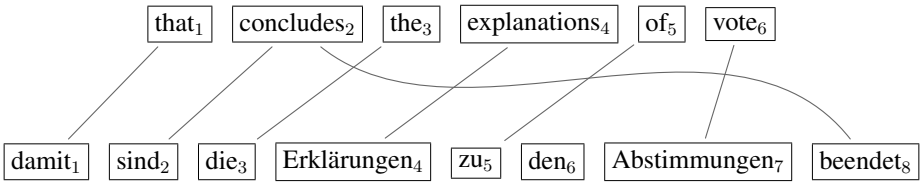

Figure 6.3: Word-aligned sentence pair.

last rule applied must not contain any discontinuities. The final weighted sentential form is again of the form $\langle s, c, (t) \rangle$.

6.3 Rule Extraction

For the string-to-string setting, our training corpus still consists of word-aligned sentence pairs $\langle e, A, f \rangle$ but there are no parses available for either side. We continue to use the notion of a consistently aligned rule span $\langle p, p_1 \cdots p_n \rangle$. Since we do not use any syntactic annotation, we do not require compatibility with any parse tree r. Hence, we can immediately start with the extraction of *initial rules* for $\langle e, A, f \rangle$. For each consistently aligned rule span $\langle p, p_1 \cdots p_n \rangle$, we can extract the rule $e(p) \rightarrow \big(f(p_1), \ldots, f(p_n)\big)$, where

- $e(p)$ is the substring of e at span p,
- $f(p_k)$ is the substring of f at span p_k for each $1 \leq k \leq n$.

For the task at hand, we depict our word-aligned sentence pair for the string-to-string setting in Figure 6.3. We once more consider our example rule span $\langle [2,4],\ [2,4]\,[8,8] \rangle$. It is still consistently aligned and as we do not require compatibility with a target parse tree, we can extract

- $e([2,4]) =$ concludes the explanations
- $f([2,4]) =$ sind die Erklärungen[2]
- $f([8,8]) =$ beendet

We display this rule and several other initial rules extracted for the sentence pair of Figure 6.3 in Figure 6.4.

Again, we derive a new extractable rule r'' by "excising" an initial rule r from another rule r' and replacing the removed part by

- the placeholder X in the source string,
- the placeholder X in the sequence of target strings, and
- linking the removed parts appropriately,

so that the substitution of r into r'' can yield r'. We illustrate this process in Figure 6.5, where we remove the middle initial rule from the topmost initial rule. The result is displayed at the bottom of said figure. Formally, the set of *extractable rules* R for a given word-aligned sentence pair $\langle e, A, f \rangle$ is the smallest set subject to the following two conditions:

- Each initial rule is in R and thus extractable.
- For every initial rule r and extractable rule $r' \in R$, any rule r'', into which we can substitute r to obtain r', is in R and thus extractable.

To restrict the set of extracted rules, we apply the same constraints as for the string-to-tree rule extraction (see Section 5.3) on a rule $s \to (t_1, \ldots, t_n)$. We once again report the number of extracted rules for all translation tasks in Table 6.1. The table shows that the number of extracted rules for string-to-string ℓMBOT increases almost threefold compared to the hierarchical SCFG baselines.

[2]In the string-to-tree setting, we had to split this span because there was no constituent in the target parse tree that matched it.

rule span $\langle [2,2],\ [2,2]\,[8,8] \rangle$:

concludes $\rightarrow$ (sind , beendet)

rule span $\langle [3,3],\ [3,3] \rangle$:

the $\rightarrow$ (die)

rule span $\langle [4,4],\ [4,4] \rangle$:

explanations $\rightarrow$ (Erklärungen)

rule span $\langle [3,4],\ [3,4] \rangle$:

the explanations $\rightarrow$ (die Erklärungen)

rule span $\langle [4,5],\ [4,5] \rangle$:

explanations of $\rightarrow$ (Erklärungen zu)

rule span $\langle [3,5],\ [3,5] \rangle$:

the explanations of $\rightarrow$ (die Erklärungen zu)

rule span $\langle [1,3],\ [1,3]\,[8,8] \rangle$:

that concludes the $\rightarrow$ (damit sind die , beendet)

rule span $\langle [2,4],\ [2,4]\,[8,8] \rangle$:

concludes the explanations $\rightarrow$ (sind die Erklärungen , beendet)

rule span $\langle [1,5],\ [1,5]\,[8,8] \rangle$:

that concludes the explanations of $\rightarrow$ (damit sind die Erklärungen zu , beendet)

rule span $\langle [2,6],\ [2,5]\,[7,8] \rangle$:

concludes the explanations of vote $\rightarrow$ (sind die Erklärungen zu , Abstimmungen beendet)

rule span $\langle [2,6],\ [2,8] \rangle$:

concludes the explanations of vote $\rightarrow$ (sind die Erklärungen zu den Abstimmungen beendet)

Figure 6.4: Some initial rules extractable for the sentence pair of Figure 6.3.

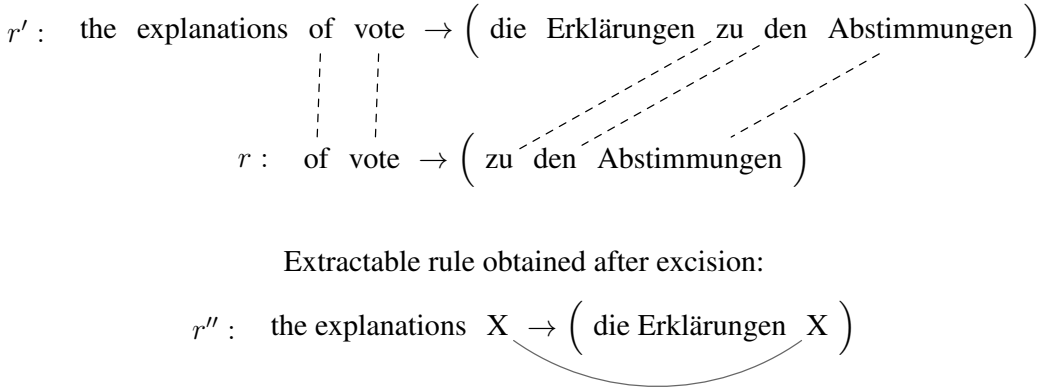

Figure 6.5: Excision of the middle initial rule from the topmost initial rule. Substituting the middle rule into the result yields the topmost rule.

System	**Number of extracted rules**		
	English-To-German	English-To-Arabic	English-To-Chinese
hierarchical SCFG	406,433,344	842,290,537	209,482,192
string-to-string ℓMBOT	1,084,007,782	2,208,445,501	682,505,767

Table 6.1: Number of extracted rules for the different rule extractions.

We present the pseudo-code for the extraction of string-to-string ℓMBOT rules in Algorithm 4. It is a further extension of the non-minimal string-to-tree algorithm (see Section 5.3). Lines 2 to 5 are exactly the same as in the string-to-tree variant. We still iterate by increasing span length over the source sentence and obtain the source string that is covered by it. In line 6 we call FINDCONSISTENTALIGNED. It differs from the string-to-tree variant as it provides us with the consistently aligned target strings (and not target tree fragments) for the current source span. Having the left- and right-hand side for this rule, we add it to R (line 7). The next step is to find unaligned words for each string in the sequence of target strings (lines 9–13). In the tree-to-tree and string-to-tree algorithm we only obtained new rules if no further discontinuities were introduced by the unaligned word(s) and the target parse tree guided this process. We still do not want to have unnecessary discontinuities by adding unaligned words. Given a discontinuous target span to which we add unaligned words, we check whether we can obtain a continuous target span. For example, consider the second to last rule displayed in Figure 6.4. Its target spans are

$[2,5]\,[7,8]$. By adding the unaligned word 'den' spanning $[6,6]$, we obtain the bottom rule of Figure 6.4 where no actual discontinuity is left as the spans $[2,5]\,[6,6]\,[7,8]$ are in fact now the continuous span $[2,8]$. We obtain extractable rules in the same manner as in the string-to-tree algorithm (lines 8 and 12). As before, we ensure that a extractable rule starts with a terminal by calling EXTRACTABLERULES with the index of the next token, i.e. $i+1$ for source span $[i,i']$. The function itself is identical to the one shown in Algorithm 3 and we therefore omit its presentation in Algorithm 4. In line 16 we output all rules that follow constraint (c).

Algorithm 4 Rule extraction for non-minimal string-to-string ℓMBOT rules

Require: word-aligned sentence pair $\langle e, A, f\rangle$
Ensure: ℓMBOT rules R

```
 1: function INITIALRULES(e, A, f)
 2:     for length ← 1, length ≤ maxSpan, length++ do        // ensures constraint (a)
 3:         for i ← 0, i < |e| − (length − 1), i++ do
 4:             i' ← i + length − 1
 5:             s ← e([i, i'])
 6:             (t_1, ..., t_n) ← FINDCONSISTENTALIGNED(A, i, i')
 7:             R[i, i'] ← R[i, i'] ∪ {r' : ⟨s → (t_1, ..., t_n)⟩}
 8:             EXTRACTABLERULES(i + 1, i', r')               // ensures constraint (e)
 9:             for all 1 ≤ k ≤ n do
10:                 ρ ← FINDUNALIGNEDWORDS(r', t_k, A)
11:                 R[i, i'] ← R[i, i'] ∪ ρ
12:                 EXTRACTABLERULES(i + 1, i', ρ)            // ensures constraint (e)
13:             end for
14:         end for
15:     end for
16:     OUTPUTRULES(R, R_E)                                   // ensures constraint (c)
17: end function
```

In Figure 6.6 we display some extractable rules for the rule span $\langle[2,6],\ [2,5]\,[7,8]\rangle$.

