Waseda University Doctoral Dissertation

A Study of Efficient Bidirectional Latent Variable Models in Machine Translation

Hao WANG

Graduate School of Information, Production and Systems
Waseda University

February 2019

"The many, as we say, are seen but not known, and the ideas are known but not seen."

Plato, The Republic

A Study of Efficient Bidirectional Latent Variable Models in Machine Translation

Hao WANG

Submitted for the degree of Doctor of Philosophy
February, 2019

Abstract

Various cross-linguistic differences exist between different languages. Such variations are called translation divergences. They may lie in the way words may be decomposed (*lexical*, word segmentation), in the way words correspond (*phrasal*, alignment), and in the way words are ordered in sentences (*structural*, syntax). The existence of translation divergences makes the straightforward transfer from source sentences into target sentences impractical.

Previous research has shown that modeling translation using latent variables, i.e., variables which are not directly observable, but for which models can be built, benefits the learning process of machine translation. Translation divergences are explicitly handled in these latent variable models. However, most of these models are asymmetric models, i.e., mono-directional (from source to target, or from target to source) or monolingual. They only partially capture some aspects of translation phenomena, leading to different performance on different language pairs. Typically, the translation of distant language pairs, like English–Japanese or Chinese–Japanese, receives lower translation scores. Apart from that, these models ignore the cross-linguistic equivalents between the source and target languages, e.g., strong correlations of some word pairs (1-1 alignments), similar underlying syntactic structures and shareable lexicons or sub-words. It is natural to exploit the bidirectional models that can make use of such equivalents.

In this dissertation, we propose a complete bidirectional latent variable framework to model translation divergences. The impact of the use of bidirectional latent

variable models is investigated through three important tasks in machine translation, namely: word segmentation, phrasal alignment, and syntactic reordering.

The methods proposed in this dissertation not only yield state-of-the-art performance but also are consistently more efficient in terms of learning time and model size. Translation experiments on distant language pairs demonstrate the efficiency of these bidirectional latent variable models.

Phrasal alignment is a fundamental latent variable in modeling phrase-based statistical machine translation (SMT). Phrase-based SMT often relies on asymmetric word alignments to extract phrasal alignments. Those asymmetric word alignments are output by some unsupervised word aligners. The accuracy of word aligners heavily affects the quality and size of phrase pairs. Given IBM models of lower complexity somehow are not reliable, word aligners have to rely on IBM models of higher complexity with long training time to obtain better alignments. These kinds of methods require an additional process of symmetrization. Therefore, we are interested in developing a phrasal alignment method that can directly output symmetric word/phrasal alignments. To deal with phrasal translation divergences, we propose a novel bidirectional phrasal alignment method for building phrase-based SMT models. This method is simple, fast and delivers small phrase tables, which takes advantages of both the strengths of IBM models and hierarchical sub-sentential alignment (HSSA). Moreover, thanks to the use of beam search, our implementation is more efficient which allows obtaining more accurate phrasal alignments.

The underlying syntactic structure is another essential latent variables. Phrase-based SMT cannot handle well the long-distance reordering problem. Recent progress in syntax-based SMT has shown that the underlying syntactic structures, like parse trees of synchronous context-free grammars (SCFGs), may benefit the learning of long-distance reorderings in the translation of distant language pairs. Although these methods commonly make use of syntactic trees to perform the translation, this does not apply to words where no syntax parser exist. Other methods using a simplified SCFGs named hierarchical phrase-based model dispense with parsers, but require large reordering rule tables. To investigate structural translation divergences, this dissertation employs the latent variable of bilingual syntactic repre-

sentation of bracketing transduction grammars (BTG). BTGs is the minimal case of SCFGs which constitutes a simple bilingual parsing model to answer this problem. It has many advantages to use BTG like its simplicity, and more important, BTG is insensitive to long-distance reorderings. Considering BTG as an effective way to represent bilingual syntactic structures, we explore the possibility of using the latent BTG derivations to control the reordering of the source or target words, either before the standard phrase-based SMT pipeline (source, preordering), or during translation (target, decoding). Our contribution consists in improved training algorithms for the top-down BTG preordering method using bootstrap aggregating with several learning techniques (mini-batch, distributed, iterative distributed and k-best list) and a latent-BTG–based decoding method.

For East Asian languages without word separators (Chinese or Japanese), word segmentation is considered an essential pre-processing step in many natural language processing tasks, including machine translation. We find that segmentation consistency and granularity of "words" influences the quality of the final translations. Different word segmentation criteria output different numbers of unique words, and consequently, having different translation scores. The underlying vocabulary, which is a crucial latent variable, controls the generating of the sequences of "words". Conventional word segmenters, e.g., Juman or Stanford Segmenter, massively produce rare words which SMT or NMT systems cannot translate. This results in lower translation scores of SMT and NMT. Researchers have recently discovered that decomposing rare words into sub-words may benefit the translation of rare words. Some words at least some sequences of characters (called sub-words) in Chinese/Japanese are equivalent. This dissertation investigates a bilingual word segmentation method with the aim of efficiently learning of translation models. The bilingual word segmentation method is based on the principle of Minimum Description Length (MDL) with two additional restrictions: finite vocabulary and minimal frequency. This constitutes a better bilingual word segmentation method for machine translation which reduces lexical divergences.

This dissertation provides a systematic solution to translation divergences existing in machine translation. Experimental results show that bidirectional latent

variable models presented in this dissertation not only yield state-of-the-art performance but also effectively reducing the problems caused by asymmetric models. This leads to more efficient learning processes in word segmentation, phrasal alignment, and reordering, while requiring shorter training times and/or having smaller model sizes.

Declaration

The work in this thesis is based on research carried out at the Graduate School of Information, Production and Systems, Waseda University. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:_____ **Date:** _____

Acknowledgements

First, I want to thank my supervisor, Yves Lepage for having me as his doctoral student. His thoughtful guidance, constant encouragements, and insistence on novelty have had shaped the way I write and do research. Since the first day of welcoming me into his lab until the last day, I am so grateful for everything he has taught me. I am thankful for not only his supervision but also for the experiences that he provided in becoming a pragmatic and qualitative researcher.

I want to thank Professor Prof. Yuki Arase (Osaka University) and Prof. Juníchi Tsujii (Microsoft Research Asia) for welcoming me as an intern in Microsoft Research Asia (Beijing), for many suggestions and discussions about not only machine translation, but also working and life experiences.

I would also like to thank Masayuki Okamoto at Knowledge Media Lab, Toshiba Corporate Research & Development Center (Kawasaki), who had supervised me nearly three months on both distance supervision with machine learning and knowledge engineering.

Thanks to Kyosuke Nishida at Media Intelligence Laboratories (Yokosuka), NTT Corporation, who had invited me twice and held me as an intern, working on reading comprehension using deep neural networks. He thought me a lot of deep learning techniques.

I would also like to thank Chenhui Chu and Long Trieu encouraging me not to give up and provide me suggestions on journal submission.

I would also like to thank my co-supervisor Professor Mizuho Iwaihara and Takayuki Furuzuki (Jinglu Hu) for agreeing to serve as members of my doctoral committee, and for carefully checking this dissertation and providing advice during my presentation.

At last, many thanks to all my friends and all the members of the Lepage laboratory.

Since this work is supported in part by the joint project of Waseda University and China Scholarship Council (CSC) under the CSC Grant No.201406890026, I am exceedingly grateful for Waseda University to remiss the tuition fee and my motherland government to support me during my doctoral course.

List of Abbreviations

MT	**M**achine **T**ranslation
SMT	**S**tatistical **M**achine **T**ranslation
NMT	**N**eural **M**achine **T**ranslation
PBSMT	**P**hrase-**B**ased **S**tatistical **M**achine **T**ranslation
HPBSMT	**H**ierarchical **P**hrase-**B**ased **S**tatistical **M**achine **T**ranslation
BTG	**B**racketing **T**ransduction **G**rammar
SBTG	**S**tochastic **B**racketing **T**ransduction **G**rammar
CFG	**C**ontext **F**ree **G**rammar
PCFG	**P**robabilistic **C**ontext **F**ree **G**rammar
SCFG	**S**ynchronous **C**ontext **F**ree **G**rammar
AER	**A**lignment **E**rror **R**ate
FRS	**F**uzzy **R**eordering **S**core
CMS	**C**omplete **M**atching **S**core
NKT	**N**ormalized **K**endall's **T**au
HSSA	**H**ierarchical **S**ub-**S**entential **A**lignment
VB	**V**ariational **B**ayes
OOV	**O**ut **O**f **V**ocabulary
MDL	**M**inimum **D**escription **L**ength
BPE	**B**yte **P**air **E**ncoding
SPM	**S**entence **P**iece **M**odel
WPM	**W**ord **P**iece **M**odel
RNN	**R**ecurrent **N**eural **N**etwork

List of Symbols

$\mathbf{T}$	Parse **T**ree
$\mathbf{E}$	**E**nglish (target) Sentence
$\mathbf{F}$	**F**oreign (source) Sentence
$\mathbf{D}$	Parse **D**erivation
$\mathbf{a}$	Word **A**lignment
$\mathbf{f}$	**F**oreign (source) Word
$\mathbf{e}$	**E**nglish (target) Word
$\mathbf{d}$	Atomic Parse **D**erivation
$\mathbf{Z}$	Latent Variable
S	**S**traight
I	**I**nverted
T	**T**erminal

Contents

List of Figures

List of Tables

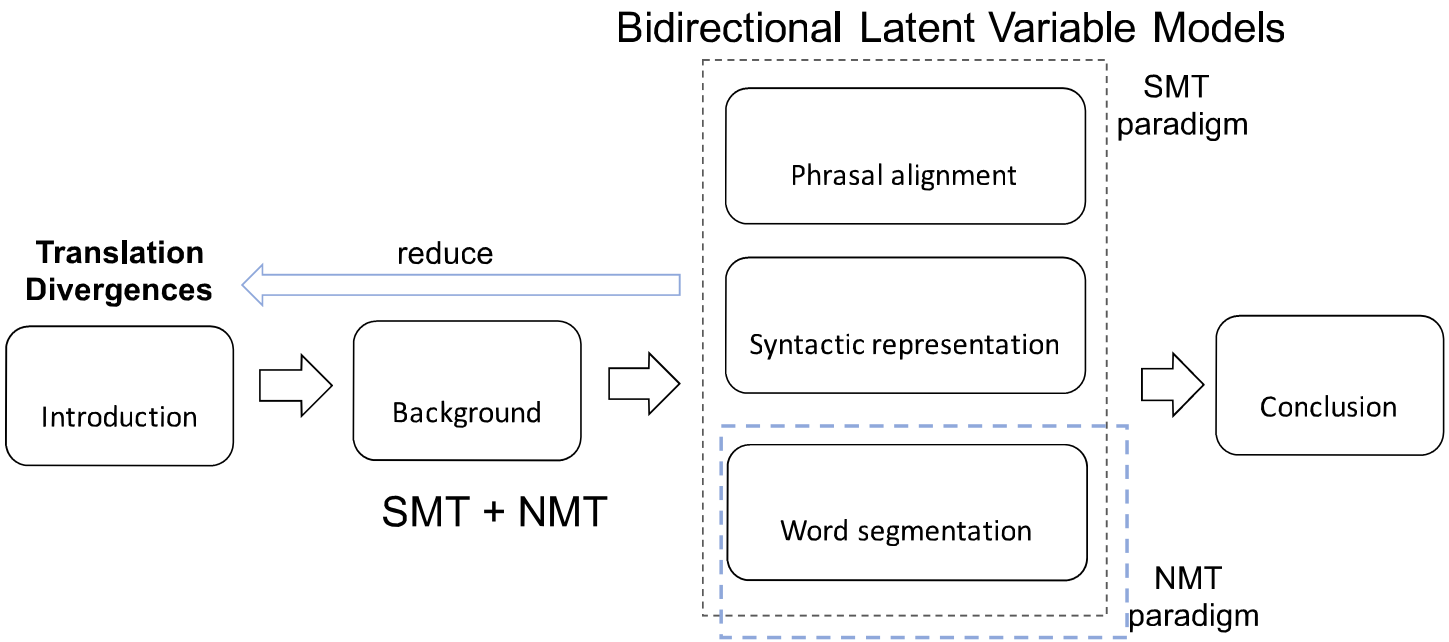


Figure 1: Structure of the dissertation.

Chapter 1

Introduction

This chapter introduces the background of the research. We describe the literature on machine translation, basic notions of translation divergence, and the way of using latent variables in machine translation. We also present an analysis of the existing problems in the translation of distant language pairs and discuss our proposal to deal with these problems.

The structure of this chapter is as follows.

- Section 1.1 introduces previous methods to machine translation and the main aspects of statistical and neural machine translation.
- In Section 1.2, we give the definition of translation divergences and classification of different types of translation divergences.
- In Section 1.3 and Section 1.4, we discuss the problem in current latent variable models and the motivation of this research.
- Section 1.5 presents our proposal, called bidirectional latent variable models. The advantages of using these bidirectional models are also discussed.
- Section 1.6 describes the structure of the thesis and highlights our contributions.

1.1 History of Machine Translation

Human languages are diverse with about 7000 languages spoken [Tyers et al., 2010]. Since the unstoppable globalization, the need for seamless communication and understanding across languages has become more and more crucial. The majority of human beings have been multilingual ($\leq$ bilingual). Therefore, one primary object of inquiry in computational linguistics is to understand various human languages using the machine, i.e., machine translation (MT).

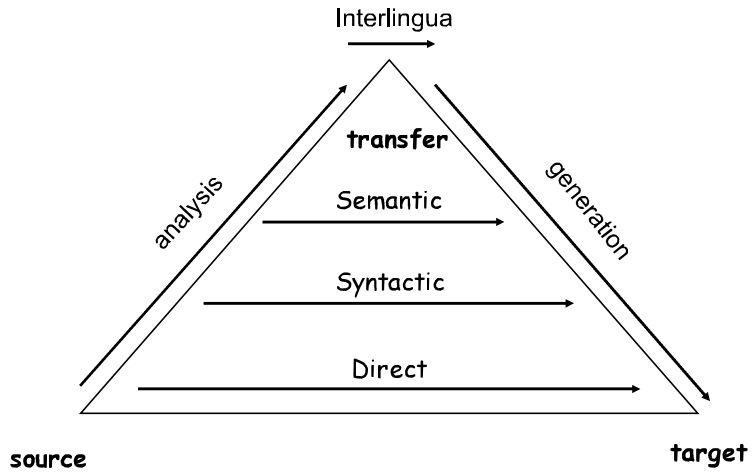


Figure 1.1: The machine translation pyramid (Vauquois' triangle).

In this dissertation, here and after, we set those problems of machine translation that we will discuss: the conversion of electronic texts from one natural language into another. Since translation divergences exist, translation is more difficult than it seems. Vauquois' triangle (see Figure 1.1) was commonly used in the linguistic rule-based machine translation community to describe the complexity of machine translation. The bottom approach consisted of a direct lexical conversion (dictionary) between a source language and a target language. Later efforts moved up the pyramid to become more and more complicated. Outside of the pyramid are preprocessing that analyze the source language and post-processing that generates in the target language. Each step up the triangle required more significant effort in the analysis of the source language and generation of the target language but reduced the effort involved in the transfer phase across languages. The ideal scenario was

a complete analysis of each sentence into a language-independent form, which can represent any meaning in any style, thus called "interlingua".

