Left-to-Right Hierarchical Phrase-based Machine Translation

by

Maryam Siahbani

M.Sc., Isfahan University of Technology, 2009
B.Sc., University of Shiraz, 2006

Dissertation Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2016

Approval

Name:	Maryam Siahbani
Degree:	Doctor of Philosophy (Computing Science)
Title:	*Left-to-Right Hierarchical Phrase-based Machine Translation*
Examining Committee:	**Dr. Nick Sumner** (chair) Assistant Professor

Dr. Anoop Sarkar
Senior Supervisor
Professor _____

Dr. Fred Popowich
Supervisor
Professor _____

Dr. Leonid Chindelevitch
Internal Examiner
Assistant Professor _____

Dr. Bill Byrne
External Examiner
Professor
Department of Engineering
University of Cambridge _____

Date Defended: 13 July 2016

Abstract

Hierarchical phrase-based translation (Hiero for short) models statistical machine translation (SMT) using a lexicalized synchronous context-free grammar (SCFG) extracted from word aligned bitexts. The standard decoding algorithm for Hiero uses a CKY-style dynamic programming algorithm with time complexity $O(n^3)$ for source input with n words. Scoring target language strings using a language model in CKY-style decoding requires two histories per hypothesis making it significantly slower than phrase-based translation which only keeps one history per hypothesis. In addition, the size of a Hiero SCFG grammar is typically much larger than phrase-based models when extracted from the same data which also slows down decoding. In this thesis we address these issues in Hiero by adopting a new translation model and decoding algorithm called Left-to-Right hierarchical phrase-based translation (LR-Hiero for short). LR-Hiero uses a constrained form of lexicalized SCFG rules to encode translation, where the target-side is constrained to be prefix-lexicalized. LR-Hiero uses a decoding algorithm with time complexity $O(n^2)$ that generates the target language output in left-to-right manner which keeps only one history per hypothesis resulting in faster decoding for Hiero grammars.

The thesis contains the following contributions:

- We propose a novel dynamic programming algorithm for the rule extraction phase. Unlike traditional Hiero rule extraction which performs a brute-force search, LR-Hiero rule extraction is linear in the number of rules.
- We propose an augmented version of LR-decoding algorithm previously proposed by Watanabe et al. [90]. Our modified LR-decoding algorithm addresses issues related to decoding time and translation quality and is shown to be more efficient than the CKY decoding algorithm in our experimental results.
- We extend our LR-decoding algorithm to capture all hierarchical phrasal alignments that are reachable in CKY-style decoding algorithms.
- We introduce a lexicalized reordering model to LR-Hiero that significantly improves the translation quality.

- We apply LR-Hiero to the task of simultaneous translation, the first attempt to use Hiero models in simultaneous translation. We show that we can perform online segmentation on the source side to improve latency and maintain translation quality.

Keywords: Statistical Machine Translation; Decoding; Hierarchical Phrase-based Translation

Dedication

To my family.

Acknowledgements

I would like to sincerely thank my senior supervisor Prof. Anoop Sarkar for his supports and gaudiness during my PhD at Simon Fraser University. I appreciate all his contribution of time, ideas, incredible patience, understanding and encouragement. I learned from every moment I spent in his research lab and courses as a student and even teaching assistant. He has always been a role model in my life.

I would also like to thank my supervisor Prof. Fred Popowich for his supports, encouragements and insightful comments. I would like to thank my thesis committee: Prof. Bill Byrne and Dr. Leonid Chindelevitch. It was my honor to have them on the committee.

I learned a great deal of what I know in machine translation and NLP from the great students in the natlang lab at SFU. I would like to thank them for all unforgettable memories they made: Baskaran Sankaran, Majid Razmara, Ravikiran Vadlapudi, Marzieh Razavi, Ramtin Mehdizadeh Seraj, Max Whitney, Ann Clifton, Milan Tofiloski, David Lindberg, Anahita Mansouri, Mark Schmidt, Jasneet Sabharwal, Rohit Dholakia, Te Bu, Golnar Sheikhshab, Mehdi Soleimani, Hassan Shaavarani, Lydia Odilinye, Fariha Naz, Bradley Ellert and Zhelun Wu.

I would also like to thank my amazing friends: Shabnam Mirshokraie, Farnaz Agahian, Hengameh Mirzaalian and Hossein Hajimirsadeghi. Thank you for your extreme supports and kindness during my life in Vancouver.

Finally, I owe my deepest thank to my family: my parents and my sisters and brothers: Soraya, Shamsi, Maliheh, Shapoor, Sadegh, Hossein, Sara, Saba and Sina. Those who always provide unconditional love, supports and encourage me to pursue my interests. They are the greatest treasures of my life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine translation (MT) is the task of automated text translation between natural languages. Translation is a challenging task which requires a thorough understanding of the source text as well as good knowledge of the target language. MT systems should formalise the whole process of translation which is an extremely difficult task since the translation process has not been completely understood yet. The introduction of statistical alignment models in 1990's [9] launched a new research direction in MT: *Statistical Machine Translation* (SMT). SMT analyses the human translations with statistical methods to extract implicit information about the translation process from corpora of translated texts.

The initial SMT models use word-based translation models [9] that not only take lexical translation into account, but also model reordering, insertion, deletion or duplication of words. The word-based models gave way to the subsequent phrase-based models [52, 61, 63], which use the translation of phrases as atomic units. In these models, phrases are any contiguous sequence of words that do not need to have any syntactic validity.

Hierarchical phrase-based machine translation (*Hiero*) [14, 15] is another prominent approach for SMT. Hiero models encode the translation correspondences in *hierarchical* phrases. The notion of hierarchy allows the Hiero models to capture the long-distance reordering between source and target languages.Additionally Hiero can model discontiguous translations as exemplified by the canonical example of translating the English word *not* as *ne* ___ *pas* in French. These properties make Hiero models more appropriate for language pairs involving complex reordering than the phrase-based models [52, 63, 15].

The syntax-based models represent the evolutionary next step in SMT. They utilize syntactic structure of source and/or target sides for translation. Researchers have proposed myriad models using different combinations such as string-to-tree, tree-to-tree and so on [92, 68, 28]. A major limitation of these models is their requirement for a syntactic parser for at least one language, which limits its application to a few high resource languages. Unlike the full syntax-based models, Hiero models do not require any linguistic parser making them much simpler to train and decode with.

Hiero translation models have been shown to attain competitive performance with other statistical frameworks [10] for several language pairs. Thus Hiero translation models are attractive choices for a wide variety of languages. The focus of this thesis is on Hiero translation models. We proposed a new translation system based on Hiero translation models which works more efficiently than the state-of-the-art Hiero translation systems.

1.1 Statistical Machine Translation

In statistical machine translation, the translation process can be considered as decoding the meaning of the source text and re-encoding it into the target language [43]. The main idea of SMT comes from decision theory. We try to translate the given input sentence according to the translation probability $p(e|f)$ (i.e. the probability a target string e is the translation of the given input string f in the source language). In SMT, the most common approach for estimating $p(e|f)$ uses the noisy channel model:

$$\begin{aligned} \hat{e} &= \arg\max_{e} p(e|f) \\ &= \arg\max_{e} p(f|e).p(e) \end{aligned} \tag{1.1}$$

Using Bayes theorem, the problem is divided into two sub-problems: $p(f|e)$ the probability that the source string is the translation of the target string, it is called *translation model*; and $p(e)$ the probability of seeing that target language string, called *language model*. The translation model is responsible for generating an adequate translation, while the language model is responsible to score the translation candidates based on the probability of seeing such a sequence of words in the target language. There are different approaches for modeling languages: the two common approaches are using n-gram models (i.e. unigram, bigram, trigram, etc.) and recently neural networks [20, 76]. Different approaches have been proposed for the translation model:

- **Word-based Models**: The early SMT models are word-based which use words as translation units, proposed by Brown et al. (1993) [9] in IBM. They proposed 5 models called IBM Models 1-5 [8, 9]. Later Och (2003) [62] proposed Model 6. There is another famous word-based model which incorporates a Hidden Markov Model over word alignments, called HMM model [88].

- **Phrase-based Models**: Och et al, (1999) [59] first proposed to use a sequence of words (i.e. phrases) as translation units rather than single words. These models are called phrase-based models which have been shown to produce remarkably better translations [52, 45, 63]. Although phrase-based models address the problem of local reordering and idiomatic expressions in word-based models, they are unable to model long-distance reorderings.

- **Hierarchical Phrase-based Models**: These models try to address the problem of complex reordering in phrase-based models by considering the structure of sentences. Chiang (2005) [14] first proposed to use hierarchical phrases in the form of synchronous context-free grammar (SCFG) [48, 1]. This approach become dominant in translation of language pairs with complex reorderings (e.g. Chinese-English).

- **Syntax-based Models**: These models incorporate the linguistic syntax of sentences in translation. Syntax-based models can be divided into two groups: synchronous-grammar-based models and tree-transducer-based models. Many synchronous grammars have been used in machine translation: SCFG [96], synchronous tree-substitution grammars [23], synchronous tree-adjoining grammars (STAG) [78, 19] and generalized multi-text grammars (GMTG) [54]. On the other hand, tree transducers have been used to create several syntax-based SMT models [92, 32, 29, 28].

1.1.1 Log-linear Models

After introducing phrase-based translation systems, the log-linear models became dominant in SMT. Different components like language model, translation model, reordering model, are used as feature functions, ϕ_i in log-linear models with associated weights, w_i [61]:

$$\begin{aligned}\hat{e} &= \arg\max_e p(e|f) \\ &= \arg\max_e \frac{exp(\sum_i w_i\phi_i(e,f)}{\sum_{e'}(\sum_i w_i\phi_i(e',f))} \\ &= \arg\max_e exp(\sum_i w_i\phi_i(e,f))\end{aligned} \tag{1.2}$$

Using a log-linear model we can easily integrate arbitrary feature functions to the model. Associated to each feature function there is a distinct weight that signifies its importance in the model.

1.2 Hierarchical Phrase-based Translation

Hiero models translation using a lexicalized synchronous context-free grammar extracted from word aligned bitexts. The SCFG rules are directly extracted from the phrase-alignments of a bitext by replacing smaller source-target phrases within larger biphrases with a non-terminal X. The extraction places some restrictions to control the size and complexity of the final grammar. The size of a Hiero SCFG grammar is typically larger than phrase-based models extracted from the same data creating challenges in rule extraction and decoding time especially for larger datasets [75].

Using the extracted SCFG rules, the source and target sentences can be generated simultaneously in a derivation process. Hiero typically uses a CKY-style chart-parsing algorithm [17] for decoding. Given a source sentence f, the decoder search for the best scoring derivation obtained by

applying rules in SCFG grammar. The final translation is the target side generated by this derivation. All derivations are scored using a log-linear model.

The decoder parses the source sentence with a modified version of CKY parser. The target side of the derivations are generated simultaneously which are translation candidates for the input source sentence. All derivations are scored using rule parameters, languages model score and some other features (c.f. Section 3.3).

The derivations starts from the leaves corresponding to the smallest spans on the source sentence (all words) and the lowest level cells in the CKY chart and proceeds bottom-up. For each cell in the CKY chart, all applicable rules are identified and like the monolingual parsing the non-terminals in these rules are mapped to some entries in the antecedent cells. The target sides of the rules yield the translation for the source spans (corresponding to the cell) and the translation candidates for the input source sentence are obtained from the top-most cell.

Scoring the target language output using a language model within CKY-style decoding requires two histories per hypothesis, one on the left edge of each span and one on the right, due to the fact that the target side is not generated in left to right order, but rather built bottom-up from sub-spans. This leads to complex problems in efficient language model integration and requires state reduction techniques [36, 37].

1.3 LR-Decoding for Hiero

Decoding in SMT, is the task of finding the best translation for a given source sentence. Since decoding is an online task thus preserving high translation accuracy with low translation time is an essential for decoding algorithms. The decoders use language models to ensure that the output translation is grammatically correct, hence computing the language model score is a crucial part of the process, but the most expensive one as well. Monotonically generating translation, in left-to-right manner aka *LR-decoding*, just requires a single language model (LM) history for each hypothesis which dramatically speeds up the decoding process. There have been many attempts to develop LR-decoders for different translation models: phrase-based [42, 31], syntax-based [40, 24], hierarchical phrase based [90] (we will discuss about LR-decoding for Hiero in Chapter 3).

LR-decoding for Hiero simplifies the decoder complexity and language model scoring by generating the target-side in strict left to right order [90]. The target side of the SCFG rules is constrained to be prefix-lexicalized aka *GNF rules*[1]. However, as we will show this is not as crippling a restriction as it might seem for machine translation if effectively combined with appropriate features that control distortion and reordering.

The LR-decoding algorithm could avoid the shortcomings of Hiero such as faster decoding, reduction in the grammar size and the simplified left-to-right language model scoring. It means LR-decoding has the potential to replace CKY decoding for Hiero. Despite these attractive properties,

[1]Greibach Normal Form (GNF), although the synchronous grammar is not in this normal form, rather only the target side is prefix lexicalized as if it were in GNF form. (c.f. Chapter 2)

we show that the original LR-decoding algorithm for Hiero proposed by Watanabe et al. (2006) [90] does not perform to the same level of the standard CKY-based Hiero with cube pruning (Chapter 3).

1.4 Contributions

In this dissertation we propose a new translation system, called Left-to-Right Hierarchical Phrase-based Translation (*LR-Hiero*). LR-Hiero uses GNF SCFG rules (target side constrained to the form of GNF) to encode translation. We propose new algorithms for different phases of LR-Hiero. The detailed contributions of this work are highlighted as follows[2]:

- **A novel algorithm for rule extraction phase.** Hiero uses a brute-force search algorithm to extract SCFG rules from word-aligned bitexts. Although many constraints are applied to simplify the rule extraction, this phase is still a bottleneck in training process of Hiero. We propose a novel dynamic programming algorithm to extract GNF rules which runs in linear time (Chapter 2).

- **An augmented LR-decoding algorithm.** We propose an augmented LR-decoding algorithm which is demonstrably more efficient than the state-of-the-art CKY Hiero; we find that it is approximately four times faster (Chapter 3). We introduce new features that significantly improve the translation quality compared to the original LR-decoding algorithm by [90].

- **Improving LR-decoding algorithm.** We introduce two more improvements to LR-Hiero decoder. We introduce the notion of *queue diversity* to the cube pruning algorithm to solve potential search errors in LR-Hiero decoder. CKY-based decoders can capture some complex phrasal re-orderings which cannot be captured by LR-Hiero decoder. We extend the LR-Hiero decoder to capture all theses hierarchical phrasal alignments that are reachable in CKY decoders. The modifications significantly improve the translation quality on different language pairs while being much faster (Chapter 4).

- **Lexicalized reordering model for LR-Hiero.** The lexicalized reordering model is an important model in phrase-based translation systems which helps to capture complex reorderings in language pairs like Chinese-English. We introduce a lexicalized reordering model to LR-Hiero which leads to significant improvement on translation quality (Chapter 5).

- **LR-Hiero in simultaneous translation.** We apply LR-Hiero for the task of simultaneous translation (Chapter 6). In simultaneous translation, the translation should be generated incrementally as the user speaks/enters the source input. Previous translation services proposed for real-time translation, are mainly phrase-based as they generate the translation in left-to-right order. This is the first time Hiero type models are used for simultaneous translation. We obtain a very fast simultaneous translation system with high accuracy.

[2]These contributions have been reported in these publications (or submissions): [80, 82, 81, 79].

Chapter 2

Expressive Hierarchical Rule Extraction for Left-to-Right Translation

In this chapter we focus on the rule extraction phase for left-to-right hierarchical phrase based translation (LR-Hiero). Hiero rule extraction induces an exhaustive search for each phrase-pair which results in excessively larger models. We present a novel dynamic programming algorithm for extracting prefix lexicalized target side rules for LR-Hiero.

2.1 Introduction

Hierarchical phrase-based translation (Hiero) [15] uses a lexicalized synchronous context-free grammar (SCFG) extracted from word and phrase alignments of a bitext. Similarly LR-Hiero uses lexicalized SCFG formalism to model translation. To simplify target generation, in LR-Hiero, SCFG rules are constrained to be prefix-lexicalized on target side, aka Greibach Normal Form (GNF). Throughout this thesis we abuse the notation for simplicity and use the term GNF grammars for such SCFGs. This constraint drastically reduces the size of grammar for LR-Hiero compared to Hiero grammar [80]. Although any monolingual context-free grammar can be converted to Greibach Normal Form, there is no algorithm to convert an arbitrary SCFG to a weakly equivalent SCFG with rules constrained to be prefix-lexicalized on the target side.

The usual Hiero rule extraction heuristic applies constraints on the length of initial phrase pairs considered for rule extraction, number and configuration of non-terminals in order to avoid excessively large grammars. Thus, extracted rules cannot capture all possible alignments on language pairs with complex reordering.

In addition, allowing more non-terminals in the rules is not practical in Hiero because the computational complexity of CKY decoders increase exponentially with the increase in the rank of the grammar (that is, the number of non-terminals permitted in the right hand side of the CFG rules). However, the decoder in LR-Hiero can efficiently apply these types of rules while keeping quadratic

time complexity by using a variant of the dotted rules used in the Earley parsing algorithm for parsing monolingual CFGs. We will discuss LR-Hiero decoding algorithm in Chapter 3.

Standard Hiero rule extraction is a brute-force search algorithm which considers all possible replacement of sub-phrases with non-terminals. Despite the constraints on rule configuration, rule extraction is still a bottleneck and it is usually achieved by way of parallelization and optimization. Increasing the length of initial phrase pairs or number of non-terminals exponentially increases the time complexity. In this chapter we propose a dynamic programming algorithm for GNF rule extraction that is linear in the output length (the number of GNF rules). We first explain a heuristic rule extraction [15] method which is used in Hiero (Section 2.2). Then we explain our GNF rule extraction algorithm for LR-Hiero (Section 2.3). This algorithm efficiently extracts SCFG rules for LR-Hiero, therefore it can be used for extracting sentence level rules or rules with arbitrary number of non-terminals.

2.2 Hiero Rule Extraction

2.2.1 Hiero Grammar

Hiero translation system uses SCFG formalism which allows the decoder to generate the source and target synchronously. The decoder successively rewrites the non-terminals in the production rules starting from a top-level rule rooted at S. A grammar G in Hiero is a special form of SCFG which is defined as: $G = (T, N, R, R_g)$ where T and N are the set of terminal and non-terminals in G. Typically there are two types of non-terminals in Hiero grammars: X and S, where S is the start symbol. The set of production rules, R, are defined as:

$$X \rightarrow \langle \gamma, \alpha, \sim \rangle, \gamma, \alpha \in X \cup T^{+} \tag{2.1}$$

where γ and α are strings of terminals and non-terminals in source and target sides respectively and the $\sim$ denotes the alignment of non-terminals in the source and target (co-indexed not-terminal pairs are written synchronously). Unlike typical SCFGs, the rules are lexicalized on the right hand side with at least one aligned word pair in source and target.

The production rules are combined to derive the start symbol S by using the glue rules R_g. Hiero typically uses two glue rules:

$$\begin{aligned} S &\rightarrow \langle X_1, X_1 \rangle \\ S &\rightarrow \langle S_1 X_2, S_1 X_2 \rangle \end{aligned} \tag{2.2}$$

The glue rules allow monotone combination of phrases. They are usually used when no rules could match or the span length is beyond the length of phrases that production rules have been extracted from [1].

2.2.2 Heuristic Rule Extraction

Hiero uses a heuristic approach to extract lexicalized SCFG rules from phrase-pairs. Similar to phrase-based models, training the Hiero models begins from word alignments and the generation of aligned phrase-pairs [15]. Given a word-aligned sentence pair $\langle \bar{f}^J, \bar{e^I}, A \rangle$[2], with alignments A, a source-target sequence pair $\langle \bar{f}, \bar{e} \rangle$ is a phrase pair iff:

$$\begin{aligned} &\forall k, \exists k' \text{ such that } (k, k') \in A, \text{ where } k \in [i, j] \text{ and } k' \in [i', j'] \\ &\nexists k, k' \text{ such that } (k, k') \in A, \text{ where } k \in [i, j] \text{ and } k' \notin [i', j'] \\ &\nexists k, k' \text{ such that } (k, k') \in A, \text{ where } k \notin [i, j] \text{ and } k' \in [i', j'] \end{aligned} \tag{2.3}$$

where $\bar{f}$ and $\bar{e}$ covers spans $[i, j]$ and $[i', j']$ on source and target sentences respectively. Under this definition phrase-pairs might have un-aligned boundary words; we call them *loose phrase-pairs* throughout this thesis. Hiero systems usually restrict the rule extraction to *tight phrase-pairs*, phrase-pairs with aligned boundaries. This constraint controls the number of extracted Hiero rules, which would otherwise be substantially higher.

After extracting the initial phrase-pairs from each sentence-pair, the rule extraction algorithm proceeds as follows. First each initial phrase-pair, $x = \langle \bar{f}, \bar{e} \rangle$ is designated as a rule:

$$X \rightarrow \langle \bar{f}, \bar{e} \rangle \tag{2.4}$$

These rules are also called *terminal rules*. Let $x' = \langle \bar{f}', \bar{e'} \rangle$ be a sub-phrase pair of the rule (2.4) such that $\bar{f} = \bar{f}_p \bar{f}' \bar{f}_s$ and $\bar{e} = \bar{e_p} \bar{e'} \bar{e_s}$. A new rule can be extracted from rule (2.4) by introducing a new non-terminal X in both source and target sides (covering the spans f' and e'):

$$X \rightarrow \langle \bar{f}_p X_1 \bar{f}_s, \bar{e_p} X_1 \bar{e_s} \rangle \tag{2.5}$$

Using the same index for non-terminal X on both source and target sides allows them to be rewritten synchronously[3]. The process of replacing smaller phrase pairs with a new non-terminal continues until no new rule can be extracted.

For example, consider the German-English phrase pair in Figure 2.1. Given this initial phrase-pair, the Hiero rule extraction heuristic would first extract a terminal rule:

[1] It is usually called *maximum phrase-pair length*.

[2] J and I indicate the source and target sentence lengths respectively.

[3] Note that $x' = \langle \bar{f}', \bar{e'} \rangle$ is a valid phrase-pair satisfying the constraints in Equation 2.3.

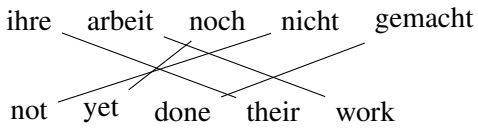


Figure 2.1: A German-English phrase-pair with word alignments.

$$X \rightarrow \langle \text{ihre arbeit noch nicht gemacht, not yet done their work} \rangle \tag{2.6}$$

By substituting a sub-phrase $\langle$ ihre arbeit, their work $\rangle$ with a new non-terminal on source and target sides, a new rule is extracted:

$$X \rightarrow \langle X_1 \text{noch nicht gemacht, not yet done} X_1 \rangle \tag{2.7}$$

Equation 2.8 shows some of the SCFG rules extracted by the Hiero rule extraction heuristic for this phrase pair.

$$\begin{aligned}
&X \rightarrow \langle X_1 \text{arbeit noch nicht gemacht, not yet done} X_1 \text{work} \rangle \\
&X \rightarrow \langle \text{ihre} X_1 \text{noch nicht gemacht, not yet done their} X_1 \rangle \\
&X \rightarrow \langle X_1 \text{noch nicht gemacht, not yet done} X_1 \rangle \\
&X \rightarrow \langle X_1 \text{noch nicht} X_2 \text{, not yet} X_2 X_1 \rangle \\
&X \rightarrow \langle \text{ihre arbeit} X_1 \text{gemacht}, X_1 \text{done their work} \rangle \\
&X \rightarrow \langle \text{ihre arbeit} X_1, X_1 \text{their work} \rangle \\
&X \rightarrow \langle \text{ihre} X_1 \text{noch nicht} X_2 \text{, not yet} X_2 \text{their} X_1 \rangle
\end{aligned} \tag{2.8}$$

Hiero models apply several constraints on the extracted rules in order to limit the grammar size and reduce the time complexity of the CKY decoder. The rules are filtered by removing those violating any of the following constraints:

- Only tight phrase-pairs are allowed to be used as initial phrase-pair.
- *Maximum phrase length* for initial phrase-pairs is 10 (both source and target sides). The extracted rules are limited to at most 5 or 7 terms (terminals and non-terminals) on the source side.
- At most two non-terminals are allowed in a rule.
- No-adjacent non-terminals are allowed in the source side. This avoids *spurious* ambiguities during decoding[4].
- The rules must be lexicalized with at least one aligned source-target word pair.

[4] *Spurious ambiguity*: distinct derivations having the same translation yield with identical feature values.

Hiero rule extraction algorithm is a brute-force search which considers all possible replacement of sub-phrases with non-terminals and then filtered them. Increasing the length of initial phrase pairs or number of non-terminals exponentially increases the time complexity.

2.2.3 Learning Rule Parameters

To learn the rule parameters such as conditional translation probabilities $p(\bar{e}|\bar{f})$ and $p(\bar{f}|\bar{e})$, we need to compute the rule counts first. However each sentence pair in the parallel corpus can be generated through many derivations. Chiang (2007) adapt the heuristics used in phrase-based models to estimate rule distributions in Hiero.