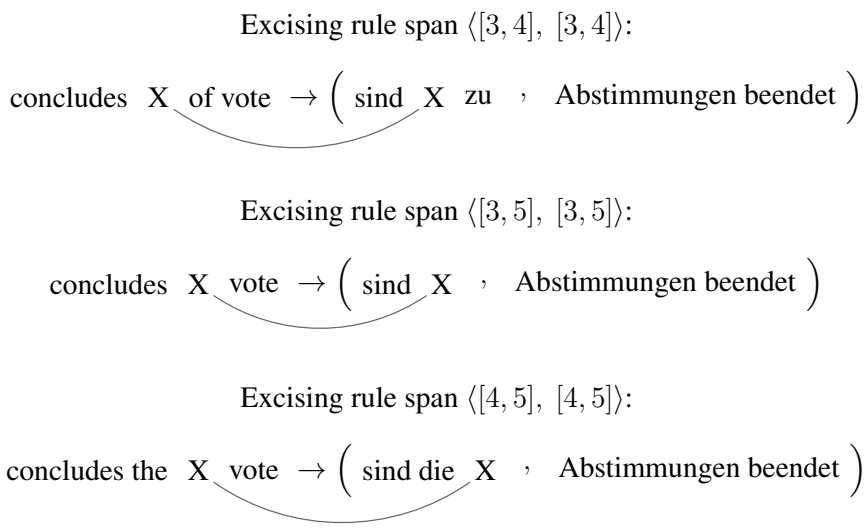

Figure 6.6: Some extractable rules for the rule span $\langle [2,6],\ [2,5]\ [7,8] \rangle$.

6.4 Translation Model

Again, the task of the decoder is to find the best corresponding target language translation $\hat{f}$ of the source language sentence e licensed by the translation model and the language model. We use once again the log-linear model as introduced in Section 4.3 to tune the features $h_m(\cdot)$ with weights λ_m scored on sentential forms $\langle s, (t) \rangle$ of our extracted ℓMBOT M such that s reads e and t reads f.

We use the MBOTMOSES decoder (Braune et al., 2013) which uses a CYK+ chart parsing algorithm using a standard X-style parse tree which is sped up by cube pruning (Chiang, 2007) with integrated language model scoring. The model uses the same ten features as described in Section 3.4.

6.5 Experimental Results

In this section we report the setup and results as presented by Seemann et al. (2015b). We experimentally evaluate the ℓMBOT system in the string-to-string setting. As a baseline system, we use the hierarchical SCFG model of MOSES. We chose

	English to German	**English to Arabic**	**English to Chinese**
training data	7th Europarl Koehn (2005)	MultiUN Eisele and Chen (2010)	
training data size	≈ 1.8M sentence pairs	≈ 5.7M sentence pairs	≈ 1.9M sentence pairs
language model	5-gram SRILM Stolcke (2002)		
add. LM data	WMT 2013	Arabic in MultiUN	Chinese in MultiUN
LM data size	≈ 57M sentences	≈ 9.7M sentences	≈ 9.5M sentences
tuning data	WMT 2013	cut from MultiUN	NIST 2002, 2003, 2005
tuning size	3,000 sentences	2,000 sentences	2,879 sentences
test data	WMT 2013 Bojar et al. (2013)	cut from MultiUN	NIST 2008 NIST (2010)
test size	3,000 sentences	1,000 sentences	1,859 sentences

Table 6.2: Summary of the used resources.

to perform experiments for the translation directions English-to-German, English-to-Arabic, and English-to-Chinese.

6.5.1 Setup

For better comparison, we once again use exactly the same resources as in Section 4.4 and Section 5.5 for the evaluation. We summarize the experimental setup in Table 6.2.

Since there are no parses for the target language side, the German text is now simply tokenized and true-cased. For Arabic, the text is tokenized with MADA (Habash et al., 2009) and transliterated according to Buckwalter (2002). The Chinese sentences were word-segmented using the Stanford Word Segmenter (Chang et al., 2008). As in the string-to-tree setting, the English (source) side of the training data was tokenized and true-cased. After the preprocessing steps, we obtained a word-aligned parallel corpus, to which we applied the rule extraction as described in Section 6.3 together with the baseline rule extraction provided by MOSES (Koehn et al., 2007; Hoang et al., 2009).

	BLEU		
System	**En-De**	**En-Ar**	**En-Zh**
string-to-string SCFG (baseline)	**17.00**	51.71	18.74
string-to-string ℓMBOT	16.57	—	18.60
phrase-based Moses	16.80	51.90	18.09

Table 6.3: BLEU evaluation results for all 3 translation tasks. Bold baseline result is a statistically significant improvement over our system (at confidence $p < 1\%$).

6.5.2 Evaluation

In this section, we first compare all systems to each other using the BLEU score (Papineni et al., 2002). We also present the results obtained by a phrase-based system (Koehn et al., 2003). All systems were tuned for BLEU on the tuning data, and we report the BLEU scores obtained by the tuned systems on the test sets. The ℓMBOT-based systems were evaluated against their corresponding syntax component (Hoang et al., 2009) of the MOSES toolkit, which implements hierarchical (string-to-string) rule extraction. All of them follow essentially the procedure outlined in Chiang (2005), which was also the basis for our string-to-string rule extraction.

We performed large scale experiments on three major translation tasks, namely English-to-German (En-De), English-to-Arabic (En-Ar), and English-to-Chinese (En-Zh). The goal was to evaluate the string-to-string ℓMBOT system. Unfortunately, the rule table of the string-to-string ℓMBOT for the English-to-Arabic translation task — although already filtered on the given input — was too large to load into 500GB of main memory.

Let us now discuss the results for this setting. The experiments for the English-to-German and English-to-Chinese translation task show that our string-to-string ℓMBOT system does not improve performance in these cases. Indeed, the analysis presented below suggests that the string-to-string rules are flexible enough to achieve high coverage even without the need for a sequence of target strings. This is slightly disappointing as Galley and Manning (2010) incorporated discontinu-

ous phrases into a phrase-based system, and their evaluation on Chinese-to-English showed significant improvements over a standard phrase-based baseline as well as over a hierarchical baseline. However, the differences are generally not large (0.43 points and 0.14 points), and even the phrase-based system achieves similar performance.

6.5.3 Analysis of Discontinuity

Naturally, we want to identify whether discontinuous rules were used at all when translating the test sets. We again inspect the statistics on the rules used in the sentential forms. Table 6.4 shows the statistics and we use the already established abbreviations and types. We can confirm that the number of discontinuous rules is indeed marginal. On the English-to-German translation task only 1.1% of 33,962 rules are discontinuous. The English-to-Chinese system also only uses 2.3% discontinuous rules (out of 25,575 rules). This is a large drop compared to the numbers for the string-to-tree setting where 27% and 30% discontinuous rules were used, respectively. In fact, these numbers constitute the new lower bound across the different settings we have explored so far. Until now, the non-minimal tree-to-tree setting had the lowest numbers with 12% and 4% discontinuous rules (see Section 4.4.2). Furthermore, only rules with a sequence of at most three target strings were used in both translation tasks. We believe that the low use of discontinuous string-to-string rules can be explained by the absence of linguistic annotations. Without them, the rules become extremely flexible, thus removing the need for discontinuous ℓMBOT rules in this setting. Since the number of used discontinuous rules is so low, it can be assumed that essentially the same rules were used during decoding when comparing the ℓMBOT system to the baseline. This would also explain their comparable BLEU scores.

Language pair	Type	Lex	Struct	Total
English-to-German	cont.	29,972	3,600	33,572
	discont.	259	131	390
English-to-Chinese	cont.	15,769	9,208	24,977
	discont.	108	490	598

Table 6.4: Number of rules per type used when decoding test (Lex = lexical rules; Struct = structural rules; [dis]cont. = [dis]continuous).

6.6 Summary

We have extended the existing rule extraction techniques for shallow local multi bottom-up tree transducers to the last logical setting. We designed an algorithm that extracts string-to-string ℓMBOT rules, thus using no syntactical information on either side. Naturally, we evaluated the new rule extraction together with two other systems in two large scale translation tasks (English-to-German and English-to-Chinese). The BLEU scores show comparable results for the ℓMBOT system and the baseline with a difference of 0.43 and 0.14 points. An analysis of the rules used to decode the test sets shows that discontinuous rules are hardly ever used. So when compared to the SCFG baseline essentially the same performance is obtained. Most likely, hierarchical rules are flexible enough to handle most common forms of discontinuity without the need to explicitly represent it in its rules.

Chapter 7

ℓMBOTs across all Settings

In this chapter, we present additional analyses of the ℓMBOT systems across the different settings. We start by presenting how many full sentential forms were obtained and by trying to approximate the use of discontinuous rules as truly discontinuous using an inspection of the number of used glue rules. We also present a linguistic analysis for the English-to-German translation tasks on which we gained significant improvements in translation quality. Furthermore, we point out which kinds of construction an ℓMBOT system can handle well.