The first exploration in translation using the machine is the 701 translator [Sheridan, 1955] developed at Georgetown and IBM in the 1950s. From that time, machine translation became a sub-field research in computational linguistics. Since its inception, different paradigms for MT came and gone through many periods of great development. From different points of view including linguistics and machine learning, state-of-the-art approach mainly falls into two categories: example/rule-based methods and statistical/neural-based methods. There are two main differences between them:

- The former relies on manually encoded knowledge, i.e., grammar rules, or translation knowledge base.
- The latter performs translation based on statistical models in contrast to the former. It treats the translation of natural language as a machine learning problem.

Statistical machine translation (SMT) [Brown et al., 1990, Och, 2003, Koehn et al., 2003, Chiang, 2005] had dominated academic MT research over the last fifteen years. The common and general setting of SMT is to learn how to translate from a large parallel corpus, which relies on statistical models learned from parallel corpora. Generally, this is typically a machine learning framework: we have an input $\mathbf{F}$ (the source sentence, e.g., Japanese), an output $\mathbf{E}$ (the target sentence, e.g., English), and a probability model P trying to produce the correct output $\mathbf{E}$ for each given input.

$$\mathbf{E} = P(\mathbf{F}) \tag{1.1.1}$$

Note that the process of segmentation is not modeled explicitly.

Progress within the SMT approach, however, has been less dramatic compared to the recent development of neural machine translation. Neural machine translation (NMT) [Bahdanau et al., 2014, Sutskever et al., 2014a] is a recently proposed approach to machine translation. Unlike the traditional SMT, instead of integration of multiple different models, NMT aims at building a single neural network that the

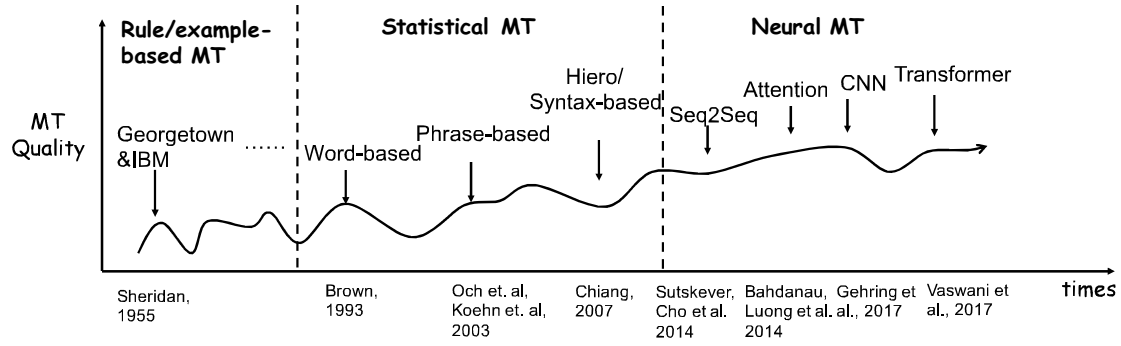


Figure 1.2: History of machine translation.

language model and the translation model can be jointly trained. NMT methods promise better sharing of statistical evidence between similar words and inclusion of rich context. In both of the statistical and neural MT approaches, the decoder is an essential component that employs statistical/neural models to translate from the source language into the target language automatically. Theoretically, we separate SMT with NMT as two different paradigms given the architectures are different, but the idea of learning the translation probability model with the machine is the same.

The description of the machine translation pyramid also suits for recent SMT and NMT, where NMT locates near the semantic floor, and SMT locates between the direct floor to the semantic floor. Figure 1.2 shows the progress since the birth of MT until now. It shows how MT has changed over the years.

1.2 Translation Divergences

English is distant to East Asian languages(such as Chinese and Japanese) than it is to Western European languages (such as French and German). For example, an English sentence significantly differs with the corresponding Japanese sentence from aspects of written forms and similarities of syntax. In linguistics, such a language pair is usually called "distant language pair" given that these languages belong to distinct language families.

For current machine translation approaches, learning the translation of such a distant language pair is challenging and problematic. The translation quality as

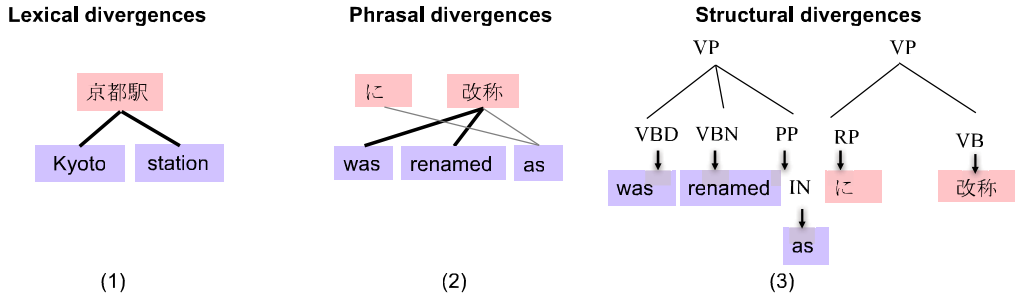


Figure 1.3: Classification of translation divergences.

measured by translation accuracy metrics is not yet satisfying. The straightforward transfer from source sentences into target sentences is impractical, because of the existence of translation divergences, i.e., cross-linguistic differences. Such differences may lie in the way words may be decomposed (word segmentation), word order (syntax), and consequently the way words correspond (phrasal alignment).

We limit the scope of translation divergences which we will focus on to the lexical-semantic divergences defined in [Dorr, 1994]. We first classify translation divergences into three types, as shown in Figure 1.3.

- **Lexical divergences**

  The first divergence type is *lexical*: in Figure 1.3 (1), English words are written in Latin script (English alphabet), while sharply differs from Japanese, which uses "Kanji" script. They use different alphabets. Multiple possible translations may exist for one source word.

- **Phrasal divergences**

  The second divergence type, *phrasal*, exploiting outside of the unit of "word", where the way words are ordered and the way of word decomposition, as Figure 1.3 (2) shows. For instance, we frequently find that expressing source word meaning in a single target word is difficult.

- **Structural divergences**

  The last divergence type, *structural*, mainly involves the structural differences in grammar or syntax as in Figure 1.3 (3).

Although the divergences were named differently in the literature on MT, a number of researchers have been working on this issue seeking a systemic solution.

1.3 Latent Variable Models

Figure 1.4: Direct learning.

Current machine translation approaches commonly treat machine translation as a machine learning problem based on conditional probabilistic models. The general translation model defines a conditional probability distribution over the target translations of a given source sentence. It describes the mapping between the source sentence and target sentences in a direct way (see Figure 1.4). These models are typically trained using large parallel corpora which contain observable examples of source sentences aligned with their translations (target sentences). Often we can see and measure the observed variables, i.e., **F** and **E**, but we cannot directly observe or measure the constructs of $\mathbf{F} \rightarrow \mathbf{E}$ themselves. A natural question raises:

Are there any unobservable or "latent" variables as Figure 1.5, *which control the constructs to correct?*

Here, latent[1] refers to variables that are not directly observed but are rather inferred (through a statistical or neural model) from other observed variables. Intrinsically, we assume the existence of useful latent structures and classes which handle such cross-linguistic differences.

Previous research has shown that modeling the translation problem using the latent variables, i.e., variables which are not directly observable, but for which models can be built, benefits the learning process in machine translation. The reason is that the latent variables may capture the underlying cross-linguistic phenomena

[1]from Latin: present participle of *lateo* ("lie hidden"), where hidden = latent = unobserved.

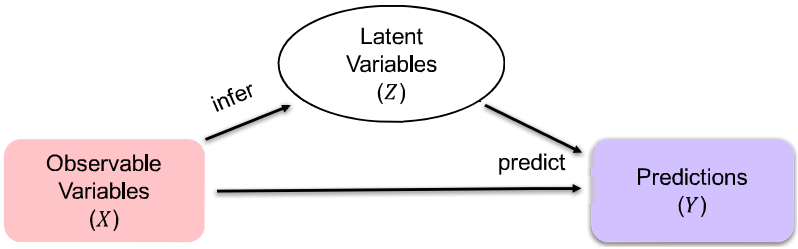


Figure 1.5: Indirect learning.

where translation models being constructed. These latent variable models mainly involve three intermediate processes, namely, segmentation, alignment, and syntax transformation.

1.4 Drawbacks of Asymmetric Models

The majority of latent variable models are asymmetric models, i.e., mono-directional (from source to target, or from target to source), or monolingual. These model might only partially capture some aspects of translation, leading to different performance on different language pairs. Typically, the translation of distant language pairs, like English–Japanese or Chinese–Japanese, exhibits lower translation scores compared with English–French. In other words, current latent variable models cannot deal very well with the translation of distant language pairs.

For example, in phrasal alignment, previous methods extract phrase pairs in a two-step process. In a first step, they rely on word alignment models, i.e., IBM models (from 1 to 5) and HMM models to obtain initial asymmetric hard word alignments. This requires a bi-directional training procedure in a second step and an additional process of symmetrization, resulting in long training times and large translation tables.

In syntax-based reordering, problems caused by asymmetric models occur when transferring from source structures into target structures. Two syntactic parse trees may be quite distinct for corresponding sentences. Such syntax-based models require a large number of phrase-structure rules to represent various reorderings.

Lexical divergences may also exist in word segmentation due to the use of dif-

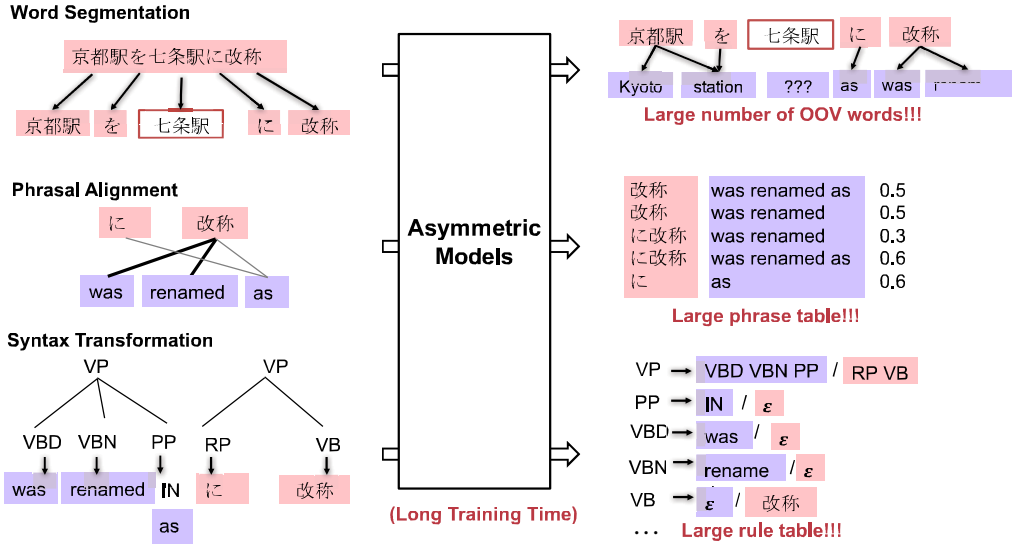


Figure 1.6: Existing problems in current approaches based on asymmetric models.

ferent segmentation standards. Different segmenters for different languages may produce words in different quantity which MT systems cannot align or translate. This results in different translation scores in SMT or NMT.

In addition to the above problems, mono-directional models such models might be not optimal for the final task of machine translation because they may neglect to use equivalents across languages. These equivalents can appear at word level and above word level, when translating from one language into another. It sounds natural to design bidirectional models which would exploit these equivalents with symmetry in mind.

1.5 A Systematic Solution: Efficient Bidirectional Latent Variable Models

In this dissertation, we propose to model translation divergences using bidirectional latent variable models. "Bidirectional" means that these latent variables should convey mostly the same information for both the source and the target sides, if and only if these conversions are bidirectional procedures. Bidirectional latent variable models are a complete framework which aims at reducing bilingual language divergences

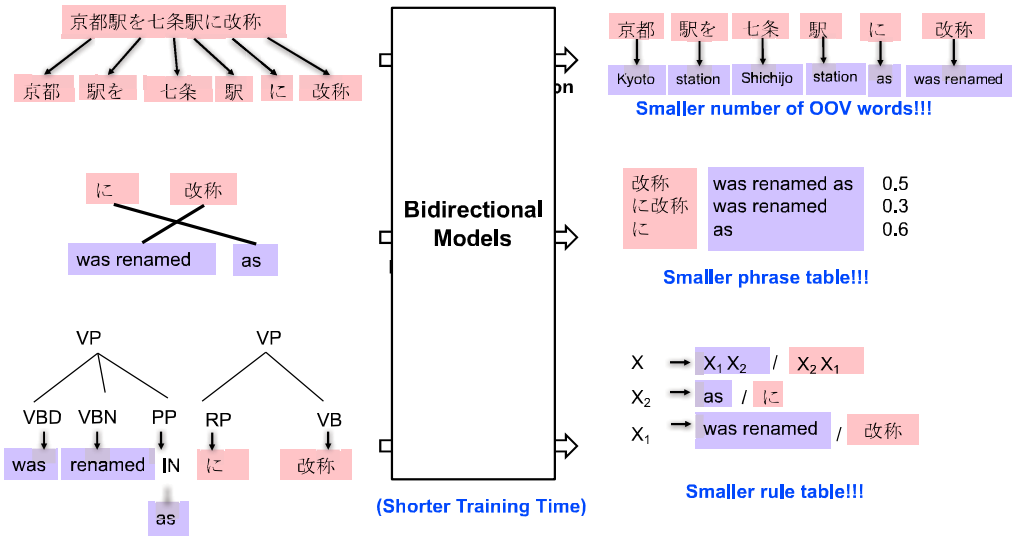


Figure 1.7: Efficiency of bidirectional latent variable models.

systematically. This dissertation investigates the impact of the use of bidirectional latent variable models in three important tasks for machine translation, namely: phrasal alignment, syntactic reordering and word segmentation. The efficiency of the bidirectional latent variable models proposed in this dissertation not only yield state-of-the-art performance but they are also consistently more effective for the translation of distant language pairs, like English–Japanese or Chinese–Japanese.