A unit count is considered for each phrase-pair which is distributed equally to all extracted rules from the phrase-pair. The rule counts $c(\bar{f}, \bar{e})$ are then summed up across all phrase-pairs in the parallel corpus (training data). Then the translation probabilities $p(\bar{e}|\bar{f})$ and $p(\bar{f}|\bar{e})$ are computed by relative frequency estimation of the counts.

This heuristic for rule count estimation was first proposed by Bod (1998) for estimating the parameters of the probabilistic tree substitution grammars (PTSG) for parsing. The heuristic was later adapted to many SMT models [45, 68, 28, 15].

2.3 LR-Hiero Rule Extraction

2.3.1 LR-Hiero Grammar

LR-Hiero uses a constrained lexicalized SCFG. The target-side rules are constrained to be prefix lexicalized, for simplicity called *GNF rules*[5]:

$$X \rightarrow \langle \gamma, \bar{b}\, \beta \rangle \tag{2.9}$$

where γ is a string of non-terminal and terminal symbols, $\bar{b}$ is a string of terminal symbols and β is a possibly empty sequence of non-terminals. This ensures that as each rule is used in a derivation, the target string is generated from left to right.

To overcome data sparsity and obtain better generalization, four new rules are generated for each terminal rule $\langle \bar{f}, \bar{e} \rangle$ which we call glue rules for the sake of simplicity. The glue rules allow reordering as well as monotone combination of phrases :

$$X \rightarrow \langle \bar{f} X_1, \bar{e} X_1 \rangle \tag{2.10}$$

$$X \rightarrow \langle X_1 \bar{f}, \bar{e} X_1 \rangle \tag{2.11}$$

$$X \rightarrow \langle X_1 \bar{f} X_2, \bar{e} X_1 X_2 \rangle \tag{2.12}$$

$$X \rightarrow \langle X_1 \bar{f} X_2, \bar{e} X_2 X_1 \rangle \tag{2.13}$$

[5]Greibach Normal Form (GNF). Just the target side is prefix lexicalized (GNF form), not the synchronous grammar.

2.3.2 Initial Phrase Pair Extraction

Hiero limits the length of initial phrase pairs (usually to 10 words), therefore Hiero models typically use phrase-pair extraction phase in phrase-based models. Given a parallel corpus, source-target and target-source word alignments are obtained using an aligner like GIZA++ [63]. Then bidirectional alignments are symmetrized using some heuristic alignment strategy such as union or intersection [62]. Aligned phrase-pairs are then extracted using the alignment template approach [63], such that no words inside the phrase pair are aligned to words outside the phrase pair (the constraints of loose phrase-pairs Equation 2.3).

The time complexity of extracting all phrase pairs of maximum length K for a word-aligned sentence pair of length n is $O(Kn)$. This algorithm is not efficient for extracting long phrase pairs (as long as the length of sentences). Therefore, we use a modified version of the algorithm by Zhang et al. (2008) [94] to extract all phrase pairs.

Uno et al. (2000) [86] propose a $O(n + K)$ time algorithm for computing all K common intervals of two different permutations of length n. This algorithm was later modified [56] to be computed in $O(n)$. Zhang et al. (2008) [94] generalize this to SMT setting and propose a linear time algorithm for phrase-pair extraction. In this algorithm, word aligned sentence pairs are maximally decomposed and encoded as a compact alignment tree. The contiguous blocks of the alignment are captured as the nodes in the alignment tree and the tree structure. For example, phrase pair in Figure 2.2 is shown in the form of decomposed alignment tree in Figure 2.3. The italicized nodes form a left-branching chain in the alignment tree and the sub-spans of this chain also lead to alignment nodes that are not explicitly captured in the tree [94].

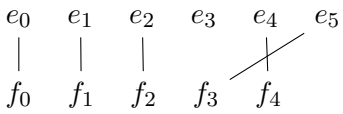


Figure 2.2: Example phrase pair with alignments.

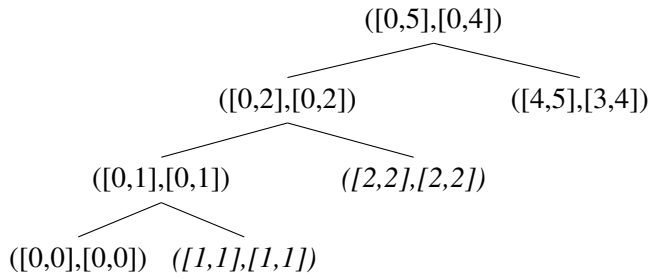


Figure 2.3: Decomposed alignment tree for the example alignment in Fig. 2.2.

2.3.3 GNF Extraction

We first explain the rule extraction algorithm using a working example, then discuss correctness of the algorithm. Let $pp = (\bar{f}, \bar{e})$ be a source-target phrase pair, where $\bar{f}$ and $\bar{e}$ are corresponding phrases on source and target sides. We define *largest right sub-phrase*, for any target interval $[i, j]$, as the largest phrase pair (in terms of length of target side) with right boundary j on the target side, and denote it by $LRS[(i, j)]$:

$$\begin{aligned} &LRS[(i,j)] = argmax_{(\bar{f},\bar{e})\in S(i,j)}|\bar{e}| \\ &S(i,j) = \{(\bar{f},\bar{e})|(\bar{f},\bar{e}) \in \mathcal{P}, |\bar{e}| < |j-i|, \bar{e}.end() = j\} \end{aligned} \tag{2.14}$$

where $\mathcal{P}$ is a set of all phrase pairs, $|\bar{e}|$ denotes length of $\bar{e}$, $\bar{e}.end()$ returns the last index of the span corresponding to $\bar{e}$ (in the target sentence). S is empty for phrase pair with target spans of length one ($i = j$). Note that $LRS[(i, j)] = None$ if the target word on index j is unaligned. In Figure 2.4, the $LRS[(1, 5)]$ is ⟨ noch nicht gemacht, not yet done ⟩[6]. LRS can be precomputed for all span lengths in $O(n^2)$, where n is target sentence length (routine ***RightSubPhrases*** in Algorithm 1). Figure 2.4 (b) shows the chart of LRS computed by ***RightSubPhrases*** for the sentence pair shown in Figure 2.4 (a). Each cell corresponds to a span on the target side.

Algorithm 1 shows the pseudocode for GNF rule extraction. It is a dynamic programming algorithm that extracts GNF rules for phrase pairs (gradually from small to large phrase pairs). It works bottom up and fills a chart, R, on the target sentence. Each cell $R_{i,j}$ keeps all rules that can cover a phrase pair $pp = (\bar{f}, \bar{e})$, where $\bar{e}$ corresponds to span [i, j] on the target sentence[7]. At the end, it returns $\mathcal{R}$ which is the union of all LR-Hiero rules for all target spans (i.e. all possible phrase pairs).

First, routine *ExtractPhrases* extracts all phrase pairs $\mathcal{P}$ and sorts them increasingly based on their target length (line 2). LRS is computed for all target spans by ***RightSubPhrases***. Then, in a *for* loop on all phrase pairs, the chart of the rules will be filled in a bottom up manner, by small to large spans (lines 5-23). For each initial phrase pair a terminal rule is created and added to $R_{i,j}$ (line 8). Then, using rules from smaller phrase pairs, more rules are generated (line 11-21).

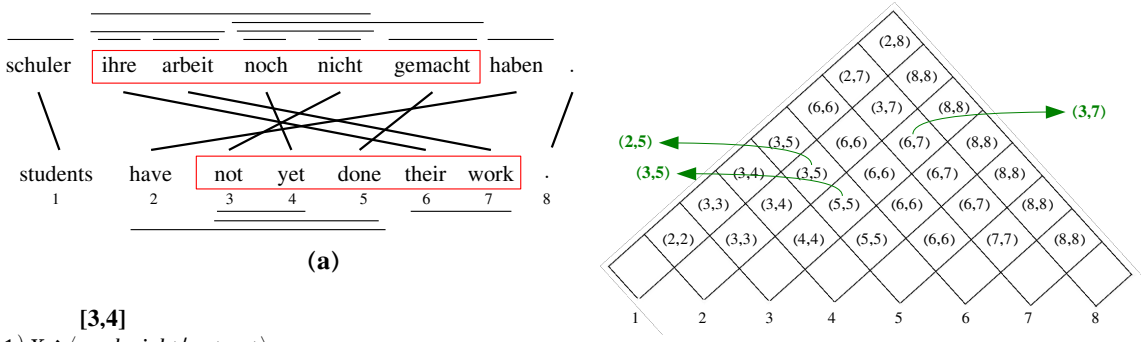
The largest right sub-phrase, pp' is obtained for initial phrase pair pp in line 12 (note that t is set to the right boundary of pp (i.e. j) at the beginning). The target span of pp', $[k, t]$, is used to retrieve rules for pp', stored in $R_{k,t}$. Replacing each rule of pp' in our $curr\_rule$[8], (***Substitute*** routine) results in a new rule for pp (lines 16-18). And as the last rule that can be generated using pp', the whole pp' in $curr\_phr$ is replaced with a non-terminal (line 19).

All rules for pp which includes rules from pp' have been generated, thus we can safely replace pp' with a non-terminal and continue to generate more rules by replacing other parts of pp with

[6] In Figure 2.4, we identify phrase pairs to target spans, $LRS[(1, 5)] = (3, 5)$.

[7] If there is not such a phrase pair, $R_{i,j}$ will be left empty.

[8] It is the initial phrase pair pp at the beginning.



[3,4]

1) $X \rightarrow \langle$ *noch nicht / not yet* $\rangle$
2) $X \rightarrow \langle X_1$ *nicht / not* $X_1 \rangle$

[6,7]

1) $X \rightarrow \langle$ *ihre arbeit / their work* $\rangle$
2) $X \rightarrow \langle$ *ihre* X_1 */ their* $X_1 \rangle$

[3,5]

1) $X \rightarrow \langle$ *noch nicht gemacht / not yet done* $\rangle$
2) $X \rightarrow \langle$ *noch nicht* X_1 */ not yet* $X_1 \rangle$
3) $X \rightarrow \langle X_2$ *nicht* X_1 */ not* $X_2 X_1 \rangle$

[3,7]

i=3, t=7
LRS(3,7)=[6,7]
k=6

curr_rule $\langle$ *ihre arbeit noch nicht gemacht / not yet done their work* $\rangle$

1) $X \rightarrow \langle$ *ihre arbeit noch nicht gemacht / not yet done their work* $\rangle$
2) $X \rightarrow \langle$ *ihre* X_1 *noch nicht gemacht / not yet done their* $X_1 \rangle$
3) $X \rightarrow \langle X_1$ *noch nicht gemacht / not yet done* $X_1 \rangle$

t=5
LRS(3,5)=[3,5]
k=3

curr_rule $\langle X_1$ *noch nicht gemacht / not yet done* $X_1 \rangle$

4) $X \rightarrow \langle X_1$ *noch nicht* X_2 */ not yet* $X_2 X_1 \rangle$
5) $X \rightarrow \langle X_1 X_3$ *nicht* X_2 */ not* $X_3 X_2 X_1 \rangle$

t=2

curr_rule $\langle X_1 X_2 / X_2 X_1 \rangle$

(c)

[2,8]

1) $X \rightarrow \langle$ *ihre arbeit noch nicht gemacht haben . / have not yet done their work .* $\rangle$
2) $X \rightarrow \langle$ *ihre arbeit noch nicht gemacht haben* X_1 */ have not yet done their work* $X_1 \rangle$
3) $X \rightarrow \langle$ *ihre* X_2 *noch nicht gemacht haben* X_1 */ have not yet done their* $X_2 X_1 \rangle$
4) $X \rightarrow \langle X_2$ *noch nicht gemacht haben* X_1 */ have not yet done* $X_2 X_1 \rangle$
5) $X \rightarrow \langle X_2$ *noch nicht* X_3 *haben* X_1 */ have not yet* $X_3 X_2 X_1 \rangle$
6) $X \rightarrow \langle X_2$ *haben* X_1 */ have* $X_2 X_1 \rangle$

$\langle X_2 X_3 X_1 / X_3 X_2 X_1 \rangle$

(d)

Figure 2.4: GNF rule extraction for a German-English sentence pair. (a) bars above (below) the source (target) words indicate phrase-pairs. (b) LRS chart for this sentence, filled by *RightSub-Phrase* (green arrows shows some cells corresponding to phrase pairs which are updated during rule extraction). The span above each set of rules shows the target side of the corresponding phrase pair. (c) Extracting rules for span [3,7]: rule #2 is created using rules of span [6,7], #3 replacing [6,7] with non-terminal, rules #4, #5 created from span [3,5]. Invalid rules are shown in grey. (d) Extracting rules for span [2,8].

non-terminals. $curr\_rule$ is updated to the last rule, t is updated to the index span of pp' on the target (line 21)[9]. The algorithm repeats the loop to find another sub-phrase pair in $curr\_rule$, and

[9] $[i, t]$ always shows the lexical part of the target side.

Algorithm 1 GNF Rule Extraction

Input $\mathbf{f}(f_1 \dots f_n), \mathbf{e}(e_1 \dots e_m), \mathbf{A}$ ($\mathbf{A}$ is alignment)
$\mathcal{P} =$ ExtractPhrases$(\mathbf{f}, \mathbf{e}, \mathbf{A})$ (generate all possible phrase pairs, sorted in increasing order of length of target side)
$LRS =$ RightSubPhrases$(\mathcal{P}, m)$ (precompute largest right sub-phrases)
$\mathcal{R} = \{\}$
for $pp \in \mathcal{P}$ **do**
  $(i, j) = \bar{e}_{pp}$ (target span of pp)
  $R_{i,j} = \{\}$ (rules for target span [i,j])
  $curr\_rule = pp$ (create a terminal rule)
  AddRule($R_{i,j}$, $curr\_rule$)
  $t = j$
  while $t \geq i$ **do**
    $pp' = LRS[(i, t)]$
    if pp' is None **then**
      break
    $(k, t) = \bar{e}_{pp'}$ (target span of pp')
    for each $r \in R_{k,t}$ **do**
      $r' =$ Substitute($curr\_rule$, pp', r)
      AddRule($R_{i,j}$, r')
    $curr\_rule =$ Substitute($curr\_rule$, pp', X) (replace subphrase with a non-terminal)
    AddRule($R_{i,j}$, $curr\_rule$)
    $t = k - 1$
  $LRS[(i, j)] = pp$ (update LRS)
  Add $R_{i,j}$ to $\mathcal{R}$
return $\mathcal{R}$

RightSubPhrases$(\mathcal{P}, m)$
  $LRS = \{\}$
  for $l = 2, \dots, m$ **do**
    for $i = 1, \dots, m - l$ **do**
      $j = i + l - 1$
      if $\exists\, pp \in \mathcal{P}, \bar{e}_{pp} == (i + 1, j)$ **then**
        $LRS[(i, j)]$= pp
      elif $(i + 1, j) \in LRS$ **then**
        $LRS[(i, j)]$= $LRS[(i + 1, j)]$
  return LRS

continues until no sub-phrase pair is found (or we reach the beginning of the target phrase (t equals i)). When all the rules for pp are computed, $LRS[(i, j)]$ is updated to pp so that it can be used in larger phrases. In fact $LRS[(i, j)]$ should always show the largest right subphrase whose rules have already been extracted. Note that only if $[i, j]$ corresponds to a phrase-pair, $LRS[(i, j)]$ is updated (in Figure 2.4(b), some updated cells are shown in green).

Figure 2.4(c) shows how algorithm extracts rules for span [3,7]: at first $curr\_rule$ is equal to the initial phrase pair. Rule #1 is a terminal rule; $LRS[3, 7]$ is $\langle ihre\ arbeit,\ their\ work \rangle$

(target span [6,7]), rule #2 is generated using rules for [6,7]; rule #3 is the result of replacing [6,7] with a non-terminal. Then $curr\_rule$ and t are updated. $LRS[3,5]$ is sub-phrase pair $\langle noch\ nicht\ gemacht,\ not\ yet\ done\rangle$, (rules for this phrase pair have been already computed and consequently $LRS[3,5]$ has been updated to ([4,6],[3,5])). Rules #4 and #5 are generated using rules for span [3,5]. Then $curr\_rule$ and t are updated. As no lexical item remains in the target side of $curr\_rule$, the algorithm stops.

AddRule verifies the rule configuration like the number of non-terminals and non-adjacent non-terminals on the source side. If the rule is valid, it is added to the corresponding cell, $R_{i,j}$ (e.g. rule #5, for span [3,7], is not valid because of adjacent non-terminals on the source side).

2.3.4 Correctness

We show that for a given phrase pair, this algorithm extracts all possible Hiero style SCFG rules which are in GNF format on the target side (the same as Hiero brute-force rule extraction). Given a phrase pair $pp = (\bar{f}, \bar{e})$ with target span $[i,j]$, $LRS([i,j])$ shows the largest subproblem that can be optimally used to generate rules for pp, denoted by $R_{i,j}$.
Optimal structure: $R_{i,j}$ consists of two disjoint sets

$$\begin{aligned} R_s &= \{r | r = Substitute(pp, pp', r') \forall r' \in R_{i',j}\} \\ R_x &= \{r | r.pos(pp') = X\} \end{aligned} \tag{2.15}$$

where $r \in R_{i,j}$ is a rule, $pp' = LRS([i,j])$ is a sub-phrase of pp and its target span is $[i,j]$. $r.pos(pp')$ denotes the interval of pp' in r. R_s is the set of rules obtained by replacing pp' in pp with each rule of $R_{i',j}$, while R_x is the set of rules having non-terminal X in position of pp' (in source and target side). GNF rules on the target side (Equation 2.9), end with a non-terminal (if there is one) and there is no lexical item between non-terminals. Assuming this we can consider two states for each $r \in R_{i,j}$: (a) r has some lexical term in $r.pos(pp')$; (b) r has a non-terminal in the location of pp'. Case (a) is equal to set R_s: this type of rules can have non-terminal just in the interval of pp' (because any non-terminal out of pp' violates GNF format on the target side). And if there is a rule of this type it corresponds to a rule in $R_{i',j}$. Consequently case (b) is equal to set R_x. It means that any $r \notin R_s$, should replace a non-terminal instead of pp' (otherwise violates GNF format on the target side). Computing R_x corresponds to a smaller problem $[i, i'-1]$ (let's define $t = i'-1$) which can be solved in a similar way. If $[i,t]$ is target side of a phrase pair (like $[3,5]$ in Figure 2.4(c)), we just need to use rules in $R_{i,t}$ to generate more rules and keep all valid rules. Otherwise we repeat the process: find the largest sub-problem, $LRS[i,t] = (k,t)$, use $R_{k,t}$ to generate more rules, then replace $[k,t]$ in the rule with a non-terminal, update the rule and continue. It stops when target side is entirely covered by non-terminals (Figure 2.4(d) shows an example of this type).

	Corpus	Train/Dev/Test
Cs-En	Europarl(v7) + CzEng(v0.9); News commentary(nc) 2008&2009; nc 2011	7.95M/3000/3003
De-En	Europarl(v7); WMT2006; WMT2006	1.5M/2000/2000
Zh-En	HK parallel-tex + GALE ph-1; MTC parts 1&3; MTC part4	2.3M/1928/919

Table 2.1: Corpus statistics in number of sentences. Tuning and test sets for Chinese-English has 4 references.

Using this optimal structure, we iteratively solve the problem in three steps: (1) find the largest sub-problem ($LRS(i, j)$), (2) use its solution to generate some rules (R_s), (3) reduce the problem to a smaller problem (R_x).

Unaligned words in the target language (not present in our example) make the computation of LRS more complex. For example if target word index j is unaligned, then $LRS[i, j]$ for all $i < j$ will be empty and the algorithm stops without considering subphrases at the left side of the unaligned word. To avoid this problem, unaligned words on the target side will be attached to the closest left phrase pair (if it exists) during computation of LRS.

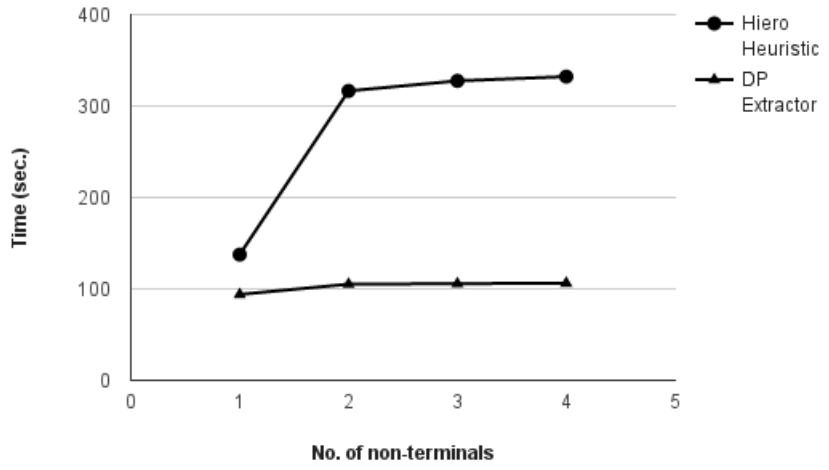
2.4 Experiments

To evaluate our rule extraction algorithm, we use it to extract the grammar for LR-Hiero on three language pairs: German-English (De-En), Czech-English (Cs-En) and Chinese-English (Zh-En). Table 2.1 shows the details of datasets we use.

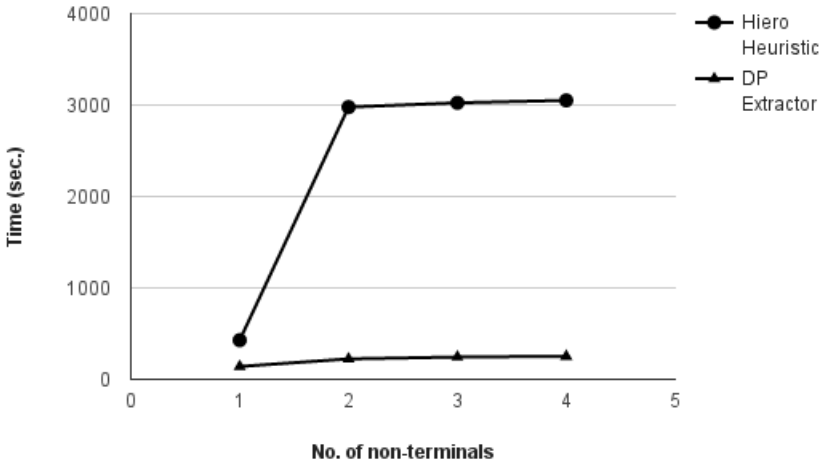
As baseline we use Kriya [75], an open-source implementation of Hiero in Python (available on `https://github.com/sfu-natlang/Kriya`) which performs comparably to other open-source Hiero systems. We use rule extraction of Kriya to extract SCFG rules (Hiero) and modify it to extract GNF rules (LR-Hiero) by applying more constraints on during search. Our own implementation for GNF rule extraction described in Section 2.3 is also in Python which allows use to make a fair comparison on speed.

To evaluate the performance of our dynamic programming (DP) rule extraction algorithm, we compare it to the Hiero rule extraction algorithm that uses brute-force search. Figure 2.5 shows the extraction time of different rule extraction algorithms for extracting grammars with various configurations and settings. We use 10000 sentence pairs randomly selected from German-English parallel data. Figure 2.5(a) and (b) illustrates the effect of the rule arity (number of non-terminals) in the case of using initial phrase-pair of length at most 10 and the full sentence length, respectively. These diagrams show that our GNF rule extraction algorithm works much faster than the Hiero rule extraction algorithm. We can see that increasing the number of non-terminals drastically increases

the extraction time in Hiero rule extraction algorithm, while it slightly increases running time in the DP extraction algorithm. Due to the constraints applied on the rule format, the number of rules having more than 2 non-terminals is very limited, therefore extracting these rules slightly affects the running time.



(a)



(b)

Figure 2.5: Comparing the effect of number of non-terminals and length of initial phrase pairs on the speed of rule extraction algorithms. Rule extraction time in terms of seconds (German-English task) for initial phrase-pairs of length at most 10 (a) and full sentence length (b). Note that these to algorithms are used to extract the same set of GNF rules.

We use our rule extraction algorithm to extract GNF rules from all initial phrase pairs (any length), rule arity 1 to 4, maximum source rule length 10. Like Hiero, we filter out rules with adjacent non-terminals on the source side. Terminal rules are constrained to maximum source rule

Model (msl)	Cs-En	De-En	Zh-En
SCFG (7)	1,961.6	858.5	471.8
GNF (7)	306.3	116.0	100.9
GNF-4 (10)	380.9	214.9	190.0

Table 2.2: Model sizes in millions of rules. Maximum source length (msl) is shown in brakets.

length 7. We use a rule count estimation heuristic similar to Hiero. Table 2.2 shows model sizes for LR-Hiero (GNF), Hiero (SCFG) and GNF grammar with at most 4 non-terminals (GNF-4). Typical Hiero rule extraction excludes phrase-pairs with unaligned words on boundaries (loose phrases). We include loose phrase-pairs as terminal rules in all GNF grammars.

In Chapter 4 we use all grammars shown in Table 2.2 for decoding in both Hiero and LR-Hiero translation system. Please see Section 4.4.2 for the results and discussion on the effect of different grammars on the performance of the translation systems.