7.1 Analysis of ℓMBOT Glue Rules

As explained in Section 2.1.1 and illustrated in Section 2.1.3, there are different kinds of glue rules. These different kinds enable us to derive additional statistics. First, we present in Table 7.1 an overview of how many complete sentential forms were obtained in the different settings.[1] Overall, we note that those numbers are not very high. For the English-to-German and the English-to-Chinese translation tasks, a strong decrease is visible from the minimal tree-to-tree to the non-minimal tree-to-tree to the non-minimal string-to-tree setting. Interestingly, for the translation into

[1] Note that the glue rules of a string-to-string translation system do not allow for such an analysis.

Setting	Complete Sentential Forms		
	En-De	**En-Ar**	**En-Zh**
minimal t-to-t ℓMBOT + SCFG	13.1%	—	—
minimal t-to-t ℓMBOT	8.3%	22.6%	13.5%
non-minimal t-to-t ℓMBOT	4.5%	21.8%	2.0%
non-minimal s-to-t ℓMBOT	0.7%	20.1%	0.6%

Table 7.1: Percentage of sentences with complete sentential forms.

Arabic, we note almost identical percentages of complete sentential forms which is probably due to the fact that the test set is excised from the training corpus.

We analyzed the use of discontinuous rules in Sections 3.5.2, 4.4.3, 5.5.3 and 6.5.3. As pointed out across these sections, at present, the analysis tools are not able to distinguish whether a discontinuous rule was actually used discontinuously or rather continuously. We try to approximate by inspecting the number of ℓMBOT glue rules used to decode the test sets. Recall that the application of a ℓMBOT glue rule always results in continuous translation (see explanation on page 59) and we exploit this fact here. Hence, we first count all ℓMBOT glue rules and all rules with two or more target tree fragments, and then compute the percentage of discontinuous rules that are used sequentially. In Table 7.2 we present those numbers. Note that the numbers are collected over the complete test sets. In general, those numbers show that the percentage increases with the introduction of non-minimal rules as well as with the move from trees to strings on the source language side. We note the highest numbers on the English-to-German translation task, while for the translation into Chinese, the lowest numbers are collected. This is interesting insofar, as this is not mirrored in the number of complete sentential forms. For both translation tasks, those numbers are almost equal. In particular, while for English-to-German and English-to-Chinese a strong increase of 24% and 35%, respectively, from the minimal to the non-minimal tree-to-tree setting is noted, there is a moderate increase of 14% for the translation into Arabic.

Additionally, we want to find out whether only rules with more than n target tree fragments were glued. We presented the numbers of rules with n target tree

Setting	En-De	En-Ar	En-Zh
minimal t-to-t ℓMBOT + SCFG	72%	—	—
minimal t-to-t ℓMBOT	56%	54%	34%
non-minimal t-to-t ℓMBOT	80%	68%	71%
non-minimal s-to-t ℓMBOT	99%	97%	91%
non-minimal s-to-s ℓMBOT	93%	—	95%

Table 7.2: Percentage of rules with 2 or more target tree fragments being sequentially applied.

fragments in previous sections (see Figures 3.3, 4.5, 5.4, and 6.4). Our analysis script allows us to distinguish ℓMBOT glue rules according to the number of target tree fragments which they glue together. Hence, we can estimate for all rules with n target tree fragments how many of those were glued. In the minimal setting, we do not find such evidence for all three translation tasks. For example, a rule with 6 target tree fragments was (probably) used discontinuously for translating into German and Arabic, while on the English-to-Chinese translation task, a rule with 7 target tree fragments was used discontinuously. In the non-minimal tree-to-tree setting, all rules with 6 or more target tree fragments were glued for the English-to-Arabic translation task. For the translation into Chinese, all rules with 5 or more target tree fragments were glued. But for the English-to-German translation task, we did not find a pattern. In the non-minimal string-to-tree setting, we find that all rules with 4 or more target tree fragments were glued for all three language pairs.

7.2 Qualitative Analysis

In this section we present an analysis of the English-to-German translations generated by decoding the test sets. This analysis is done for the experiments in which we gained significant improvements on BLEU over the respective baselines. We show translations for both systems and how we obtain better results. Furthermore, we point out linguistic phenomena that are well handled by our ℓMBOT systems.

7.2.1 Minimal Tree-to-Tree ℓMBOT

We start with the evaluation of the minimal tree-to-tree ℓMBOT with additional minimal SCFG rules. All examples are taken from the translation of the test set used for automatic evaluation as presented in Section 3.5.2. We present some evidence that shows how the presence of discontinuous target tree fragments can help model certain linguistic phenomena on the English-to-German translation task. We identified (a) the realization of relative pronouns, reflexive pronouns, and pronominal adverbs, (b) the realization of expletives, (c) the translation of gerunds, (d) the translation into multi-word expressions, and (e) local and long distance reordering to be well realized by discontinuous rules. Ungrammatical constructions are enclosed in brackets and marked with a star.

An analysis of the sentential forms shows that ℓMBOT takes advantage of rules having several target tree fragments. Through its ability to use these discontinuous rules, our system correctly translates English verbs into German ones that require either reflexive pronouns or pronominal adverbs. For example, '*focuses*' translates into the German '*konzentriert sich*' or the German verb '*besteht*' requires a pronominal adverb '*darauf*' (for the English '*insist*'). Similarly, our system handles the generation of necessary relative pronouns after commata well. Pronouns such as 'that' or 'whose' are systematically translated into both '*,*' and '*dass*' or '*,*' and '*deren*'. In Figure 7.1 we show the realization of a verb requiring a pronominal adverb as well as a relative pronoun after a comma. The translation in Figure 7.2 shows the realization of a required reflexive pronoun. The discontinuous rules that were used in those cases are displayed in Figure 7.3.

Similar rules for verbs requiring an expletive *"it"* in German (e.g. *"managed"* translates to *"es geschafft"*; *"allow"* to *"es ermöglichen"*) are used in the sentential forms. Furthermore, there are rules that translate gerunds like *"commemorating"* into *"zum gedenken an"* or *"overcoming"* into *"die Überwindung"*. The translation of a single English word into a German multi-word expression is also well handled by using rules that transform e.g. *"questioned"* into *"in frage gestellt"* or *"given"* into

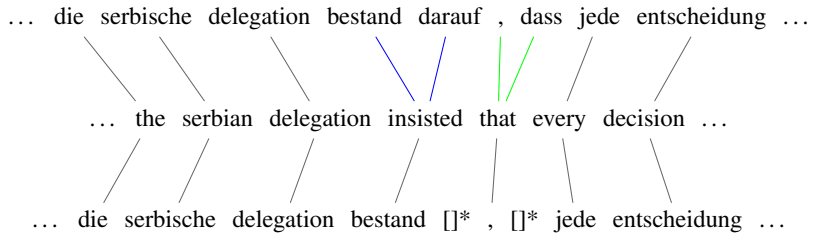

Figure 7.1: Relative Clause marked in green and pronominal adverb marked in blue (top: our system, bottom: baseline)

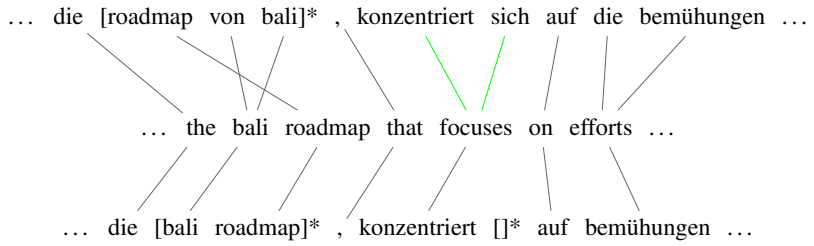

Figure 7.2: Reflexive Pronoun (top: our system, bottom: baseline)

Figure 7.3: ℓMBOT rules generating a relative pronoun, reflexive pronoun, and pronominal adverb.

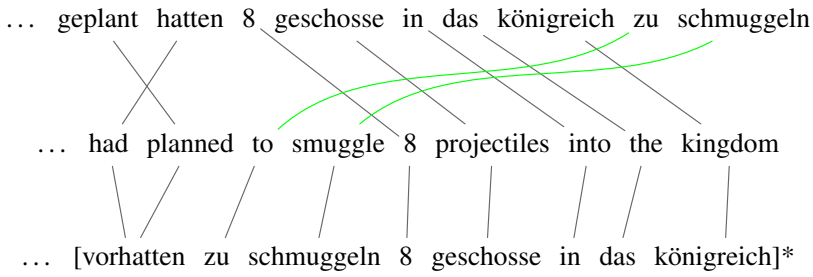

Figure 7.4: Verbal Reordering (top: our system, bottom: baseline)

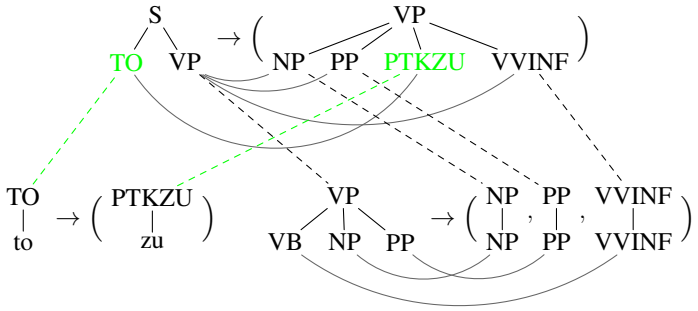

Figure 7.5: Used ℓMBOT rules for verbal reordering shown in Figure 7.4.