Figure 1.8 illustrates four main challenges within the Japanese–English translation task: segmentation, alignment, reordering. For example, in languages that use word separators, like English, words can be easily recognized by splitting a sentence using these separators. However, languages like Chinese or Japanese do not make use of any explicit word delimiter. Identifying word boundaries is more laborious. To learn a phrase-based translation model for SMT, we need to know word alignments in a source-target sentence pair. Different word orders is another problem need to solve. Generally, word order in languages differs variously. For example, the order of subject (S), verb (V) and object (O) in a Japanese or Korean sentence, in which we called SOV languages, is different from English. Therefore, we can decompose the translation model as a chain of latent variable sub-models. Let $\mathbf{Z}$ denotes the latent variables hidden in Equation 1.1.1,

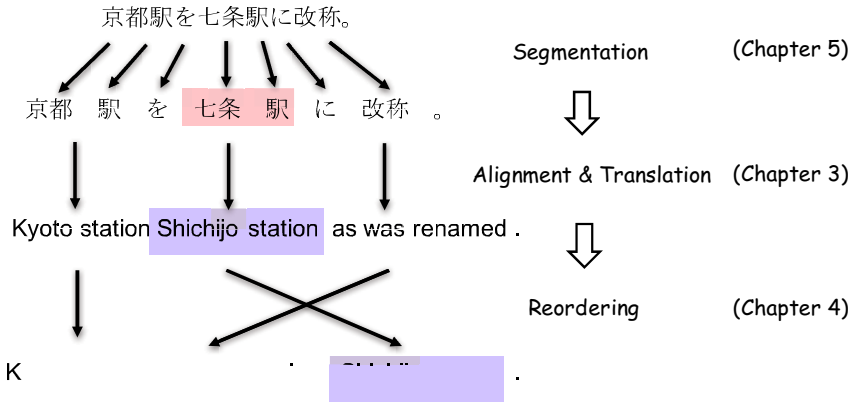


Figure 1.8: Example of phrase-based translation from the source (Japanese) sentence into a target (English) sentence.

$$P(\mathbf{E}|\mathbf{F}) = P(\mathbf{E}, \mathbf{Z}|\mathbf{F}) \tag{1.5.2}$$

$$= P(\mathbf{E}, \{\mathbf{Z}_{seg}, \mathbf{Z}_{reord}, \mathbf{Z}_{align}\}|\mathbf{F}) \tag{1.5.3}$$

$$= P(\mathbf{E}|\mathbf{Z}_{seg}, \mathbf{F})P(\mathbf{Z}_{seg}|\mathbf{F}) \tag{1.5.4}$$

$$= P(\mathbf{E}|\mathbf{Z}_{[seg,reord]}, \mathbf{F})P(\mathbf{Z}_{[seg,reord]}|\mathbf{Z}_{seg}, \mathbf{F})P(\mathbf{Z}_{seg}|\mathbf{F}) \tag{1.5.5}$$

$$= P(\mathbf{E}|\mathbf{Z}_{[seg,reord,align]}, \mathbf{F}) \times P(\mathbf{Z}_{[seg,reord,align]}|\mathbf{Z}_{[seg,reord]}, \mathbf{F})$$
$$\times P(\mathbf{Z}_{[seg,reord]}|\mathbf{Z}_{seg}, \mathbf{F}) \times P(\mathbf{Z}_{seg}|\mathbf{F}) \tag{1.5.6}$$

where $\mathbf{Z}$ can be any latent variable, e.g., segmentation $\mathbf{Z}_{seg}$, reordering $\mathbf{Z}_{reord}$ or alignment $\mathbf{Z}_{align}$. $\mathbf{Z}$ bridges the gap between the source language and the target language, thus it is a bilingual latent variable model.

This bilingual latent variable model is capable of modeling these translation divergences that do not decompose into word-for-word translation. The sharp difference between our proposal and the previous methods is that they are asymmetric models while our models are bidirectional. The advantages of the proposed models are illustrated in the Figure 1.7. The efficiency of these methods are independently evaluated given the following aspects:

(a) the translation accuracy of the end-to-end translation systems;

(b) the training time required for these processes, such as phrasal alignment, sen-

tence reordering and word segmentation;

(c) the amount of memory needed to hold the program and the produced models.

1.6 Overview of the Dissertation

The rest of this dissertation is organized as follows. Figure 1 illustrates the structure of the dissertation.

In Chapter 2, we introduce the context and background of machine translation and two dominant paradigms: SMT and NMT. We give a brief overview of the architectures in standard phrase-based SMT, syntax-based SMT and sequence-to-sequence NMT. The following chapters present our proposal for handling various translation divergences by using bidirectional latent variable models.

Chapter 3 presents a novel symmetric phrasal alignment method. It incorporates IBM models with hierarchical sub-sentential alignment method for symmetric phrasal alignment. Thus, this method enables automatically generating symmetric alignment directly from a parallel corpus. We evaluate the proposed method in various language pairs. Machine translation experiments for distant language pairs like English–Japanese as well as other similar language pairs like English–French, Spanish-Portuguese are performed. We evaluate the efficiency of the proposed method in terms of alignment error rate, training time and translation quality.

In Chapter 4, we investigate Bracketing Transduction Grammars, as a bidirectional latent variable model for syntax-based SMT. We start our work with a state-of-the-art method, called the top-down BTG-based preordering method, then boost the accuracy of structure predictions by reducing the insensitivity of initial alignment noise. We first investigate how latent derivations in response to reordering, then adopt various multi-processing techniques, so that the latent BTG parse trees predicted are more accurate. We conduct experiments on preordering for phrase-based SMT to test the effectiveness from the aspects of the quality of reorderings and translations. Experiments on extending syntax-based SMT systems with these reordering models also show promising results in the English-Japanese task.

Chapter 5 describes a bilingual word segmentation method using the principle of

Minimum Description Length (MDL). The vocabulary of behind the segmentation is a crucial latent variable for both SMT and NMT. We propose to impose two additional restrictions: minimum frequency and limited vocabulary, into MDL during inference of vocabulary. Our proposal yields a compressed dictionary of words for both the source and target languages. This allows sharing vocabulary and word embedding across languages in an encoder-decoder architecture. To prove the efficiency of our proposal, we compare with baseline systems using some pre-trained supervised word segmenters for both Chinese–Japanese and English–Japanese translation tasks. Such a method also benefits the translation model learning in seq2seq NMT.

Chapter 6 summarizes and concludes this dissertation. The main contribution of this dissertation are the proposed method involving several bidirectional latent variable models. This dissertation provides a systematic solution to translation divergences. Finally, the possible directions for future work are discussed.

Chapter 2

Background

For a better understanding of the following chapters, this chapter reviews the basic notions of SMT and NMT.

- Section 2.1 introduces the standard structure of the phrase-based statistical machine translation system.
- In Section 2.2, we go through the basics of formal language theory, and its application in SMT, called syntax-based SMT.
- Section 2.3 describes the recent developments in NMT, typically sequence-to-sequence NMT, one of the recent very popular NMT models.

2.1 Phrase-based Statistical Model

Statistical machine translation is one of the approaches to machine translation. It considers translation as a machine learning problem based on conditional probabilistic models. Different methods have been proposed, including word-based [Brown et al., 1988], phrase-based [Koehn et al., 2003], hierarchical-phrase-based [Chiang, 2007], tree-to-string-based [Liu et al., 2006] and tree-to-tree-based [Zhang et al., 2007].

Among these methods, phrase-based SMT is the most widely used and easily built one, which translate with continuous fragments, i.e. phrases, as the basic units processed at each time step. The basic process of translation in SMT is called decoding. According to the noisy channel formulation, it consists in a searching process, where the most probable target (or English) sentence $\tilde{\mathbf{E}}$ is selected from all candidates $\hat{\mathbf{E}}$ given a source (or foreign) sentence $\mathbf{F}$:

$$\tilde{\mathbf{E}} = \arg\max_{\mathbf{E}\in\hat{\mathbf{E}}} P(\mathbf{E}|\mathbf{F}) \tag{2.1.1}$$

$$= \arg\max_{\mathbf{E}\in\hat{\mathbf{E}}} \frac{P(\mathbf{F}|\mathbf{E})P(\mathbf{E})}{P(\mathbf{F})} \tag{2.1.2}$$

$$= \arg\max_{\mathbf{E}\in\hat{\mathbf{E}}} P(\mathbf{F}|\mathbf{E})P(\mathbf{E}) \tag{2.1.3}$$

where $P(\mathbf{E},\mathbf{F})$ estimates conditional probability of any $\mathbf{E}$ given $\mathbf{F}$. Current phrase-based SMT models decomposes $P(\mathbf{E}|\mathbf{F})$ into a combination of sub-models $P_i(\mathbf{E}|\mathbf{F})$ within a log-linear framework as:

$$P(\mathbf{E}|\mathbf{F}) = \prod_{\lambda_i} \lambda_i P_i(\mathbf{E},\mathbf{F}) \tag{2.1.4}$$

$$= \lambda_{TM} P_{TM}(\mathbf{F},\mathbf{E}) \times \lambda_{RM} P_{RM}(\mathbf{F},\mathbf{E}) \times \lambda_{LM} P_{LM}(\mathbf{E}) \tag{2.1.5}$$

This formula consists of three sub-models: a phrase-pair–based translation model $P_{TM}(\mathbf{F},\mathbf{E})$, a language model $P_{LM}(\mathbf{F})$ and a reordering model $P_{RM}(\mathbf{F},\mathbf{E})$. The λ's are the model weights of each sub-model. $P(\mathbf{F},\mathbf{E})$ is decomposed further into $P_{TM}(\mathbf{F},\mathbf{E})$ and $P_{RM}(\mathbf{F},\mathbf{E})$.

State-of-the-art SMT methods [Chiang et al., 2009] has been further improved with the recent developments in machine learning. These methods employ large

statistical models developed using highly sophisticated linguistic knowledge. For instance, instead of using a single conditional probability of $P_{TM}(\mathbf{F}|\mathbf{E})$, current SMT methods allow the use of multiple translation models, various kinds of reordering models, even any number of other models. MT researchers are not satisfied with the traditional way of translation probability estimation using naïve Bayes with the EM algorithm anymore.

2.1.1 Language Model

The language model (LM) is the essential component of any SMT systems, which helps an SMT system to find the right word order. For example, the incremental statistical language models provide the the probability that a given the word will occur next, based on the preceding words:

$$P_{LM}(\text{renamed as} \mid \text{kyoto station was}) > P_{LM}(\text{ as renamed} \mid \text{kyoto station was })$$

The LM literature includes count-based language models and continuous-space language models. Count-based language models are based on Markov chains, where the number of previous states (words) is the order of the model. Since zero probability occurs when the query n-gram has not appeared in the corpus, some smoothing [Chen and Goodman, 1996] and back-off [Kneser and Ney, 1995] techniques are used. A major problem of Markovian LM is that any dependency beyond the window is ignored. This reason leads to the idea of applying Neural Networks to the problem of LM. [Mikolov et al., 2010] proposed continuous-space language models later. Training language model needs large amounts of text in the target language.

2.1.2 Translation Model

The type of the translation models may range largely for different SMT frameworks: word pair [Brown et al., 1988], phrase pair [Koehn et al., 2003], hierarchical phrase-based pair [Chiang, 2007], tree-to-string templates [Liu et al., 2006] and tree-to-tree templates [Zhang et al., 2007]. However, all these translation models require word-aligned parallel texts. In particular, phrase-based SMT requires a phrase translation table to map the source phrases into the target phrases. At the beginning of the

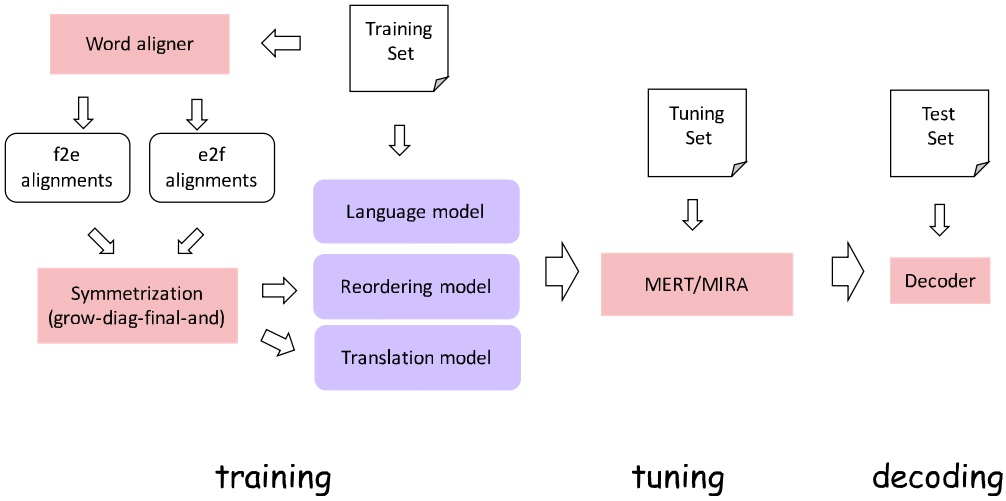


Figure 2.1: Standard architecture of a phrase-based statistical machine translation system.

training process, word alignment is typically not available. By treating it as a hidden variable, [Och and Ney, 2003] showed how to the expectation maximization (EM) algorithm to iteratively learn word alignment and translation model parameters from a parallel corpus aligned only at the sentence level. They then extract phrase pairs which are consistent with the learned word alignments. Probabilities (e.g., phrase translation probability, lexical weighting) for each phrase pair can be estimated given that alignment. Since the translation model is represented as a phrase table, the quality of word alignments has a great influence on the quality of translations. Hence, it is an important issue to improve word alignment for SMT. We will address this issue in Chapter 3.

2.1.3 Reordering Model

Initially, long distance word reordering was not explicitly modeled by phrase-based translation models. To overcome this limitation, a phrase distortion model d is incorporated into the phrase-based model. This model assigns a probability under each condition of permutation: *monotonic*, *reverse* and *jumping*. Since we will discuss these problems in Chapter 4, we skip the details for now.

2.1.4 Training, Tuning, Testing

The main steps in the development of an SMT system are training, tuning and decoding. Figure 2.1 illustrates how phrase-based SMT models are developed. As shown in Figure 2.1, a log-linear model is employed. The overall translation probability of a candidate sentence relies on the scores obtained from each sub-model. Therefore, building an SMT is, in fact, a machine learning problem. Traditionally, optimization is done on *n-best* lists of multiple decoder runs. The popular minimum error rate training (MERT) method was proposed by [Och, 2003]. One problem with MERT is that it is difficult to tune a large number of features. Other methods like the Margin Infused Relaxed Algorithm (MIRA) [Chiang et al., 2009] or *k-best* MIRA [Cherry and Foster, 2012] support more features. With careful tuning, PB-SMT can achieve acceptable performance.

With the trained model, an SMT system translates using a translation engine, called "decoder". It consists of two processes: retrieving the translation fragments (phrase pair) and generating the optimal target sentence satisfying **Equation 2.1.3**.

The target sentence is always built *left-to-right*, while the input sentence positions can be covered in different orders. Because searching over the space of all possible translations would be NP-hard, SMT decoders employ heuristic search algorithms to only explore a promising subset of the search space. In particular, state-of-the-art PBSMT systems rely on the beam-search algorithm [Tillmann and Ney, 2003] with the limitation of the reordering constraints to reduce the decoding complexity. Besides, in hierarchical phrase-based (Hiero/syntax-based) SMT, cube pruning [Chiang, 2005] is widely used in hierarchical phrase-based decoders to reduce the decoding complexity, performing the decoding using the CYK algorithm from bottom to up. We will address this issue in Chapter 4.

2.2 Syntax-based Model

Noam Chomsky said:

"Language is a process of free creation; its laws and principles are fixed, but the manner in which the principles of generation are used is free and infinitely varied."

To understand the above sentence, we start with some basic notions in formal language theory.

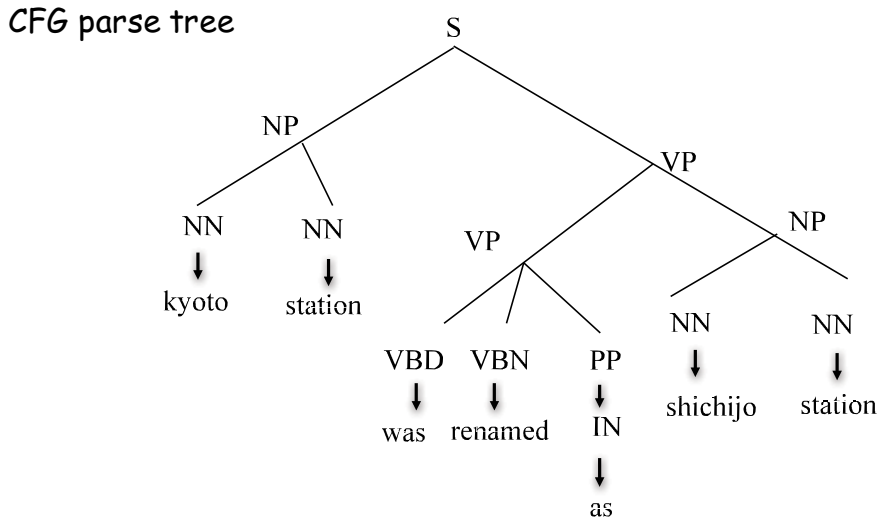


Figure 2.2: CFG parse tree for an English sentence.