2.5 Related Work

Many approaches have been developed to improve SCFG rules for Hiero. Some of the works have employed generative methods using Bayesian techniques to induce SCFG [6, 5, 47, 74] directly from bilingual data without word alignments. de Gispert et al. (2010) [18] extract rules based on posterior distributions provided by the HMM word-to-word alignment model, rather than a single alignment which is used in original Hiero. Most of these approaches restrict the grammar to rules with one or at most two non-terminals to be able to use the grammar in decoding [6, 18, 74].

Recently Levenberg et al. (2012) [47] propose an approach to learn grammars with an unrestricted number of non-terminals but do not use the grammar directly in the decoder. The obtained SCFG rules are used to obtain the word alignments rather than the SCFG rules for decoding. An unrestricted number of non-terminals makes the induced grammar unusable in CKY based decoders.

Zhang et al. (2008) [94] encode the word aligned sentence pairs as a normalized decomposition tree (a hierarchical representation of all the phrase pairs in linear time) which yields a set of minimal Hiero (SCFG) rules. They discuss that the method can be modified to extract all Hiero rules. But the algorithm is just applied as an analytical tool for aligned bilingual data.

Syntax-based translation systems, tree-to-tree [21], tree-to-string [50, 38] and string-to-tree [28], extract sentence level rules, but they extract rules from parse trees (on source or target) rather than word aligned sentence pairs which we discussed in this chapter.

Braune et al. (2012) [7] extend Hiero by extracting an additional and separate set of rules for long-distance reorderings. They modify Hiero extractor based on some analysis on long-distance German-to-English movement and filter them based on linguistic information. New rules are applied to long spans (11 to 50) but do not improve translation quality in terms of BLEU (in some case BLEU scores get reduced by 0.4). However they show that their approach helps in terms of

improving the reordering between source and target (using LRscore [4] evaluation scores and some manual evaluation).

2.6 Summary and Conclusion

We propose a dynamic programming algorithm for GNF rule extraction that is linear in the number of GNF rules. We use the sentence level GNF rules with different number of non-terminals in the LR-decoder and analyze the effect of these rules in LR-Hiero translation system on different language pairs. New rules with more non-terminals improve the alignment coverage (24% on average) on language pairs with more complex reordering, while it marginally affects the decoding speed. Using rules with more non-terminals is a promising approach in Hiero translation systems which is practical using LR-decoding.

Chapter 3

Left-to-Right Decoding for Hierarchical Phrase-based Translation

Watanabe et al. (2006) [90] propose a promising decoding algorithm for hierarchical phrase-based translation (Hiero). It generates the target sentence by extending the hypotheses only on the right edge which is called left-to-right (LR) decoding. LR-decoding has complexity $O(n^2b)$ (in practice) to translate an input sentence of n words and beam size b, compared to $O(n^3b)$ for the CKY algorithm. It requires a single language model (LM) history for each target hypothesis rather than two LM histories per hypothesis as in CKY. In this chapter we present an augmented LR decoding algorithm that builds on the LR-decoding algorithm by Watanabe et al. (2006) [90]. Unlike that algorithm, using experiments over multiple language pairs we show two new results: our LR decoding algorithm provides demonstrably more efficient decoding than CKY-based Hiero decoder, four times faster; and by introducing new distortion and reordering features for LR decoding, it maintains the same translation quality (as in BLEU scores) obtained by phrase-based and CKY Hiero with the same translation model.

3.1 Introduction

Hiero [15] models translation using a lexicalized synchronous context-free grammar (SCFG) extracted from word aligned bitexts. Typically, CKY-style decoding is used for Hiero with time complexity $O(n^3)$ for source input with n words. Scoring the target language output using a language model within CKY-style decoding requires two histories per hypothesis, one on the left edge of each span and one on the right, due to the fact that the target side is not generated in left to right order, but rather built bottom-up from sub-spans. This leads to complex problems in efficient language model integration and requires state reduction techniques [36, 37]. The size of a Hiero SCFG grammar is typically larger than phrase-based models extracted from the same data creating challenges in rule extraction and decoding time especially for larger datasets [75].

In contrast, the LR-decoding algorithm could avoid these shortcomings such as faster time complexity, reduction in the grammar size and simplified left-to-right language model scoring.

Despite these attractive properties, we show that the LR-decoding algorithm proposed by Watanabe et al. (2006) [90] does not perform to the same level of the standard CKY Hiero with cube pruning (see Table 3.3). In addition, the current LR decoding algorithm does not obtain BLEU scores comparable to phrase-based or CKY-based Hiero models for different language pairs (see Table 3.4). In this chapter we propose modifications to the LR decoding algorithm that addresses these limitations and provides, for the first time, a true alternative to the standard Hiero that uses CKY-based decoder.

We introduce a new extended version of the LR-decoding algorithm presented in [90] which is demonstrably more efficient than the CKY Hiero algorithm. We measure the efficiency of the LR-decoder in a way that is independent of the choice of system and programming language by measuring the number of language model queries. Although more efficient, the new LR-decoding algorithm suffered from lower BLEU scores compared to Hiero CKY decoder. Our analysis of left to right decoding showed that it has more potential for search errors due to early pruning of good hypotheses. This is unlike bottom-up decoding (CKY) which keeps best hypotheses for each span. To address this issue, we introduce two novel features into LR-Hiero that deal with reordering and distortion. Our experiments show that LR-decoding with these features using prefix lexicalized target side rules (GNF) equals the scores obtained by CKY decoding with the same set of rules and phrase-based translation system. It performs four times fewer language model queries on average, compare to CKY Hiero decoding with Hiero SCFG rules: 6466.7 LM queries for CKY Hiero (with cube pruning) compared to 1500.45 LM queries in LR Hiero (with cube pruning), while translation quality suffers by only about 0.67 in BLEU score on average, across two different language pairs.

3.2 Left-to-Right Decoding for Hiero

Hierarchical phrase-based SMT [14, 15] uses a synchronous context free grammar (SCFG), where the rules are of the form $X \rightarrow \langle \gamma, \alpha \rangle$, where X is a non-terminal, γ and α are strings of terminals and non-terminals.

Chiang (2007) [15] places certain constraints on the extracted rules in order to simplify decoding (c.f./ Section 2.2 for the details of rule extraction algorithms).

3.2.1 GNF Rules

Watanabe et al. (2006) [90] propose a left-to-right decoding algorithm that generates the target hypotheses left to right, but for synchronous context-free grammar (SCFG) as used in Hiero. The target-side rules are constrained to be prefix lexicalized (c.f. Section 2.3 for GNF rule extraction algorithms).

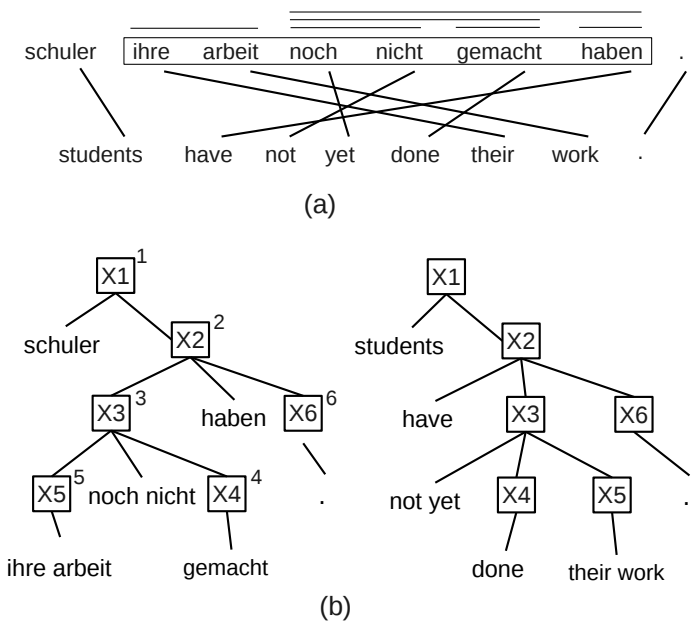


Figure 3.1: (a): A word-aligned German-English sentence pair. The bars above the source words indicate phrase-pairs having at least two words. (b): its corresponding left-to-right target derivation tree. Superscripts on the source non-terminals show the indices of the rules (see Figure 3.2) used in derivation.

Figure 3.1(a) shows a word-aligned German-English sentence with a phrase pair $\langle$*ihre arbeit noch nicht gemacht haben* , *have not yet done their work*$\rangle$ that will lead to a SCFG rule. Given other smaller phrases (marked by bars above the source side), we extract a GNF rule[1]:

$$X \rightarrow \langle X_1 \text{ noch nicht } X_2 \text{ haben}, \text{have not yet } X_2\ X_1 \rangle \tag{3.1}$$

It might appear that the restriction that target-side rules be GNF is a severe restriction on the coverage of possible hypotheses compared to the full set of rules permitted by the Hiero extraction heuristic. However there is some evidence in the literature that discontinuous spans on the source side in translation rules is a lot more useful than discontinuous spans in the target side (which is disallowed in the GNF). For instance, Galley et al. [31] do an extensive study of discontinuous spans on source and target side and show that source side discontinuous spans are very useful but removing discontinuous spans on the target side only lowers the BLEU score by 0.2 points (using the Joshua SMT system on Chinese-English). Removing discontinuous spans means that the target side rules have the form: $uX, Xu, XuX', XX'u$, or uXX' of which we disallow $Xu, XuX', XX'u$.

Zhang et al. (2012) [95] also conduct a study on discontinuous spans on source and target side of Hiero rules and conclude that source discontinuous spans are always more useful than discontinuities on the target side with experiments on four language pairs (Chinese-English, French-English, German-English and Spanish-English). As we shall also see in our experimental results (see Ta-

[1] LR-Hiero rule extraction excludes non-GNF rules such as $X \rightarrow \langle X_1 \text{ noch nicht gemacht } X_2,\ X_2 \text{ not yet done } X_1 \rangle$.

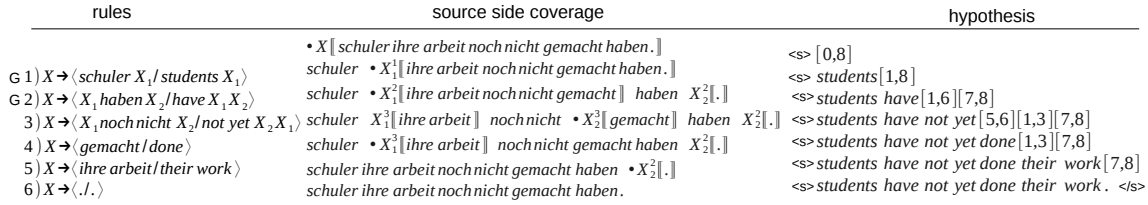


Figure 3.2: Illustration of the LR-decoding process in Figure 3.1. (a) Rules pane shows the rules used in the derivation (glue rules are marked by G) (b) Decoder state using Earley dot notation (superscripts show rule#) (c) Hypotheses pane showing translation prefix and ordered list of yet-to-be-covered spans.

ble 3.4) we can get close to the BLEU scores obtained using the full set of Hiero rules by using only target lexicalized rules in our LR-decoder.

3.2.2 LR-Decoding

LR-decoding uses a top-down depth-first search, which strictly grows the hypotheses in target surface ordering. Search on the source side follows an Earley-style search [22], the dot jumps around on the source side of the rules based on the order of non-terminals on the target side. This search is integrated with beam search or cube pruning to efficiently find the k-best translations.

Several important details about the algorithm of LR-decoding are implicit and unexplained in [90]. In this section we describe the LR-decoding algorithm in more detail than the original description in [90]. We explain our own modified algorithm for LR-Hiero decoder with cube pruning in Section 3.2.3.

Algorithm 2 shows the pseudocode for LR decoding. Decoding the example in Figure 3.1(b) is explained using a walk-through shown in Figure 3.2. Each partial hypothesis h is a 4-tuple (h_t, h_s, h_{cov}, h_c): consisting of a translation prefix h_t, a (LIFO-ordered) *list* h_s of uncovered spans, source words coverage set h_{cov} and the hypothesis cost h_c. The initial hypothesis is a null string with just a sentence-initial marker $\langle s \rangle$ and the list h_s containing a span of the whole sentence, $[0, n]$. The hypotheses are stored in stacks $S_0, \ldots, S_n$, where each stack corresponds to a coverage vector of same size, covering same number of source words [45].

At the beginning of beam search the initial hypothesis h_0 is added to the decoder stack S_0 (line 6 in Algorithm 2). Hypotheses in each decoder stack are expanded iteratively, generating new hypotheses, which are added to the latter stacks corresponding to the number of source words covered. In each step it pops from the LIFO list h_s, the span $[u, v]$ of the next hypothesis h to be processed.

All rules that match the entire span $[u, v]$ are then obtained efficiently via pattern matching [51]. *GetSpanRules* addresses possible ambiguities in matched rules to the given span $[u, v]$. For example, given a rule r, with source side $r_s : \langle X_1 \text{ the } X_2 \rangle$ and source phrase $p : \langle$*ok, the more the better*$\rangle$. There is ambiguity in matching r to p. *GetSpanRules* returns a distinct matched rule for each possible matching.

Algorithm 2 LR-Decoding

1: Input sentence: $\mathbf{f} = f_0 f_1 \dots f_n$
2: $\mathcal{F} = \text{FutureCost}(\mathbf{f})$ (Precompute future cost for spans)
3: **for** $i = 0, \dots, n$ **do**
4: $S_i = \{\}$ (Create empty stacks)
5: $h_0 = (\langle s \rangle, [[0, n]], \emptyset, \mathcal{F}_{[0,n]})$ (Initial hypothesis 4-tuple)
6: Add h_0 to S_0 (Push initial hyp into first Stack)
7: **for** $i = 0, \dots, n-1$ **do**
8: **for** each h in S_i **do**
9: $[u, v]$ = pop(h_s) (Pop first uncovered span from list)
10: R = GetSpanRules($[u, v]$) (Extract rules matching the entire span $[u, v]$)
11: **for** $r \in R$ **do**
12: $h' = \text{GrowHypothesis}(h, r, [u, v], \mathcal{F})$ (New hypothesis)
13: Add h' to S_l, where $l = |h'_{cov}|$ (Add new hyp to stack)
14: **return** $\arg\max(S_n)$

15: **GrowHypothesis**$(h, r, [u, v], \mathcal{F})$
16: $h' = (h'_t = \emptyset, h'_s = h_s, h'_{cov} = \emptyset, h'_c = 0)$
17: $r_X = \{X_j, X_k, \dots | j \lhd k \lhd \dots\}$ (Get non-terminals in surface order)
18: **for** each X in reverse(r_X) **do**
19: push($h'_s, span(X)$) (Push uncovered spans to LIFO list)
20: $h'_t = \text{Concatenate}(h_t, r_t)$
21: $h'_{cov} = \text{UpdateCoverage}(h_{cov}, r_s)$
22: $h'_c = \text{ComputeCost}(g(h'), \mathcal{F}_{\neg h'_{cov}})$
23: **return** h'

The *GrowHypothesis* routine creates a new candidate by expanding given hypothesis h using rule r and computes the complete hypothesis score including language model score. Since the target-side rules are in GNF, the translation prefix of the new hypothesis is obtained by simply concatenating the terminal prefixes of h and r in same order (line 20). *UpdateCoverage* updates source word coverage set using the source side of r. The h_s list is built by pushing the non-terminal spans of rule r in a reverse order (lines 17 and 18). The reverse ordering maintains the left-to-right generation of the target side.

In the walk-through in Figure 3.2, the derivation process starts by expanding the initial hypothesis h_0 (first item in the right pane of Figure 3.2) with the rule (rule #1 in left pane) to generate a new partial candidate having a terminal prefix of $\langle s \rangle$ *students* (second item in right pane). The second item in the middle pane shows the current position of the parser employing Earley's dot notation, indicating that the first word has already been translated. Now the decoder considers the second hypothesis and pops the span $[1, 8]$. It then matches the rule (#2) and pushes the spans $[1, 6]$ and $[7, 8]$ into the list h_s in the reverse order of their appearance in the target-side rule. At each step the new hypothesis is added to the decoder stack S_l depending on the number of covered words in the new hypothesis (line 13 in Algorithm 2).

Algorithm 3 LR-Hiero Decoding with Cube Pruning

1: Input sentence: $\mathbf{f} = f_0 f_1 \dots f_n$
2: $\mathcal{F}$ = FutureCost($\mathbf{f}$) (Precompute future cost for spans)
3: $S_0 = \{\}$ (Create empty initial stack)
4: $h_0 = (\langle s \rangle, [[0, n]], \emptyset, \mathcal{F}_{[0,n]})$ (Initial hypothesis 4-tuple)
5: Add h_0 to S_0 (Push initial hyp into first Stack)
6: **for** $i = 1, \dots, n$ **do**
7: $cubeList = \{\}$ (MRL is max rule length)
8: **for** $p = max(i - \text{MRL}, 0), \dots, i - 1$ **do**
9: $\{G\}$ = Grouped(S_p) (Group based on the first uncovered span)
10: **for** $g \in \{G\}$ **do**
11: $[u, v] = g_{span}$
12: R = GetSpanRules($[u, v]$)
13: **for** $R_s \in R$ **do**
14: $cube = [g_{hyps}, R_s]$
15: Add $cube$ to $cubeList$
16: S_i = Merge($cubeList, \mathcal{F}$) (Create stack S_i and add new hypotheses to it, see Figure 3.3)
17: **return** $\arg\max(S_n)$

18: **Merge**($CubeList, \mathcal{F}$)
19: $heapQ = \{\}$
20: **for** each (H, R) in $cubeList$ **do**
21: $[u, v]$ = span of rule R
22: h' = GrowHypothesis($h_1, r_1, [u, v], \mathcal{F}$) (from Algorithm 2)
23: push($heapQ, (h'_c, h', [H, R])$)
24: $hypList = \{\}$
25: **while** $|heapQ| > 0$ and $|hypList| < K$ **do**
26: $(h'_c, h', [H, R])$ = pop($heapQ$)
27: push($heapQ, GetNeighbours([H, R])$)
28: Add h' to $hypList$
29: **return** $hypList$

For pruning we use an estimate of the future cost[2] of the spans uncovered by current hypothesis together with the hypothesis cost. The future cost is precomputed (line 2 Algorithm 2) in a way similar to the phrase-based models [44] using only the terminal rules of the grammar. The ComputeCost method (line 22 in Algorithm 2) uses the usual log-linear model and scores a hypothesis based on its different feature scores $g(h')$ and the future cost of the *yet to be covered* spans ($\mathcal{F}_{\neg h'_{cov}}$). Time complexity of left to right Hiero decoding with beam search is $O(n^2 b)$ in practice where n is the length of source sentence and b is the size of beam [40].

3.2.3 LR-Hiero Decoding with Cube Pruning

[2]Watanabe et al. [90] also use a similar future cost, even though it is not discussed in the paper (p.c.).

The Algorithm 2 presented earlier does an exhaustive search as it generates all possible partial translations for a given stack that are reachable from the hypotheses in previous stacks. However only a few of these hypotheses are retained, while the majority of them are pruned away. The cube pruning technique [15] avoids the wasteful generation of poor hypotheses that are likely to be pruned away by efficiently restricting the generation to only high scoring partial translations.

We modify the cube pruning for LR-decoding that takes into account the next uncovered span to be translated indicated by the Earley's dot notation. The Algorithm 3 shows the pseudocode for LR-decoding using cube pruning. The structure of stacks and hypotheses and computing the future cost is similar to Algorithm 2 (lines 1-5). To fill stack S_i, it iterates over previous stacks (line 8 in Algorithm 3) [3].

All hypotheses in each stack S_p (covering p words on the source-side) are first partitioned into a set of groups, $\{G\}$, based on their first uncovered span (line 9) [4].

Each group g is a 2-tuple (g_{span}, g_{hyps}), where g_{hyps} is a list of hypotheses which share the same first uncovered span g_{span}. Rules matching the span g_{span} are obtained from routine *GetSpanRules*, which are then grouped based on unique source side rules (i.e. each R_s contains rules that share the same source side s but have different target sides). Each g_{hyps} and possible R_s[5] create a cube which is added to $cubeList$.

In the LR-Hiero decoder, each hypothesis is developed with only one uncovered span, therefore each cube always has just two dimensions: (1) hypotheses with the same number of covered words and similar first uncovered span, (2) rules sharing the same source side.

In Figure 3.3(a), each group of hypotheses, g_{hyps}, is shown in a green box (in stacks), and each rectangle on the top is a cube. Figure 3.3 is using the example in Figure 3.2.

The *Merge* routine is the core function of cube pruning which generates the best hypotheses from all cubes [15]. For each possible cube, (H, R), the best hypothesis is generated by calling $GrowHypothesis(h_1, r_1, \textit{span}, \mathcal{F})$ where h_1 and r_1 are the best hypothesis and rule in H and R respectively (line 21). Figure 3.3 (b) shows a more detailed view of a cube (shaded cube in Figure 3.3(a)). Rows are hypotheses and columns are rules which are sorted based on their scores.

The first best hypotheses, h', along with corresponding score, h'_c and corresponding cube, (H, R) are placed in a priority queue, $heapQ$ (triangle in Figure 3.3). Iteratively the best hypothesis is popped from the queue (line 26) and its neighbours in the cube are added to the priority queue (using $GetNeighbours([H, Q])$). It continues to generate all K best hypotheses. Using the cube pruning technique, each stack is filled with K best hypotheses without generating all possible hypotheses in each cube.

[3] As the length of rules are limited (at most MRL), we can ignore stacks with index less than $i - \text{MRL}$

[4] The beam search decoder in Phrase-based system [39, 44, 73] groups the hypotheses in a given stack based on their coverage vector. But this idea does not work in LRHiero decoding in which the expansion of each hypothesis is restricted to its first uncovered span. We have also tried another way of grouping hypotheses: group by all uncovered spans, h_s. Our experiments did not show any significant difference between the final results (BLEU score), therefore we decided to stick to the simpler idea: using first uncovered span for grouping.

[5] Note that, just rules whose number of terminals in their source side is equal to $i - p$ can be used.

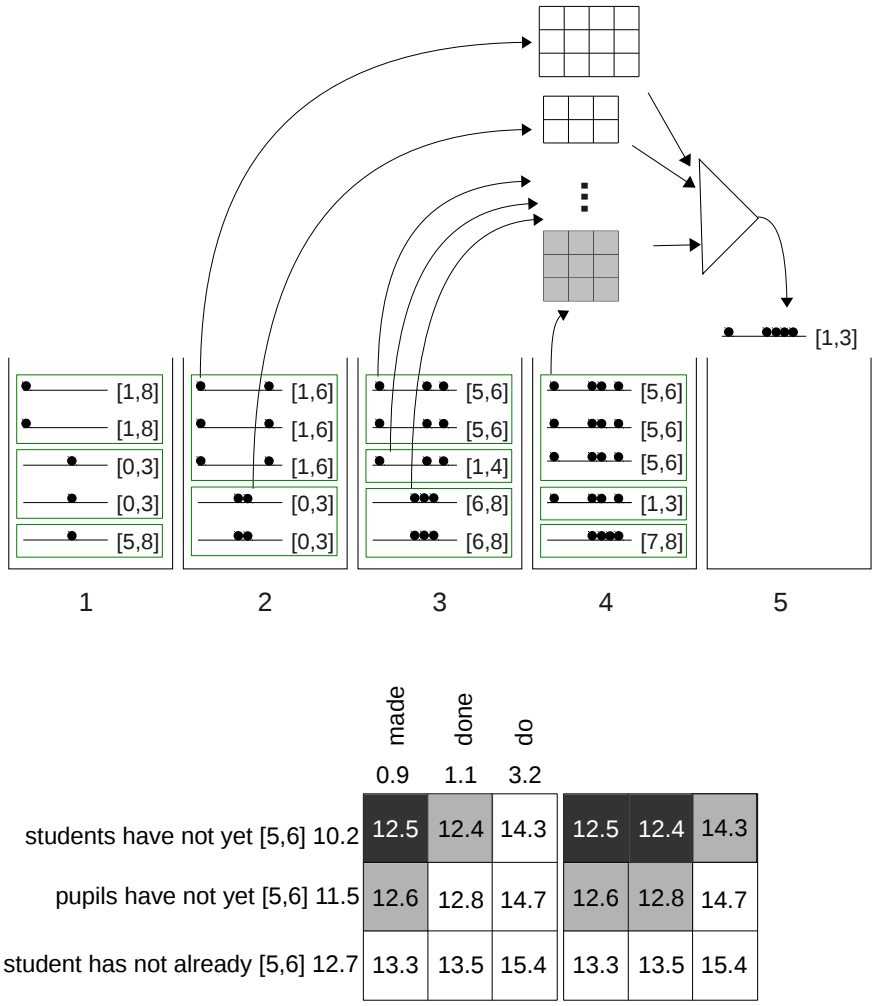


Figure 3.3: Example of generating hypotheses in cube pruning using Figure 3.2: (a) Hypotheses in previous stacks are grouped based on their first uncovered span, and build cubes (grids on top). Cubes are in different sizes because of different number of rules and group sizes. Cubes are fed to a priority queue (triangle) and new hypotheses are iteratively popped from the queue and added to the current stack, S_5. (b) Generating hypotheses from a cube. The top side of the grid denotes the target side of rules sharing the same source side (R_s) along with their scores. Left side of the grid shows the hypotheses in a same group, their first uncovered span and their scores. Hypothesis generated from row 1 and column 1 is added to the queue at first. Once it is popped from the queue, its neighbours (in the grid) are subsequently added to the queue.