"in Anbetracht". In general, these phenomena are not actually discontinuous, but rather just realized at different levels of the parse trees. These structures make it hard for a SCFG system to extract similar rules.

Finally, we show the ability of our system to correctly reorder multiple segments in the source sentence where the baseline translates those segments sequentially. An analysis of the generated sentential forms shows that our system produces the correct translation by taking advantage of rules with discontinuous units on the target language side. Our first example is shown in Figure 7.4. To achieve this translation, our system begins by translating '((*smuggle*)$_{\text{VB}}$ (*eight projectiles*)$_{\text{NP}}$ (*into the kingdom*)$_{\text{PP}}$)$_{\text{VP}}$' into the discontinuous sequence composed of (i) '(*acht geschosse*)$_{\text{NP}}$' ; (ii) '(*in das königreich*)$_{\text{PP}}$' and (iii) '(*schmuggeln*)$_{\text{VP}}$'. In a second step, we assemble all sequences in a rule with a continuous right-hand side and, at the same time, insert the word '(*zu*)$_{\text{PTKZU}}$' between '(*in das königreich*)$_{\text{PP}}$' and '(*schmuggeln*)$_{\text{VP}}$'. The rules used are shown in Figure 7.5.

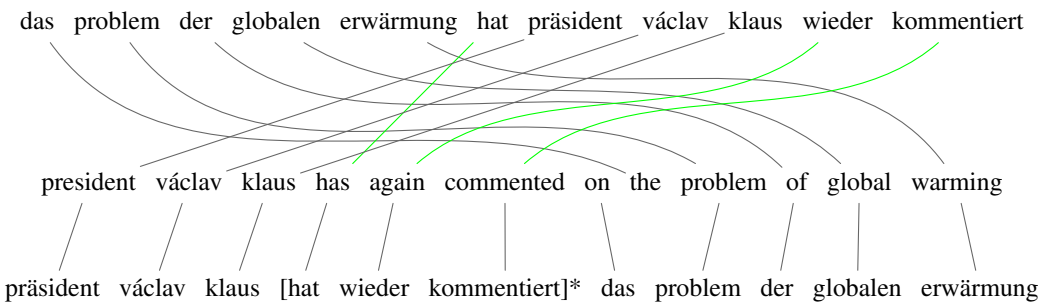

Figure 7.6: Verbal Reordering (top: our system, bottom: baseline)

The second example (Figure 7.6) illustrates an even more complex reordering. First, we translate '((*again*)$_{\text{ADV}}$ *commented on* (*the problem of global warming*)$_{\text{NP}}$)$_{\text{VP}}$' into the discontinuous sequence composed of (i) '(*das problem der globalen erwärmung*)$_{\text{NP}}$'; (ii) '(*wieder*)$_{\text{ADV}}$' and (iii) '(*kommentiert*)$_{\text{VPP}}$'. In a second step, we translate the auxiliary '(*has*)$_{\text{VBZ}}$' by inserting '(*hat*)$_{\text{VAFIN}}$' into the sequence. We thus obtain, for the input segment '((*has*)$_{\text{VBZ}}$ (*again*)$_{\text{ADV}}$ *commented on* (*the problem of global warming*)$_{\text{NP}}$)$_{\text{VP}}$', the sequence (i) '(*das problem der globalen erwärmung*)$_{\text{NP}}$'; (ii) '(*hat*)$_{\text{VAFIN}}$'; (iii) '(*wieder*)$_{\text{ADV}}$'; (iv) '(*kommentiert*)$_{\text{VVPP}}$'. In a last step, the constituent '(*president václav klaus*)$_{\text{NP}}$' is inserted between the discontinuous units '(*hat*)$_{\text{VAFIN}}$' and '(*wieder*)$_{\text{ADV}}$' to form the continuous sequence '((*das problem der globalen erwärmung*)$_{\text{NP}}$ (*hat*)$_{\text{VAFIN}}$ (*präsident václav klaus*)$_{\text{NP}}$ (*wieder*)$_{\text{ADV}}$ (*kommentiert*)$_{\text{VVPP}}$)$_{\text{TOP}}$'. The rules used to achieve this translation are shown in Figure 7.7.

7.2.2 Non-minimal String-to-Tree ℓMBOT

In this section, we inspect some English-to-German translations generated by the SCFG baseline and our ℓMBOT system in order to provide some evidence for linguistic constructions that our system handles better. Again, we identified (a) the realization of reflexive pronouns, pronominal adverbs, and relative pronouns, (b) the realization of expletives, (c) the translation of gerunds, (d) the translation of fixed expressions, and (e) local and long distance reordering to be well realized by discontinuous rules. In this setting, a lot of rules realize a combination of above stated

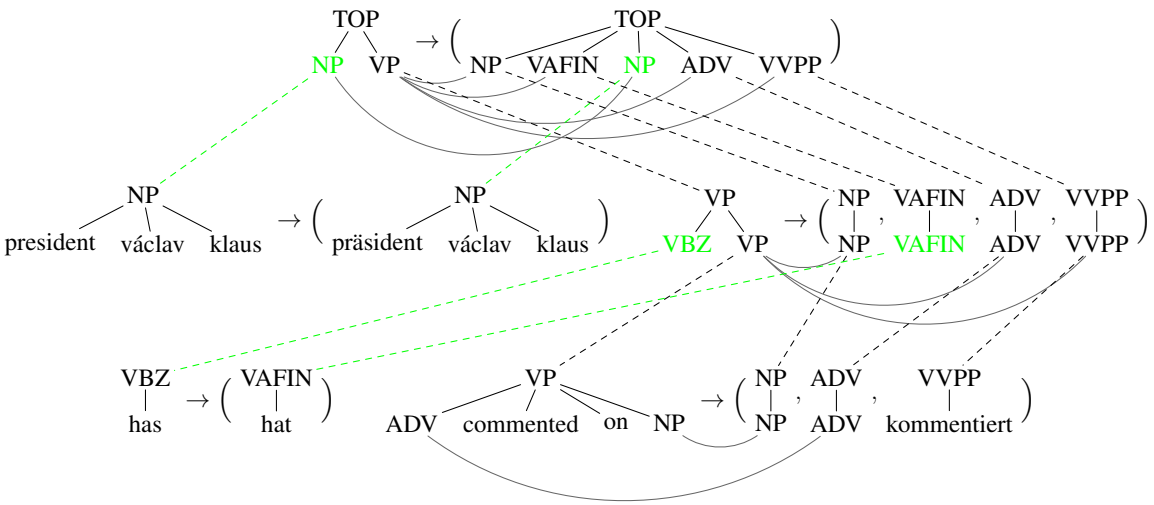

Figure 7.7: Used ℓMBOT rules for verbal reordering shown in Figure 7.6.

linguistic constructions. In most cases, these translations are better than in the baseline system. All examples are (parts of) translations of sentences from the test data used in Section 5.5.2 for automatic evaluation. Again, ungrammatical constructions are enclosed in brackets and marked with a star. We focus on instances that seem relevant to the new ability of having a string on the left-hand side of the rules by showcasing some examples of string-to-tree rules that allow the consistent translation of large segments of the input sentence which are not possible in a tree-to-tree system.

We start with an example (see Figure 7.8) showing the realization of a verb with its required reflexive pronoun '*sich*'.

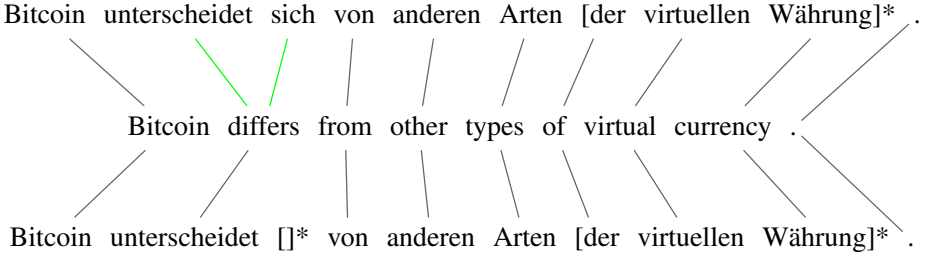

Figure 7.8: Realization of a verb with reflexive pronoun marked in green (top: our system; bottom: baseline).

Here the baseline drops the reflexive pronoun *sich*, which is correctly realized by the ℓMBOT system. The rule used is displayed in Figure 7.9.

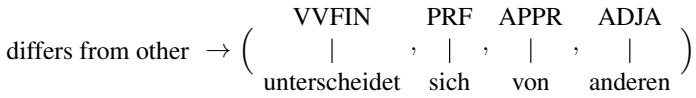

Figure 7.9: Rule realizing the reflexive pronoun of Figure 7.8.