Definition. 1 Context Free Grammar (in Chomsky Normal Form) A context-free grammar (CFG) $G = (\mathcal{V}, \Sigma, R, S)$ is a 4-tuple (see **Figure 2.2**), where $\mathcal{V}$ is a finite set; R is a finite relation; Σ is a finite set of terminals, disjoint from $\mathcal{V}$. S is the start symbol. In Chomsky Normal Form, if each rule $\alpha \to \beta$ in R takes one of the two following forms:

- $X \to X_1 X_2$ where $X \in \mathcal{V}, X_1 \in \mathcal{V}, X_2 \in \mathcal{V}$.
- $X \to Y$, where $X \in \mathcal{V}, Y \in \Sigma$.

Hence each rule in the grammar either consists of a non-terminal X rewritten as exactly two non-terminal symbols, $Y_1 Y_2$; or a non-terminal X is rewritten as exactly one terminal symbol Y. A CFG provides a simple and mathematically precise mechanism for describing the methods by which phrases in some natural language are built from smaller blocks, capturing the "block structure" of sentences in a natural way.

Let us move to the bilingual case, see Figure 2.3, in which the English parse tree is linked with the corresponding Japanese parse tree. Each link in the figure stands

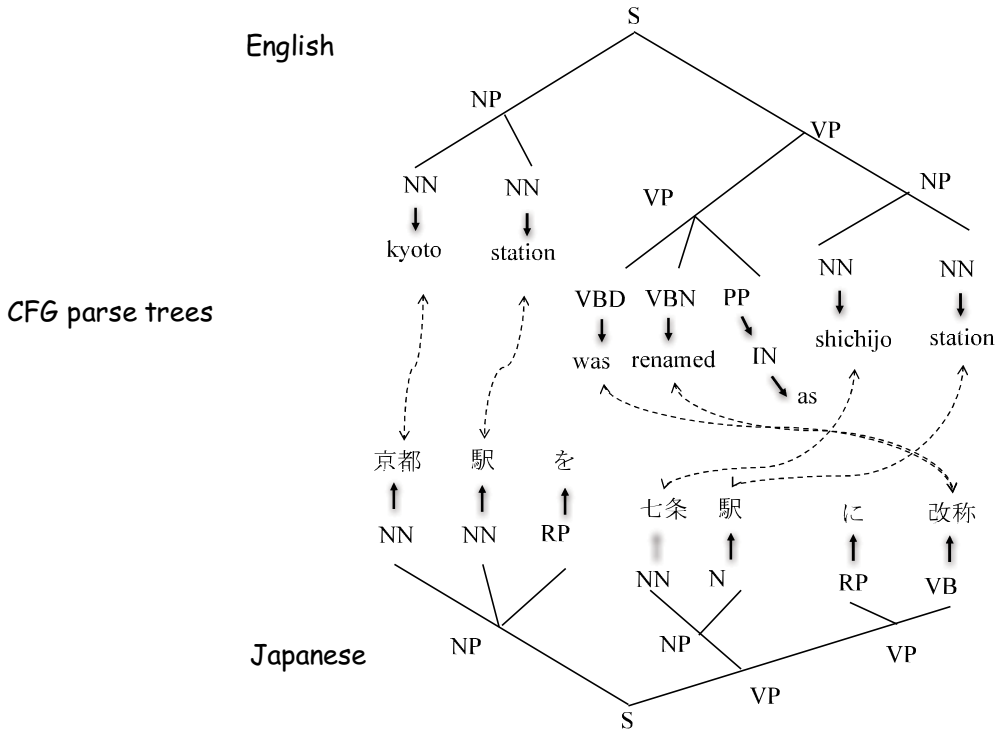


Figure 2.3: Linked CFG parse trees. *top*: English; *bottom*: Japanese.

for a word-to-word correspondence. These two trees have the similar underlying structure, but the order of constituents differs in the two languages.

Definition. 2 Synchronous Context-free Grammar (SCFG) An SCFG is defined as a 5 tuple (see Figure 2.4): $G = (\mathcal{V}, \mathcal{W}_1, \mathcal{W}_2, R, S)$, where $\mathcal{W}_1$ is a finite set, the set of *terminal* words in the source language and $\mathcal{W}_2$ is a finite set, the set of terminal words of the target language. The difference between SCFG and CFG is that the rewriting rules R are bilingual.

Definition. 3 Hiero Grammars [Chiang, 2005] propose a simplified version of SCFG, named Hiero Grammars, in which all *non-terminal symbols* are represented with a single symbol X but noted in serial, i.e., $\mathcal{V} = \{X\}$. Figure 2.4 shows the parse tree using the following rule:

$X \rightarrow X_1$ was renamed as X_2/X_1をX_2 に 改称.

Definition. 4 Bracketing Transduction Grammars (BTG) (or **Inversion Transduction Grammar** (ITG)) [Wu, 1997] is defined by the 5-tuple: $G = (\mathcal{V}, \mathcal{W}_1, \mathcal{W}_2, R, S)$. Similarly to SCFG, $\mathcal{V}$ is a finite set of *non-terminal symbols*

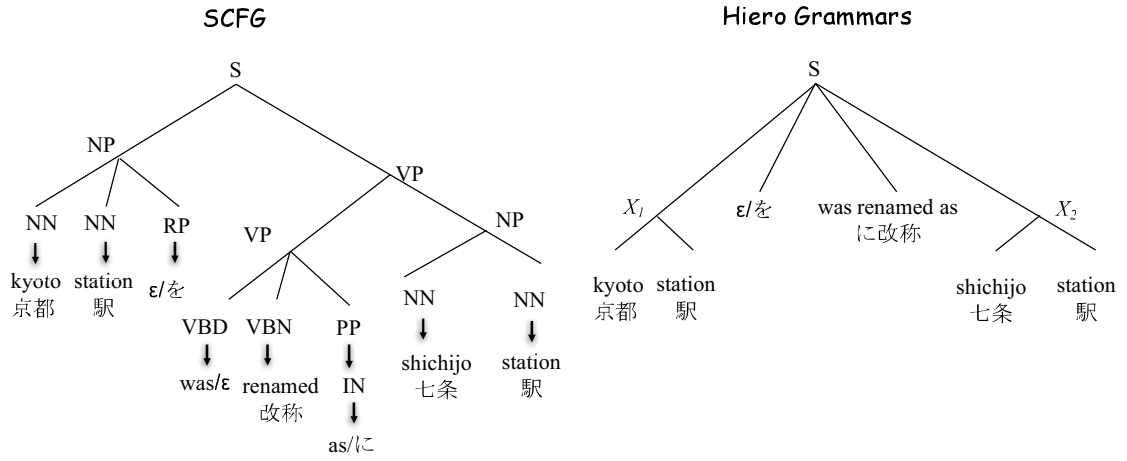
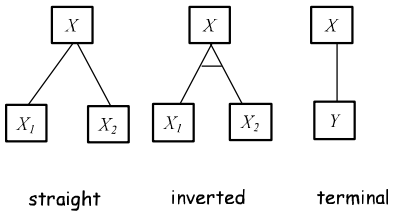


Figure 2.4: An example of parse trees for SCFGs and Hiero Grammars.

shared by the two languages which only contains two *non-terminal symbols* X_1, X_2, i.e., $\mathcal{V} = \{X_1, X_2\}$, and a single terminal symbol Y.

BTG can describe a structurally correlated pair of languages. In contrast, to address the linguistic divergences in each language, BTG try to model two different languages with the same inherent grammar. Similar to non-deterministic push-down transducers described in [Savitch, 1982], BTG have an additional *inverted* orientation, which allows the production of right-hand side in the *right-to-left* direction. The simplest formulation of BTG contains three simple generation rules R: straight—S, inverted—I and terminal—T as follows:



$$S : X \rightarrow \quad [X_1 X_2] \quad (2.2.6)$$

$$I : X \rightarrow \quad \langle X_1 X_2 \rangle \quad (2.2.7)$$

$$T : X \rightarrow \quad Y_e / Y_f \quad (2.2.8)$$

X_1, X_2 and X are non-terminal symbols, Y_f and Y_e are terminal strings, and [ ] denotes the same order for the two non-terminals in two languages, $\langle\rangle$ denotes the inversion case[1]. BTGs implement a crossing constraint inherently. At each level,

[1]Where $\gamma \in R$ and we compress the terminal rules (L_1 singleton $\gamma \rightarrow e/\epsilon$ and L_2 singleton $\gamma \rightarrow \epsilon/f$) into the default terminal rule T.

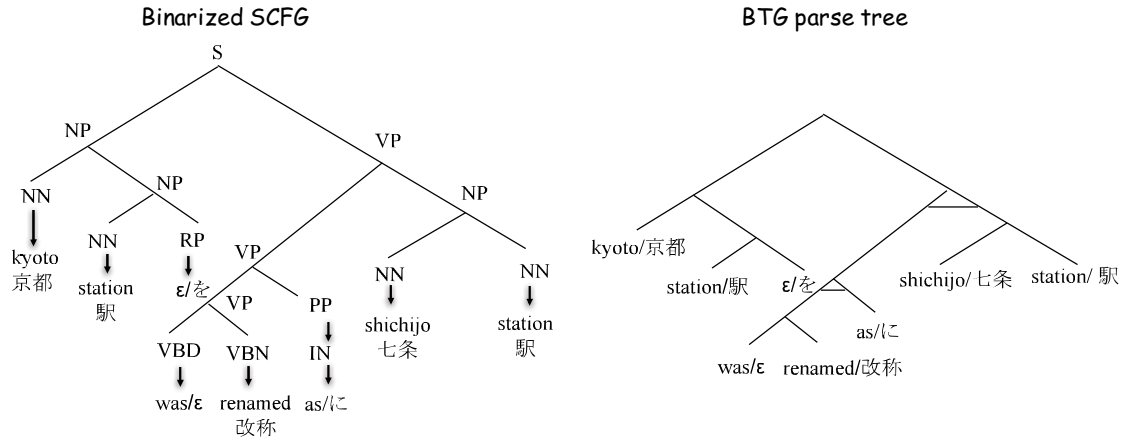


Figure 2.5: An example of a parse tree for binarized SCFGs and its corresponding BTG parse tree.

subtrees are permitted to cross in exact *inverted* order or *straight* order. This restriction greatly reduces matching flexibility during parsing. However, BTG formalism somehow limits the permutation of constituents, [Wu, 1997] shows that BTGs cannot generate 2 out of 24 permutations. One of them is $B_2D_4A_1C_3$ to $A_1B_2C_3D_4$ (2413 pattern). There also may exist more than one BTG tree to represent the same word reordering. E.g., the word reordering $C_3B_2A_1$ to $A_1B_2C_3$ has two possible BTG trees:

1. $X_{123} \rightarrow \langle X_{12}C_3 \rangle; X_{12} \rightarrow \langle A_1B_2 \rangle$

2. $X_{123} \rightarrow \langle A_1X_{23} \rangle; X_{23} \rightarrow \langle B_2C_3 \rangle$

Therefore, BTG is a simplified well-formed SCFG. It meets Chomsky Normal Form. Table 2.1 shows the mapping from binarized SCFG rules to BTG rules. We draw Figure 2.5 to make the connection between BTGs and SCFGs. The parse trees in Figure 2.5 are generated simultaneously.

Figure 2.5 allows visualizing the structure commodities of two sentences more clearly. BTG offers a compact representation with the bracketing notation:

[[**kyoto/京都 station/駅**] ⟨ ⟨ [**was/**ε **renamed/改称**] **as/に**⟩ [**shichijo/七条 station/駅**] ⟩]

Definition. 5 Probabilistic Context-free Grammar (PCFG) [Eddy and

Table 2.1: Binarized SCFG rules and their corresponding BTG rules.

Binarized SCFG rules		↔	BTG rules	
S	→ NP VP | NP VP		X	$\rightarrow [X_1 X_2]$
NP	→ NN NP | NN NP		X	$\rightarrow [X_1 X_2]$
NP	→ NN ADP | NN ADP		X	$\rightarrow [X_1 X_2]$
VP	→ VP NP | NP VP		X	$\rightarrow \langle X_1 X_2 \rangle$
VP	→ VP PP | PP VP		X	$\rightarrow \langle X_1 X_2 \rangle$
VP	→ VBD VBN | VBD VBN		X	$\rightarrow [X_1 X_2]$
PP	→ IN | IN		X	$\rightarrow Y$

Durbin, 1994,Charniak, 1997,Smith and Johnson, 2007] A PCFG includes a context-free grammar $G = (\mathcal{V}, \Sigma, R, S)$ and a parameter $\mathrm{q}(\alpha \rightarrow \beta)$ for each rule $\alpha \rightarrow \beta \in R$. In the left-most derivation, the parameter $\mathrm{q}(\alpha \rightarrow \beta)$ stanfs for the conditional probability of choosing the rule, i.e., $\alpha \rightarrow \beta$, given which the non-terminal being expanded, i.e., α. For any $X \in \mathcal{V}$, we have the constraint

$$\sum_{\alpha \rightarrow \beta \in R : \alpha = X} \mathrm{q}(\alpha \rightarrow \beta) = 1 \tag{2.2.9}$$

In addition, $\mathrm{q}(\alpha \rightarrow \beta) \geq 0$ for any $\alpha \rightarrow \beta \in R$.

These definitions are also appropriate for the bilingual case, e.g., stochastic BTG (SBTG) [Wu, 1997] and SCFG, in which a probability is associated with each bilingual rule.

Given a sentence pairs, the corresponding bilingual parse tree $\mathbf{T} \in \mathbf{T}_G$ that consists of SCFG rules $\alpha_1 \rightarrow \beta_1$, $\alpha_2 \rightarrow \beta_2$, . . . , $\alpha_n \rightarrow \beta_n$, the probability of $\mathbf{T}$ under PCFG is just the product of the probabilities of the rules used to produce the parse tree:

$$P(\mathbf{T}) = \prod_{i=1}^{n} \mathrm{q}(\alpha_i \rightarrow \beta_i) \tag{2.2.10}$$

Parsing a source sentence $\mathbf{F}$ and a target sentence $\mathbf{E}$ with a PCFG consists in finding the find the highest scoring parse tree $\tilde{\mathbf{T}}$ that produce the sentence pair

$(\mathbf{F}, \mathbf{E})$.

$$\tilde{\mathbf{T}} = \underset{\mathbf{T} \in \mathbf{T}_G}{\operatorname{argmax}} P(\mathbf{T}) \quad \mathbf{T} \rightarrow (\mathbf{F}, \mathbf{E}) \tag{2.2.11}$$

The goal of syntax-based machine translation techniques is to incorporate an explicit representation of syntax into the statistical systems to obtain the best translation while not requiring intensive human efforts.

2.3 Neural Network Model

In this section, we give a brief introduction to NMT and a classification of NMT models.