Figure 3.3 (b) shows the derivation of the two best hypotheses from the cube. The best hypothesis of this cube which is likely created from the best hypothesis and rule (left top most entry) is popped at first step. Then, *GetNeighbours* calls *GrowHypothesis* to generate the next potential best hypotheses of this cube (neighbours of the popped entry which are shaded in Figure 3.3(b)). These hypotheses are added to the priority queue. In the next iteration, the best hypothesis is popped from all candidates in the queue and algorithm continues.

3.3 Features

We use the following standard SMT features for the log-linear model of LR-Hiero: relative-frequency translation probabilities $p(f|e)$ and $p(e|f)$, lexical translation probabilities $p_l(f|e)$ and $p_l(e|f)$, a language model probability, word count and phrase count. In addition we also use the glue rule count and the two reordering penalty features employed by Watanabe et al. [90, 89]. These features compute the *height* and *width* (span size of the entire subtree) of all subtrees which are *backtraced* in the derivation of a hypothesis. A non-terminal X_i is pushed into the LIFO list of a partial hypothesis; it's *backtrace* refers to the set of non-terminals that must be popped before X_i.

In Figure 3.1(b), X_2 has two subtrees X_3 and X_6, where X_3 should be processed before X_6. The subtree rooted at X_3 in Figure 3.1(b) has a height of 2 and span $[1, 6]$ having a width of 5. Similarly, X_4 should be backtraced before X_5 and has height and width of 1. Backtracing applies only for rules having at least two non-terminals. Thus the total height and width penalty for this derivation are 3 and 6 respectively.

However, the height and width features do not distinguish between a rule that reorders the non-terminals in source and target from one that preserves the ordering. Rules #2 and #3 in Figure 3.2 are treated equally although they have different orderings. The decoder is thus agnostic to this difference and would not be able to exploit this effectively to control reordering and instead would rely on the partial LM score. This issue is exacerbated for glue rules, where the decoder has to choose from different possibilities without any way to favour one over the others. Instead of the rule #2, the decoder could use its reordered version $\langle X_1\ haben\ X_2, have\ X_2\ X_1 \rangle$ leading to a poor translation.

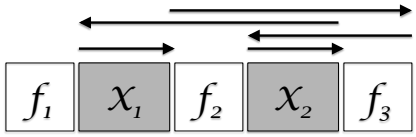
The features we introduce can be used to learn if the model should favour monotone translations at the cost of re-orderings or vice versa and hence can easily adapt to different language pairs. Further, our experiments (see Section 5.4) suggest that the features h and w are not sufficient by themselves to model reordering for language pairs exhibiting very different syntactic structure. Below we introduce new features aimed at modeling reordering phenomena and propose the use of (slightly modified) distortion features for LR-Hiero.

3.3.1 Distortion Features

Our distortion features are inspired by their namesake in phrase-based system, with some modifications to adapt the idea for the discontiguous phrases in LR-Hiero grammar (GNF rules).

Consider a rule $r = \langle \gamma, \bar{b}\ \beta \rangle$, with the source term γ being a mixed string of terminals and non-terminals. Representing the non-terminal spans and each sequence of terminals in γ as distinct *items*, our distortion feature counts the total length of *jumps* between the items during Earley parsing.

Figure 3.4 (a) explains the computation of our distortion feature for an example rule r. Let $I = [I_0, \ldots, I_k]$ be the *items* denoting the terminal sequences and non-terminal spans with I_0 and I_k being dummy items ($\vdash$ and $\dashv$ in Figure) marking the left and right indices of the rule r in input



(a)

$r : \langle f_1 X_1 f_2 X_2 f_3, tX_2 X_1 \rangle \quad I = [\vdash, f_1, f_2, f_3, X_2, X_1, \dashv]$

$r : \langle X_1 \textit{noch nicht} X_2 / \textit{not yet} X_2 X_1 \rangle$

$_1\textit{ihre}_2 \textit{arbeit}_3 \textit{noch}_4 \textit{nicht}_5 \textit{gemacht}_6$

$I = [(1,1),(3,5),(5,6),(1,3),(6,6)]$

(b)

Figure 3.4: (a) Distortion feature computation using a rule r. (b) Example of distortion computation for applying r_3 on phrase $\langle$*ihre arbeit noch nicht gemacht haben*$\rangle$. subscripts between words show the indices which are used to build I. Distortion would be: $d = 2 + 0 + 5 + 3$.

sentence f. Other items are arranged by their realization order on the target-side with the terminal sequences preceding non-terminal spans. The items for the example rule are shown in Figure 3.4 (a). The distortion feature is computed as follows:

$$d(r) = \sum_{j=1}^{k} |I_j^{\mathcal{L}} - I_{j-1}^{\mathcal{R}}| \tag{3.2}$$

where superscripts refer to position of left ($\mathcal{L}$) and right ($\mathcal{R}$) edge of each item in the source sentence f. These are then aggregated across the rules of a derivation D as: $d = \sum_{r \in D} d(r)$. For each item I_j, we count the jump from the end of previous item to the beginning of the current. In Figure 3.4 (a) the jumps are indicated by the arrows above the rule. Figure 3.4 (b) shows an example of distortion computation for r_3 and phrase $\langle$*ihre arbeit noch nicht gemacht haben*$\rangle$ from Figure 3.2.

Since the glue rules are likely to be used in the top levels (possibly with large distortion) of the derivation, we would want the decoder to learn the distortion for regular and glue rules separately. We thus use two distortion features for the two rule types and we call them d_p and d_g.

These features do not *directly* model the source-target reordering, but only capture the source-side jumps. Furthermore they apply for both monotone and reordering rules. We now introduce a new feature for exclusively modelling the reordering.

3.3.2 Reordering Feature

This feature simply counts the number of *reordering rules*, where the non-terminals in source and target sides are reordered. Thus $r_{\langle\rangle} = \text{rule}(D, \langle\rangle)$, where $\text{rule}(D, \langle\rangle)$ is the number of reordering rules in D. Similar to width and height, this feature is applied for rule having at least two non-terminals. This feature is applied to regular and glue rules.

3.4 Experiments

We conduct different types of experiments to evaluate LR-Hiero decoding developed by cube pruning and integrating new features into LR-Hiero system for two language pairs: German-English (de-en) and Czech-English (cs-en). Table 6.3 shows the dataset details.

3.4.1 System Setup

In our experiments we use four baselines as well as our implementation of LR-Hiero (written in Python):

- **Hiero:** we used Kriya, our open-source implementation of Hiero in Python, which performs comparably to other open-source Hiero systems [75]. Kriya can obtain statistically equal BLEU scores when compared with Moses [44] for several language pairs [70, 10].
- **Hiero-GNF:** where we use Hiero decoder with the restricted LR-Hiero grammar (GNF rules).
- **LR-Hiero:** our implementation of LR-Hiero [90] in Python.
- **phrase-based:** Moses [44]
- **LR-Hiero+CP:** LR-Hiero decoding with cube pruning.

We use a 5-gram LM trained on the Gigaword corpus and use KenLM [35] for LM scoring during decoding. We tune weights by minimizing BLEU loss on the dev set through MERT [60] and report BLEU scores on the test set. We use comparable pop limits in each of the decoders: 1000 for Moses and LR-Hiero and 500 with cube pruning for Hiero and LR-Hiero+CP. Other extraction and decoder settings such as maximum phrase length, etc. were identical across settings so that the results are comparable.

Table 3.2 shows how the GNF grammar used in LR-Hiero is much smaller than SCFG grammar (Hiero).

3.4.2 Time Efficiency Comparison

To evaluate the performance of LR-Hiero decoding with cube pruning (LR-Hiero+CP), we compare it with three baselines: (i) Hiero, (ii) Hiero-GNF, and (iii) LR-Hiero (without cube pruning) with two different beam sizes 500 and 1000. When it comes to instrument timing results, there are lots

	Corpus	Train/Dev/Test
cs-en	Europarl(v7), CzEng(v0.9); News commentary	7.95M/3000/3003
de-en	Europarl(v7); News commentary	1.5M/2000/2000

Table 3.1: Corpus statistics in number of sentences

Model	cs-en	de-en
Phrase-based	233.0	77.2
Hiero	1,961.6	858.5
LR-Hiero	230.5	101.3

Table 3.2: Model sizes (millions of rules). We do not count glue rules for LR-Hiero which are created at runtime as needed.

of system level details that we wish to abstract away from, and focus only on the number of "edges" processed by the decoder. In comparison of parsing algorithms, the common practice is to measure the number of edges processed by different algorithms for the same reason [57]. By analogy to parsing algorithm comparisons, we compare the different decoding algorithms with respect to the number of calls made to the language model (LM) since that directly corresponds to the number of hypotheses considered by the decoder. A decoder is more time efficient if it can consider fewer translation hypotheses while maintaining the same BLEU score. All of the baselines use the same wrapper to query the language model, and we have instrumented the wrapper to count the statistics we need and thus we can say this is a fair comparison. For this experiment we use a sample set of 50 sentences taken from the test sets.

Table 3.3 shows the results in terms of average number of language model queries and times in milliseconds.

3.4.3 Reordering Features

To evaluate the new reordering features proposed to LR-Hiero (Section 3.3.2), LR-Hiero+CP with new features is compared to all baselines. Table 3.4 shows the BLEU scores of different models in two language pairs. The baseline [90] model uses all the features mentioned therein but is worse than both phrase-based and Hiero baselines by up to 2.3 BLEU points.

All the reported results are obtained from a single optimizer run. However we observed insignificant changes in different tuning runs in our experiments. We find a gain of about 1 BLEU point when we add a single distortion feature d and a further gain of 0.3 BLEU when we split the distortion feature for the two rule types (d_p and d_g). The last line in part two of Table 3.4 shows a consistent gain of 1.6 BLEU over the LR-Hiero baseline for both language pairs. It shows that LR-Hiero maintains the BLEU scores obtained by "phrase-based" and "Hiero-GNF".

Model	cs-en #queries / time(ms)	de-en #queries / time(ms)
Hiero	5,679.7 / 16.12	7,231.62 / 20.33
Hiero-GNF	4,952.5 / 14.71	5,858.74 / 18.23
LR-Hiero (1000)	46,333.21 / 163.6	83,518.63 / 328.11
LR-Hiero (500)	24,141.03 / 97.61	42,783.12 / 192.23
LR-Hiero+CP	**1,303.2 / 4.2**	**1,697.7 / 5.67**

Table 3.3: Comparing average number and time of language model queries.

Model	**cs-en**	**de-en**
Phrase-based	20.32	24.71
Hiero	20.64	25.52
Hiero-GNF	20.04	24.84
LR-Hiero	18.30	23.47
LR-Hiero + reordering feats	20.20	24.90
LR-Hiero + CP + reordering feats	20.15	24.83
Hiero-GNF + reordering feats	20.52	25.09
Hiero + reordering feats	**20.77**	**25.72**

Table 3.4: BLEU scores. The rows are grouped such that each group use the same model. The last row in part 2 of table shows LR-Hiero+CP using our new features in addition to the baseline Watanabe features (line *LR-Hiero baseline*). The last part shows Hiero using new reordering features. The reordering features used are d_p, d_g and $r_{\langle\rangle}$. LR-Hiero+CP has a beam size of 500 while LR-Hiero has a beam size of 1000, c.f. with the LM calls shown in Table 3.3.

We performed statistical significance tests using two different tools: Moses bootstrap resampling and MultEval [16]. The difference between "LR-Hiero+CP+reordering feat" and three baselines: "phrase-based", "Hiero-GNF", "LR-Hiero+reordering feat" are not statistically significant even for p-value of 0.1 for both tools.

To investigate the impact of the proposed reordering features with other decoder or models, we add these features to both Hiero and Hiero-GNF[6]. The last part of Table 3.4 shows the performance CKY decoder with different models (full Hiero and GNF) with the new reordering features in terms of BLEU score. The results show that these features are helpful in both models. Although, they do not make a big difference in Hiero with full model, they can alleviate the lack of non-GNF rules in Hiero-GNF.

Nguyen et al. (2013) [58] integrate traditional phrase-based features: distortion and lexicalized reordering into Hiero as well. They show that such features can be useful to boost the translation quality of Hiero with the full rule set. Nguyen et al. (2013) [58] compute the distortion feature in a different way, only applicable to CKY. The distortion for each cell is computed after the translation

[6]Feature $r_{\langle\rangle}$ is defined for SCFG rules and cannot be adopted to phrase-based translation systems; and Moses uses distortion feature therefore we omit Moses from this experiment.

for non-terminal sub-spans is complete. In LR-decoding, we compute distortion for rules even though we are yet to translate some of the sub-spans. Thus our approach computes the distortion incrementally for the untranslated sub-spans which are later added. Unlike [58], our distortion feature can be applied to both LR and CKY-decoding (Table 3.4). We have also introduced another reordering feature (Section 3.3.2) not proposed previously.

3.5 Summary and Conclusion

We provided a detailed description of left-to-right Hiero decoding, many details of which were only implicit in [90]. We presented an augmented LR decoding algorithm that builds on the original algorithm in [90] but unlike that algorithm, using experiments over multiple language pairs we showed two new results: (i) Our LR decoding algorithm provides demonstrably more efficient decoding than Hiero and the original LR decoding algorithm in [90]. And, (ii) by introducing new distortion and reordering features for LR decoding we show that it maintains the BLEU scores obtained by phrase-based and Hiero-GNF.

Hiero uses standard Hiero-style translation rules capturing better reordering model than prefix lexicalized target-side translation rules used in LR-Hiero. Our LR-decoding algorithm is 4 times faster in terms of LM calls while translation quality suffers by about 0.67 in BLEU score on average.

Unlike Watanabe et al. (2006) [90], our new features can easily adapt to the reordering requirements of different language pairs. We also introduce the use of future cost in decoding algorithm which is an essential part in decoding. We have shown in this chapter that left-to-right (LR) decoding can be considered as a potential faster alternative to CKY decoding for Hiero-style machine translation systems.

In future work, we plan to apply lexicalized reordering models to LR-Hiero. It has been shown to be useful for Hiero in some languages therefore it is promising to improve translation quality in LR-Hiero which suffers from lack of modeling power of non-GNF target side rules. We also plan to extend the glue rules in LR-Hiero to provide a better reordering model. We believe such an extension would be very effective in reducing search errors and capturing better reordering models in language pairs involving complex reordering requirements like Chinese-English.

Chapter 4

Search Errors and Alignment Coverage in Left-to-Right Decoding

In Chpater 3 we discussed a left-to-right algorithm for LR-Hiero and showed that LR-decoder is more efficient than CKY based decoders. However LR-decoder is unable to capture some hierarchical phrase alignments reachable using CKY decoding and suffers from lower translation quality as a result. In this chapter we introduce some improvements to the LR-decoder to overcome this alignment coverage issue and fix some potential search error in LR-Hiero.

4.1 Introduction

Watanabe et al. (2006) [90] propose a left-to-right (LR) decoding algorithm for Hiero which uses beam search and runs in $O(n^2b)$ in practice where n is the length of source sentence and b is the size of beam [40]. However, this decoding algorithm does not perform well in comparison to current state-of-the-art Hiero and phrase-based translation systems (c.f. Section 3.4). In Chapter 3, we propose an augmented LR decoding which address these limitations in translation quality and time efficiency. This Algorithm is used in LR-Hiero.

Although LR-Hiero performs much faster than Hiero in decoding and obtains BLEU scores comparable to phrase-based translation system, there is still a notable gap between Hiero and LR-Hiero (Section 3.4). In this chapter, we inspect this gap and propose some solutions to improve the LR-Hiero decoding algorithm and fill this gap.

Using instructive examples, we show that Hiero CKY decoder can capture some complex phrasal re-orderings that are observed in language pairs such as Chinese-English that LR-Hiero cannot (c.f. Section 4.3). We extend the LR-Hiero decoder to capture all the hierarchical phrasal alignments that are reachable in CKY decoder.

Unlike Hiero, in LR-Hiero the decoder requires *future cost* to have a fair comparison in pruning hypotheses. We show that using future cost leads to search error in cube pruning. To alleviate this issue, we introduce *queue diversity* to the cube pruning algorithm (c.f. Section 4.2).

rules	hypotheses
	$\langle s \rangle$[0, 15]
G 1)⟨Taiguo shi X_1/Thailand X_1⟩	$\langle s \rangle$ Thailand [2,15]
G 2)⟨yaoX_1/wants X_1⟩	$\langle s \rangle$Thailand wants [3,15]
G 3)⟨liyong X_1/to utilize X_1⟩	$\langle s \rangle$Thailand wants to utilize [4,15]
4)⟨zhe bi qian X_1/this money X_1⟩	$\langle s \rangle$Thailand wants to utilize this money [7,15]
5)⟨X_1zhuru geng duo X_2/ to inject more X_2X_1⟩	$\langle s \rangle$Thailand wants to utilize this money to inject more [12,15][7,9]
6)⟨liudong X_1/circulating X_1⟩	$\langle s \rangle$Thailand wants to utilize this money to inject more circulating [13,15][7,9]
G 7)⟨zijin X_1/capital X_1⟩	$\langle s \rangle$Thailand wants to utilize this money to inject more circulating capital [14,15][7,9]
8)⟨./.⟩	$\langle s \rangle$Thailand wants to utilize this money to inject more circulating capital . [7,9]
9)⟨xiang jingji/to the economy⟩	$\langle s \rangle$Thailand wants to utilize this money to inject more circulating capital . to the economy$\langle /s \rangle$

Figure 4.1: The process of translating the Chinese sentence in Figure 4.3(b) in LR-Hiero. Left side shows the rules used in the derivation. Glue rule are marked by G (see chapter 2). The hypotheses column shows the translation prefix and the ordered list of yet-to-be-covered spans.

We evaluate the modified decoder on three language pairs and show that LR-Hiero can reach the translation scores comparable to CKY-Hiero in two language pairs, and reduce the gap between Hiero and LR-Hiero on the third one.

4.2 LR Decoding with Queue Diversity

As we mentioned in the previous chapters, in LR-Hiero we use GNF grammar which is a constrained form of SCFG: $X \rightarrow \langle \gamma, \bar{b}\, \beta \rangle$ where γ is a string of non-terminal and terminal symbols, $\bar{b}$ is a string of terminal symbols and β is a possibly empty sequence of non-terminals. Using GNF grammar ensures that as each rule is used in a derivation, the target string is generated from left to right. We discussed the rule extraction algorithm in Chapter 2.

LR-Hiero decoder, that we describe in Chapter 3, uses a top-down depth-first search, which strictly grows the hypotheses in target surface ordering. Search on the source side follows an Earley-style search [22], the dot jumps around on the source side of the rules based on the order of nonterminals on the target side. This search is integrated with beam search or cube pruning to find the k-best translations.

Given an input source string, each source side non-terminal is instantiated with the legal spans, e.g. if there is a SCFG rule $\langle aX_1, a'X_1 \rangle$ and if a only occurs at position 3 in the input then this rule can be applied to span $[3, i]$ for all i, $4 < i \leq n$ for input of length n and source side X_1 is instantiated to span $[4, i]$.

Algorithm 4 shows the pseudocode for LR-Hiero decoder with cube pruning (CP). In this section, we will introduce the notion of *queue diversity* to LR-Hiero decoder (function *Merge* in Algorithm 4). We first briefly explain the LR-Hiero decoder algorithm. Algorithm 4 is similar to Algorithm 3 in Chapter 3 and the modified lines have been highlighted.

Algorithm 4 LR-Hiero Decoding

Input sentence: $\mathbf{f} = f_0 f_1 \dots f_n$
$\mathcal{F} =$ FutureCost($\mathbf{f}$) (Precompute future cost[1] for all spans of the source sentence)
$S_0 = \{\}$ (Create empty initial stack)
$h_0 = (\langle s \rangle, [[0, n]], \emptyset, \mathcal{F}_{[0,n]})$ (Initial hypothesis 4-tuple)
Add h_0 to S_0 (Push initial hyp into first Stack)
for $i = 1, \dots, n$ **do**
  $cubeList = \{\}$ (MRL is max rule length)
  for $p = max(i - \text{MRL}, 0), \dots, i - 1$ **do**
    $\{G\}$ = Grouped(S_p) (based on the first uncovered span)
    for $g \in \{G\}$ **do**
      $[u, v] = g_{span}$
      R = GetSpanRules($[u, v]$)
      for $R_s \in R$ **do**
        $cube = [g_{hyps}, R_s]$
        Add $cube$ to $cubeList$
  S_i = Merge($cubeList, \mathcal{F}$) (Create stack S_i and add new hypotheses to it, see Figure 4.2)
return $\arg\max(S_n)$

Merge($CubeList, \mathcal{F}$)
  $heapQ = \{\}$
  for each (H, R) in $cubeList$ **do**
    $hypList$ = **getBestHypotheses**($(H, R), \mathcal{F}, d$) (d best hypotheses of each cube)
    for each h' in $hypList$ **do**
      push($heapQ, (h'_c, h', [H, R])$) (Push new hyp in queue)
  $hypList = \{\}$
  while $|heapQ| > 0$ and $|hypList| < K$ **do**
    $(h'_c, h', [H, R])$ = pop($heapQ$) (pop the best hypothesis)
    push($heapQ, GetNeighbours([H, R])$) (Push neighbours to queue)
    Add h' to $hypList$
  return $hypList$

4.2.1 LR-Hiero Decoder

Figure 4.1 shows a worked out example of how the decoder works. Each partial hypothesis h is a 4-tuple (h_t, h_s, h_{cov}, h_c): consisting of a translation prefix h_t, a (LIFO-ordered) *list* h_s of uncovered spans, source words coverage set h_{cov} and the hypothesis cost h_c. The initial hypothesis is a null string with just a sentence-initial marker $\langle s \rangle$ and the list h_s containing a span of the whole sentence, $[0, n]$. The hypotheses are stored in stacks $S_0, \dots, S_n$, where S_p contains hypotheses covering p source words just like in stack decoding for phrase-based SMT [45].

To fill stack S_i we use hypotheses in stacks S_p, where $i - \text{MRL} \leq p < i$. As the length of rules are limited (at most MRL), we can ignore stacks with index less than $i - \text{MRL}$. Hypotheses in each stack

[1] The future cost is precomputed in a way similar to the phrase-based models [44] using only the terminal rules of the grammar.

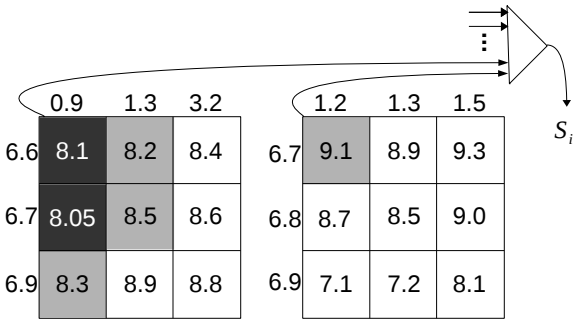


Figure 4.2: Cubes (grids) are fed to a priority queue (triangle) and generated hypotheses are iteratively popped from the queue and added to stack S_i. Lower scores are better. Scores of rules and hypotheses appear on the top and left side of the grids respectively. Shaded entries are hypotheses in the queue and black ones are popped from the queue and added to S_i.

are partitioned into a set of groups $\{G\}$, based on their first uncovered span (line 9). Each group g is a 2-tuple (g_{span}, g_{hyps}), where g_{hyps} is a list of hypotheses which share the same first uncovered span g_{span}. Rules matching the span g_{span} are obtained from routine *GetSpanRules*. Each g_{hyps} and possible R_s (unique source side rule) create a cube which is added to $cubeList$ (line 15). Generated cubes (*cubeList*) are given to the *Merge* routine to generate the K best hypotheses using cube pruning. Decoding finishes when the last stack, S_n, has been filled.

4.2.2 Cube Pruning with Queue Diversity

The routine *Merge* in Algorithm 4 is the main part of cube pruning. *Merge* is responsible to creates the best hypotheses for each stack, given the list of all cubes (*cubeList*). Figure 4.2 shows a detailed view of the *Merge* routine and how it works. In each cube, rows correspond to hypotheses and columns correspond to rules (all rules share the same source side, but have different target sides). Both hypotheses (rows) and rules (columns) are sorted based on their scores. Cube pruning assumes that the best hypothesis of a cube, (H, R), is created from the best hypothesis and rule (leftmost and topmost entry of each cube in Figure4.2) and the next best hypotheses are the neighbours of that entry.

$GetBestHypotheses((H, R), \mathcal{F}, d)$ uses hypotheses H and rule set R to produce new hypotheses[2]. The first best hypothesis, h' along with its score h'_c and the corresponding cube (H, R) create a 3-tuple which is placed in a priority queue $heapQ$ (triangle in Figure 4.2 and line 23 in Algorithm 4). The best hypotheses in the queue are popped (line 26) iteratively. After popping each hypothesis, its neighbours in the corresponding cube are generated and added to the priority queue (line 27). This continues until K best hypotheses are generated and added to *hypList*.

The main idea behind hypotheses generation in cube pruning is that the best possible hypothesis for each cube is generated by combining the best old hypothesis and the best rule (sorted based on their scores). This assumption holds if the final score of new hypotheses is the summation of

[2]This function is described in details in Chapter 3.

old hypothesis score and the rule score. While in addition to hypothesis and rule scores, we need to compute the language model (LM) score for the new partial translation[3]. Adding the LM score violates the hypotheses generation assumption of CP and it can cause search errors.