In Figure 7.10, we show how a particle verb as well as its necessary pronominal adverb is realized. The English verb '*assuming*' translates into the German '*davon ausgehen*'. Due to word order of the sentence, it is necessary to split the German verb into '*gehen*' and '*aus*' as well as realizing the pronominal adverb inbetween those parts as shown in Figure 7.10. This translation is achieved by the rule of Figure 7.11. Furthermore, this rule allows for the correct translation of '*that*' into '*, dass*'.

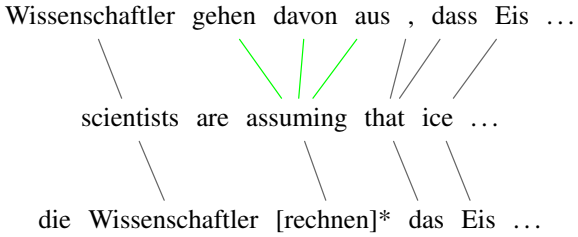

Figure 7.10: Example translation for a particle verb with pronominal adverb (top: our system; bottom: baseline).

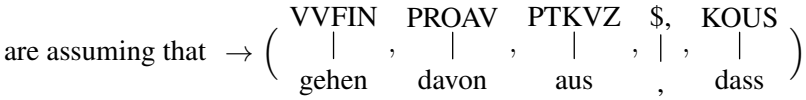

Figure 7.11: ℓMBOT rule with split particle verb and necessary pronominal adverb for the translation shown in Figure 7.10.

Next, we show a translation in Figure 7.12 in which our system correctly translates an English segment that constitutes a fixed expression in both languages. The conversion is achieved by one ℓMBOT rule. The baseline drops the construction

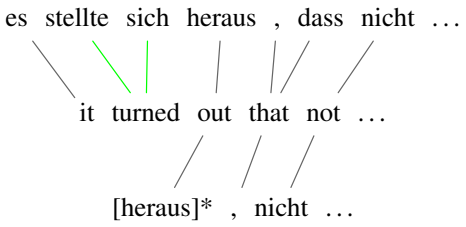

Figure 7.12: A translation showcasing a fixed expression in both English and German (top: our system; bottom: baseline).

almost completely whereas the large string-to-tree rule of Figure 7.13 allows our ℓMBOT system to avoid that drop. Again, the required reflexive pronoun *sich* is realized as well as the necessary comma before the conjunction *dass*.

Figure 7.13: ℓMBOT rule for the fixed expression shown in Figure 7.12.

A similar example is shown in Figure 7.14. The baseline does not generate a verb whereas the ℓMBOT system uses a large rule to translate the expression '*don't worry*' into its correct German counterpart.

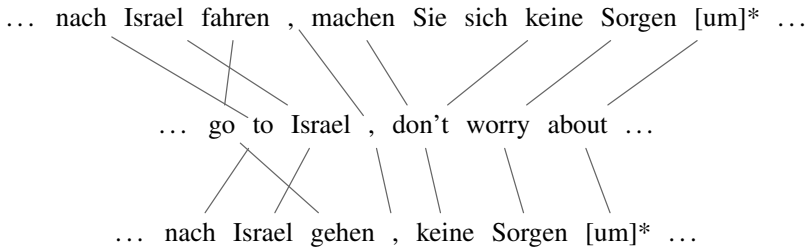

Figure 7.14: Translation of the expression 'don't worry'.

The ℓMBOT rule responsible for the correct translation of the verbal segment *don't worry* is displayed in Figure 7.15. Note how the required reflexive pronoun *sich* is realized.

Our manual analysis of the baseline translations showed that verbal drop is a common mistake, whereas it only rarely happens in the ℓMBOT system.

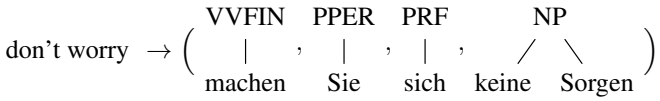

Figure 7.15: ℓMBOT rule used for translation shown in Figure 7.14.

Moreover, we observe that the ℓMBOT system handles reorderings better. In the following example shown in Figure 7.16, it correctly reorders the words whereas the baseline fails. The correct local reorderings are achieved with the help of the rules depicted in Figure 7.17, which translate large parts of the input sentence using several discontinuous target tree fragments. The top rule allows for the reordering of '*is*$_V$ *difficult*$_{ADJ}$' into '*schwierig*$_{ADJ}$ *ist*$_V$' while the bottom rule reorders '*[to interpret]*$_{VC}$ *[the data]*$_{NP}$' into '*[die Daten]*$_{NP}$ *[zu interpretieren]*$_{VC}$' and, additionally, models the required commata before this subordinate clause.

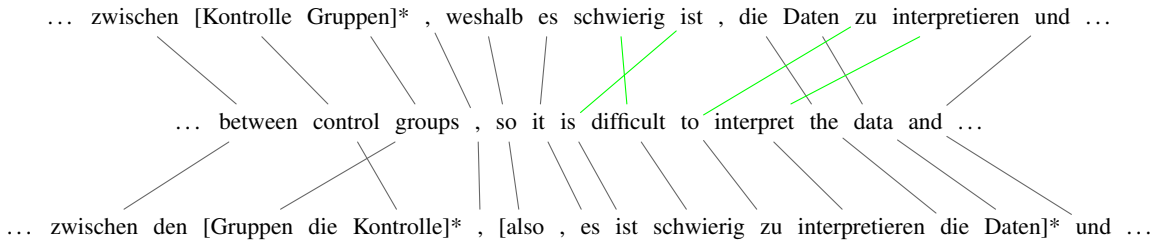

Figure 7.16: Translation with local reorderings marked in green (top: our system; bottom: baseline).

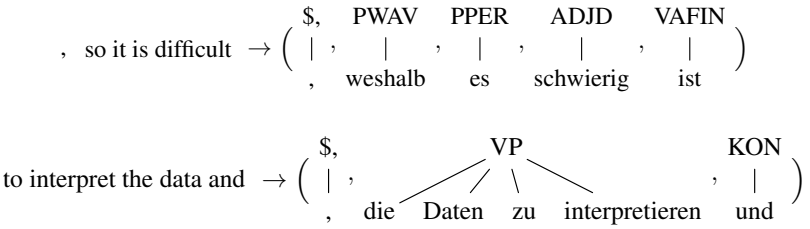

Figure 7.17: Rules enabling the better local reorderings shown in Figure 7.16.

In Figure 7.18, we depict another example of local reordering which is of a more complex nature. The example shows how our system reorders the verbal phrase into the German verb final position as required for subordinate clauses. The base-

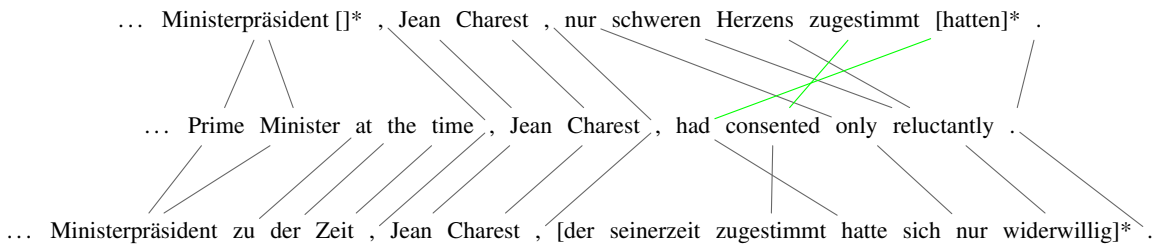

Figure 7.18: Translation with correct complex reordering into verb final position.

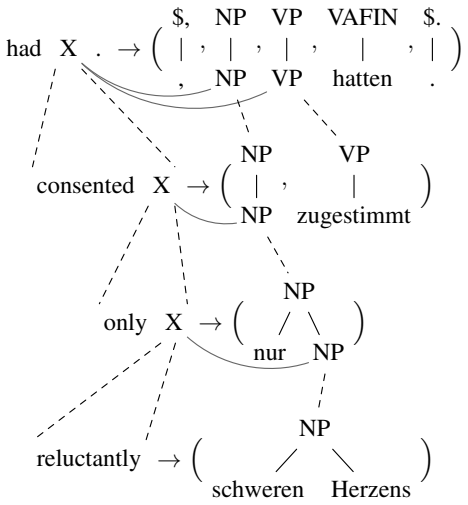

Figure 7.19: Rules enabling the complex reordering of Figure 7.18.

line completely fails to correctly reorder the translation of '*had consented only reluctantly*'. The combination of several discontinuous ℓMBOT rules depicted in Figure 7.19 successfully achieves the reordering, but selects the wrong verb form as indicated in the translation.

Another feature of the ℓMBOT system is its power to perform long distance reordering with the help of several discontinuous target tree fragments. The next example (Figure 7.20) shows how our system reorders the verbal phrase '*to protect [...]*' into the German '*[...] zu schützen*' over a long distance. The lexical choices of the ℓMBOT system are wrong, but it correctly reorders the English segment '*the main goal [...] is to protect the system from corruption*' using its discontinuous rules displayed in Figure 7.21. The baseline fails to reorder the verbal complex.