2.3.1 Sequence-to-Sequence Model

Recently, sequence-to-sequence (seq2seq) neural machine translation systems have achieved great performances in large-scale translation tasks [Sutskever et al., 2014b, Bahdanau et al., 2015, Cho et al., 2014, Luong et al., 2015a]. Unlike traditional phrase-based statistical machine translation (SMT) [Koehn et al., 2003] that perform translation by breaking up source sentences into multiple chunks and makes use of phrase pairs for translation, neural machine translation treats words as atomic units for processing. It employs a single neural network to train the joint (translation/reordering/language) model, instead of using the plain sentence. This kind of neural networks requires word embedding vectors [Mikolov et al., 2013] to give better semantic representations for the input. Nowadays, NMT has dominated as a powerful alternative to conventional SMT models given its outperformance over other translation models in many cases [Bentivogli et al., 2016].

In general, seq2seq NMT is an end-to-end approach to machine translation. The most successful approach illustrated by [Sutskever et al., 2014b, Bahdanau et al., 2015, Cho et al., 2014, Luong et al., 2015a] share the similar sequence-to-sequence transduction model. In these models, the source sentence is encoded into a variable-length representation with a bi/mono-directional recurrent neural network (RNN) [Sutskever et al., 2014b]. The translation is then generated with another RNN from left to right. Sometimes, it is equipped with a soft-attention mechanism [Bahdanau

et al., 2015, Luong et al., 2015a]. Recurrent networks are typically parameterized as long short-term memory networks (LSTM) [Hochreiter and Schmidhuber, 1997, Pham et al., 2013] or gated recurrent units (GRU) [Cho et al., 2014]. Since residual connections or skip connections between layers encourage gradient flow, such RNNs are commonly a stack of several layers [Wu et al., 2016, Zhou et al., 2016].

Seq2seq NMT models commonly make use of 'words' as atomic units in consideration of computational complexity. They train a joint reordering and translation (alignment) model for performing translation. Given a source sentence $\mathbf{F} = (f_1, \ldots, f_m)$, assuming $\mathbf{E} = (e_1, \ldots, e_n)$ is the corresponding target sentence, the translation probability of a target sentence $\mathbf{E}$ is parameterized by a neural network model using the chain rule:

$$P(\mathbf{E}|\mathbf{F}) = \prod_{j=1}^{n} p(e_j|f_0, e_1, \ldots, e_{j-1}; f_1, \ldots, f_m) \tag{2.3.12}$$

$$= \prod_{j=1}^{n} p(e_j|e_{<j}, \mathbf{F}) \tag{2.3.13}$$

with j being the time step during decoding. The decoder predicts each new target word e_j given the source sentence encoding and the decoded target sequence so far.

At time j, it computes the probability of the next state through a softmax layer with weights $\boldsymbol{W_s}$ and the attentional hidden state $\tilde{\boldsymbol{h}}_j$ output by the attention layer as:

$$p(e_j|e_{<j}, \mathbf{F}) = \text{softmax}(\boldsymbol{W_F}\tilde{\boldsymbol{h}}_j) \tag{2.3.14}$$

Considering the structure on the encoder side, in the framework of bidirectional LSTM, forward states and backward states may benefit the representation using RNN. To this end, the encoder concatenates both the forward/backward hidden states $\overrightarrow{h}_i$ and $\overleftarrow{h}_i$ so as to obtain a complete source hidden state h_i.

$$h_i = \text{concat}(\overleftarrow{h}_i; \overrightarrow{h}_i) \tag{2.3.15}$$

Here, $\overrightarrow{h}_i$ or $\overleftarrow{h}_i$ are the RNN hidden units (i.e., an LSTM network) in each direction. For example, $\overrightarrow{h}_i$ is abstractly computed based on the previous state $\overrightarrow{h}_{i-1}$ as:

$$\overrightarrow{h_i} = f(\overrightarrow{h}_{i-1}, \mathbf{F}) \tag{2.3.16}$$

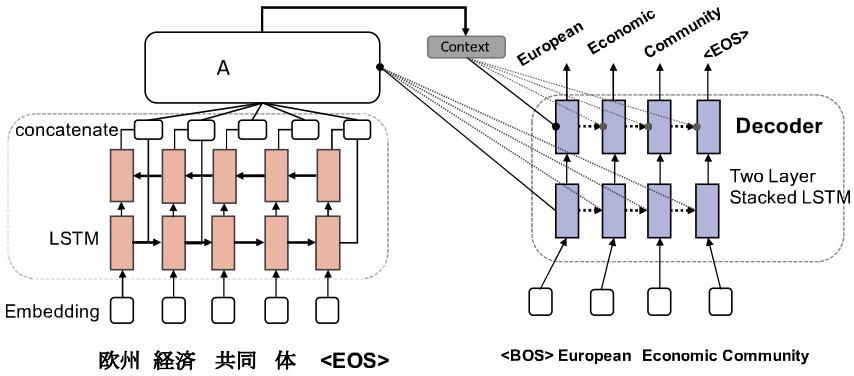


Figure 2.6: Bidirectional LSTM encoder-decoder architecture with attention mechanism for NMT. Given the Japanese input, the NMT system output the corresponding English translation.

On the decoder side, the RNN is slightly different. There are two layers in a decoder. The first la la lobal alignment information for the whole source sentence from the attention mechanism. In the attention mechanism of [Luong et al., 2015a], an additional attention layer is incorporated. Let $\tilde{\boldsymbol{h}}_j$ stand for the attentional hidden state and h'_j represent the target hidden state produced at time j. Given the source-side context vector $\boldsymbol{c}_j$, $\tilde{\boldsymbol{h}}_j$ is computed using the following formula:

$$\tilde{\boldsymbol{h}}_j = \tanh(\boldsymbol{W_c}[c_j; h'_j]) \tag{2.3.17}$$

c_j is the weighted average that depends on alignment score $\boldsymbol{a}_j(i)$ that weights the source word F_i with the target word E_j in a soft alignment matrix:

$$c_j = \sum_{i=1}^{I} \boldsymbol{a}_j(i) h_i \tag{2.3.18}$$

In this model, the length of $\boldsymbol{a}_j$ is variable. It is equal to the number of time steps on the source side. It can be computed using a simple dot-product function [Luong et al., 2015a]:

$$\boldsymbol{a}_j(i) = \frac{\exp\left(h'^{\top}_j h_i\right)}{\sum_{i'} \exp\left(h'^{\top}_j h_i\right)} \tag{2.3.19}$$

Training this kind of model is effective, because both source and target words are known. The training objective function aims at minimizing the negative log-

likelihood of the translation probability, which is defined as:

$$J_t = \sum\nolimits_{(\mathbf{F},\mathbf{E})\in\mathbf{C}} -\log\ P(\mathbf{E}|\mathbf{F}) \tag{2.3.20}$$

with $\mathbf{C}$ being our parallel training corpus. However, when decoding a new sentence, the summation operation will iterate over all target words in the vocabulary. This procedure is much time-consuming. The time complexity increases in proportion to the size of the target vocabulary. Thus it is worth noticing that it is necessary to limit the vocabulary size on a large corpus. It will be the goal of Chapter 5.

In our experiments, we utilise the NMT system that follows the encoder-decoder architecture with the attention mechanism [Luong et al., 2015a]. The encoder is a bidirectional RNN with LSTM, and the decoder are two-layer stacked RNN [Sutskever et al., 2014a] with LSTM. Figure 2.6 illustrates the architecture of the used model in details.

2.3.2 Other Models

Other successful neural models proposed after that such as the convolutional neural network (CNN) encoder model [Gehring et al., 2017] and self-attention (transformer) model [Vaswani et al., 2017] are competitive with recurrent network alternatives. These models also still follow the encoder-decoder architecture, but mainly differ in the structure of the encoder or decoder. Our proposal should be well suited these models as well.

Chapter 3

Exploiting Bidirectional Latent Variable Models in Phrase-based SMT: Phrasal Alignment

This chapter focuses on the study of phrasal alignment for distant language pairs. We present a novel symmetric phrasal alignment method, taking the advantages of both IBM model 2 and the hierarchical sub-sentential alignment method. The main content of this chapter is on the basis of the following papers:

- Wang, H. and Lepage, Y. (2016b). Yet another symmetrical and real-time word alignment method: Hierarchical sub-sentential alignment using F-measure. In *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation (PACLIC 30)*, pages 143–152. (Best Paper Award)

- Wang, H. and Lepage, Y. (2016a). Combining fast_align with hierarchical sub-sentential alignment for better word alignments. In *Proceedings of the 6th Workshop on Hybrid Approaches to Translation (Hytra 6), the 26th International Conference on Computational Linguistics (COLING 2016)*, pages 1–7

- Wang, H. and Lepage, Y. (2017b). Hierarchical sub-sentential alignment with IBM models for statistical phrase-based machine translation. 自然言語処理 *(Journal of Natural Language Processing)*, 24(4):619–646

Phrasal alignment is a fundamental latent variable in phrase-based SMT from which translation models are built. Conventional methods rely on asymmetric models of high complexity, resulting in long training times and large model sizes. Hence, a fast aligner which directly outputs symmetrical alignments is very attractive. This chapter describes a novel symmetric phrasal alignment method based on a bidirectional latent variable model meeting the demand for the fast symmetric phrasal aligner. Compared to other state-of-the-art methods, our contributions are located in:

1. **Parameter Initialization**

   We propose a novel fast approximate estimation method for IBM model 2 based on Variational Bayes [Riley and Gildea, 2012] which less sensitive to noise caused by function words. This leads to (b) faster training and (a) better estimation of initial word-to-word translation probabilities.

2. **Hierarchical Phrasal Alignment**

   The main shortcoming of the original HSSA implementation is that it may yield locally optimal solutions due to the greedy heuristic used. We promote alignment accuracy by replacing the greedy search with beam search. This helps to deal with undetermined alignments and to find the best global alignments. This strategy (a) significantly reduces alignment errors.

3. **Symmetric Phrasal Extraction**

   Our proposal is intrinsically bidirectional, dispensing with additional processes of symmetrization. During the alignment process, it forces the final alignments under the symmetric constraints. This leads to (a) more accurate alignments and (c) smaller phrase tables.

This chapter is organized as follows.

- Section 3.1 briefly introduces the background of phrasal alignment and state-of-the-art alignment methods. We also point out the existing problems.
- Section 3.2 reviews related works on word alignment. The traditional methods of Viterbi alignment and phrase extraction heuristics are introduced.
- Section 3.3 presents our proposed method. The symmetric phrasal alignments are jointly extracted using a hybrid method. We justify the proposed method with mathematical principles.
- In Section 3.4, we give the details of the experimental settings and experimental results on word alignment and machine translation. The evaluation results show that the proposal, i.e., the latent bidirectional phrasal alignment model has shorter training times and smaller translation tables, which is beneficial to phrase-based SMT.
- In Section 3.5, we summarize our work on this research and highlight our contributions.

3.1 Background

Phrasal alignment is a fundamental pre-processing step in phrase-based SMT. The traditional way to extracting phrasal alignments constitutes in a three-step process: *word alignment* (asymmetric), *symmetrization* and *phrase extraction*. Since the quality of phrase alignment depends on the quality of word alignment, high accuracy word aligners are essential.

To obtain high-accuracy word alignments, most of the previous methods are based on generative models, like the IBM models of [Brown et al., 1993] or the HMM-based model of [Vogel et al., 1996]. These unsupervised methods commonly estimate model parameters using the EM algorithm [Dempster et al., 1977]. [Liang et al., 2006] classified them into two groups:

- sequence-based models (IBM model 1, 2 and HMM model)
- fertility-based models (IBM model 3, 4 and 5)

F lity-based (high-level) models are more difficult to implement than simpler sequence-based models and often become intractable. Although only using sequence-based models can produce word alignments, one of the major problems is *discontiguous alignment*. To explain this new term, we use an example in Figure 3.1, under the definition of IBM models for alignment, we can map multi-indices of the source side to the same index of the target side, and it is possible that some indices on the target side are not being mapped. Phrase-based SMT requires that translation fragments, i.e. phrase pairs, are contiguous when building the translation hypothesis, it relies more on alignments of multi-word units (*many-to-many*). The existence of discontiguous alignments prevents the sequence-based models being directly used to phrase extraction. This weakness will have a significant influence on the quality of the final translation. [Liang et al., 2006, Wang et al., 2016] point out discontiguous alignments produced by sequence-based models increase the difficulty and complexity in the decoding step.

Given the drawback of discontiguity in sequence-based models, high-level fertility-based models were developed. Sequence-based models often serve as the sub-components for parameter initialization in the fertility-based models. The fertility-based models

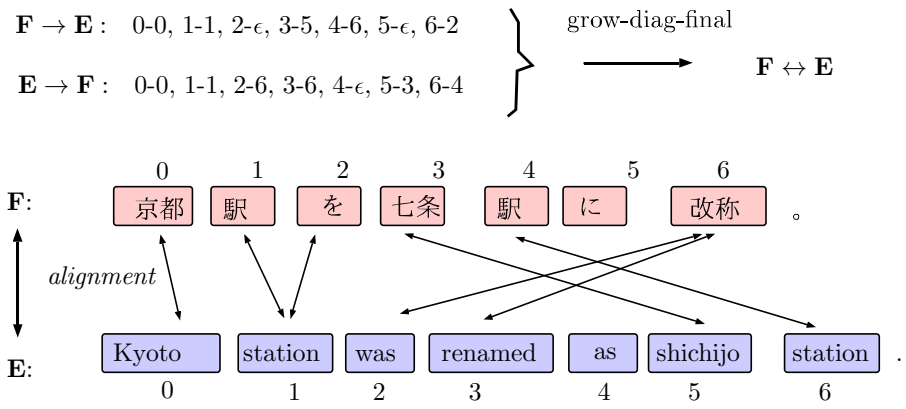


Figure 3.1: Example of bidirectional word-to-word alignments between a source (Japanese) sentence and a target (English) sentence.

are expected to generate more word alignments to fill the empty cells that are close the cells with the high probability being aligned in alignment matrices. These models have shown state-of-the-art results, but we cannot ignore the fact that high-level IBM model 3 and the aboves are often criticized for their complexity, which has been reflected in the time needed for training.

Aiming at simple and effective models that scale well, [Dyer et al., 2013] develop a variation of IBM model 2 (i.e., a sequence-based model) to output word alignment. This dispenses the use of high complicated fertility-based models. Their implementation, called (`fast_align`[1]) is faster to train than the traditional methods, and that it produces alignments that lead to comparable translation quality. Their model reparameterizes IBM Model 2 with a log-linear framework combining both position distribution and alignment distribution. This model favours alignment points close to the diagonal in an alignment matrix. This assumption is flawless. However, such a position-featured method does not suit the alignment of distant language pairs. [Ding et al., 2015] demonstrate that `fast_align` does not surpass `GIZA++` on Japanese–English and English–German experiments, because word orders in both language pairs are distinct.

Phrase-based SMT relies on asymmetric (mono-directional) word alignment meth-

[1]https://github.com/clab/fast_align

ods. This may be not suitable for a bilingual task like machine translation. To obtain symmetric alignments, [Och and Ney, 2003] propose several strategies training models in both forward and reverse directions, whereafter merging the outcome of monodirectional alignments with some symmetrization heuristics. Among these heuristics, the *grow-diag-final-and* heuristic (GDFA) has been shown to be most effective for phrase extraction for PB-SMT [Wu and Wang, 2007]. Other methods, contrary to heuristics, train the alignment model to maximize the agreement between two directional word alignments as the work of [Liang et al., 2006]. Some researchers also work on symmetric word alignment, using Bayesian techniques such as Gibbs sampling [Mermer and Saraclar, 2011]. Such work is beyond the scope of this dissertation because our goal is to train phrasal alignments using the minimal time cost. Therefore, it is natural to look for the methods which directly outputs symmetric phrasal alignments.