Figure 4.2 illustrates this issue and how it results in search error. The numbers on the boundaries of cubes[4] are the hypothesis (left) and rule (top) scores. The number in each entry shows the final score (including LM score) of the hypothesis generated from hypothesis and rule corresponding to the row and column of the entry. Note that in cube pruning, we do not generate all possible hypotheses. At each step just the shaded entries have been generated which have been added to the queue.

The entry on the top-left corner of the right cube has a score worse than many other hypotheses in the same cube (after adding LM score). Since the score of this hypothesis is worse than all hypotheses in the left cube, all hypotheses in the right cube will be ignored. This type of search error hurts LR-Hiero more than Hiero, due to the fact that hypotheses scores in LR-Hiero also rely on the future cost. While Hiero uses CKY decoder which fills a CKY chart. Each entry in CKY chart contains hypotheses for the same span, therefore Hiero does not need to use future cost for ranking or pruning the hypotheses.

To solve this issue in LR-Hiero, we introduce the notion of *queue diversity* which is indicated by parameter d in $GetBestHypotheses((H, R), \mathcal{F}, d)$ in Algorithm 4. As shown in Figure 4.2, in any cube the top-left entry might not be the best possible hypothesis. However, if we generate a few more hypotheses of each cube, we might access better hypotheses in the cube which are blocked by a bad hypothesis in the top-left entry. The idea is to generate d best hypotheses from each cube and add them all to the priority queue at the begining (line 21-23). This parameter guarantees that each cube will produce at least d candidate hypotheses for the priority queue. In the standard cube pruning for LR-Hiero $d = 1$.

We call it *queue diversity*, since we apply the idea of diversity at queue level, before generating K best hypothesis. We fill each stack differently from Hiero CKY decoder, thus queue diversity is different from lazy cube pruning [67] or cube growing [39, 87, 91]. In the experimental results we show that adding queue diversity reduces the search error for some language pairs.

4.3 Capturing Missing Alignments

Figure 4.3(a) and Figure 4.3(b) show two examples of a common problem in LR-Hiero decoding. The process of translating the Chinese sentence in Figure 4.3(b) to English, are shown in Figure 4.1. The problem occurs in step 5 of Figure 4.1 where rule #5 is matched to span $[7, 15]$. LR-Hiero decoder maintains a stack (*last-in-first-out*) of yet-to-be-covered spans for each partial hypothesis (h_s in Algorithm 4). To develop new hypotheses from a given partial hypothesis, the decoder should match some rules to the first uncovered span. In this example, span $[7, 15]$ in Step 5. LR-

[3]The target side of rule, r_t, is concatenated to the old translation prefix, h_t.

[4]In LR-Hiero, there are only two dimensions. We abuse the notation for simplicity and use the term cube.

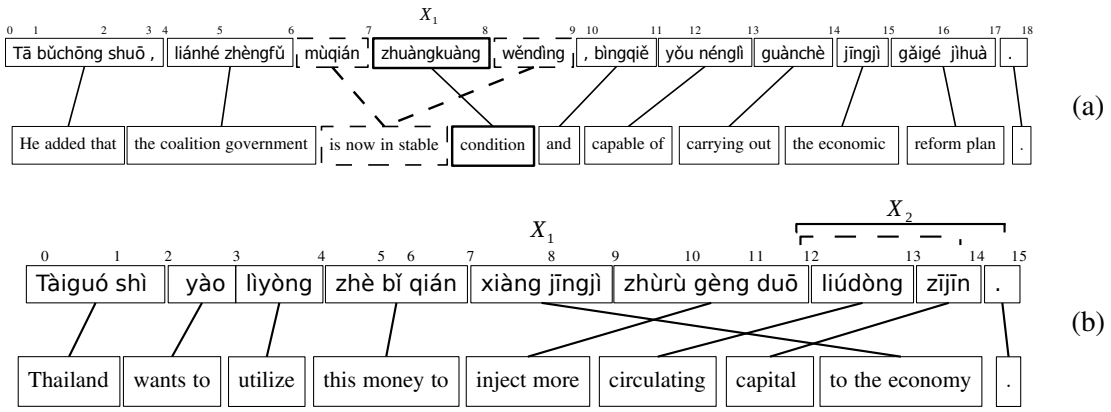


Figure 4.3: Two Chinese-English sentence pairs from devset data in our experiments. (a) Correct rule cannot be matched to [6,18], our modifications match the rule to the first subspan [6,9]. (b) LR-Hiero decoder detects a wrong span for X_2, [12,15]. We modify the rule matching algorithm so that X_2 is matched to three subspans: [12,13], [12,14] and [12,15]. Each match of X_2 corresponds to a distinct hypothesis, therefore this rule results in three hypotheses.

Hiero matches rule #5 to span $[7, 15]$, therefore X_2 is forced to match to span $[12, 15]$. This rule reorders the translation of span $[7, 9]$ (corresponding to X_1) and $[12, 15]$ which at the end leads to an incorrect translation in Step 9.

However, if we use Hiero for translating this sentence using the same set of rules, the CKY decoder will be able to generate the correct translation for span. Since the CKY-decoder works bottom-up, it first generates the best translation for each source span and then combines them. In this example, CKY decoder can use rule#5 to translate span $[7, 14]$. Then it simply combines the translation of $[7, 14]$ with the translation of spans $[0, 7]$ and $[14, 15]$ using glue rules (monotonic combination).

Figure 4.3(a) shows another issue in decoding. In this example the first two phrases can be translated monotonically using rules in the form of $\langle \bar{b} X_1, \bar{b}' X_1 \rangle$, where $\bar{b}$ and $\bar{b}'$ are strings of terminal symbols in the source and target respectively. First rule $\langle$ta buchong shuo , X_1, he added that $X_1\rangle$ is matched to the span of whole sentence, $[0, 18]$ (the initial uncovered span). By matching this rule the non-terminal X_1 is matched to span $[4, 18]$ which creates the new uncovered span for the obtained partial hypothesis. Later in expanding the second partial hypothesis, span $[4, 18]$ is matched by rule $\langle$lianhe zhengfu X_1, the coalition government $X_1\rangle$ and the non-terminal X_1 in this rule is matched to span $[6, 18]$. For span $[6, 18]$, we have to apply rule $\langle$muqian X_1 wending, is now in stable $X_1\rangle$ to obtain the correct translation. But this rule cannot be matched to span $[6, 18]$ and the decoder fails to generate the correct translation. Again, CKY decoder in Hiero can apply this rule to span $[6, 9]$ and generate the correct translation for this span. Then the translation of this span can be monotonically concatenated to span $[0, 6]$ and later on to the remaining spans ($[9, 18]$).

In both these cases, Hiero has no difficulty in reaching the target sentence using *the same GNF rules*. This fact that, in LR-Hiero we have to process spans in the same order as they added to the

the stack of yet-to-be-covered spans, h_s, applies severe constraints to the decoding process. While bottom-up decoders such as Hiero CKY can capture the alignments in Figure 4.3.

4.3.1 Augmenting GNF Rules

We extend the LR-Hiero decoder to handle the cases discussed in previous section, by augmenting the GNF grammar and make it more expressive. Generally GNF Rules in LR-Hiero can be grouped into three types based on the right boundary of the source and target side. These groups are shown in Equation 4.1.

$$\begin{aligned} &\text{(a)}\ \langle \gamma \bar{a}, \bar{b} \beta \rangle \\ &\text{(b)}\ \langle \gamma X_n, \bar{b} \beta X_n \rangle \\ &\text{(c)}\ \langle \gamma X_n, \bar{b} \beta X_m \rangle \end{aligned} \tag{4.1}$$

where γ is a string of terminals and non-terminals, $\bar{a}$ and $\bar{b}$ are terminal sequences of source and target respectively, β is a possibly empty sequence of non-terminals and X_n and X_m are different non-terminals[5]. We will discuss about the issues that LR-Hiero decoder might face using each type and then propose a solution to resolve these issues.

- **Type (a)**: rules end at a terminal (lexical term). These rules can only be matched to spans ending at the same word. Example (a) in Figure 4.3 shows an issue caused by this type of rules.

- **Type (b)**: rules end at the same non-terminal, X_n, on both source and target sides. These type of rules model the monotone concatenation of any spans match to X_n to the first part of rule (before X_n).

- **Type (c)**: rules end at different non-terminals on source and target sides. Unlike type (b), these rules reorder the last non-terminal on the source side, X_n. Figure 4.3 (b) is an example of mis-matching this type of rules.

To resolve the possible issues, we propose to augment the GNF grammar by creating new rules corresponding to each extracted GNF rule. The new rules are created by adding a new non-terminal X_r to the end of GNF rules (both source and target sides). Note that the new rules are created on the fly during decoding. The new non-terminal X_r is responsible to match the right boundary. Equation 4.2 shows the new rules.

[5] In rule type (c) X_n will be a part of β and X_m will be a part of γ.

Model	Cs-En	De-En	Zh-En
Hiero	318	351	187
LR-Hiero	278	300	132
LR-Hiero+(abc)	338	361	174

Table 4.1: No. of sentence covered in forced decoding of a sample of sentences from the devset.

$$\begin{aligned}
&\text{(a)}\ \langle \gamma\bar{a}, \bar{b}\beta\rangle \Rightarrow \langle \gamma\bar{a}X_r, \bar{b}\beta X_r\rangle \\
&\text{(b)}\ \langle \gamma X_n, \bar{b}\beta X_n\rangle \Rightarrow \langle \gamma X_n X_r, \bar{b}\beta X_n X_r\rangle \\
&\text{(c)}\ \langle \gamma X_n, \bar{b}\beta X_m\rangle \Rightarrow \langle \gamma X_n X_r, \bar{b}\beta X_m X_r\rangle
\end{aligned} \tag{4.2}$$

X_n and X_m are different non-terminals distinct from X_r. The extra non-terminal X_r enables the decoder to add a new yet-to-be-covered span to the bottom of the stack, h_s, in generating each new partial hypothesis [6]. This allows the decoder to match any two adjacent spans and simulate monotonic glue rule, $\langle SX, SX\rangle$ in Hiero CKY decoder.

In an experiment, we translated sentences of devset data using forced decoding to evaluate the effect of augmenting GNF rules on alignment coverage[7]. Table 4.1 compares the number of sentences that can be translated by LR-Hiero and Hiero, in forced decoding mode. It shows that augmenting GNF rules improves the coverage by 31% for Chinese-English and more than 20% for the other two language pairs.

4.4 Experiments

The experimental results in this chapter is divided into two parts: (i) experiments to evaluate modified LR-Hiero decoder proposed in this chapter (Section 4.4.1); (ii) experiments to evaluate the effect of GNF rules extracted using GNF extraction algorithm proposed in Chapter 2. We use three language pairs in all experiments (Table 6.3): German-English (De-En), Czech-English (Cs-En) and Chinese-English (Zh-En).

4.4.1 Improvements on LR-Hiero Decoder

We use a 5-gram LM trained on the Gigaword corpus and use KenLM [35] for LM scoring during decoding. We tune weights by minimizing BLEU loss on the dev set through MERT [60] and report BLEU scores on the test set. We use pop limit (beam size) 500 for Hiero and LR-Hiero. Other extraction and decoder settings such as maximum phrase length, etc. are identical across settings so that results are comparable. To make the results comparable we use the same feature set for

[6] For the sake of simplicity, in rule type (b) we can merge X_n and X_r as they are in the same order on both source and target side.

[7] c.f. Section 4.4 for experimental settings.

	Corpus	Train/Dev/Test
Cs-En	Europarl(v7) + CzEng(v0.9); News commentary(nc) 2008&2009; nc 2011	7.95M/3000/3003
De-En	Europarl(v7); WMT2006; WMT2006	1.5M/2000/2000
Zh-En	HK + GALE phase-1; MTC part 1&3; MTC part 4	2.3M/1928/919

Table 4.2: Corpus statistics in number of sentences. Tuning and test sets for Chinese-English has 4 references.

Model	Cs-En	De-En	Zh-En
Hiero	1,961.6	858.5	471.9
LR-Hiero	266.5	116.0	100.9

Table 4.3: Model sizes (millions of rules).

all baselines which includes standard features of Hiero: two relative-frequency probabilities $p(e|f)$ and $p(f|e)$, two lexically weighted probabilities $lex(e|f)$ and $lex(f|e)$, language model probability, word penalty, phrase penalty, and glue rule penalty. We add distortion features (separated for regular and glue rules in LR-Hiero) proposed in Chapter 3 [80] to all translation systems including Hiero.

We use 3 baselines: (i) our implementation of Watanabe et al. (2006) [90] LR decoding for Hiero which use beam search: LR-Hiero (beam); (ii) LR-Hiero decoder proposed in Chapter 3: (LR-Hiero+CP); (iii) Kriya, an open-source implementation of Hiero in Python, which performs comparably to other open-source Hiero systems [75]: (Hiero).

Table 4.3 shows model sizes for LR-Hiero (GNF) and Hiero (SCFG). Typical Hiero rule extraction excludes phrase-pairs with unaligned words on boundaries (loose phrases[8]). In these experiments, we use the Hiero rule extraction heuristic, but exclude non-GNF rules. We include loose phrase-pairs as terminal rules.

Table 4.4 shows the translation quality of different systems in terms of BLEU score. Row 3 is repeated from Chapter 3 which is presented in [80] [9]. As we discussed in Section 4.2.2, LR-Hiero+CP suffers from severe search errors on Zh-En (1.5 BLEU) but using queue diversity (QD=15) we fill this gap. We use the same QD(=15) in next rows for Zh-en. For Cs-En and De-En we use regular cube pruning (QD=1), as it works as well as beam search (compare rows 4 and 2).

We measure the benefit of the new modified rules from Section 4.3: (*ab*): adding modifications for rules type (a) and (b); (*abc*): modification of all rules. We can see that for all language pairs

[8] c.f. Section 2.2.2 for the formal definition of loose phrases.

[9] We report results on Cs-En and De-En in [80]. Row 4 is the same translation system as row 3 (LR-Hiero+CP). We achieve better results than our previous work [80] (row 4 vs. row 3) due to bug corrections and adding loose phrases as terminal rules.

Model	Cs-En	De-En	Zh-En
Hiero	20.77	25.72	27.65
LR-Hiero (beam) [90]	20.72	25.10	25.99
LR-Hiero+CP [80]	20.15	24.83	-
LR-Hiero+CP (QD=1)	20.68	25.14	24.44
LR-Hiero+CP (QD=15)	-	-	26.10
LR-Hiero+CP+(ab)	20.88	25.22	26.55
LR-Hiero+CP+(abc)	20.89	25.22	26.52

Table 4.4: BLEU scores for different baselines and modified LR-Hiero. QD=15 indicates we use $d = 15$ in Algorithm 4. We use QD=15 for Zh-En in last three rows.

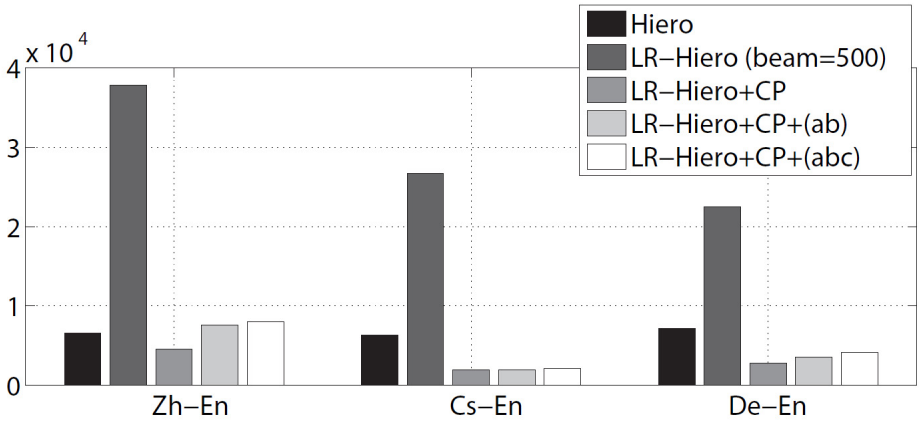


Figure 4.4: Average number of language model queries.

(ab) constantly improves performance of LR-Hiero, significantly better than LR-Hiero+CP and LR-Hiero (p-value<0.05) on Cs-En and Zh-En, evaluated by MultEval [16]. But modifying rule type (c) does not show any improvement. Since, augmenting type (c) rules with a new non-terminal introduces spurious ambiguity.

Figure 4.4 shows the results in terms of average number of language model queries on a sample set of 50 sentences from test sets. All of the baselines use the same wrapper to KenLM [35] to query the language model, and we have instrumented the wrapper to count the statistics. In Section 3.4 we showed that LR-Decoder with beam search by Watanabe et al. [90] does not perform at the same level of state-of-the-art Hiero (more LM calls and less translation quality). As we can see in this figure, adding new modified rules slightly increases the number of language model queries on Cs-En and De-En so that LR-Hiero still works 2 to 3 times faster than Hiero. On Zh-En, LR-Decoder works significantly better than LR-Hiero decoder which runs with cube pruning (row 2 and 4 in Table 4.4). It is mainly due to the search error that we discussed in Section 4.2.2. We apply queue diversity (QD=15) which reduces search errors and improves translation quality but increases the number of hypothesis generation on the other hand.

Model (msl)	**Cs-En**	**De-En**	**Zh-En**
SCFG (7)	1,961.6	858.5	471.8
GNF (7)	306.3	116.0	100.9
GNF-4 (10)	380.9	214.9	190.0

Table 4.5: Model sizes in millions of rules. Maximum source length (msl) is shown in brakets.

Model	Cs-En	De-En	Zh-En
Hiero	20.77	25.72	27.65
LR-Hiero [90]	20.72	25.05	25.99
LR-Hiero+CP	20.52	25.07	26.10
LR-Hiero+CP (GNF-1)	20.38	24.20	25.81
LR-Hiero+CP (GNF-2)	20.49	25.32	25.92
LR-Hiero+CP (GNF-3)	20.50	25.34	26.13
LR-Hiero+CP (GNF-4)	20.50	25.34	26.10

Table 4.6: BLEU scores for Hiero, LR-Hiero and LR-Decoder [90]. GNF-x: GNF grammars with at most x non-terminals using the proposed rule extraction algorithm.

Comparing Table 4.4 with Figure 4.4 shows that our modifications to LR-Hiero decoder significantly improves the BLEU scores compared to previous LR decoder for LR-Hiero. We obtain comparable results to Hiero for Cs-En and De-En and remarkably improve results on Zh-En, while at the same time making 2 to 3 times less LM calls on Cs-En and De-En compared to CKY-Hiero.

4.4.2 GNF Rule Extraction

In this section we evaluate GNF rules extracted using the LR-Hiero rule extraction algorithm proposed in Chapter 2[10]. We use initial phrase pairs of any length (as long as sentence length) to extract rules (arity 1 to 4) and maximum source rule length 10. Terminal rules are constrained to maximum source rule length 7. To make the comparison easier, we repeat the Table 2.2 from Chapter 2 here. Table 4.5 shows model sizes for GNF grammar extracted by Hiero rule extraction heuristic (GNF), Hiero (SCFG) and GNF grammar extracted by GNF rule extraction with at most 4 non-terminals (GNF-4). We include loose phrase-pairs as terminal rules in GNF grammars.

To evaluate the GNF rule extraction, we use all grammars in LR-Hiero and compare them with SCFG grammar in Hiero. The experimental setting is the same as experiments in Section 4.4.1[11].

Table 4.6 shows the BLEU score for different decoders and grammars. The last 4 rows are GNF grammar with 1 to 4 non-terminals extracted by our rule extraction. To show how adding more non-terminals affect the alignment coverage, we translate the devset sentences with different grammars in forced decoding mode. We use CKY decoding for SCFG and LR-Hiero decoder for

[10]The results in this section has been reported in [81]. Some of the experiments on LR-Hiero has been repeated which leads to minor changes in some results.

[11]We use queue diversity (QD=15) for Chinese-English in LR-Hiero.

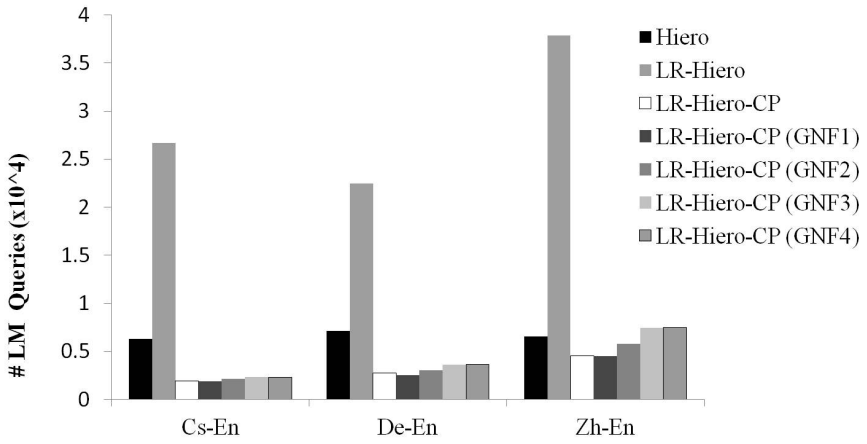


Figure 4.5: Average number of language model queries. (GNF4) denotes new GNF grammar with 4 non-terminals.

GNF grammars. Table 4.1 shows the size of the reachable subset by forced decoding for different grammars. It shows that adding more non-terminals considerably improves the alignment coverage on De-En and Zh-En (average 24%).

Comparing Tables 4.6 and 4.1 is interesting. While adding rules with more than 2 non-terminals does not change BLEU score it improves the alignment coverage. In our analysis we notice that LR-Decoder rarely uses rules with 3 or 4 non-terminals in the K-best list. It is probably because, rules with less non-terminals are generally more frequent and hypotheses which use them have got higher scores during decoding. Here we just use Hiero and LR-Hiero standard features which are not designed for rules with more complex reordering. The next step is to elaborate features for rules with 3 and 4 non-terminals[12].

To evaluate the effect of the grammars on the decoding process in terms of speed, we use the number of language model calls since that directly corresponds to the number of hypotheses considered by the decoder, consequently the speed of decoder. Figure 4.5 shows the results in terms of average number of language model queries and times in milliseconds on a sample set of 50 sentences from test sets.

4.5 Summary and Conclusion

In this chapter we introduce two improvements to LR-Hiero decoder. We add queue diversity to the cube pruning algorithm for LR-Hiero to solve the search error issue. We show that CKY-based decoders can capture some complex phrasal re-orderings (observed in language pairs such as Chinese-English) which cannot be captured by the LR-Hiero decoder. We extend the LR-Hiero decoder to capture all these hierarchical phrasal alignments that are reachable in CKY decoders.

[12]In another experiment not reported here, we extract rules with unlimited number of non-terminals and source rule length for Cs-En (while we keep non-adjacent non-terminals on the source side). But filtering rules on dev and test sets results in rules with at most 5 non-terminals.

Our experimental results show that these modifications improve translation quality of LR-Hiero to reach comparable (slightly better) BLEU scores to LR-Hiero on Czech-English and significantly improve the BLEU score on Chinese-English language pair.

Chapter 5

Lexicalized Reordering Model for Left-to-Right Hierarchical Phrase-based Translation

Phrase-based and hierarchical phrase-based (Hiero) translation models differ radically in the way reordering is modeled. Lexicalized reordering models play an important role in phrase-based MT and such models have been added to CKY-based decoders for Hiero. In Chapter 3, we proposed a promising decoding algorithm for Hiero (LR-Hiero) that visits input spans in arbitrary order and produces the translation in left to right (LR) order which leads to far fewer language model calls and leads to a considerable speedup in decoding. In Chapter 4 we introduced an augmented version of LR-Hiero decoder to capture more hierarchical phrasal alignment and fix some potential search errors. In this chapter we introduce a novel lexicalized reordering model (LRM) for LR-Hiero and show that it improves translation quality for Czech-English, Chinese-English and German-English.

5.1 Introduction

The phrase-based translation system generates the translation from left-to-right, by selecting a phrase on the source sentence and concatenating its translation to the growing target sentence. It continues until all source words are translated. A distortion penalty is used to penalize deviation from the monotone translation (no reordering between source and target) [45, 63]. Identical distortion penalties for different types of phrases ignore the fact that certain phrases (with certain words) were more likely to reorder than others.

State-of-the-art phrase based translation systems directly address this issue by applying *lexicalized reordering models* (LRM) to capture the reordering of phrase pairs [84, 44, 30, 31]. LRM uses word aligned data to determine how each source-target phrase pair tends to be reordering during decoding. LRM condition reordering probabilities on the words of each phrase pair. These models distinguish three orientations with respect to the previously translated phrase: *monotone* (M), *swap*

(S), and *discontinuous* (D), which are primarily designed to handle local re-orderings of neighbouring phrases.

Hierarchical phrase-based translation (Hiero) [15] uses hierarchical phrases for translations which are represented in the form of a lexicalized synchronous context-free grammar (SCFG). Non-terminals in the SCFG rules correspond to gaps in phrases which are recursively filled by other rules (phrases). The SCFG rules are extracted from word and phrase alignments of a bitext. Hiero uses CKY-style decoding which parses the source sentence with time complexity $O(n^3)$ and synchronously generates the target sentence (translation).