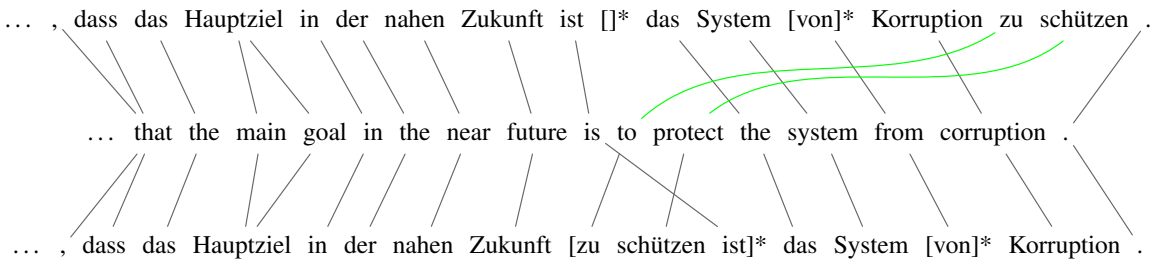

Figure 7.20: Translation with long distance reordering of verbal phrase.

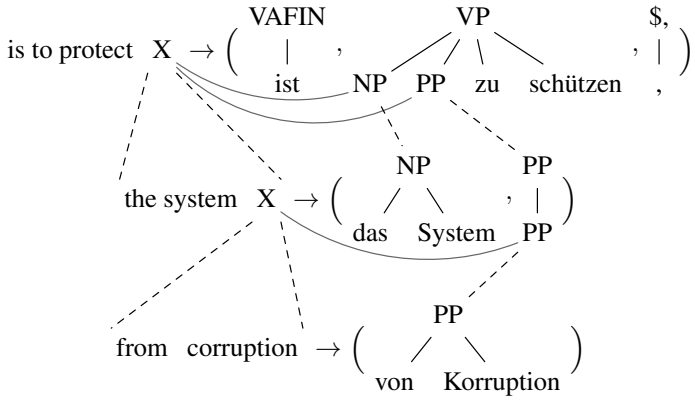

Figure 7.21: ℓMBOT rules enabling the long distance reordering of Figure 7.20.

Another successful long distance reordering is shown in Figure 7.22. Our system uses again several discontinuous ℓMBOT rules to move the German translation of the verb '*endure*' in the verb final position over a long distance. Figure 7.23 shows

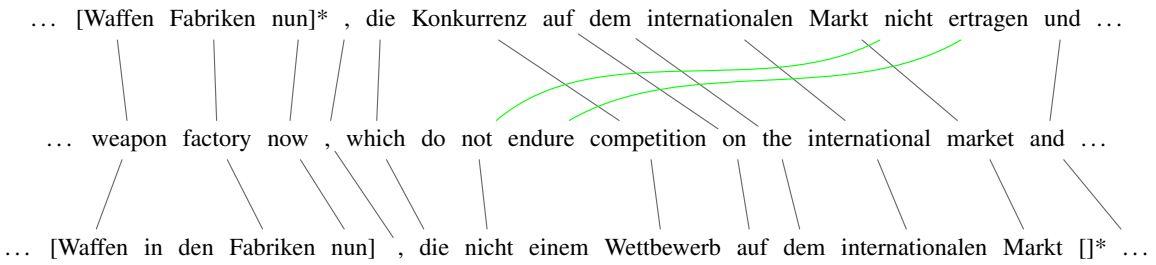

Figure 7.22: Showing a translation that correctly reorders the verb into verb final position (top: our system; bottom: baseline).

the rules which enable the ℓMBOT system to produce the correct reordering. Note how '*endure X*' is reordered into '*NP ertragen*' and the rule on top adds the nega-

tion '*nicht*' between the NP and the VP in its right-hand side.

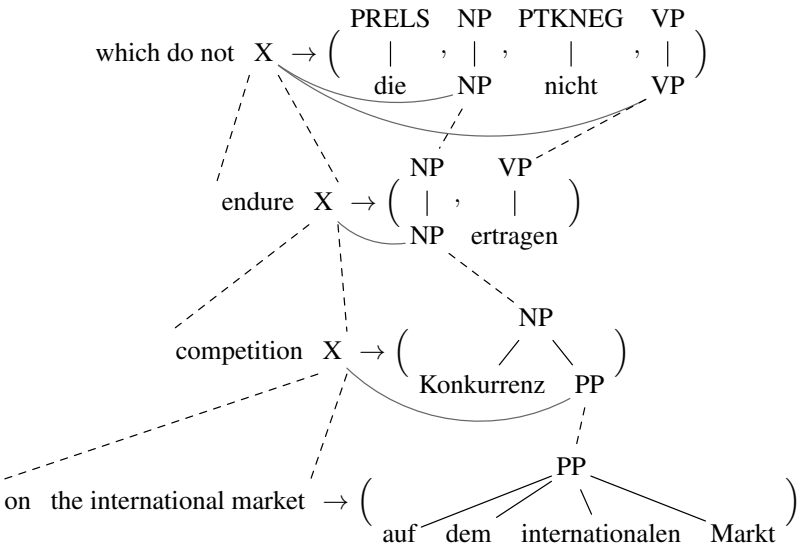

Figure 7.23: ℓMBOT rules used for the long distance reordering shown in Figure 7.22.

7.3 Summary

We presented an evaluation of ℓMBOT systems across all settings. We started by showing for how many sentences in each test set a complete sentential form was obtained by our system. We find that there is a drop for the English-to-German and the English-to-Chinese translation task when moving from minimal rules to non-minimal rules. Also the move from trees to strings on the left-hand side of our rules resulted in another drop. But for the English-to-Arabic translation task, the reported numbers are almost equal.

Next, we inspected the used ℓMBOT glue rules to decide whether discontinuous rules were actually used discontinuously or rather continuously. Our analysis showed that there is an increase of used ℓMBOT glue rules when moving from the minimal to the non-minimal setting as well as moving on to strings on the left-hand side of our rules. A further analysis revealed that in the string-to-tree setting and

for all three translation tasks, the rules with more than two target tree fragments were glued. Since our systems achieved significant improvements over the respective baselines in this setting, we conclude that those rules most probably realize linguistic patterns that do not need to be used discontinuously. In the non-minimal tree-to-tree setting, rules with more than 4 and more than 3 target tree fragments were glued for the English-to-Arabic and English-to-Chinese translation tasks, respectively. For the translation into German we did not find a pattern. Also for the minimal tree-to-tree setting we could not find a pattern in either translation task.

Finally, we presented an qualitative analysis of our English-to-German translation systems which obtained significant improvements over their respective baselines. We found that our translation system systematically realizes certain aspects of the German language, like the realization of reflexive pronouns or local and long distance reordering to be well realized by rules with more than two target tree fragments.

Chapter 8

Conclusion

In this chapter we first summarize our contributions and conclude with an overview of future work.

8.1 Contributions

This thesis contributes to the field of syntax-based statistical machine translation. We introduced a translation model that is based on multi bottom-up tree transducers. In contrast to the traditional continuous SCFG models widely used in syntax-based SMT, an ℓMBOT translation model allows discontinuities on the target language side. A decoder that can handle discontinuous translations was already available (Braune, 2015) as well as a minimal rule extraction algorithm (Maletti, 2011), but the algorithm was neither implemented nor evaluated yet. This work closes the gap. The evaluation of the minimal tree-to-tree ℓMBOT translation system led to further models based on ℓMBOTs with different amounts of linguistic annotation. In the following, we will recap our contributions.

Tree-to-Tree Models We implemented the rule extraction of Maletti (2011) and obtained a set of minimal tree-to-tree ℓMBOT rules which were combined with a

set of minimal SCFG rules. The evaluation on an English-to-German translation task showed significant improvement over a tree-to-tree SCFG baseline. An analysis of the rules used to decode the test set showed that discontinuous rules play indeed a role to translate the sentences.

In general, minimal rules are not well suited for machine translation. Hence, we presented and implemented the first parameterized non-minimal tree-to-tree ℓMBOT rule extraction algorithm. We evaluated the non-minimal tree-to-tree ℓMBOT system and the minimal tree-to-tree ℓMBOT system (without additional minimal SCFG rules) on three different translation tasks. The non-minimal ℓMBOT systems achieve the expected improvements over the minimal ℓMBOT systems but do not beat either SCFG baseline. Our analysis of the rules used to decode the test sets showed a strong shift from discontinuous structural rules (minimal rules) to discontinuous lexical rules (non-minimal rules).

String-to-Tree Models With the encouraging findings from the evaluation of the tree-to-tree ℓMBOTs, we decided to move from syntactic parse trees to simple strings on the source language side. It is well known that tree-to-tree models are too restrictive to achieve good translation results and string-to-tree models yield better translation quality in general. Hence, we presented and implemented a non-minimal string-to-tree ℓMBOT rule extraction algorithm. We evaluated it on the same three translation tasks as for the tree-to-tree models and obtained significant improvements over all string-to-tree SCFG baselines. The scores show increases from 0.66 to 0.78 BLEU points.