Another challenge in phrasal alignment, especially for distant languages, is modeling different word permutations in the source and target sentences. Since the diversity of natural languages, e.g., English and Japanese, the word orders of the source and target sentences may be entirely different. There is no substantial relationship between the next alignment and the previous alignments in a sequence. This contradicts the position assumption in the HMM model and IBM model 2.

3.2 Related Works

3.2.1 Viterbi Alignment

State-of-the-art word alignment models apply statistical estimation to obtain the most possible/Viterbi alignments, which have massive parameters (e.g., word translation probabilities) including the desired hidden alignment variables. In these models, word alignment is treated as a hidden variable [Och and Ney, 2003]. The problem of translation is defined as:

$$Pr(\mathbf{E}|\mathbf{F}) = p(\mathbf{E}, \mathbf{a}|\mathbf{F}) \tag{3.2.1}$$

$$= \sum_{\mathbf{a}} p(e, \mathbf{a}|f) \tag{3.2.2}$$

where **a** is the latent alignment that stands for the mapping from a source position i to a target position a_i (asymmetric or mono-directional). The symbol $p(\cdot)$ denotes general probability distributions. m, n are the lengths of the source and target sentences respectively. For example, in IBM model 2, to find the best alignment, we train the model towards maximizing the likelihood on a parallel corpus as:

$$p(\mathbf{E}, \mathbf{a}|\mathbf{F}) = \prod_{j=1}^{n} \sum_{i=0}^{m} \theta(e_j|f_{a_j}) \delta(a_j = i|j, m, n) \tag{3.2.3}$$

where $\mathbf{F} = f_1, \ldots, f_m$ represents the source sentence and $\mathbf{E} = e_1, \ldots, e_n$ represents the target sentence. The translation scores $\theta(e_j|f_{a_j})$ are parameterized by an appropriate local conditional probability distribution[2].

$$\delta(a_j = i|j, m, n) = \begin{cases} p_0 & i = 0 \\ (1 - p_0) \times h(a_j = i|j, m, n) & i \neq 0 \end{cases} \tag{3.2.4}$$

For the distortion function $\delta(a_j = i|j, m, n)$, [Liang et al., 2006] summarize the most popular IBM models as follows:

$$h(a_j = i|j, m, n, a_{j-1} = i') \propto \begin{cases} 1 & \text{IBM 1} \\ \phi(\frac{i}{m}, \frac{j}{n}) & \text{IBM 2} \\ \phi(i, i') & \text{HMM} \end{cases} \tag{3.2.5}$$

In the above formula, i' denotes the last alignment point $(i', j-1)$ at the end of the current alignment sequence. Typically, IBM model 1 assumes a uniform distribution for $p(a)$, which actually means that the word order of the sentences are considered irrelevant. This is clearly not true in real translated sentences for most language pairs. However, a_j and a_{j-1} tend to be strongly correlated. It is reasonable for the case in English-French, but not for English-Japanese. To capture the dependency between a_j and a_{j-1}, most research on word alignment has assumed some version of a word order model. For example, [Dyer et al., 2013] use a variation of IBM model 2 where the observation is made that

$$i/m \approx j/n \tag{3.2.6}$$

[2] p_0 is a null alignment probability.

This is a very rough but efficient approximation. Another kind of models is HMM [Vogel et al., 1996] which directly models the relationship as:

$$P(a_j - a_{j-1} = x|m) \tag{3.2.7}$$

This describes the length x of "jump" in the source sentence when moving one word forward in the target sentence, conditioned on the source sentence length n. However, sentences in Japanese and English tend to have the different word orders.

3.2.2 Symmetrization

Based on the above model, a Viterbi alignment model returns mono-directional alignments ($\mathbf{F} \rightarrow \mathbf{E}$ or vice-versa) which maximizes the following formula:

$$\hat{\mathbf{a}} = \hat{a}_1^n = \underset{a_j \in \{\text{all alignments}\}}{\text{argmax}} \sum_{\mathbf{j=1}}^{n} p(e_j, a_j = i|f_i) \tag{3.2.8}$$

The reason of discontinuity in word alignments is simple, because conflicts exist in two mono-directional Viterbi alignments. Although the influence of discontinuity can be reduced by combining two sets of alignments $\hat{\mathbf{a}} = \hat{a}_1^n$ ($\mathbf{F} \rightarrow \mathbf{E}$) and $\hat{\mathbf{b}} = \hat{b}_1^m$ ($\mathbf{E} \rightarrow \mathbf{F}$) into one alignment matrix using the *grow-diag-final-and* algorithm, there may still exist some unaligned target words. Given the merged alignment, we can easily factor the lexical translation probabilities. Obviously, the existence of unaligned target words makes the process of phrase extraction to produce more translation fragments, and it is reasonable to allow *null-to-1* alignments when iterating over all possible boundaries in target side. This strategy largely increases the size of the extracted phrase table. To alleviate this problem, we force to align each source word with at least one target word and vice versa. This forbids the appearance of any unaligned source/target words. In our experiments, we found that this strategy is effective and that it reduces the size of the phrase tables produced.

3.2.3 BTG-based Word Alignment

There has been some interest in using BTGs for the purpose of alignment [Zhang and Gildea, 2005, Wang et al., 2007, Xiong et al., 2010, Neubig et al., 2012b]. Ini-

tially, the BTG formalism [Wu, 1995] offers a simplified case of synchronous context-free grammars, which are efficient for synchronous parsing. [Cherry and Lin, 2007] demonstrated that using BTGs improves word alignment.

The biggest barrier for applying BTG with Viterbi alignment is the time complexity of CYK parsing ($O(n^6)$, for the bilingual case). It is hard to deal with long sentences or large grammars in practice because the complexity of word alignment grows exponentially with the length of the source and the target sentences. Previous research attempts to reduce the computational complexity of BTG parsing with some pruning methods. [Zhang and Gildea, 2005] propose as so-called *tic-tac-toe* pruning method in which they extend BTGs with additional lexical information based on IBM model 1 Viterbi probability. [Haghighi et al., 2009] investigate pruning based on the posterior predictions from two joint estimated models. [Li et al., 2012] present a simple beam search algorithm for searching Viterbi BTG alignments.

Phrase extraction using discontiguous word-to-word alignments pivotally may have problems, because it is incapable of dealing with certain translation phenomena. The constraint of BTGs provides a natural, alternative way to reduce the search space for phrase extraction. By estimating the joint phrase alignment relation directly, it eliminates the need for any of the conventional heuristics (see Section 3.2.2) to generate contiguous word alignments. Following this idea, [Neubig et al., 2011b] propose a model-based heuristic for phrase extraction under the constraint of BTGs.

3.2.4 Hierarchical Sub-sentential Alignment

Hierarchical sub-sentential alignment (HSSA) [Lardilleux et al., 2012] was first introduced as a compliment for `Anymalign`[3]. Given the soft alignment matrix built using the parameters `Anymalign` output, HSSA takes all cells in the soft alignment matrix into consideration. This relies on a precise criterion to determine a good partition similarly as image segmentation.

HSSA makes use of an unsupervised clustering criterion called *normalized cuts* [Shi and Malik, 2000, Zha et al., 2001], or *Ncut* for short, to recursively segment

[3]https://anymalign.limsi.fr/

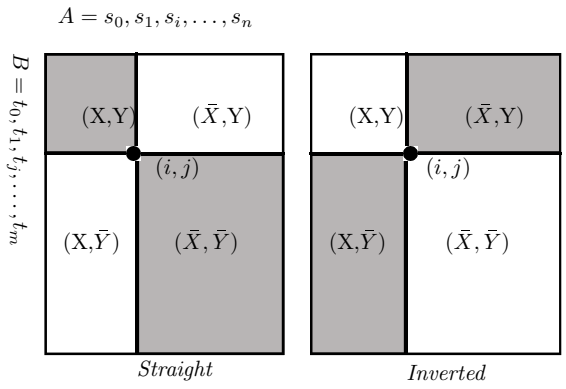


Figure 3.2: Bipartitioning example for hierarchical sub-sentential alignment method.

the matrix into two parts. During the segmentation processing, HSSA is supervised by the BTG constraint to select the search scope of next level on the diagonal (rep: anti-diagonal) corresponding exactly to the case of *straight* (rep: *inverted*.) HSSA terminates when all words in the source and target sentences are aligned and generate symmetric alignments at the same time.

Consider a source phrase $A:\ X\,\overline{X}$ and a target phrase $B:\ Y\,\overline{Y}$ (see Figure 3.2), which can be split at source index i and target index j in a dichotomic way. The sub-spans X, $\overline{X}$ in source side are corresponding to Y, $\overline{Y}$ or $\overline{Y}$, Y. According to the definition of [Shi and Malik, 2000], assuming the segmentation option as $\gamma \in \{0, 1, 2\}$ (diagonal, anti-diagonal and terminal), the association within groups $asso(A, B)$ and the risk of cutting *cut* at point (i, j) is defined by the following formula:

$$asso(A, B) = \sum_{f \in A} \sum_{e \in B} w(f, e) \tag{3.2.9}$$

where $w(f, e)$ stands for the weighted score that f is aligned with e.

$$cut(i, j|\gamma = 0) = \underbrace{asso(\ X,\ \overline{Y})}_{\text{left}} + \underbrace{asso(\ \overline{X},\ Y)}_{\text{right}}, \qquad \textit{straight} \tag{3.2.10}$$

$$cut(i, j|\gamma = 1) = \underbrace{asso(\ X,\ Y)}_{\text{left}} + \underbrace{asso(\ \overline{X},\ \overline{Y})}_{\text{right}}, \qquad \textit{inverted} \tag{3.2.11}$$

The optimal bi-partitioning of such a matrix is the one that minimizes this *cut* value. However, the minimum cut criterion favors cutting small sets of isolated nodes in the graph, which is counterintuitive, so [Shi and Malik, 2000] propose *Ncut* as a

measure for total normalized association within groups for a given partition. In our case, *Ncut* can be defined as:

$$Ncut(i,j|\gamma) = \frac{cut(i,j|\gamma)}{cut(i,j|\gamma) + 2 \times cut_{\text{left}}(i,j|\bar{\gamma})} + \frac{cut(i,j|\bar{\gamma})}{cut(i,j|\bar{\gamma}) + 2 \times cut_{\text{right}}(i,j|\bar{\gamma})}$$

Each possible splitting point (i, j) in the matrix divides the parent matrix into 4 sub-matrices ($X\ Y$, $X\overline{Y}$, $\overline{X}\ Y$, $\overline{X\ Y}$). Either the two sub-matrices on the diagonal ($X\ Y$, $\overline{X\ Y}$) or the two sub-matrices on the anti-diagonal ($X\overline{Y}$, $\overline{X}\ Y$) will be explored recursively on the next layer, referring to γ equal to 0 or 1. Recursive segmentation consists in determining these indices (i, j) which minimize $Ncut(i, j|\gamma)$ or $Ncut(i, j|\bar{\gamma})$ over all possible indices.

HSSA seeks a criterion, called *Ncut*, in a recursive partition algorithm, minimizing the disassociation between the blocks unaligned while maximizing the associations within the blocks aligned. It is faster than the original BTG method $O(n^6)$. The worst case time complexity of top-down HSSA is cubic $O(m \times n \times \min(m, n))$ and the best case is $O(m \times n \times \log \min(m, n))$ in the length of the input sentence pair.

3.3 Proposal: Hierarchical Symmetric Phrasal Alignment

Recent work has shown that bidirecitonal latent variable models, such as phrasal alignment based on Bracket Transduction Grammars (BTG) [Wu, 1995], effectively constrains the search space of distortion in word alignment [Zens et al., 2004, Zhang and Gildea, 2005, Haghighi et al., 2009, Riesa and Marcu, 2010].

In this chapter, aiming at the high quality of symmetric phrasal alignments, we propose a novel yet simple symmetric BTG-based phrasal alignment method to solve the discontiguous word-to-word alignment problem. This model does not require training two mono-directional word alignment. Thus it is a bidirectional model. The proposed method also delivers smaller phrase tables compared with other state-of-the-art methods.

We consider that an efficient phrasal alignment method should address both the problems of discontiguous alignment in word alignment and unbalanced phrase pair in phrase extraction. In this chapter, we present a joint (word alignment + extraction) method for phrasal alignment.

We concentrate our attention on hierarchical phrasal alignment with additional BTG constraint. Instead of using synchronous parsing to search for Viterbi BTG alignments as in [Li et al., 2012], we employ HSSA [Lardilleux et al., 2012] to generate symmetric BTG blocks. The combination of HSSA and lower-level IBM models enables a joint training of the phrasal model for phrase-based SMT. This model allows to directly extract symmetric word alignments with the restriction that phrasal alignments must be constructed under the BTG constraints. Compared with previous BTG-based unsupervised/supervised methods for word alignment, our proposal takes a distinct two-phase process:

1. This symmetric method starts with adjacency matrices initialized using lexical translation probabilities. These probabilities are obtained via a fast approximate estimation, which relies on a variation of IBM model 2;

2. Then we apply an improved beam search version of the bi-partitioning algorithm used in [Lardilleux et al., 2012].

Thus, our approach performs word alignment and symmetrization jointly. It can be regarded as hybridization of BTG parsing and IBM models, which indeed is a bidirectional model. Figure 3.3 characterizes the differences between our proposal and the standard approach.

3.3.1 Building Soft Alignment Matrices Using Lower IBM Models

Given a source sentence $\mathbf{F} = f_1^m = f_1, \ldots, f_i, \ldots, f_m$ and a target sentence $\mathbf{E} = e_1^n = e_1, \ldots, e_j, \ldots, e_n$, the alignment associations between $\mathbf{F}$ and $\mathbf{E}$ can be regarded as a contingency matrix [Matusov et al., 2004,Moore, 2005,Liu et al., 2009], noted $\mathcal{M}$. m is the length of the source sentence in words and n the length on the target side. [Liu

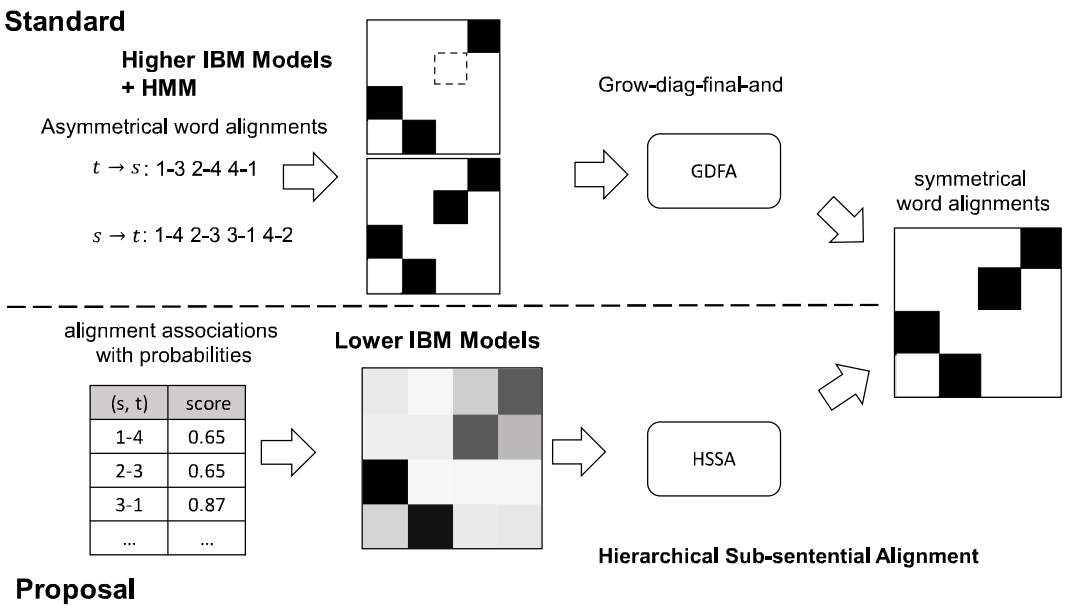


Figure 3.3: Comparison of standard bidirectional word alignment pipeline and our proposal.

et al., 2009] employ a weighted matrix that consists of the cells that correspond to an arbitrary word pair to extract phrase pairs. In such a matrix, each cell has been assigned with a probability score, which measures the confidence that two words are aligned. Following this definition, we define a function w which measures the probability of the alignment between any word pair (f_i, e_j). A weighted cell (i, j) in the alignment matrix presents the symmetric alignment between word f_i and e_j.