Left-to-right hierarchical translation system (LR-Hiero) also uses translation rules in the form of SCFG grammar. LR-Hiero uses an augmented version of the left-to-right (LR) decoder first proposed by Watanabe et al. (2006) [90] for Hiero. This decoder is an Earley style parser which parses the source sentence using SCFG rules but generates translation in left-to-right manner. To simplify target generation, SCFG rules are constrained to be prefix-lexicalized on target side aka Greibach Normal Form (GNF). Restricting the target generation in one way (left-to-right) which just requires a single language model (LM) history for each hypothesis effectively helps speed up decoding process [80, 82].

We showed that GNF grammars can get close to the BLEU scores obtained by full set of Hiero rules (SCFG) for some language pairs (c.f. Chapter 4). However, in language pairs with more complex reordering like Chinese-English, discontinuous spans on the target side play an important role in modeling the reordering. Therefore removing all non-GNF rules can hurt the translation quality.

In Chapter 3, we introduce a new distortion feature to Hiero and LR-Hiero which significantly improves translation quality in LR-Hiero and improves Hiero results to a lesser extent. Nguyen et al. (2013) [58] integrate a distortion and lexicalized reordering feature with a CKY-based Hiero decoder significantly improving the translation quality. In their approach, each partial hypothesis during decoding is mapped into a phrase-based translation path to compute the reordering and distortion features. However they use an LRM trained for phrase-based MT [31] which cannot be applied to all Hiero rules. Cao et al. (2014)[11] propose an approach to directly train LRM for Hiero rules. These two approaches are designed for CKY-decoding and cannot be applied to LR-Hiero.

To improve the reordering model we introduce lexicalized reordering model (LRM) to LR-Hiero. We show that augmenting LR-Hiero by lexicalized reordering model significantly improve the translation quality in different language pairs.

5.2 Lexicalized Reordering Model

The main idea in phrase-based LRM is to divide possible reorderings into three orientations that can be easily determined during decoding and also from word-aligned sentence pairs (parallel corpus). Given a source sentence $\mathbf{f}$, a sequence of target language phrases $\mathbf{e} = (\bar{e_1}, \ldots, \bar{e_n})$ is generated

by the decoder, and a phrase alignment $\mathbf{a}= (a_1, \ldots a_n)$ defines a source phrase $\bar{f_{a_i}}$ for each target phrase $\bar{e_i}$.

For each phrase-pair, the orientations are described in terms of the previously translated source phrase $f_{a_{i-1}}$ (*prev*) and the next source phrase to be translated $f_{a_{i+1}}$ (*next*):

- Monotone (M): *next* follows *prev* (immediately).
- Swap (S): *prev* follows *next* (immediately).
- Discontinuous (D): *next* and *prev* are not adjacent in the source sentence.

Usually *next* and *prev* are defined for both directions left-to-right and right-to-left in generating translation. we only discuss left-to-right here, but the right-to-left case is similar (but symmetrical).

The probability of an orientation given a phrase pair $\langle \bar{f}, \bar{e} \rangle$ can be estimated using relative frequency:

$$P(o|\bar{f}, \bar{e}) = \frac{cnt(o, \bar{f}, \bar{e})}{\sum_{o' \in \{M,S,D\}} cnt(o', \bar{f}, \bar{e})} \tag{5.1}$$

where, $o \in \{M, S, D\}$ and *cnt* is computed on word-aligned parallel data (count phrase-pairs and their orientations). Given the sparsity of the orientation types, we may use smoothing.

As the decoder develops a new hypothesis by translating a source phrase, f_{a_i}, it assesses the implied orientation, o_i, (respect to a_{i-1}) to verify if the reordering does make sense. During decoding, for each phrase pair $\langle \bar{f_{a_i}}, \bar{e_i} \rangle$, only one orientation is activated (the probability is greater than zero):

- $o_i = M$ if $a_i - a_{i-1} = 1$
- $o_i = S$ if $a_i - a_{i-1} = -1$
- $o_i = D$ if $|a_i - a_{i-1}| \neq 1$

The log of this probability is easily folded into the linear models (one log-linear weight for each orientation) that guide the decoder.

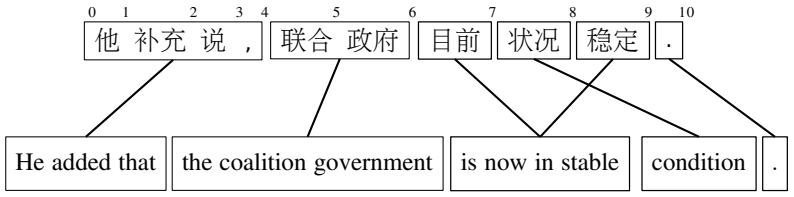
5.3 Lexicalized Reordering Model for LR-Hiero

LR-Hiero uses a subset of the Hiero SCFG rules where the target rules are in Greibach Normal Form (GNF): $\langle \gamma, \bar{e}\, \beta \rangle$ where γ is a string of non-terminal and source words, $\bar{e}$ is a target phrase and β is a possibly empty sequence of non-terminals. We abuse notation slightly and call this a GNF SCFG grammar. The LR-Hiero decoder develops a new hypothesis by applying a GNF rule , $r = \langle \gamma, \bar{e}\, \beta \rangle$, to an untranslated span of the source sentence (determined by the SCFG)[1]. The translation prefix of the new hypothesis is generated by appending the target side of the applied rule, $\bar{e}$, to the translation prefix of the old hypothesis. Figure 5.1 shows a walk-through of decoding

[1] *Uncovered span*. The order for translating source spans is determined by the grammar.

rules	hypotheses $\langle h_t, h_s, h_c \rangle$
	$\langle$ <s> , $[[0,10]]$, 0 $\rangle$
(1)	$\langle$ <s> *He added that* , $[[4,10]]$, 4.3 $\rangle$
(2)	$\langle$ <s> *He added that the coalition government* , $[[6,10]]$, 7.7 $\rangle$
(3)	$\langle$ <s> *He added that the coalition government is now in stable* , $[[7,8][9,10]]$, 11.2 $\rangle$
(4)	$\langle$ <s> *He added that the coalition government is now in stable condition* , $[[9,10]]$, 13.4 $\rangle$
(5)	$\langle$ <s> *He added that the coalition government is now in stable condition.* </s> , $[]$, 14.3 $\rangle$

Figure 5.1: The process of translating a Chinese (Figure 5.2) sentence to English using LR-Hiero. Left side shows the rule Numbers used in each step of creating the derivation (Rules are shown in Figure 5.2). The hypotheses column shows 3-tuple partial hypotheses: the translation prefix, h_t, the ordered list of yet-to-be-covered spans, h_s, and cost h_c.



rules	phrase pairs
1) $X \rightarrow \langle$ 他 补充 说 , X_1 / *He added that* $X_1 \rangle$	1) $\langle$ 他 补充 说 , / *He added that* $\rangle$
2) $X \rightarrow \langle$ 联合 政府 X_1 / *the coalition government* $X_1 \rangle$	2) $\langle$ 联合 政府 / *the coalition government* $\rangle$
3) $X \rightarrow \langle$ 目前 X_1 稳定 X_2 / *is now in stable* $X_1 X_2 \rangle$	3) $\langle$ 目前 * __ * 稳定 / *is now in stable* $\rangle$
4) $X \rightarrow \langle$ 状况 / *condition* $\rangle$	4) $\langle$ 状况 / *condition* $\rangle$
5) $X \rightarrow \langle$ ./. $\rangle$	5) $\langle$ ./. $\rangle$

Figure 5.2: A word-aligned Chinese-English sentence pair on the top (from devset data used in experiments.) The rules pane shows the rules used for decoding in Figure. 5.1; Phrase-pairs pane showing the set of source-target phrase pairs created by removing the non-terminals from the rules.

the example in Figure 5.2. The target generation in LR-Hiero is analogous to phrase-based MT. Given an input sentence **f**, the output translation is a sequence of contiguous target-language phrases $\mathbf{e} = (\bar{e_1}, \ldots, \bar{e_n})$ incrementally concatenated during decoding. We can define a phrase alignment $\mathbf{a} = (a_1, \ldots a_n)$ which aligns each target phrase, $\bar{e_i}$ to a source phrase f_{a_i} corresponding to source side of a rule, r_i used at step i. But unlike target, source phrases can be discontiguous. For each rule $r_{a_i} = \langle \gamma_i, \bar{e_i}\, \beta_i \rangle$, the source phrase f_{a_i} is created by removing all non-terminals on the right or left boundaries of γ_i and then replacing the other non-terminals with a gap[2] (Function *removeNonTerm* in Algorithm 5). Figure 5.1 illustrates the process of translating a Chinese-English sentence pair by LR-Hiero.

5.3.1 Training

Here we discuss our approach to compute $P(o|\bar{f}, \bar{e})$, the probability of an orientation given phrase pair of a rule, $r.p = \langle \bar{f}, \bar{e} \rangle$, on word-aligned data using relative frequency. We assume that phrase

[2]We replace the non-terminals with a unique symbol *__*.

Algorithm 5 Training LRM

Input
  sentence pair: $\langle \mathbf{f}, \mathbf{e} \rangle$
  rule: $r = \langle \gamma, \bar{e}\beta \rangle$
$\bar{f} = removeNonTerm(\gamma)$ (remove non-terminals in src of r)
for each $\langle \bar{f}', \bar{e}' \rangle$ in $\langle \mathbf{f}, \mathbf{e} \rangle$ **do**
  if $s - t' = 1$ **then**
    if $u - v' = 1$ **then**
      $orientation = M$
    elif $v - u' = 1$ **then**
    $orientation = S$
    else
    $orientation = D$
  else
    $orientation = D$
return $orientation$

$\bar{e}$ spans the word range $s \ldots t$ in the target sentence and the phrase $\bar{f}$ spans the range $u \ldots v$ in the source sentence[3].

For a given phrase pair $\langle \bar{f}, \bar{e} \rangle$, we set orientation to $o = M$ if there is a phrase pair,$\langle \bar{f}', \bar{e}' \rangle$, where its target side appears just before the target side of the given phrase ($s = t' + 1$) and its source side also appears just before $\bar{f}$, or $u = v' + 1$. Orientation is $o = S$ if there is a phrase pair where its target side appears just before $\bar{e}$ ($s = t' + 1$) but its source side also appears just after $\bar{f}$ ($v = u' - 1$). Otherwise the orientation is $o = D$.

Algorithm 5 shows how to determine the orientation for a rule in a sentence pair from training data[4]. We consider phrases of any length to compute orientation. Note that although the phrase pair extracted from the rules can be discontinuous, we just consider continuous phrases in the sentence pair to compute orientation.

We use relative frequency to build smoothed maximum likelihood estimates of orientation probabilities. Once orientation counts for rules (phrase-pairs obtained form rules) are collected from the bitext, the probability model $P(o|\bar{f}, \bar{e})$ is estimated using recursive MAP smoothing as discussed in [13]:

[3]The phrase $\bar{e}'$ spans the word range $s' \ldots t'$ and similarly for the source phrases.
[4]The given rule exists in the sentence pair.

$$
\begin{aligned}
P(o|\bar{f},\bar{e}) &= \frac{cnt(o,\bar{f},\bar{e}+\alpha_t P_t(o|\bar{e})+\alpha_s P_s(o|\bar{f})}{\sum_{o'} P(o'|\bar{f},\bar{e})+\alpha_t+\alpha_s} \\
P_s(o|\bar{f}) &= \frac{\sum_{\bar{e'}} cnt(o,\bar{f},\bar{e'})+\alpha_g P_g(o)}{\sum_{\bar{e'},o'} cnt(o',\bar{f},\bar{e'})+\alpha_g} \\
P_t(o|\bar{e}) &= \frac{\sum_{\bar{f'}} cnt(o,\bar{f'},\bar{e})+\alpha_g P_g(o)}{\sum_{\bar{f'},o'} cnt(o',\bar{f'},\bar{e})+\alpha_g} \\
P_g(o) &= \frac{\sum_{\langle\bar{f},\bar{e}\rangle} cnt(o,\bar{f},\bar{e})+\alpha_u/3}{\sum_{\langle\bar{f},\bar{e}\rangle,o'} cnt(o',\bar{f},\bar{e})+\alpha_u}
\end{aligned}
\tag{5.2}
$$

where the α parameters can be tuned empirically[5].

5.3.2 Decoding

Phrase-based LRM uses local information to determine orientation for a new phrase pair, $\langle\bar{f}_{a_i},\bar{e}_i\rangle$, during decoding [44, 84]. For left-to-right order, $\bar{f}_{a_i}$ is compared to the previously translated phrase $\bar{f}_{a_{i-1}}$ to find orientation and compute the score. For instance, if $\bar{f}_i$ is the left adjacent of $\bar{f}_{i-1}$, the orientation is swap with respect to the previous phrase, then feature $P(o_i = S|\bar{f}_i,\bar{e}_i)$ in the current translation hypothesis is activated.

Galley et al. (2008) [30] introduce the hierarchical phrase reordering model (HRM) which increases the consistency of orientation assignments. In HRM, the emphasis on the previously translated phrase is removed and instead a compact representation of the full translation history, as represent by a shift-reduce stack, is used to determine orientation during decoding. Once a source span is translated, it is shifted onto the stack; if the two spans on the top are adjacent, then a reduction merges the two. During decoding, orientations are always determined with respect to the top of this stack, rather than the previously translated phrase.

LR-Hiero is a hierarchical phrase-based model that uses SCFG rules[6] for translation. Although we reduce the SCFG rules to phrase pairs for training the reordering model, the rules are used for decoding and the order of translating source phrases (spans) are determined by the non-terminals in the SCFG rules. Therefore we cannot simply rely on the previously translated phrase to compute the reordering scores during decoding.

Since LR-Hiero uses lexicalized glue rules [90], non-terminals can be matched to very long spans on the source sentence. It makes LRM in LR-Hiero comparable to HRM in phrase-based MT. However, we cannot rely on the full translation history like HRM, since translation model is a SCFG grammar encoding reordering information.

We employ a shift-reduce approach to find a compact representation of the recent translated source spans which is also represented by a stack, S, for each hypothesis. However, S always

[5]Following [13] we use $\alpha_* = 10$.

[6]LR-Hiero uses a restricted form on SCFG rules, in which the target is in GNF format.

Algorithm 6 LRM in Decoding

Input
  hypothesis: $hyp = (S, r, feat)$
  rule: $r = (\bar{f}, \bar{e}, feat)$
$hyp' = (S' = \{\}, None, hyp.feat)$ (initialize new hypothesis)
$r_p = hyp.r$ (the rule which was used to generate hyp)
if $r.\bar{f}$ right adjacent S **then**
  $active\_M(hyp'.feat, r.feat)$
  $S' = update(S, r.\bar{f})$
elif $r.\bar{f}$ left adjacent S **then**
  $active\_S(hyp'.feat, r.feat)$
  $S' = update(S, r.\bar{f})$
elif $r.\bar{f}$ right adjacent $r_p.\bar{f}$ **then**
  $active\_M(hyp'.feat, r.feat)$
  $S' = update(r_p.\bar{f}, r.\bar{f}, S)$
elif $r.src$ left adjacent $r_p.\bar{f}$ **then**
  $active\_S(hyp'.feat, r.feat)$
  $S' = update(r_p.\bar{f}, r.\bar{f}, S)$
else
  $active\_D(hyp'.feat, r.feat)$
  $S' = update(S, r.\bar{f})$
$hyp'.r = r$
return hyp'

update$(pre\_span, new\_span, hyp\_span = None)$
  if $hyp\_span \neq None$ **then**
    if hyp_span covers new_span **then**
      return $add(hyp\_span, new\_span)$
  if $(pre\_span$ covers $new\_span)$ **or** $(pre\_span$ adjacent $new\_span)$ **then**
    return $add(pre\_span, new\_span)$
  return new_span

contains just one source span (which might be discontiguous), unlike HRM which always maintain all previous translated spans. As the decoder applies a rule, r_i, the corresponding source phrase $r_i.\bar{f}$ is compared respect to the span in S to determine the orientation. If they are adjacent or S covers the span $r_i.\bar{f}$, they are reduced. Otherwise stack is set to $S = r_i.\bar{f}$. Orientation of $r_i.\bar{f}$ is computed with respect to S but if they are not adjacent (*M* or *S*), we still need to consider the possible local reordering with respect to the previous rule $r_{i-1}.\bar{f}$. For instance, in Figure 5.3, rules #5,#4 are monotone, while both are covered by the current span in S. This algorithm runs in $O(1)$ since it needs a limited number of comparisons to update S and compute orientation, unlike HRM which needs to maintain a sequence of contiguous spans in the stack and runs in linear time.

Algorithm 6 shows how to compute reordering scores during decoding in LR-Hiero. There are two inputs: a hypothesis, hyp, and a rule, r, that should be integrated into hyp to generate a new

rules	$r_i.\bar{f}$	O_i	S
	$\{-1\}$		[(-1)-(-1)]
1) $\langle$0 1 2 3 4 X_1/ under such circumstance $X_1\rangle$	$\{0,1,2,3,4\}$	M	[(-1)-4]
2) $\langle 5\ X_1/,\ X_1\rangle$	$\{5\}$	M	[(-1)-5]
3) $\langle 6\ X_1\ 11/\text{when}\ X_1\rangle$	$\{6,11\}$	M	[(-1)-11]
4) $\langle 7\ 8\ X_1/\text{the right of life}\ X_1\rangle$	$\{7,8\}$	D	[(-1)-11]
5) $\langle$9 10/was deprived$\rangle$	$\{9,10\}$	M	[(-1)-11]
6) $\langle 12\ X_1/,\ X_1\rangle$	$\{12\}$	M	[(-1)-12]
7) $\langle 13\ 14\ X_1/\text{it can only}\ X_1\rangle$	$\{13,14\}$	M	[(-1)-14]
8) $\langle 15\ 16 X_1 18/\text{take violence} X_1\rangle$	$\{15,16,18\}$	M	[(-1)-18]
9) $\langle$17/to$\rangle$	$\{17\}$	D	[(-1)-18]

Figure 5.3: Computing correct orientation for each rule during decoding in LR-Hiero for the example in Figure 5.4. **rules**: the rules used in the derivation. $r_i.\bar{f}$: the position of rule's lexical terms in the source sentence; O_i: the identified orientation. S is the recent translated source span (possibly discontinuous). At each step O_i is identified by comparing $r_i.\bar{f}$ to S in the previous step or last translated source phrase $r_{i-1}.\bar{f}$.

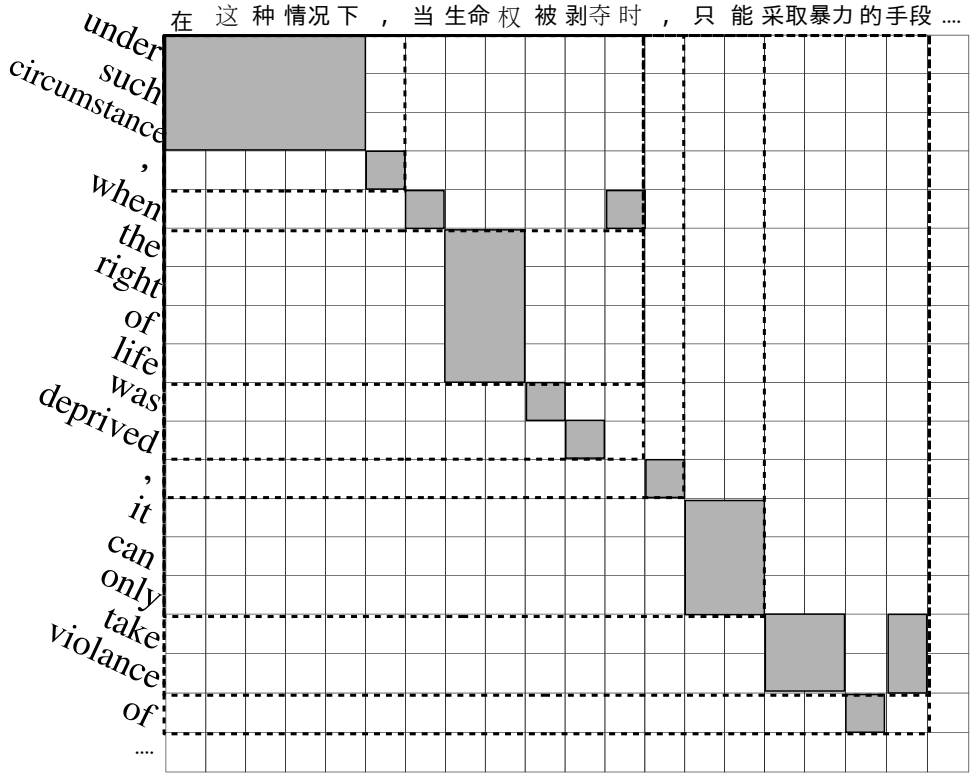


Figure 5.4: An example showing that the shift-reduce algorithm can capture local reorderings like: *the right of life* and *was deprived*.

hypothesis, hyp'[7]. The information required (from the input hypothesis) to compute the reordering scores is stored in a 3-tuple $(S, r, feat)$: S is the stack containing the previously translated source

[7]We skip the complete decoding algorithm here. It is described in Algorithm 3, Chapter 3.

	Corpus	Train/Dev/Test
Cs-En	Europarl(v7) + CzEng(v0.9); News commentary(nc) 2008&2009; nc 2011	7.95M/3000/3003
De-En	Europarl(v7); WMT2006; WMT2006	1.5M/2000/2000
Zh-En	HK + GALE phase-1; MTC part 1&3; MTC part 4	2.3M/1928/919

Table 5.1: Corpus statistics in number of sentences. Tuning and test sets for Chinese-English has 4 references.

span, r is the last rule that has been used to generate hyp, $feat$ is a feature vector. The given rule also is a 2-tuple $(\bar{f}, feat)$: $\bar{f}$ is the simplified source phrase (span on the source sentence) of the rule, $feat$ is a feature vector containing reordering scores for the phrase-pair of this rule. The new hypothesis, hyp' is initialized with an empty stack, S' and a copy of the given hypothesis feature vector, $hyp.feat$. Procedures *active_M*(), *active_S*() and *active_D*() activate monotone, swap and disjoint reordering score respectively (i.e. add the corresponding score from $r.feat$ to the new hypothesis feature vector $hyp'.feat$).

Figure 5.3 illustrates the application of shift-reduce approach to compute orientation for initial decoding steps of a Chinese-English sentence pair shown in Figure 5.4. For the sake of simplicity, we show source words in the rules with the corresponding index in the source sentence. S and $r_i.\bar{f}$ for the initial hypothesis are set to -1, corresponding to the start of sentence symbol, making it easy to compute the correct orientation for spans at the beginning of the input (with index 0).

5.4 Experiments

We evaluate lexicalized reordering model and new glue rules for LR-Hiero decoder on three language pairs: German-English (De-En), Czech-English (Cs-En) and Chinese-English (Zh-En). Table 6.3 shows the corpus statistics for all language pairs.

We train a 5-gram LM on the Gigaword corpus using KenLM [35] and use it for decoding. The weights in the log-linear model are tuned by minimizing BLEU loss through MERT [60] on the dev set for each language pair and the report BLEU scores on the test set.

We use three baselines in our experiments:

- **Hiero:** we used Kriya, an open-source implementation of Hiero in Python, which performs comparably to other open-source Hiero systems [75]. Kriya can obtain statistically equal BLEU scores when compared with Moses [44] for several language pairs [70, 10].
- **phrase-based:** Moses [44] with and without lexicalized reordering features.
- **LR-Hiero:** LR-Hiero decoding with cube pruning[8].

[8]Following Chapter 4 we use queue diversity for Chinese-English task. In these experiments we use $QD = 10$.

Model	Cs-En	De-En	Zh-En
Phrase-based	20.32	24.71	25.68
+ Lexicalized reordering	20.74	**25.99**	26.61
Hiero	20.77	25.72	**27.65**
LR-Hiero	20.52	24.96	25.73
+ Lexicalized reordering	**20.86**	25.44	26.57

Table 5.2: Comparing the translation accuracy in terms of BLEU scores for different baselines and LR-Hiero with lexicalized reordering model.

Model	Cs-En	De-En	Zh-En
Hiero	6279.3	7152.3	6524.7
LR-Hiero + Lexicalized reordering	**2015.1**	**2908.3**	**2225.7**

Table 5.3: Comparing Translation time in terms of average No. of language model queries.

Pop limit for Hiero and LR-Hiero is 500 and beam size for Moses is 1000. Other extraction and decoder settings such as maximum phrase length, etc. were identical across settings. To make the results comparable we use the standard SMT features for log-linear model of all baselines: relative-frequency translation probabilities $p(f|e)$ and $p(e|f)$, lexical translation probabilities $p_l(f|e)$ and $p_l(e|f)$, a language model probability, word count, phrase count and distortion[9]. For Hiero and LR-Hiero, we also use the glue rule count.

Tables 5.2 compares the performance of different translation systems in terms of translation quality (BLEU). In all language pairs lexicalized reordering model improves the translation quality of LR-Hiero. We observe that lexicalized reordering model have similar effect on both phrase-based translation system and LR-Hiero. In Czech-English, LRM gets the best results but not significantly better than LR-Hiero without LRM but for the other language pairs LRM significantly improves the LR-Hiero results (p-value<0.05, evaluated by MultEval [16]).