Having obtained such significant improvements, we naturally wanted to apply our model on other language pairs for which we expected a gain from discontinuous rules. We decided to evaluate on an English-to-Russian and an English-to-Polish translation task. But for those target languages, only dependency parsers are available which might return non-projective dependency parse trees. This is due to the rather free word order of these languages, which do not allow for a constituent-

like tree representation. Consequently, we applied a (non-projective) dependency parser on our target languages and applied a lifting technique that not only projectivizes the non-projective parses but also documents the lifts in the labels. Next, we transformed the dependency tree structures into constituent-like tree structures. Then we were able to obtain our ℓMBOT systems. On the English-to-Russian translation task, we achieved a significant improvement of 1.47 BLEU points over the string-to-tree SCFG baseline. For the translation into Polish we even gained an impressive improvement of 2.14 BLEU points, thus confirming our claim that a translation system based on ℓMBOTs is quite suitable for such translation tasks.

String-to-String Models For all different syntax-based settings we have explored so far, we noticed that a hierarchical SCFG system achieves the best translation quality in terms of BLEU. As a consequence, we moved from syntactic parse trees to strings on the target language side as well, hoping that the discontinuous rules contribute in this setting. We presented and implemented the third parameterized rule extraction algorithm to obtain a set of (non-minimal) string-to-string ℓMBOT rules. We trained once again our translation system on the already established three translation tasks. Unfortunately, the rule-table for the English-to-Arabic was too large to load into main memory. The evaluation on the English-to-German and English-to-Chinese translation tasks revealed that our ℓMBOT achieves comparable results compared to the hierarchical SCFG baseline system. For the translation into German, the baseline outperforms our system by 0.43 BLEU points and by 0.14 BLEU points on the English-to-Chinese translation task. An analysis of the rules used to decode the test sets showed that hardly any discontinuous rules were used. It seems that string-to-string SCFG rules are flexible enough and do not need the additional flexibility introduced by discontinuities.

Analysis of ℓMBOTs across all Settings We showed for how many sentences in each test set a complete sentential form was obtained by our system. We noticed a drop for the English-to-German and the English-to-Chinese translation task when

moving from minimal rules to non-minimal rules. Another drop became obvious with the move from trees to strings on the left-hand side of our rules. But for the translation into Arabic, the numbers were almost equal for all settings.

We try to estimate the use of discontinuous rules as truly discontinuous by inspecting the used ℓMBOT glue rules. Our analysis revealed an increase of glued discontinuous rules when moving from the minimal to the non-minimal setting as well as moving on to strings on the left-hand side of our rules. Next, we further analyzed whether discontinuous rules with a certain number of target tree fragments were glued. In the string-to-tree setting and for all three translation tasks, all rules with more than two target tree fragments were glued. For the minimal tree-to-tree setting we could not find a pattern in either translation task. In the non-minimal tree-to-tree setting, we found a pattern for the English-to-Arabic and English-to-Chinese translation tasks but not for the translation into German.

Finally, we presented a qualitative analysis of our English-to-German translation systems which obtained significant improvements over their respective baselines. We found that our translation system systematically realizes certain aspects of the German language, like the realization of reflexive pronouns and pronominal adverbs, well by rules with more than two target tree fragments. In the string-to-tree setting we also notice how fixed expressions are well translated. Furthermore, our system is able to correctly model local and long distance reorderings.

Software and Analysis tools We publicly release all our software and analysis tools to enable the SMT community to experiment and improve on our work.

8.2 Future Work

Restriction on Target Tree Fragments Our analysis for the different translation tasks showed that some languages use discontinuous rules with up to a specific number of target tree fragments. It is possible to implement a restriction that ex-

tracts only rules with up to n target tree fragments and specify n for the desired target language.

Given this restriction, it would also be possible to allow only rules with two target tree fragments or a sequence of two target strings . The experiments for the discontinuous tree-to-tree setting of Sun et al. (2009) and the discontinuous string-to-string setting of Kaeshammer (2015) showed significant improvements over a corresponding baseline with a strict limitation to two discontinuities per rule. As we did not beat the baselines in the tree-to-tree and string-to-string setting, maybe we can gain improvements with this restriction.

English-to-German translation task For the English-to-German translation tasks we removed the full parse tags which we obtained from BitPar (Schmid, 2004) to avoid data sparsity. Our manual inspection of translations for this language pair (see Section 7.2) showed that the lexical choices of the ℓMBOT system are sometimes wrong. It could help the translation quality, if some information from the full tags like number and gender would be kept.

Incorporating Dependency Parses It would also be interesting to experiment with dependency parses where the lifts are not documented. Since there were not that many non-projective structures present for both Russian and Polish, it could be that the same results can be obtained.

Composition All three variants of our parameterized (non-minimal) rule extraction algorithms apply a heuristic to obtain a set of rules. Further research could investigate a rule extraction that first extracts minimal rules only and obtains from those additional rules by composition. Galley et al. (2006) use this approach to improve the performance of Galley et al. (2004). Furthermore, Williams and Koehn (2012) use composition for the extraction of MOSESGHKM rules.

Tree-to-String Setting Another point of research could be the implementation of a tree-to-string variant. Especially for the translation direction into English, this could provide insights whether a language with “simple” grammar can benefit from discontinuous rules.

Bibliography

Aho, A. V. and Ullman, J. D. (1969). Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3(1):37–56.

Almaghout, H., Jiang, J., and Way, A. (2011). CCG contextual labels in hierarchical phrase-based SMT. In *Proc. 15th EAMT*, pages 281–288. European Association for Machine Translation.

Ambati, V. and Lavie, A. (2008). Improving syntax driven translation models by re-structuring divergent and non-isomorphic parse tree structures. In *Proc. 8th AMTA*, pages 235–244. Association for Machine Translation in the Americas.

Ambati, V., Lavie, A., and Carbonell, J. (2009). Extraction of syntactic translation models from parallel data using syntax from source and target languages. In *Proc. MT Summit XII*, pages 190–197. Association for Machine Translation in the Americas.

Arnold, A. and Dauchet, M. (1982). Morphismes et bimorphismes d'arbres. *Theoretical Computer Science*, 20(1):33–93.

Baroni, M., Bernardini, S., Ferraresi, A., and Zanchetta, E. (2009). The WaCky Wide Web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Bojar, O., Buck, C., Callison-Burch, C., Federmann, C., Haddow, B., Koehn, P., Monz, C., Post, M., Soricut, R., and Specia, L. (2013). Findings of the 2013 Workshop

on Statistical Machine Translation. In *Proc. 8th WMT*, pages 1–44. Association for Computational Linguistics.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L., and Tamchyna, A. (2014). Findings of the 2014 Workshop on Statistical Machine Translation. In *Proc. 9th WMT*, pages 12–58. Association for Computational Linguistics.

Bojar, O. and Hajič, J. (2008). Phrase-based and deep syntactic English-to-Czech statistical machine translation. In *Proc. 3rd WMT*, pages 143–146. Association for Computational Linguistics.

Brants, S., Dipper, S., Eisenberg, P., Hansen-Schirra, S., König, E., Lezius, W., Rohrer, C., Smith, G., and Uszkoreit, H. (2004). TIGER: Linguistic interpretation of a German corpus. *Research on Language and Computation*, 2(4):597–620.

Braune, F. (2015). *Decoding Strategies for syntax-based Statistical Machine Translation*. PhD thesis, Universität Stuttgart.

Braune, F., Seemann, N., Quernheim, D., and Maletti, A. (2013). Shallow local multi bottom-up tree transducers in statistical machine translation. In *Proc. 51st ACL*, pages 811–821. Association for Computational Linguistics.

Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Buch-Kromann, M. (2005). *Discontinuous Grammar — A dependency-based model of human parsing and language learning*. PhD thesis, Copenhagen Business School.

Buckwalter, T. (2002). Arabic transliteration. `http://www.qamus.org/transliteration.htm`.

Callison-Burch, C., Koehn, P., Monz, C., and Schroeder, J. (2009). Findings of the 2009 Workshop on Statistical Machine Translation. In *Proc. 4th WMT*, pages 1–28. Association for Computational Linguistics.

Chang, P.-C., Galley, M., and Manning, C. D. (2008). Optimizing Chinese word segmentation for machine translation performance. In *Proc. 3rd WMT*, pages 224–232. Association for Computational Linguistics.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. 43rd ACL*, pages 173–180. Association for Computational Linguistics.

Chen, S. F. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard University.

Chiang, D. (2005). Hierarchical phrase-based translation. In *Proc. 43rd ACL*, pages 263–270. Association for Computational Linguistics.

Chiang, D. (2006). An introduction to synchronous grammars. In *Proc. 44th ACL*. Association for Computational Linguistics. Part of a tutorial given with Kevin Knight.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Chiang, D. (2010). Learning to translate with source and target syntax. In *Proc. 48th ACL*, pages 1443–1452. Association for Computational Linguistics.

DeNeefe, S., Knight, K., Wang, W., and Marcu, D. (2007). What can syntax-based MT learn from phrase-based MT? In *Proc. 2007 EMNLP*, pages 755–763. Association for Computational Linguistics.

Eisele, A. and Chen, Y. (2010). MultiUN: A multilingual corpus from United Nation documents. In *Proc. 7th LREC*, pages 2868–2872. European Language Resources Association.

Eisner, J. (2003). Learning non-isomorphic tree mappings for machine translation. In *Proc. 41st ACL*, pages 205–208. Association for Computational Linguistics.