Let $\mathcal{M}$ be a soft bidirectional alignment matrix, i.e., a weighted adjacency matrix, which represents the graph of the sentence pair. Each pair of words in such a graph is connected with weighted edges between the nodes. Formally, we define a soft link $l = (f, e)$ to exist if f and e are probable translations. There are many ways to define the weights. Perhaps, the most simple way that can be imaged is using the posterior probabilities of IBM model 1. Given the existence of the data sparsity for counting, we apply Laplace smoothing to handle the unseen alignments. We also assign a smoothing parameter $p_0 = 10^{-4}$ in place of small values to force $l(f, e) \geq p_0$.

$$l(f,e) = \begin{cases} p_0 & \text{if } \ l = null \\ \sqrt{l(f|e) \times l(e|f)} & \text{otherwise} \end{cases} \tag{3.3.12}$$

After having computed the posterior probabilities using the EM algorithm, we obtain symmetrical scores, i.e. $l(f, e)$, by taking the geometric mean of the lexical translation probabilities in both directions: $l(f|e)$ and $l(e|f)$. Approximately, we use $l(f, e)$ instead of $w(f_i, e_j)$. Since we also aim at saving time, a faster and efficient training is essential. [Haghighi et al., 2009, Liu et al., 2010] propose different ways to perform word alignment with supervised BTG models. While they achieved promising results, such methods rely on a great number of features. On the contrary, our joint model is purely unsupervised. Other works like training with agreement [Liang et al., 2006] either requires more time or results in a computationally expensive process.

[Moore, 2005] points out that the simple lower-level IBM models have many disadvantages: it is sensitive to rare words and over-weighs high-frequency words, e.g., function words. For this reason, it is necessary to incorporate Variational Bayes (VB) [Riley and Gildea, 2012] into our model. We assume the distribution of the target vocabulary to be a Dirichlet distribution and apply VB with a symmetric Dirichlet prior as α to infer the translation probabilities in the M step of EM algorithm[4].

$$l(e|f) \sim \text{Dirichlet}(\alpha) \tag{3.3.13}$$

As an alternative, other probability models for estimation are also available, for example, [Zhang and Gildea, 2005] change the original IBM model with the max operator. Given asymmetrical alignments in both directions, reestimating the Viterbi probabilities is very fast. We found that such a process significantly reduces the size of the models learned. Furthermore, symmetrization heuristics, like *grow-diag-final-and*, can also be applied before re-estimating. In our experiments, we found that this process benefits in the alignment quality.

For a word pair (f_i, e_j), the position information (i, j) is a very important term in IBM model 2 or HMM model. We design our model as a log-linear framework to take the position information as a complementary component. It is expected to work under the condition that the sentence pair contains multiple possible word

[4] [Dyer et al., 2013] show that the prior $\alpha = 0.01$ is a proper value.

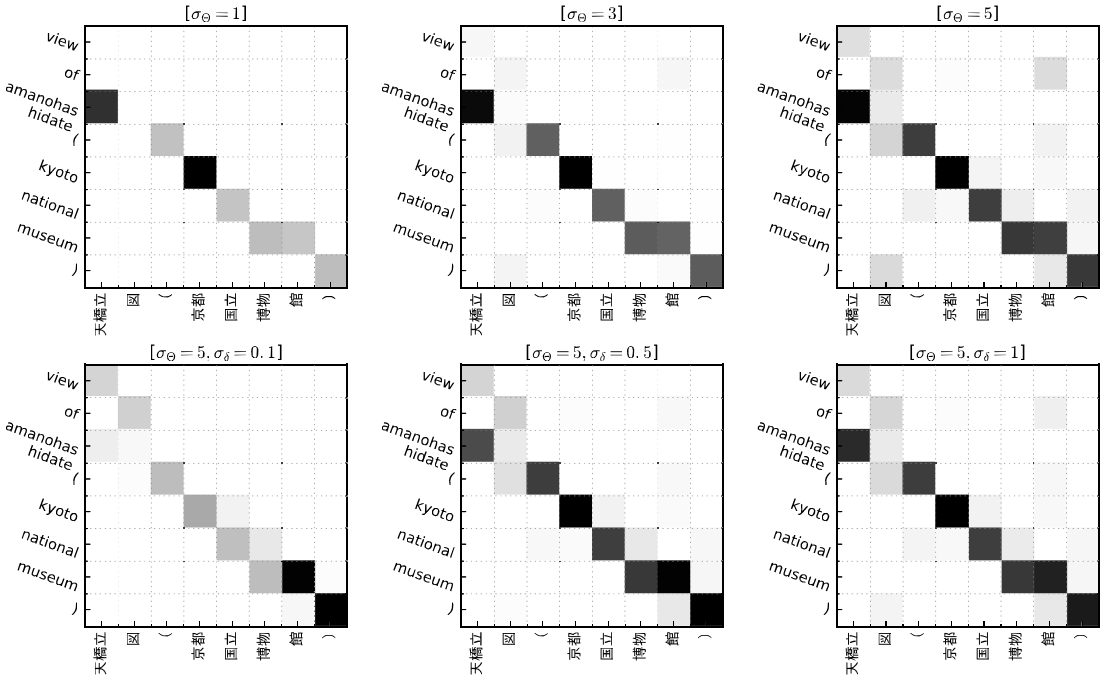


Figure 3.4: Grey-scale map of soft alignment matrices. *top*: three sub-matrices without the spatial proximity term; *bottom*: three sub-matrices with the spatial proximity term.

translation pairs for an identical (f, e). In other words, the case when $(f_i, e_j) = (f_{i'}, e_{j'})$. An effective way to define the alignment score $w(f_i, e_j)$ is to take the product of a feature translation term (translation probability) and spatial proximity term (relative position similarity) as in [Shi and Malik, 2000]:

$$w(f_i, e_j) = e^{\frac{\theta(f_i, e_j)}{\sigma_\theta}} \times \begin{cases} p_0 & \text{if } h(i,j,m,n) \geq r \\ e^{\frac{\delta(i,j,m,n)}{\sigma_\delta}} & \text{otherwise} \end{cases} \tag{3.3.14}$$

where w measures the strength of the translation link between a source word and a target word (f_i, e_j).

$$\theta(f_i, e_j) = \log(l(i,j)) \tag{3.3.15}$$

$$\delta(i,i,m,n) = \log(1 - h(i,j,m,n)) \tag{3.3.16}$$

$\theta(f_i, e_j)$ is a translation model and $\delta(i,i,m,n)$ is a distortion model. r is a threshold in the range $[0.5, 0.9]$, which depends on the language. σ_θ and σ_δ are hyper-parameters. To compute the value of $h(i,j,m,n)$, we assume $h(i,j,m,n) = |i/m -$

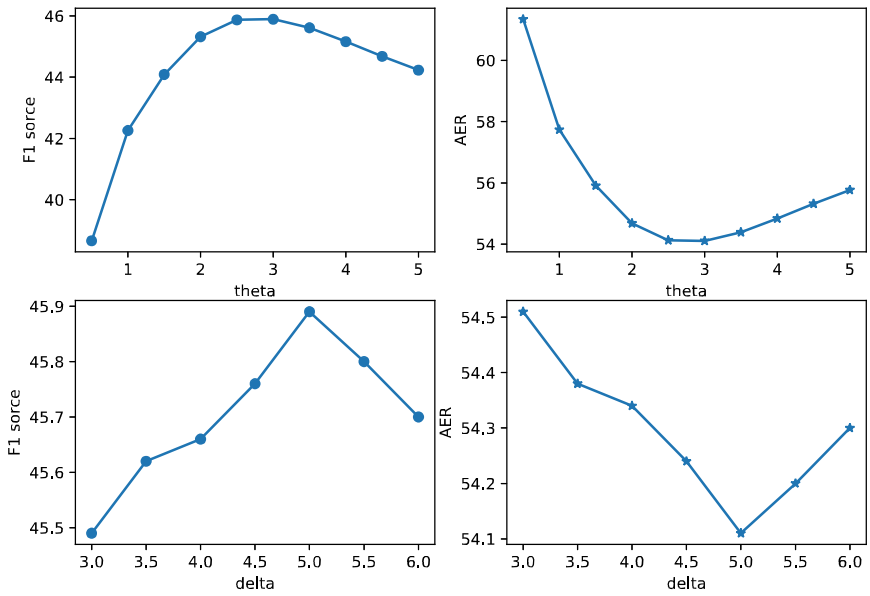


Figure 3.5: Determining the proper θ and δ value according to F1 score and AER on the KFTT corpus. When θ is equal to 3 and δ is equal to 5, it achieves the highest accuracy of word alignments.

$j/n|$. Figure 3.4 shows how soft alignment change when changing these hyper-parameters.

Although this is not necessary, we adjust values to a specified range $w(j,i) \in [{p_0}^2, 1)$. Since *Ncut* is already a normalized score, it does not require any normalization. The hyper-parameters σ_θ and σ_δ are fixed at the beginning of experimentation by maximizing the *Recall* in the preliminary experiments. Tuning such hyper-parameters for each language pairs would be expensive, but in practice, it is possible to obtain reasonable results without careful tuning. Figure 3.5 shows how these hyper-parameters affect the quality of bidirectional word alignments.

3.3.2 Efficient Hierarchical Phrasal Alignment with Beam Search

[Lardilleux et al., 2012] employ greedy (best-1) parsing to find the optimal *Ncut* at each layer. Experimentally, we found that the strategy of greedy parsing used in the original HSSA probably miss the best global derivation. To find the best derivation $\tilde{\mathbf{D}}$, we first define a score function $Score()$ aiming at the minimal value:

$$\tilde{\mathbf{D}} = \underset{\mathbf{D}}{\operatorname{argmin}}\ Score(\mathbf{D}_{Ncut}|\mathbf{F},\mathbf{E}) = \underset{\mathbf{D}}{\operatorname{argmin}}\ Score(\mathbf{D}_{Ncut}|\mathcal{M}) \tag{3.3.17}$$

D_{Ncut} stands for the parser derivation obtained according to $Ncut$. It can be shown that $Ncut$ indeed is a function of the arithmetic mean of two F-measures, notes F_{avg} [Wang and Lepage, 2016c] (see Appendix A.1). In fact, the derivation either $\mathbf{D}_{F_{avg}}$ or $\mathbf{D}_{Ncut}$ for two sub-matrices (X, Y) and $(\bar{X}, \bar{Y})$ is the same:

$$Ncut(i,j|\gamma) \propto 1 - \frac{F_1(X, Y) + F_1(\overline{X}, \overline{Y})}{2} = 1 - F_{avg} \tag{3.3.18}$$

The F_1 score is computed as:

$$F_1(X,Y) = 1 - \frac{asso(\overline{X}, Y) + asso(X, \overline{Y})}{2\times\ asso(X, Y) + asso(\overline{X}, Y) + asso(X, \overline{Y})} \tag{3.3.19}$$

With this interpretation, minimizing $Ncut$ is equivalent to maximizing F_{avg} . Intuitively, it suffices to replace $Ncut$ with F_{avg} to derive the following formula. The probability of a parsing tree or the probability of a sequence of derivations $\mathbf{D} = \{d_0, \ldots, d_K\}$ for the best word alignment $\hat{\mathbf{a}}$ based on the derivation $\tilde{\mathbf{D}}$ can be defined as:

$$\tilde{\mathbf{D}} = \underset{\mathbf{D}}{\operatorname{argmax}}\ Score(\mathbf{D}_{F_{avg}}|\mathcal{M}) = \underset{d_k\in\mathbf{D}}{\operatorname{argmax}}\ \prod_{k=0}^{K} F_{avg}(d_k) \tag{3.3.20}$$

$$\hat{\mathbf{a}} = Proj(\tilde{\mathbf{D}}) \tag{3.3.21}$$

$Proj()$ is a projection function which produces the final word-to-word alignments from the leaves of the BTG parser tree. Let d_k denote the operation of derivation at step k, in which d_k is defined as a triple $< i, j, \gamma >$. i stands for the index of the splitting point on the source side and j stands for the index of the splitting point on the target side. Figure 3.6 shows the top-down parsing with beam search. The incremental parsing algorithm used is presented in Algorithm 1 (see Appendix B).

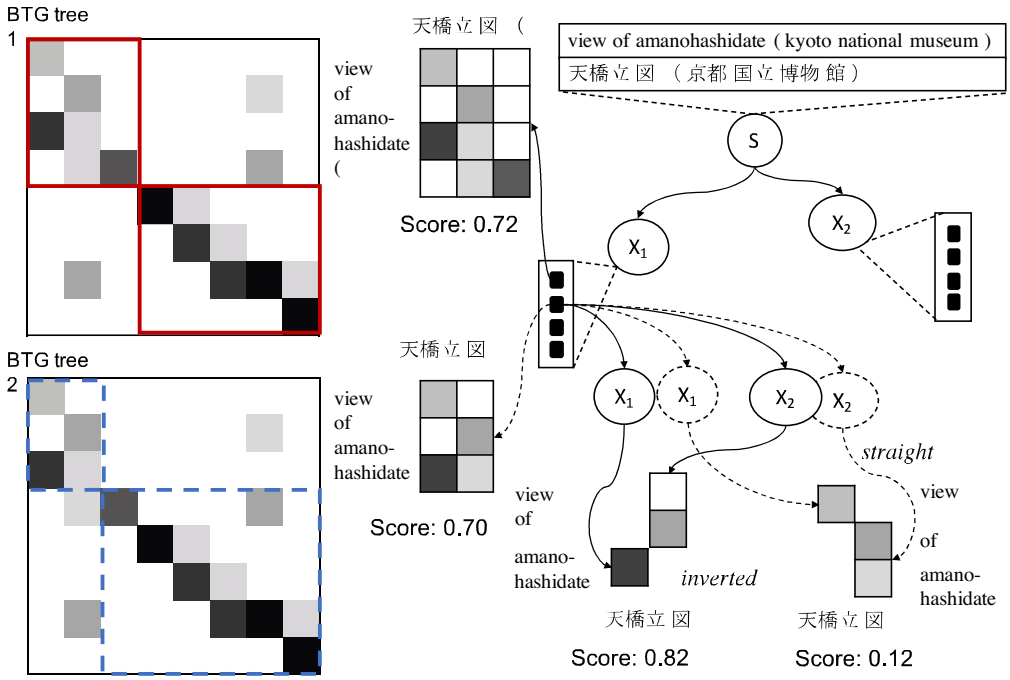


Figure 3.6: Hierarchical sub-sentential alignment with beam search as Top-down BTG forest parsing.

The parser works in the following way. Consider that the incremental parser has a parser hypothesis at each step. We define the hypothesis as a four-tuple $\langle P, Q, v, c \rangle$.

- P is a stack of the unsolved blocks.
- Q is a list of the previous derivations $\{d_0, \ldots, d_{k-1}\}$.
- v records the current score.
- c is *true* when terminated (stack P is empty).