Tables 5.2 compares the performance of different translation systems in terms of decoding speed. By analogy to parsing algorithm comparisons, we compare the different decoding algorithms with respect to the number of calls made to the language model (LM) since that directly corresponds to the number of hypotheses considered by the decoder. A decoder is more time efficient if it can consider fewer translation hypotheses while maintaining the same BLEU score. The same wrapper is used to query the language model, and we have instrumented the wrapper to count the statistics we need and thus we can say this is a fair comparison. For this experiment we use a sample set of 50 sentences taken from the test sets.

Table 3.3 shows the results in terms of average number of language model queries and times in milliseconds. We can see LR-Hiero+LRM still works 3 times faster than Hiero in terms of number of LM calls

[9]Two distortion features for Hiero and LR-Hiero models, as defined in Section 3.3.

5.5 Summary and Conclusion

In this chapter we proposed a novel lexicalized reordering model for the left-to-right variant of Hiero called LR-Hiero. The model was different from lexicalized reordering models proposed for phrase-based or Hiero-style translation systems. We showed that our lexicalized reordering model significantly improved the translation quality of LR-Hiero on three different language pairs.

Chapter 6

Simultaneous Translation using Hierarchical Phrase-based Translation Models

In this chapter we focus on simultaneous translation using the Hiero translation models. Previous simultaneous translation approaches either use a separate segmenter and a traditional machine translation decoder or just make the decoder decide how to segment and translate by its own. We propose a new approach for automatic simultaneous translation by integrating segmentation and incremental decoding. We investigate different methods to generate training data for the segmentation model. We augment the left-to-right hierarchical translation decoder and integrate it with our segmentation model to build an incremental translation system. This is the first use of hierarchical phrase-based translation models in simultaneous translation.

6.1 Introduction

In simultaneous translation the incoming speech stream is segmented and translated incrementally to reduce the latency. Mainly there are two approaches for simultaneous translation task: *sentence segmentation* and *incremental decoding*, also called *stream decoding*.

In incremental decoding, incoming words are fed into the decoder one-by-one, and the decoder updates its internal state. The decoder is responsible to decide when to begin the translation process and when to output the translation. Incremental decoding algorithms have been proposed for phrase-based [46, 73] translation, and recently for hierarchical phrase-based [25] and syntax-based [65] translation systems.

Actual speech translation systems automatically estimate the sentence boundaries using period estimation methods [69]. Many recent researches in simultaneous translation focus on sentence segmentation, where the input is already splitted into sentences. Hence, the main task is to further split the sentences into shorter subsequences of words which we call *segments*. The sentence segmenta-

tion approach first splits the input sequence into segments. As soon as a segment is recognized, it is given to a decoder to generate the translation for that segment.

Different methods have been proposed for sentence segmentation. Some early works use prosodic boundaries [26, 3], Rangarajan et al. (2013) [69] use classification models to predict punctuation marks, and some research leverages the reordering probabilities of phrases to predict the segment boundaries [27, 93, 79].

More recently methods have been proposed to explicitly optimize translation accuracy in sentence segmentation [64, 77]. These methods provide annotated data which can be used in the training the segmentation models, but they have never been used in end-to-end simultaneous translation.

In this work, we focus on sentence segmentation for simultaneous translation. We model the segmentation task as a classification problem and investigate different methods to provide labeled data for training the segmentation model (Section 6.2). While the input sentence is given (word by word), the segmenter is queried for each word to determine whether a segment is completed or not.

Hierarchical phrase-based translation model is a prominent approach for SMT, usually comparable to or better than conventional phrase-based systems. However, previous translation services proposed for real-time translation environments, are mainly phrase-based [26, 73, 3, 93, 64]. Since a phrase-based decoder generates translations in a left-to-right manner which is more suited than the CKY based decoding algorithm used in Hiero decoders which requires the entire input sentence before generating the translation. We propose to use LR-Hiero for simultaneous translation which uses hierarchical phrase-based translation model while generates the translation in left-to-right manner.

Segmentation-based simultaneous translation approaches typically use a traditional decoder to translate each input segment individually. We adapt LR-Hiero decoder to incrementally translate the input sentence (stream of words) given the segment boundaries (Section 6.3). We evaluate our incremental translation system on the speech translation of TED talks on two language pairs: English-French and English-German.

6.2 Sentence Segmentation

The segmentation task is usually modeled as a binary classifier which is called for each input word to determine if it is a segment boundary or not. To train the segmentation classifier we need some training data in the form of sentences with labeled words showing if a word is a segment boundary or not. For each sentence $\mathbf{f} = \langle f_1 \dots f_J \rangle$ different possible segmentations exist which grow exponentially with the length of the sentence. Finding the best segmentation can be quite difficult, as it requires a brute-force search over all possible segmentations which is intractable.

Different heuristics have been proposed to efficiently solve this problem. We will describe two recent approaches: *alignment-based* and *translation-based* heuristics. These heuristics use parallel data on the source and target languages of the simultaneous translation task to effectively label the training data for the segmentation model. We define $\mathcal{C} = \langle F, E \rangle$ as a parallel corpus of source and target sentences used to extract traning data.

6.2.1 Alignment-based Heuristic

The idea behind alignment-based segmenters is to split the input sentence into segments which can be translated to the target language monotonically [93, 79]. To achieve this goal, the segmentation task is simplified to find segmentations of the source sentences where reordering just occurs inside segments but not across segments.

To find such segmentations we can leverage word alignment models. Given the word alignment $\mathbf{a} = \langle a_1 \dots a_J \rangle$ for a sentence pair $\langle \mathbf{f}, \mathbf{e} \rangle$, we can segment the source sentence $\mathbf{f} = \langle f_1 \dots f_J \rangle$, into a set of K phrases $\mathbf{s}$:

$$s_k = \langle j_{k-1}, j_k \rangle \qquad \forall k = 1 \dots K, j_0 = 1 \tag{6.1}$$

where j_k is the end position of source phrase or segment s_k. To restrict the reorderings inside the segments, we should extract segments where $a_{j_{k-1}} < a_{j_k}$ for all k. This segmentation results in a phrase alignment for the sentence pair $\langle \mathbf{f}, \mathbf{e} \rangle$ called *monotonic phrase alignment*. Figure 6.1 shows word alignment matrix and monotonic phrase alignment for an English-German sentence pair.

Monotonic phrase alignment for a sentence pair can be found in linear time, given the word alignment. Experimentally, it has been shown that translation quality improves significantly with longer phrases [45]. Therefore to avoid too short segments which results in word-to-word translation, the segmentation algorithm is forced to extract segments of length more than μ[1].

We can train a word alignment model over a parallel corpus containing the parallel data of the translation task and $\mathcal{C}$. It provides us the oracle word alignment for $\mathcal{C}$ which can be used to extract labeled data for the segmentation classifier.

6.2.2 Translation-based Heuristic

The translation-based segmentation strategy [64, 77] focuses on obtaining the segmentation points which are the least harming to the translation accuracy. This heuristic performs the segmentation using an iterative greedy approach which tries to find the most harmless segmentation point to the translation accuracy at each iteration and add it to the previously discovered ones. Segmentation points are described using a set of features. Different kinds of features can be used such as bigram POS tags, lexical terms, parsing related features and etc. Each feature is used as a metric to recognize segmentation point in a given input sentence. For instance, suppose we find a feature which is a bigram POS tag: *NNS-IN*. Using this feature we can find a segmentation point at index 10 in the English-German sentence in Figure 6.1 (the segmentation point is surrounded by two words *engineers* and *in* with POS tags NNS and IN). This approach is applied on a given parallel corpus and outputs an optimal set of features, called FVS, based on the input corpus[2].

[1] μ is usually set to 4 [93, 79].

[2] The output FVS results in the best segmentation strategy (a set of segmentation points) for the input parallel corpus in terms of translation quality.

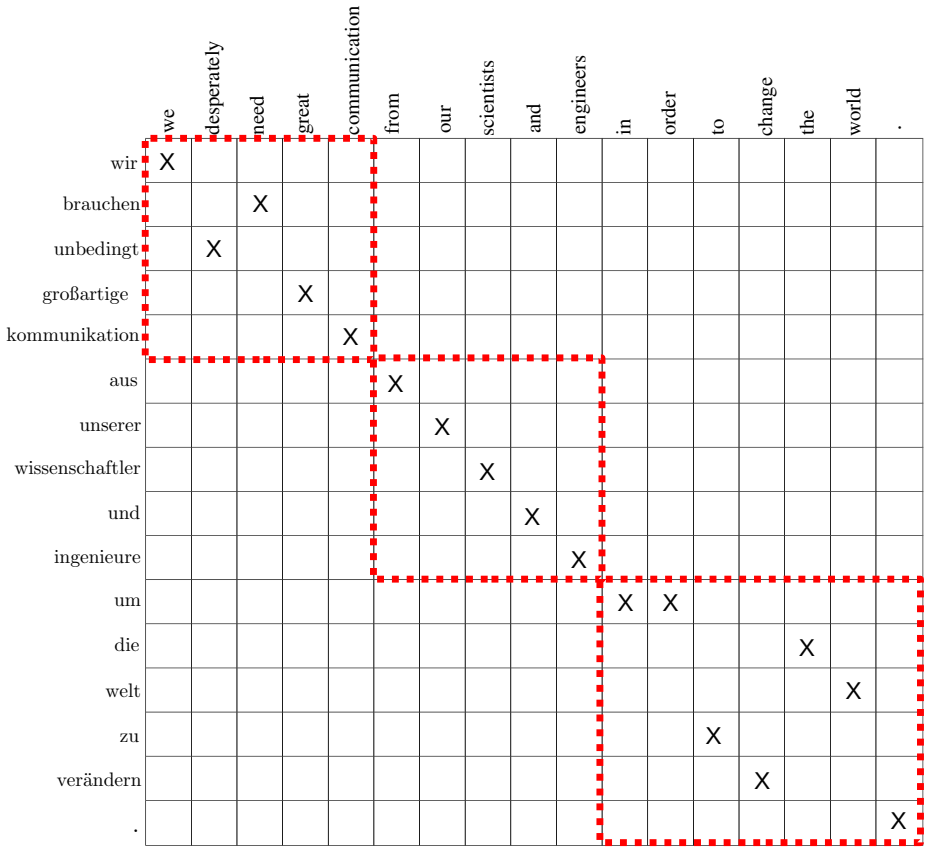


Figure 6.1: Word alignment matrix for an English-German sentence pair. Monotone phrases are shown in dashed lines.

Given a parallel corpus $\mathcal{C} = \{\mathcal{F}, \mathcal{E}\}$ and a given number of segmentation points, K, the translation-based segmentation heuristic first extracts the FVSs over the corpus and excludes the ones occurring more than K times. Take the feature frequencies $\langle c_1 \dots c_m \rangle$ to be the related occurring frequencies of the m remaining features. As an example, if $c_2 = 7$ it means the second feature has happened 7 times over the corpus and if it belongs to our selected FVS to build the segmentation model, we will put 7 segmentation points in the source corpus wherever this second feature appears in the source sentences. The translation-based segmentation heuristic tries to find a feature set s containing $l(\leq m)$ features according to the least harmful segmentation criterion for the translation accuracy where Equation (6.2) holds.

$$\sum_{i=1}^{l} c_{s_i} = K \tag{6.2}$$

The translation accuracy in the previous works [64, 77] was defined as the summation over the sentence-level BLEU score [49] of the translations of segmented sentences. Our primarily experiments show that the sentence-level BLEU tends to oversegment some sentences in the corpus and leave the other sentences untouched. To overcome this issue, we propose to use corpus-level BLEU

to measure translation accuracy. The corpus-level BLEU gives a general view over the corpus therefore it alleviate the tendency to localize the segmentation.

Oda et al. (2014) [64] propose a dynamic programming algorithm to find an optimal feature set s. They initialize the feature set as empty ($s = \{\}$), then greedily choose a feature which adding it to the corpus causes the least translation loss on the parallel corpus (in terms of average sentence level-BLEU). Once a feature has been chosen, all the points exhibiting that feature are segmented at the same time. They take advantage of dynamic programming (DP) to implement their approach. DP is used to build larger sets of segmentation points from smaller sets. This approach is called Greedy-DP (GDP).

However, this approach does not model the trade-off between accuracy and latency. This trade-off is crucial in designing a simultaneous translation system. To model this trade-off we use the *Pareto Optimality* approach proposed by [77]. This approach finds all Pareto-Optimal points (i.e. those segmentation strategies that are equally optimal, in terms of latency and translation quality which heres is BLEU) to generate training data for the segmentation model. In this approach, the algorithm iteratively goes over the corpus and examines the available features by computing the difference of translation accuracy (Δ) before and after applying each available feature to the source corpus. The features which cause least translation loss (the smallest Δ) are selected as candidate points [64]. Among them, the feature which causes the least latency or the highest throughput which we define in Equation 6.3 is selected to be added to the feature set s.

$$\frac{\#Translated\ Segments}{Total\ Translation\ Time} \tag{6.3}$$

Shavarani et al. (2015) [77] use sentence-level BLEU score to compute translation accuracy and still suffer from tendency to localize the segmentation to a few sentences in the data set. We propose to use corpus-level BLEU within the Pareto optimality approach.

6.2.3 Segmentation Model

Given a set of sentences along with the gold segmentations, we can prepare the training data for the segmentation model. For each segment in the gold segmentation we create a positive training example corresponding to the whole segment and a set of negative examples corresponding to each smaller segment. For example for a segment $\langle i, j\rangle$, the positive example is (i, j), and negative examples are $[(i, i+1), (i, i+2), \ldots (i, j-1)]$.

Having the training data, we train a binary classifier (a log-linear model) based on different feature sets. Basic features, used in [93], are: the last word of the segment (candidate segment boundary), the position of the boundary in the sentence, and the candidate segment length (Set1). We use this model as our baseline. We propose to use additional features containing different information which can be partitioned into:

- **Part of Speech tags**: The first group uses Part Of Speech (POS) tags of the candidate segment as features. We considered the last three POS tags in a segment and also bigrams and trigrams

Set1: Word, Position, Length	"engineers", 9, 5
Set2: + POS tags	[NNS],[CC-NNS],[NN-CC-NNS]
Set3: + Cross POS tag	[NNS-IN]
Set4: + Reordering	0.8904, 0.6

Table 6.1: Feature sets and an example (for segment "from our scientist and engineers" in the English sentence in Figure 6.1)

of the POS tags for each segment (Set2). In addition to these features we consider POS bigram surrounding the segment boundary (Set3) and two features to model the reordering (Set4).

- **Decoder Status**: We hypothesize that the previous state of the decoder might have useful information. This was the motivation for the next group of features. These features can be seen as feedback from decoder about its output. Feedback from the decoder includes: language model score (lm), translation probabilities $p(e|f)$ and $p(f|e)$ and two lexically-weighted translation probabilities (tm_0,tm_1,tm_2,tm_3), and the model score (c). We normalize these values and use them as features in segmentation model.

- **Reordering Features**: The lexicalized reordering model [44] of phrase-based translation system determines the orientation of phrases with respect to the previous phrase, monotone (M), swap (S) and discontinuous (D). We expect the segments to be monotonically ordered. For each segment, we define two reordering features corresponding to the monotone feature orientation of the first and last phrases of the segment[3]. To compute the feature values we use lexicalized reordering model of Moses [44] for monotone orientation of both left-to-right and right-to-left (corresponding to the first and last phrases of the segment).

Table 6.1 shows an example for POS-based and reordering-based features defined on the second segment ("from our scientist and engineers") of the stream "we desperately need great communication from ..." (see Figure 6.1).

6.3 Integrated Segmentation and Decoding

As we discussed before, in sentence segmentation approaches, the input stream is segmented and for each segment the machine translation decoder is called to translate the obtained segments individually. In this approach the decoder treats each segment as an independent input, while we are translating the input stream. We propose to integrate the segmentation model and decoder. This approach can be also considered as a stream decoding method which the decoder exploit other resources beyond just decoding cues.

Hiero models encode the translation correspondences in *hierarchical* phrases, unlike the phrase-based models that use contiguous translation phrases. The notion of hierarchy allows the Hiero

[3]We consider the longest phrase which is available in the phrase-table.

models to capture long-distance reordering between source and target languages unlike phrase-based models. Additionally they also model discontiguous translations, e.g. translating the English word *not* as *ne* ___ *pas* in French (with an appropriate verb form inserted between *ne* and *pas*). These properties make Hiero models more appropriate for some language pairs than phrase-based models [52, 61, 63].

Hiero uses a lexicalized synchronous context-free grammar (SCFG) extracted from word and phrase alignments of a bitext. Typically, Hiero uses a CKY-style decoding algorithm with time complexity $O(n^3)$ where the source input has n words.

Previous translation services proposed for real-time translation environments, are mainly phrase-based [26, 73, 3, 93, 64]. Since a phrase-based decoder generates translations in a left-to-right manner, it is more suited than the CKY based decoding which requires the entire input sentence before generating the translation.

We propose to use left-to-right hierarchical phrase-based translation in our simultaneous translation framework. It has been shown that left-to-right hierarchical (LR-Hiero) decoder can translate using Hiero translation model much faster than CKY Hiero decoder [80]. In addition, it generates the translation in left-to-right manner. These properties make it a suitable decoder for simultaneous translation [79]. We augment LR-Hiero decoder to incrementally translate the input and integrate it with the segmentation model.

6.3.1 LR-Hiero Decoder

LR-Hiero uses a constrained lexicalized SCFG usually called GNF grammar: $X \rightarrow \langle \gamma, \bar{b}\, \beta \rangle$, where X is a non-terminal, γ is a string of non-terminal and terminal symbols, $\bar{b}$ is a string of terminal symbols and β is a possibly empty sequence of non-terminals. Using GNF rules ensures that in derivations the target side is always generated from left to right. The rules are obtained from a word and phrase aligned bitext by replacing the smaller source-target phrase pair within a larger phrase pair with some non-terminal[4].

The decoding algorithm in LR-Hiero follows an Earley-style search [22] on the source side. The dot jumps around on the source side of the rules based on the order of nonterminals on the target side. Thus the target side derivation is strictly developed in left to right order. The search algorithm is integrated with beam search or cube pruning to find the k-best translations.

Algorithm 7 shows the pseudocode of incremental decoder for LR-Hiero. This a modified version of LR-Hiero decoder described in Chapter 3 (the modified lines have been highlighted). Each partial hypothesis h is a 4-tuple (h_t, h_s, h_{cov}, h_c) : a translation prefix h_t, a (LIFO-ordered) *list* h_s of uncovered spans, source words coverage set h_{cov} and the hypothesis cost h_c which includes future cost and a model score computed based on feature values (using a log-linear model).

The translation prefix for the initial hypothesis is the input string *history*. In the standard LR-Hiero decoder, history is a null with just a sentence-initial marker ($history = \langle s \rangle$). The initial

[4]We discussed the rule extraction algorithm in Chapter 2.

Algorithm 7 LR-Hiero Incremental Decoder

Input: $\mathbf{f} = f_0 \dots f_n, history$
$\mathcal{F} =$ FutureCost($\mathbf{f}$) (Precompute future cost for all source spans)
$S_0 = \{\}$ (Create empty initial stack)
$h_0 = (history, [[0, n]], \emptyset, \mathcal{F}_{[0,n]})$ (Initial hypothesis 4-tuple)
Add h_0 to S_0 (Push initial hyp into first Stack)
for $i = 1, \dots, n$ **do**
  $cubeList = \{\}$ (MRL is max rule length)
  for $p = max(i - \text{MRL}, 0), \dots, i - 1$ **do**
    $\{G\}$ = Grouped(S_p) (based on the first uncovered span)
    for $g \in \{G\}$ **do**
      $[u, v] = g_{span}$
      R = GetSpanRules($[u, v]$) (Extract rules matching span $[u, v]$)
      for $R_s \in R$ **do**
        $cube = [g_{hyps}, R_s]$
        Add $cube$ to $cubeList$
  S_i = Merge($cubeList, \mathcal{F}$) (Create stack S_i and add new hypotheses to it)
$h\_best = argmin_{h \in S_n} h_c$
return h_best_t

hypothesis is the history and the list h_s containing one span indicating the whole input, $[0, n]$. The hypotheses are stored in stacks $S_0, \dots, S_n$, where S_p contains hypotheses covering p source words, just like in stack decoding for phrase-based SMT [45]. Decoding process finishes when stack S_n has been filled.

6.3.2 Incremental Translation

In our simultaneous translation framework, we integrate LR-Hiero with the segmentation model. This framework is shown in Algorithm 8. The input is a stream of words ($\mathbf{f} = f_0 f_1 \dots$) which is fed to the translation system word by word. *buffer* always contains the sequence of yet-to-be-translated words (initially empty) and *history* contains the translation words previously emitted by the translation system, initially set to null string (a sentence initial marker).

For each new input word, f_i, the translation system queries the segmentation model. The content of *buffer* and f_i are given to the *Segmenter* to determine whether the sequence of words in *buffer* is a valid segment for translation or not.

Once a segment is recognized, the segment and *history* are passed to the decoder (Algorithm 7). The decoder translates the given source segment, and produces the translation output for that segment given the *history* as previously translated words. After emitting the translation to the output, *buffer* is initialized with the last input word, f_i which is not translated yet and *history* is updated with the last emitted translation. This is a simple, yet effective way to define *history*.

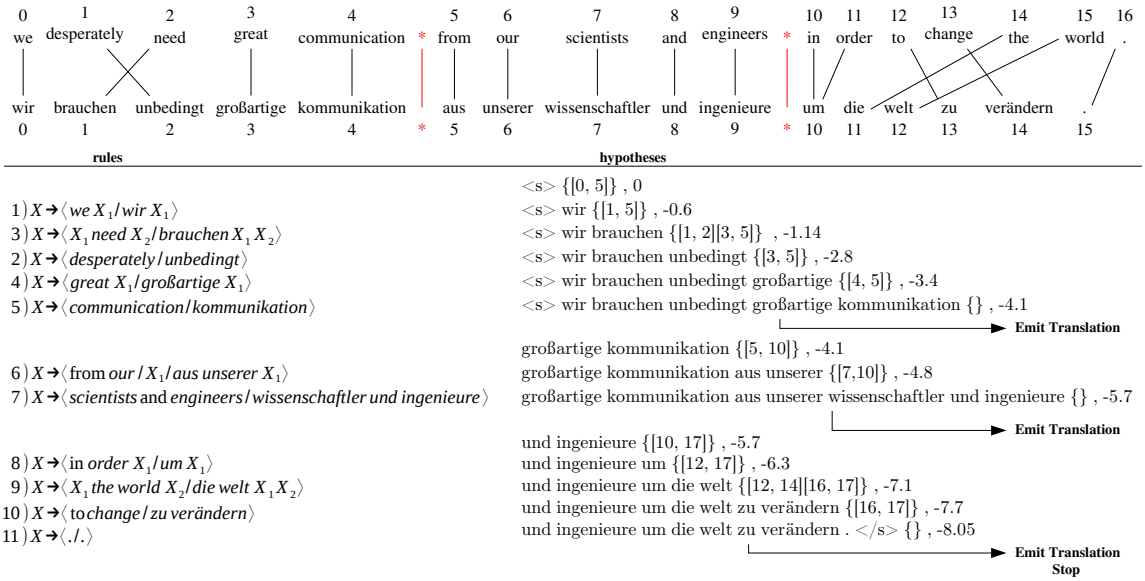


Figure 6.2: Simultaneous translation for an English-German sentence using LR-Hiero. The word alignment is shown on the top. The segmentation points are shown by red stars. On the bottom, different steps of the decoder are shown. The left side shows the rules used in the derivation.The hypotheses column shows 4-tuple partial hypotheses: the translation prefix, h_t, the ordered list of yet-to-be-covered spans, h_s, source word coverage vector, h_{cov} and cost h_c.

In this approach, the translation output is updated over time by adding the translation of the next input segments and the decoder does not change the output which is already produced and emitted. Therefore, it is an appropriate approach for speech translation.

Figure 6.2 illustrates different steps of translating the English-German sentence pair in Figure 6.1 using the simultaneous translation framework with LR-Hiero as translation system. The first 5 words are recognized as a segment and given to the decoder along with $\langle s \rangle$ as history. Then the translation is emitted and the status of the decoder (hypotheses) is preserved for the translation of next segment. The process is repeated until the end of sentence is detected.

6.4 Experimental Results

Following the *International Workshop on Spoken Language Translation* (IWSLT) shared task, we evaluate our approach on the speech translation of TED talks for English-French and English-German [12].

6.4.1 System Setup

For English-French, we use development (dev2010) and test data (tst2010) of IWSLT 2010 to tune and evaluate the translation system. We use the parallel text provided as training data of IWSLT

Algorithm 8 Simultaneous Translation

```
Input stream: f = f_0 f_1 ...
buffer = []
history = ⟨s⟩
while f_i ≠ ⟨/s⟩ do
    if Segmenter(buffer, f_i) == True then
        trans = Decoder(buffer, history)
        print trans
        buffer = [f_i]
        history = trans
    else
        Add f_i to buffer                    (Add the current word to the end of buffer)
trans = Decoder(buffer, history)             (Translate the last segment)
print trans
```

2011 and Europarl (v7) as the training data for our translation task (about 2M sentence pairs). The training data from IWSLT 2011 is used as the training set for alignment-based segmentation model (90% as training and 10% as test set). We use a 4-gram language model trained on the French corpus of WMT2011 (Europarl, News Commentary and UN documents) using KenLM [35].