Galley, M., Graehl, J., Knight, K., Marcu, D., DeNeefe, S., Wang, W., and Thayer, I. (2006). Scalable inference and training of context-rich syntactic translation models. In *Proc. 44th ACL*, pages 961–968. Association for Computational Linguistics.

Galley, M., Hopkins, M., Knight, K., and Marcu, D. (2004). What's in a translation rule? In *Proc. 2004 NAACL*, pages 273–280. Association for Computational Linguistics.

Galley, M. and Manning, C. D. (2010). Accurate non-hierarchical phrase-based translation. In *Proc. 2010 HLT-NAACL*, pages 966–974. The Association for Computational Linguistics.

Gao, Q. and Vogel, S. (2008). Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57. Association for Computational Linguistics.

Gimpel, K. (2011). Code for statistical significance testing for MT evaluation metrics. `http://www.ark.cs.cmu.edu/MT/`.

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3–4):237–264.

Habash, N., Rambow, O., and Roth, R. (2009). MADA+TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In *Proc. 2nd MEDAR*, pages 102–109. Association for Computational Linguistics.

Hanneman, G. and Lavie, A. (2013). Improving syntax-augmented machine translation by coarsening the label set. In *Proc. 2013 HLT-NAACL*, pages 288–297. Association for Computational Linguistics.

Hoang, H. (2011). *Improving Statistical Machine Translation with Linguistic Information*. PhD thesis, University of Edinburgh.

Hoang, H., Koehn, P., and Lopez, A. (2009). A unified framework for phrase-based, hierarchical, and syntax-based statistical machine translation. In *Proc. 6th IWSLT*, pages 152–159. ISCA.

Hopkins, M. and Langmead, G. (2010). SCFG decoding without binarization. In *Proc. 2010 EMNLP*, pages 646–655. Association for Computational Linguistics.

Kaeshammer, M. (2015). Hierarchical machine translation with discontinuous phrases. In *Proc. 10th WMT*, pages 228–238. Association for Computational Linguistics.

Kahane, S., Nasr, A., and Rambow, O. (1998). Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *Proc. 36th ACL*, pages 646–652. Association for Computational Linguistics.

Kallestinova, E. D. (2007). *Aspects of Word Order in Russian*. PhD thesis, University of Iowa, IA, USA.

Knight, K. and Graehl, J. (2005). An overview of probabilistic tree transducers for natural language processing. In *Proc. CICLing*, volume 3406 of *Lecture Notes in Computer Science*, pages 1–24. Springer.

Koehn, P. (2004a). Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proc. 6th AMTA*, pages 115–124. Association for Machine Translation in the Americas.

Koehn, P. (2004b). Statistical significance tests for machine translation evaluation. In *Proc. 2004 EMNLP*, pages 388–395. Association for Computational Linguistics.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proc. 10th MT Summit*, pages 79–86. Association for Machine Translation in the Americas.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA.

Koehn, P., Axelrod, A., Mayne, A. B., Callison-Burch, C., Osborne, M., and Talbot, D. (2005). Edinburgh system description for the 2005 IWSLT Speech Translation Evaluation. In *Proc. 2nd IWSLT*, pages 68–75. ISCA.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proc. 45th ACL*, pages 177–180. Association for Computational Linguistics.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proc. 2003 HLT-NAACL*, pages 48–54. Association for Computational Linguistics.

Kübler, S., McDonald, R., Nivre, J., and Hirst, G. (2009). *Dependency Parsing*. Morgan and Claypool Publishers.

Lavie, A., Parlikar, A., and Ambati, V. (2008). Syntax-driven learning of subsentential translation equivalents and translation rules from parsed parallel corpora. In *Proc. 2nd SSST*, pages 87–95. Association for Computational Linguistics.

Li, L., Xie, J., Way, A., and Liu, Q. (2014). Transformation and decomposition for efficiently implementing and improving dependency-to-string model in Moses. In *Proc. 8th SSST*, pages 122–131. Association for Computational Linguistics.

Lilin, E. (1978). *Une généralisation des transducteurs d'états finis d'arbres: les S-transducteurs*. Thèse 3ème cycle, Université de Lille.

Liu, Y., Lü, Y., and Liu, Q. (2009). Improving tree-to-tree translation with packed forests. In *Proc. 47th ACL*, pages 558–566. Association for Computational Linguistics.

Maletti, A. (2011). How to train your multi bottom-up tree transducer. In *Proc. 49th ACL*, pages 825–834. Association for Computational Linguistics.

Marcu, D., Wang, W., Echihabi, A., and Knight, K. (2006). SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. 2006 EMNLP*, pages 44–52. Association for Computational Linguistics.

Neubig, G. and Watanabe, T. (2016). Optimization for statistical machine translation: A survey. *Computational Linguistics*, 42(1):1–54.

NIST (2010). NIST 2002 [2003, 2005, 2008] open machine translation evaluation. Linguistic Data Consortium. LDC2010T10 [T11, T14, T21].

Nivre, J., Boguslavsky, I. M., and Iomdin, L. L. (2008). Parsing the SynTagRus treebank of Russian. In *Proc. 22nd CoLing*, pages 641–648. Association for Computational Linguistics.

Nivre, J., Hall, J., and Nilsson, J. (2006). Maltparser: A data-driven parser-generator for dependency parsing. In *Proc. 5th LREC*, pages 2216–2219. European Language Resources Association.

Nivre, J. and Nilsson, J. (2005). Pseudo-projective dependency parsing. In *Proc. 43rd ACL*, pages 99–106. Association for Computational Linguistics.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proc. 41st ACL*, pages 160–167. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, F. J. and Ney, H. (2004). The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proc. 40th ACL*, pages 311–318. Association for Computational Linguistics.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proc. 44th ACL*, pages 433–440. Association for Computational Linguistics.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *Proc. Int. Conf. New Methods in Language Processing*, pages 44–49. University of Manchester, Institute of Science and Technology.

Schmid, H. (2004). Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proc. 20th CoLing*, pages 162–168. Association for Computational Linguistics.

Seemann, N., Braune, F., and Maletti, A. (2015a). String-to-tree multi bottom-up tree transducers. In *Proc. 53rd ACL*, pages 815–824. Association for Computational Linguistics.

Seemann, N., Braune, F., and Maletti, A. (2015b). A systematic evaluation of MBOT in statistical machine translation. In *Proc. MT Summit XV*, pages 200–214. Association for Machine Translation in the Americas.

Seemann, N. and Maletti, A. (2015). Discontinuous statistical machine translation with target-side dependency syntax. In *Proc. 10th WMT*, pages 239–247. Association for Computational Linguistics.

Sennrich, R., Williams, P., and Huck, M. (2015). A tree does not make a well-formed sentence: Improving syntactic string-to-tree statistical machine translation with more linguistic knowledge. *Computer Speech & Language*, 32(1):27–45.

Sharoff, S. and Nivre, J. (2011). The proper place of men and machines in language technology processing Russian without any linguistic knowledge. In *Proc. Dialogue*, pages 657–670. Russian State University for the Humanities.

Shieber, S. M. and Schabes, Y. (1990). Synchronous tree-adjoining grammars. In *Proc. 13th CoLing*, pages 253–258. Association for Computational Linguistics.

Steedman, M. (2000). *The Syntactic Process*. MIT Press, Cambridge, MA, USA.

Stolcke, A. (2002). SRILM — an extensible language modeling toolkit. In *Proc. 7th INTERSPEECH*, pages 257–286. ISCA.

Sun, J., Zhang, M., and Tan, C. L. (2009). A non-contiguous tree sequence alignment-based model for statistical machine translation. In *Proc. 47th ACL*, pages 914–922. Association for Computational Linguistics.

Web-as-Corpus Consortium (2008). SDeWaC — a 0.88 billion word corpus for German. Website: `http://wacky.sslmit.unibo.it/doku.php`.

Wellington, B., Waxmonsky, S., and Melamed, I. D. (2006). Empirical lower bounds on the complexity of translational equivalence. In *Proc. 44th ACL*, pages 977–984. Association for Computational Linguistics.

Williams, P. and Koehn, P. (2012). GHKM rule extraction and scope-3 parsing in Moses. In *Proc. 7th WMT*, pages 388–394. Association for Computational Linguistics.

Wróblewska, A. and Przepiórkowski, A. (2012). Induction of dependency structures based on weighted projection. In *Proc. 4th ICCCI*, volume 7653 of LNAI, pages 364–374. Springer.

Xie, J., Mi, H., and Liu, Q. (2011). A novel dependency-to-string model for statistical machine translation. In *Proc. 2011 EMNLP*, pages 216–226. Association for Computational Linguistics.

Zhang, M., Jiang, H., Aw, A., Li, H., Tan, C. L., and Li, S. (2008). A tree sequence alignment-based tree-to-tree translation model. In *Proc. 46th ACL*, pages 559–567. Association for Computational Linguistics.

Zhang, M., Jiang, H., Aw, A. T., Sun, J., Li, S., and Tan, C. L. (2007). A tree-to-tree alignment-based model for statistical machine translation. In *Proc. 11th MT Summit*, pages 535–542. European Association for Machine Translation.

Zollmann, A. and Venugopal, A. (2006). Syntax augmented machine translation via chart parsing. In *Proc. 1st WMT*, pages 138–141. Association for Computational Linguistics.