For English-German, we use the parallel text provided as training data of IWSLT 2013 and about one million sentence pairs of Europarl (v7), selected randomly to train the translation system. We use development set 2010 and 2012 and test set 2010 of IWSLT shared task as development set to tune the translation system (LR-Hiero) and test set of IWSLT 2013 is used for final evaluation. The training data from IWSLT 2013 is used as the training set for alignment-based segmenter (90% as training and 10% as test set). We use a 5-gram LM trained on the monolingual German data provided by WMT 2013 shared task using KenLM [35].

We use pop limit 500, maximum source rule length 7 and at most 2 non-terminals in LR-Hiero. The standard feature set of LR-Hiero [80] (c.f. Section 3.3) is used in a discriminative log-linear model. We use GNF rule extraction algorithm described in Chapter 2 and include loose phrase-pairs as terminal rules. The weights in the log-linear model are tuned by minimizing BLEU loss through MERT [60] on the dev set for each language pair. In these experiments, we use the reference transcript of the utterance for dev and test sets. LR-Hiero is trained once for each language pair and used in all experiments.

We use the Stanford POS-Tagger [85] to obtain the POS tags used in training the segmenter.

6.4.2 Evaluating Features for Segmentation Modeling

We conduct some experiments to evaluate decoder-based and POS-based features for segmentation model. For the initial evaluation on features we use a segmentation model trained labeled data provided using alignment-based heuristic on English-French [5].

[5]The experiments in this section are published in [79]. The experiments on decoder-based features are expensive, therefore we limit these initial experiments to English-French.

features	P	R	F1
Basic	0.77	**0.86**	0.81
+ POS	0.7924	**0.84151**	**0.8162**
+ Decoder (all)	0.7971	0.8295	0.8129
+ Decoder (lm,tm_0,c)	**0.8085**	0.8137	0.8110
+ POS + Decoder (lm,tm_0,c)	0.8041	0.8284	**0.8161**
+ POS + Decoder (all)	0.8084	0.8137	0.8110

Table 6.2: Results of segmentation model trained using different decoder-based and POS-based features on French-English.

Table 6.2 shows the results of the segmentation model on test set (10% of IWSLT 2011 training data). As we can see both POS-based and decoder based features improve the baseline in terms of precision and F1 measure. But adding decoder-based features decrease the recall. Based on this experiment we decide to use Basic+POS features as in other experiments. This set of features outperforms all other feature sets, while it is faster, than others during decoding [6].

6.4.3 Different Labeled Data for Segmentation Modeling

In Section 6.2 we discussed two heuristics: translation-based and alignment-based, to provide training data for segmentation model. We conduct some experiments on English-German to compare these heuristics. We use Dev 2010 and 2012 and Test 2010 from IWSLT to provide the training date for segmentation model. Table 3 shows the statistics of data used in our experiments.

Task	Sentences	Tokens
MT Train	1033491	27948039
Tune	3669	74883
Seg. model Train	3669	74883
Test	1025	22026

Table 6.3: Corpus statistics in number of sentences and tokens (source side) for English-German.

In the translation-based heuristic we need the translation of all possible segments. We use the data by Shavarani et al. [77] which contains translation for all segments (translated by Moses) stored in a lattice. To evaluate segment translation quality, we use corpus level BLEU [66]. To compute the latency model, we use the *sayit*[7] script by Hal Daumé which receives the content of the segment (in text format) and estimates the time it takes from a human to say the segment in some languages: English (US), German, French, Italian, Spanish, and Japanese. We set the α regularizer coefficient to 0.5 because this value avoids selecting features with extremely high or low frequency.

For the alignment-based heuristic, we concatenate the training data of segmentation model (3669 sentence pairs) to the training data of the translation system and run GIZA++ to get the

[6]The models trained using decoder-based features need the feedback from decoder, therefore it requires to run the decoder on the corresponding input segment, given each input word.

[7]`http://www.umiacs.umd.edu/hal/sayit.py`

Labeled data Heuristic	Features	P	R	F1
Translation-based	Set1	81.38	52.56	63.87
	Set2	82.03	53.90	65.06
	Set3	97.18	69.89	81.31
	Set4	93.41	64.14	76.06
Alignment-based	Set1	71.78	62.88	67.04
	Set2	74.58	56.46	64.27
	Set3	79.78	58.39	67.43
	Set4	79.09	59.62	67.97

Table 6.4: Results of segmentation model trained on different labeled data using various feature sets on German-English.

Labeled data	Features	BLEU	Latency	No. segments
Translation-based	Set3	**20.86**	**0.311**	3313
Alignment-based	Set3	20.60	0.540	2648
	Set4	20.62	0.524	2654
Prosodic heuristic	-	**20.88**	0.514	2709
Regular translation	-	21.04	6.353	1025

Table 6.5: Results of simultaneous translation using different models for segmentation on test data (German-English).

word alignment. Then the heuristic discussed in Section 6.2 is used to extract segments. To have a fair comparison, we choose $\mu = 5$ in translation-based heuristic which provides comparable number of segments on the training data.

Table 6.4 shows the results of different segmentation models in terms of precision, recall and F1 measure on a heldout set[8] (5000 sentences randomly selected from training data of IWSLT 2013). This table compares the performance of different labeled data and feature sets. Based on results of Section 6.4.2, we choose POS-based features (Set2) and here compare it to new feature sets (Set3 and Set4 in Section 6.2.

Table 6.5 shows the results of the end-to-end simultaneous translation on test data for German-English. Base on results of previous experiments on different feature sets, we choose Set3 as the best feature set for translation-based model while on alignment-based model Set3 and Set4 have comparable results in Table 6.4, therefore we try the translation system using both models.

We implemented a heuristic segmenter based on Rangarajan et al. [69] which segments on surface clues such as punctuation marks. These segments reflect the idea of segmentation on silence frames of around 100ms in the ASR output used in [3]. The results of this heuristic (*prosodic*) has been shown in the forth row of Table 6.5.

[8]We use the set of POS tags obtained by translation-based heuristic to create the gold reference for this experiment.

In Table 6.5, latency is calculated as the total time taken to translate the whole sentence divided by the number of segments. This is the result of taking the average over 5 different runs for 50 sentences randomly selected from test set.

6.5 Discussion

Early work on speech translation uses prosodic pauses detected in speech as segmentation boundaries [26, 3]. Segmentation methods applied on the transcribed text can be divided to two categories: heuristic methods which use linguistic cues, like conjunctions, commas, etc. [69]; and statistical methods which train a classifier to predict the segmentation boundaries. Some early methods use prosodic and lexical cues as features to predict soft boundaries [53]; while most recent methods rely on word alignment information to identify contiguous blocks of text that do not contain alignments to words outside them [93, 79]. In addition to these segmentation approaches which are applied before calling the translation decoder, there is another strategy which perform the segmentation during decoding which is usually called stream or incremental decoding. Various incremental decoding approaches have been proposed for phrase-based [46, 73], hierarchical phrase-based [79, 25], and recently syntax-based [65] translation systems. In most incremental decoding algorithms, the decoder waits for more input and commits the translation when the current utterance is enough to generate a fluent translation. Oda et al. (2015) [65] propose a method to predict the future syntactic constituents and use it in generating complete parse trees. It helps to find a good point to commit the translation.

Recently some research focused on language pairs with divergent word order. Grissom et al. (2014) [33] predict sentence-final verbs using reinforcement learning. It greatly affects the delay in tasks like German-English which translates from a verb-final language to a verb-initial language. He et al. (2015) [34] design syntactic transformations to rewrite batch translations into more monotonic translations.

Some research has been conducted on human simultaneous interpretation to determine the effect of the latency and accuracy metrics on the human evaluation of the output of simultaneous translation. The results indicate that latency is not as important as accuracy [55]. This implies that we need algorithms that can make a careful choice between different segmentation decisions of the same latency to produce translations with the best translation quality possible (for that latency) which we have done in this chapter.

6.6 Summary and Conclusion

In this chapter we proposed the use of Hiero translation model for simultaneous translation for the first time. We integrate a sentence segmentation model into LR-Hiero translation system to create a new framework for simultaneous translation. We propose to use different heuristics to create training data for the sentence segmentation model based on translation accuracy and reordering

(word alignment). We investigate different sets of features for segmentation model and evaluate them based on different measures.

We show that our framework is able to produce fast yet accurate translation for simultaneous translation. Our framework works around 23 times faster than a regular translation system, while preserving a comparable translation quality.

As future work we are interested to compare our translation framework with other simultaneous MT systems and also apply it on complex word reordering language pairs like Chinese-English. We would like to improve the segmentation model and use it in regular translation tasks, we hope it results in faster translation systems.

Chapter 7

Conclusions and Future Directions

In this thesis we presented a new hierarchical phrase based translation system: left-to-right hierarchical phrase-based translation (LR-Hiero). LR-Hiero uses a constrained form of lexicalized SCFG rules to encode translation, where the target-side is constrained to be prefix-lexicalized, GNF grammar which is a small subset of Hiero SCFG grammar. LR-Hiero applies a left-to-right decoding algorithm that generates the target side in left-to-right manner which simplifies the language model integration and leads to less language model calls during decoding. Using a smaller but informative grammar and applying LR-decoding results in a faster translation system. We showed that LR-Hiero a viable alternative for Hiero.

In Chapter 2, we proposed a novel dynamic programming algorithm for rule extraction phase of LR-Hiero. Unlike traditional Hiero rule extraction which performs a brute-force search, LR-Hiero rule extraction is linear in the number of rules.

In Chapter 3 we proposed an augmented version of LR-decoding algorithm which was first proposed by Watanabe et al. [90]. The original LR-decoding does not perform to the same level of current state-of-the-art translation systems, both on time and translation quality. Our modified LR-decoding algorithm addressed these issues. We showed that this algorithm performs around four times faster than CKY decoding algorithm. We further extende the LR-decoding algorithm to capture all hierarchical phrasal alignments that are reachable in CKY-style decoding algorithms (Chapter 4).

We also introduced a lexicalized reordering model to LR-Hiero, and showed that this model significantly improved the translation quality (Chapter 5).

Finally we applied hierarchical translation model to the task of simultaneous translation for the first attempt using LR-Hiero. We obtained a very fast simultaneous translation system which is 23 times faster than a regular translation system (in terms of latency) while it maintains a comparable translation quality (Chapter 6).

7.1 Future Directions

The following directions can be explored to extend the L-Hiero frame work or to apply it in other tasks such as simultaneous translation:

- **Grammar Induction**: currently we use an approach similar to the standard Hiero to compute rule parameters. Some heuristics are used to estimate rule counts. Then, the conditional translation probabilities are computed by relative frequency estimation of the counts. It has been shown that this approach leads to poor estimation for rule parameters [6, 5, 47, 72]. This issue becomes more severe in LR-Hiero, since it uses rules extracted from phrase pairs (production rules) to build glue rules and use the same parameters for the generated glue rules. Levenberg et al. (2012) [47] propose an approach to learn grammars with unrestricted number of non-terminals but do not use the grammar directly in a decoder for translation. The obtained SCFG rules are just used for the purpose of obtaining the phrase alignments.

  For future exploration, we intend to extend the non-parametric Bayes model proposed in [47], by expanding the sampler to explore the space of word alignment broadly and more quickly. We hope it would result in a more expressive grammar with better parameter estimation. As LR-Hiero can efficiently use grammars with more than two non-terminals and preserve the quadratic time complexity, we expect using the induced grammar in decoding further improves translation output.

- **Extend Simultaneous Translation Framework**: In Chapter 6 we proposed a sentence segmentation approach and integrate it into the decoder of LR-Hiero for the task of simultaneous translation. We expanded the translation based heuristic by [64, 77] to extract labeled data to train the sentence segmentation model. In this approach based on the segment length, different set of segments and consequently different training data is created. In Chapter 6 we fixed the segment length to 5 to extract training data and train one segmentation model.

  As a future direction, we plan to extend the simultaneous translation framework. Corresponding to each segment length, extract a training data and train a separate segmentation model. By tuning a weighted combination of these segmentation models, we reach more flexibility in the final sentence segmenter. We hope it results in more accurate sentence segmentation model.

- **Neural Machine Translation**: Most of the current NMT models use an *encoder/decoder* approach, where the encoder reads and encodes the given source sentence into a fixed-length vector, which is then fed to the decoder to produce the output translation. Different neural network models, e.g. convolutional, long short term memory (LSTM), etc. have been used as encoder/decoder in NMT systems [41, 2, 83]. But all models suffer on long sentences. I believe it is due to the fact that the fixed length context vector is unable to maintain the whole translation and reordering information of source sentence.

Left-to-right translation generation is a promising idea for any translation system. This idea can be applied to NMT systems so that the network can model the reordering information between source and target languages more explicitly. To achieve this, we can leverage syntactic parse tree on source (and target) language. Socher et al. (2011) [71] use recursive auto-encoder which is built based on parse tree for paraphrase detection. We can adapt this idea to NMT.

Bibliography

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

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. Real-time incremental speech-to-speech translation of dialogs. In *Proc. of NAACL HLT 2012*, pages 437–445, 2012.

[4] Alexandra Birch and Miles Osborne. Reordering Metrics for MT. In *Proceedings of the Association for Computational Linguistics*, Portland, Oregon, USA, 2011. Association for Computational Linguistics.

[5] Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. A gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of Association of Computational Linguistics-09*, pages 782–790. Association for Computational Linguistics, 2009.

[6] Phil Blunsom, Trevor Cohn, and Miles Osborne. Bayesian synchronous grammar induction. In *Proceedings of Neural Information Processing Systems-08*, 2008.

[7] Fabienne Braune, Anita Gojun, and Alexander Fraser. Long distance reordering during search for hierarchical phrase-based smt. In *Proc. of EAMT 2012*, 2012.

[8] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. volume 16, pages 79–85, June 1990.

[9] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, June 1993.

[10] Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June 2012. Association for Computational Linguistics.

[11] Hailong Cao, Dongdong Zhang, Mu Li, Ming Zhou, and Tiejun Zhao. A lexicalized reordering model for hierarchical phrase-based translation 1. In *Proceedings of the 21st International Conference on Computational Linguistics*. Association for Computational Linguistics, 2014.

[12] Mauro Cettolo, Christian Girardi, and Marcello Federico. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy, May 2012.

[13] Colin Cherry. Improved reordering for phrase-based translation using sparse features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, USA, 2013.

[14] David Chiang. A hierarchical phrase-based model for statistical machine translation. In *In Proc. of ACL*, pages 263–270, 2005.

[15] David Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33, 2007.

[16] Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. Better hypothesis testing for statistical machine translation: controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 176–181, 2011.

[17] John Cocke. *Programming Languages and Their Compilers: Preliminary Notes*. Courant Institute of Mathematical Sciences, New York University, 1969.

[18] Adrià de Gispert, Juan Pino, and William Byrne. Hierarchical phrase-based translation grammars extracted from alignment posterior probabilities. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 545–554. Association for Computational Linguistics, 2010.

[19] Steve DeNeefe and Kevin Knight. Synchronous tree adjoining machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP 2009, pages 727–736, 2009.

[20] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M. Schwartz, and John Makhoul. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1370–1380, 2014.

[21] Yuan Ding and Martha Palmer. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL 2005, pages 541–548, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[22] Jay Earley. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, February 1970.

[23] Jason Eisner. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 2*, ACL 2003, pages 205–208, 2003.

[24] Yang Feng, Yang Liu, Qun Liu, and Trevor Cohn. Left-to-right tree-to-string decoding with prediction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1191–1200, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[25] Andrew Finch, Xiaolin Wang, Masao Utiyama, and Eiichiro Sumita. Hierarchical phrase-based stream decoding. In *Proc. of EMNLP*, Lisbon, Portugal, September 2015.

[26] Christian Fügen, Alex Waibel, and Muntsin Kolss. Simultaneous translation of lectures and speeches. *Machine Translation*, 21(4):209–252, 2007.

[27] Tomoki Fujita, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Simple, lexicalized choice of translation timing for simultaneous speech translation. In *INTERSPEECH*, pages 3487–3491, 2013.

[28] Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics*, pages 961–968. Association for Computational Linguistics, 2006.

[29] Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 273–280, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

[30] Michel Galley and Christopher D. Manning. A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP 2008, pages 848–856, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[31] Michel Galley and Christopher D. Manning. Accurate non-hierarchical phrase-based translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 966–974, Los Angeles, California, June 2010. Association for Computational Linguistics.

[32] Daniel Gildea. Loosely tree-based alignment for machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 80–87, 2003.

[33] Alvin C. Grissom II, Jordan Boyd-Graber, He He, John Morgan, and Hal Daume III. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *EMNLP*, pages 1342–1352, 2014.

[34] He He, Alvin Grissom II, John Morgan, Jordan Boyd-Graber, and Hal Daumé III. Syntax-based rewriting for simultaneous machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015.

[35] Kenneth Heafield. KenLM: Faster and smaller language model queries. In *In Proc. of the Sixth Workshop on Statistical Machine Translation*, 2011.

[36] Kenneth Heafield, Hieu Hoang, Philipp Koehn, Tetsuo Kiso, and Marcello Federico. Left language model state for syntactic machine translation. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 183–190, San Francisco, California, USA, 12 2011.

[37] Kenneth Heafield, Philipp Koehn, and Alon Lavie. Grouping language model boundary words to speed K-Best extraction from hypergraphs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, USA, 6 2013.

[38] Liang Huang. Statistical syntax-directed translation with extended domain of locality. In *Proc. of AMTA 2006*, pages 66–73, 2006.

[39] Liang Huang and David Chiang. Forest rescoring: Faster decoding with integrated language models. In *In ACL 2007*, 2007.

[40] Liang Huang and Haitao Mi. Efficient incremental decoding for tree-to-string translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 273–283, Cambridge, MA, October 2010.

[41] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. Seattle, October 2013. Association for Computational Linguistics.

[42] Philipp Koehn. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *AMTA*, pages 115–124, 2004.

[43] Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[44] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[45] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proc. of NAACL*, 2003.

[46] Muntsin Kolss, Stephan Vogel, and Alex Waibel. Stream decoding for simultaneous spoken language translation. In *INTERSPEECH*, pages 2735–2738, 2008.

[47] Abby Levenberg, Chris Dyer, and Phil Blunsom. A Bayesian Model for Learning SCFGs with Discontiguous Rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 223–232, Jeju Island, Korea, 2012. Association for Computational Linguistics.

[48] P. M. Lewis, II and R. E. Stearns. Syntax-directed transduction. *Journal of the ACM*, 15(3):465–488, July 1968.

[49] Dekang Lin and Franz Och. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *COLING 2004*, pages 501–507, 2004.

[50] Yang Liu, Qun Liu, and Shouxun Lin. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 609–616, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[51] Adam Lopez. Hierarchical phrase-based translation with suffix arrays. In *EMNLP-CoNLL*, pages 976–985, 2007.

[52] Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proc. of EMNLP-2002*, pages 133–139, 2002.

[53] Evgeny Matusov, Dustin Hillard, Mathew Magimai-doss, Dilek Hakkani-tur, Mari Ostendorf, and Hermann Ney. Improving speech translation with automatic boundary prediction. In *In Proc. Interspeech*, pages 2449–2452, 2007.

[54] I. Dan Melamed, Giorgio Satta, and Benjamin Wellington. Generalized multitext grammars. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, 2004.

[55] Takashi Mieno, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Speed or accuracy? a study in evaluation of simultaneous speech translation. In *INTERSPEECH*, 2015.

[56] Binh minh Bui Xuan, Michel Habib, and Christophe Paul. Revisiting t. uno and m. yagiura's algorithm. In *Proc. 16th International Symposium on Algorithms and Computation, in Lecture Notes in Comput. Sci*, pages 146–155. Springer, 2005.

[57] Robert C. Moore and John Dowding. Efficient bottom-up parsing. In *HLT*. Morgan Kaufmann, 1991.

[58] Thuylinh Nguyen and Stephan Vogel. Integrating phrase-based reordering features into chart-based decoder for machine translation. In *Proc. of ACL*, 2013.

[59] Franz Josef Och. An efficient method for determining bilingual word classes. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, EACL 1999, pages 71–76, 1999.

[60] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proc. of ACL*, 2003.

[61] Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL*, 2002.

[62] Franz Josef och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, March 2003.

[63] Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30, 2004.

[64] Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Optimizing segmentation strategies for simultaneous speech translation. In *The 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Baltimore, USA, June 2014.

[65] Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Syntax-based simultaneous translation through prediction of unseen syntactic constituents. In *ACL*, 2015.

[66] Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. Bleu: a method for automatic evaluation of machine translation. pages 311–318, 2002.

[67] Michael Pust and Kevin Knight. Faster mt decoding through pervasive laziness. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 141–144, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[68] Chris Quirk, Arul Menezes, and Colin Cherry. Dependency treelet translation: syntactically informed phrasal smt. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 271–279. Association for Computational Linguistics, 2005.

[69] Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. Segmentation strategies for streaming speech translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[70] Majid Razmara, Baskaran Sankaran, Ann Clifton, and Anoop Sarkar. Kriya - the sfu system for translation task at wmt-12. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT '12, pages 356–361, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[71] Richard Socher and Eric H. Huang and Jeffrey Pennington and Andrew Y. Ng and Christopher D. Manning. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems 24*. Association for Computational Linguistics, 2011.

[72] Baskaran Sankaran. *Improvements in Hierarchical Phrase-based Statistical Machine Translation*. PhD thesis, Simon Fraser University, Burnaby, Canada, 2013.

[73] Baskaran Sankaran, Ajeet Grewal, and Anoop Sarkar. Incremental decoding for phrase-based statistical machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT, 2010.

[74] Baskaran Sankaran, Gholamreza Haffari, and Anoop Sarkar. Compact rule extraction for hierarchical phrase-based translation. In *The 10th biennial conference of the Association for Machine Translation in the Americas (AMTA)*, San Diego, CA, oct 2012. Association for Computational Linguistics.

[75] Baskaran Sankaran, Majid Razmara, and Anoop Sarkar. Kriya - an end-to-end hierarchical phrase-based mt system. *The Prague Bulletin of Mathematical Linguistics (PBML)*, 97(97):83–98, apr 2012.

[76] Hendra Setiawan, Zhongqiang Huang, Jacob Devlin, Thomas Lamar, Rabih Zbib, Richard M. Schwartz, and John Makhoul. Statistical machine translation features with multitask tensor networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 31–41, 2015.

[77] Hassan S. Shavarani, Maryam Siahbani, Ramtin M. Seraj, and Anoop Sarkar. Learning segmentations that balance latency versus quality in spoken language translation. In *The 12th International Workshop on Spoken Language Translation (IWSLT 2015)*, pages 217–224, 2015.

[78] Stuart M. Shieber and Yves Schabes. Synchronous tree-adjoining grammars. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 3*, COLING 2090, pages 253–258, 1990.

[79] Maryam Siahbani, Ramtin Mehdizadeh Seraj, Baskaran Sankaran, and Anoop Sarkar. Incremental translation using a hierarchical phrase-based translation system. In *In Proceedings of the 2014 IEEE Spoken Language Technology Workshop (SLT 2014)*, Nevada, USA, December 2014.

[80] Maryam Siahbani, Baskaran Sankaran, and Anoop Sarkar. Efficient left-to-right hierarchical phrase-based translation with improved reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Seattle, USA, October 2013.

[81] Maryam Siahbani and Anoop Sarkar. Expressive hierarchical rule extraction for left-to-right translation. In *Proceedings of the 11th Biennial Conference of the Association for Machine Translation in the Americas (AMTA-2014).*, Vancouver, Canada, October 2014. Association for Computational Linguistics.

[82] Maryam Siahbani and Anoop Sarkar. Two improvements to left-to-right decoding for hierarchical phrase-based machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, October 2014.

[83] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014.

[84] Christoph Tillmann. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, HLT-NAACL-Short '04, pages 101–104, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[85] Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259, 2003.

[86] Takeaki Uno and Mutsunori Yagiura. Fast algorithms to enumerate all common intervals of two permutations. volume 26, page 2000, 2000.

[87] David Vilar and Hermann Ney. On lm heuristics for the cube growing algorithm. In *Annual Conference of the European Association for Machine Translation*, pages 242–249, Barcelona, Spain, may 2009.

[88] Stephan Vogel, Hermann Ney, and Christoph Tillmann. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING 1996, pages 836–841, 1996.

[89] Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. NTT statistical machine translation for iwslt 2006. In *Proceedings of IWSLT 2006*, pages 95–102, 2006.

[90] Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. Left-to-right target generation for hierarchical phrase-based translation. In *Proc. of ACL*, 2006.

[91] Wenduan Xu and Philipp Koehn. Extending hiero decoding in moses with cube growing. *Prague Bull. Math. Linguistics*, 98:133–, 2012.

[92] Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proceedings of the 2001 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 523–530, 2001.

[93] Mahsa Yarmohammadi, Vivek K Rangarajan Sridhar, Srinivas Bangalore, and Baskaran Sankaran. Incremental segmentation and decoding strategies for simultaneous translation. In *Proc. of IJCNLP-2013*, 2013.

[94] Hao Zhang, Daniel Gildea, and David Chiang. Extracting synchronous grammar rules from word-level alignments in linear time. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*, pages 1081–1088, Manchester, UK, 2008.

[95] Jiajun Zhang and Chenqqing Zong. A Comparative Study on Discontinuous Phrase Translation. In *NLPCC 2012*, pages 164–175, 2012.

[96] Andreas Zollmann and Ashish Venugopal. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, StatMT 2006, pages 138–141, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.