A block $([i_0, i_1), [j_0, j_1))$ covers the source words from f_{i_0} to f_{i_1-1} and the target words from e_{j_0} to e_{j_1-1}. In the beginning, the initial hypothesis contains only a symmetric block which covers all the words in the source and target sentences. Then, we split the block in each step, and decide the node type (*straight* or *inverted*) when the splitting point is determined according to the defined score function. $top_k(S)$ returns the first k-th hypotheses from stack S in terms of their scores.

The computational complexity of the top-down parsing algorithm is $O(n \times m \times k \times \log \min(m, n))$ for sentence lengths of m, n and beam size of k. Thus, the parsing

depth is less than $\log \min(m, n)$. At each iteration, each hypothesis in the history is used to generate new hypotheses. Algorithm 2 gives the details of how the parser works.

To reduce the time complexity in the calculation of $asso(A, B)$, we make use of a specialized data structure for efficient computation. For each built soft alignment matrix, a summed area table (SAT) is created for fast calculating the summation of cells in the corresponding soft alignment matrix $\mathcal{M}(m, n)$. This pre-processing step is to build a new $(m + 1, n + 1)$ matrix $\mathcal{M}'$, where each entry is the sum of the sub-matrix to the upper-left of that entry. Any arbitrary sub-matrix sum can be calculated by looking up and combining only 4 entries in the SAT. For instance, assume that A, B extends from point (i_0, j_0) to point (i_1, j_1). We have,

$$asso(A, B) = \sum_{i=i_0+1}^{i_1} \sum_{=j_0+1}^{j_1} w(i, j) = \mathcal{M}'(i_1, j_1) - \mathcal{M}'(i_0, j_1) - \mathcal{M}'(i_1, j_0) + \mathcal{M}'(i_0, j_0) \tag{3.3.22}$$

Where the summation of all cells in the block of (A, B) if using of SAT greatly reduces the time complexity from $O(m \times n)$ to $O(1)$.

3.4 Experiments

3.4.1 Data

Word Alignment Datasets

The data sets for word alignment and translation tasks are from two different corpora. For word alignment sub-task, we use:

- Hansard Corpus[5] for English–French from the NAACL-2003 shared task [Mihalcea and Pedersen, 2003]
- KFTT Corpus[6] for English–Japanese.

Table 3.1 shows statistics on these datasets. The training set includes the test set.

[5]http://web.eecs.umich.edu/ mihalcea/wpt/index.html#resources

[6]http://www.phontron.com/kftt/

Table 3.1: Statistics of Hansard Corpus and KFTT Corpus for alignment evaluation in our experiment (M: million, K: thousand).

	# of	Hansard Corpus (en-fr)	KFTT Corpus (en-ja)
	lines	1,130,551	331,109
Train	tokens	20.02 M/23.61 M	5.97 M/6.12 M
	types	68.0 K/86.6 K	138 K/114 K
	lines	447	1,235
Test	tokens	7,020/7,761	30,822/34,366
	types	1,732/1,943	4,990/4,908

Translation Datasets

For the translation task, we conduct experiments in several language pairs:

- WMT 2008 Shared Task[7]
  - English–French (en–fr)
  - English–German (en–de)
- KFTT corpus
  - English–Japanese (en–ja)
  - Japanese–English (ja–en)
- Europarl Corpus[8]
  - English–French (en–fr)
  - Spanish-Portuguese (es–pt)

The training, development, test sets in translation evaluation experiments are created separately. Table 3.2 gives statistics on the training, development and test sets.

[7]http://www.statmt.org/wmt08/shared-task.html

[8]http://www.statmt.org/europarl/

Table 3.2: Statistics of data from WMT08 Shared tasks, KFTT Corpus and Europarl Corpus v7 for translation evaluation used in our experiments.

		WMT 08		**KFTT**	**Europarl v7**	
	# of	**en-fr**	**en-de**	**en-ja**	**es-pt**	**en-fr**
Train	lines	1.28 M	1.26 M	330 K	183.3 K	183.3 K
	tokens	32.2 M/33.5 M	29.3 M/31.6 M	5.91 M/6.09 M	5.27 M/5.02 M	4.95 M/5.23 M
Dev	lines	2,000	2,000	1,166	1,000	1,000
	tokens	53.1 K/55.1 K	5.31 K/48.8 K	24.3 K/26.8 K	36.4 K/34.5 K	35.3 K/36.3 K
Test	lines	2,000	2,000	1,160	2,000	2,000
	tokens	54.3 K/56.2 K	26.7 K/28.5 K	5.43 K/5.02 K	59.6 K/58.8 K	57.6 K/61.8 K

3.4.2 Experiment Settings

We first lowercase all data sets. In the case of `GIZA++`, we train word alignments in both directions with the default settings of the standard bootstrap for IBM model 4 alignment as:

- 5 iterations for IBM model 1
- 5 iterations for HMM model
- 3 iterations for IBM model 3
- 3 iterations for IBM model 4

For `fast_align`, we run 5 iterations. Word alignments are symmetrized using *grow-diag-final-and*. Finally, we evaluate with these symmertic alignments.

For our implementation `Hieralign`[9], we run 5 iterations of parameter updates to limit the run-time to that of `fast_align`. Since re-estimating the Viterbi probability is very fast when an initial word alignment is given for reference, we also employ various methods to compute $l(f, e)$. e.g.:

- IBM1
- IBM1 using Variational Bayes (VBIBM1)

[9]https://github.com/wang-h/Hieralign

- IBM1 Viterbi with heuristic (IBM1+VBH)
- IBM1 using Variational Bayes Viterbi with heuristic (VBIBM1+VBH)
- IBM2 Viterbi with heuristic (IBM2+VBH)[10]
- IBM4 Viterbi with heuristic (IBM4+VBH)

To confirm the advantage of our proposal, some comparison with other BTG alignment methods is necessary. For this consideration, we use another open-sourced BTG-based word aligner, `pialign` [Neubig et al., 2011b] [11]. `pialign` contains a hierarchical BTG models (hier) which allows to output symmetric phrasal alignment directly. For `pialign`, we run it with 8 threads and train the model with batch size 40 and only taking 1 sample during parameter inference. We extract phrases directly from the word-to-word alignment (1-to-many, many-to-1 and many-to-many) with traditional heuristic [Koehn et al., 2003] for translation. Our BTG-style parsing is on the basic of top-down bi-partitioning. This is different from previous CKY-based models [Xiong et al., 2010, Neubig et al., 2011b]. In the work [Neubig et al., 2011b], they employed another heuristic in phrase extraction rather the standard one [Koehn et al., 2003]. To make our work traceable, we compare the phrase-table size extracted with the simple heuristic-based phrase extraction [Koehn et al., 2005] with the model-based phrase extraction approach in [Neubig et al., 2011b]. For scoring phrases, four features are used in each phrase table for all experiments: the conditional phrase probabilities in both directions, $p(f|e)$ and $p(e|f)$, lexical weighting probabilities in both directions, $p_{lex}(f|e)$ and $p_{lex}(e|f)$ [Koehn et al., 2003][12].

For translation evaluation in all experiments, the phrase-based SMT systems are standard statistical machine translation systems built by using the `Moses`[13] toolkit

[10]We use the variation of IBM model 2 [Dyer et al., 2013], which is a fast and relatively simple model.

[11]http://www.phontron.com/pialign/

[12]In the experiment reported in [Neubig et al., 2011b], three additional features are employed (the joint probability of the phrase, the average posterior probability of a span and the uniform phrase penalty)

[13]http://www.statmt.org/moses/

[Koehn et al., 2007] with Minimum Error Rate Training [Och, 2003] and a 5-gram language model learnt using `KenLM` [Heafield, 2011]. We use the training set for training translation, lexical reordering, and target language models, the development set for tuning the parameters of the log-linear model in decoding and the test set for evaluation. For phrase extraction, we employ the traditional heuristic [Koehn et al., 2003] for all methods used in experiment. In our experiment, the maximum length of phrases entered into phrase table is limited to 7. Before that, *grow-diag-final-and* was used for `GIZA++`, `fast_align`. Specially for `pialign`, we make use of `pialign:itgstats.pl` [14] to extract the final word alignments used for phrase extraction. The baselines are the PB-SMT systems with the default distortion limit set to 6. We also filter the data to remove the long sentences (more than 100 words) from the training set. We do the same pre-processing (lower-casing and tokenization) using the scripts provided in `Moses` [15] and baseline processing as WAT[16] for English-Japanese and Japanese-English. Finally, we conduct translation experiments and compare `Hieralign`, `fast_align`, `pialign` and `GIZA++` for performance and time.

3.4.3 Alignment Evaluation

Word Alignment

Figure 3.7 plots different word alignments output by different methods. We found that `pialign` outputs asymmetric word alignments (1-to-m) while `Hieralign` outputs block alignments which is symmetrical (1-to-m/m-to-1).

To evaluate the performance of our proposed method, we evaluate the performance of various alignment methods in terms of precision, recall and alignment error rate (AER) as defined in [Och and Ney, 2003]. The quality of an alignment $A = \{(j, a_j)|a_j > 0\}$ is then computed by appropriately redefined precision and recall measures:

$$recall = \frac{|A \cap S|}{|S|} \tag{3.4.23}$$

[14]https://github.com/neubig/pialign/tree/master/script

[15]https://github.com/moses-smt/mosesdecoder/tree/master/scripts/tokenizer

[16]http://lotus.kuee.kyoto-u.ac.jp/WAT/WAT2015/baseline/tools.html

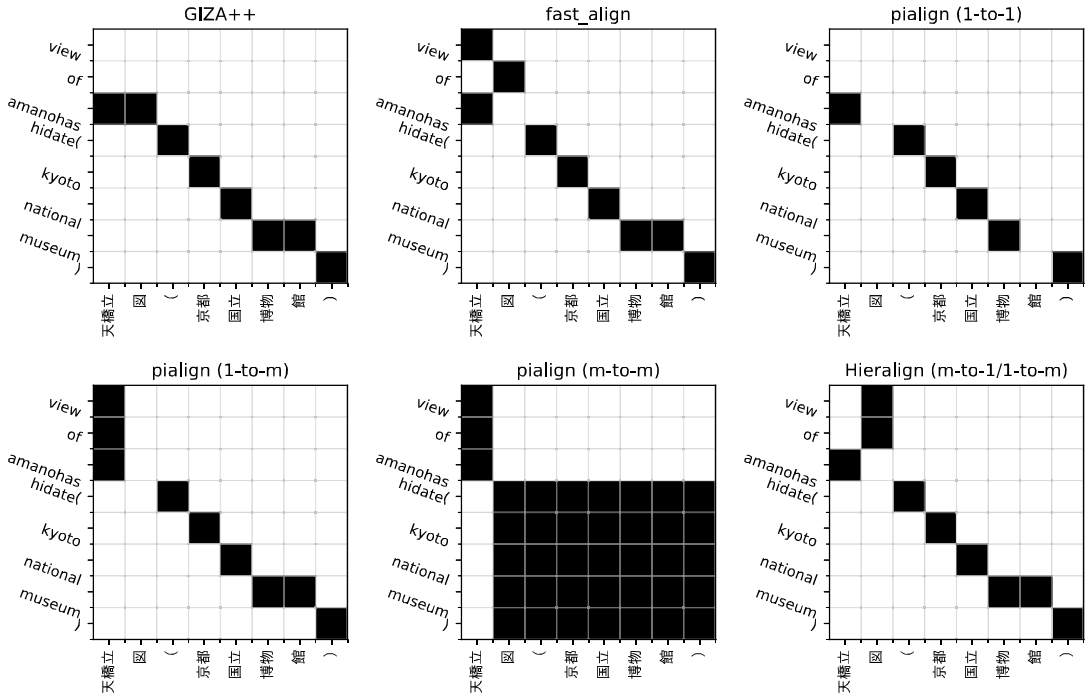


Figure 3.7: Example of alignment matrices output by `GIZA++` (+GDFA), `fast_align` (+GDFA) , `pialign` and `Hieralign` (our implementation).

$$precision = \frac{|A \cap P|}{|A|} \tag{3.4.24}$$

$$AER = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|} \tag{3.4.25}$$

where $S \sqsubseteq P$; S is *sure*/unambiguous alignment; P is *possible*/ambiguous alignment. The P label is used especially for free translations and missing function words.

"Test" means the total count of word-to-word alignments in the file output by each method ($|A|$). "Matches" means the count of "true positives" alignment found in the output of each method ($|A \cap P|$).

Table 3.3 shows the performance of various alignment profiles using the human annotated alignment data provided by the KFTT Corpus and Hansard Corpus. The first and second lines show the alignment difference using `GIZA++` + *grow-diag-final-and* and `fast_align` + *grow-diag-final-and*. From the table, we found both `GIZA++` and `fast_align` output less alignments than `Hieralign`. `Hieralign` tends to output more matches than `fast_align` but not as much as `GIZA++` from the point view of matching alignments and recall against the reference, which we can

Table 3.3: Evaluation of alignment results on Hansard Corpus and KFTT Corpus using various configuration of `GIZA++` (+GDFA), `Fast_align` (+GDFA), `pialign` and `Hieralign`.

	Hansard (French-English)					KFTT (Japanese-English)				
	Test	Prec	Rec	AER	Matches	Test	Prec	Rec	AER	Matches
asymmetric										
`fast_align`	7,845	80.66	92.62	15.27	6,328	25,368	55.49	42.17	52.08	14,076
`GIZA++`	7,709	87.99	96.14	9.21	6,783	31,342	59.48	55.85	42.39	18,641
`pialign`										
1-to-1	4,341	**95.51**	80.01	**11.96**	4,146	16,705	**74.93**	37.50	50.01	12,517
1-to-many	6,648	84.42	85.14	15.31	5,612	30,386	55.36	50.40	**47.24**	16,821
symmetric										
`pialign`										
many-to-many	16,295	51.14	**94.28**	40.29	**8,334**	57,053	33.99	**58.11**	57.11	**19,394**
`Hieralign:` `1-to-many/many-to-1` ($\sigma_\theta = 1$)										
IBM1	8,979	65.85	87.10	27.56	5,913	42,317	33.46	42.42	62.59	14,160
VBIBM1	9,016	68.23	88.86	25.39	6,152	42,465	34.39	43.76	61.49	14,605
IBM1+VBH	9,037	68.58	87.74	25.50	6,198	43,150	37.15	**48.03**	**58.11**	**16,030**
VBIBM1+VBH	9,012	68.36	88.81	25.31	6,161	42,974	**37.20**	47.90	58.12	15,986
IBM2+VBH	9,008	70.32	89.57	23.72	6,334	44,123	36.01	47.60	59.00	15,887
IBM4+VBH	8,958	**72.91**	**89.97**	**21.79**	**6,531**	43,257	35.16	45.57	60.31	15,209
`Hieralign:` `1-to-many/many-to-1` ($\sigma_\theta = 3. \sigma_\delta = 5$)										
IBM1	8,542	70.52	87.91	23.90	6,024	39,896	36.66	43.82	60.08	14,626
VBIBM1	8,540	71.02	87.72	23.62	6,008	39,107	37.22	43.82	59.75	14,627
IBM1+VBH	8,634	73.56	89.87	21.24	6,351	40,260	**42.43**	**51.18**	**53.60**	**17,082**
VBIBM1+VBH	8,638	74.09	90.56	20.66	6,400	40,214	42.10	50.72	53.99	16,929
IBM2+VBH	8,667	74.93	90.81	20.02	6,494	40,043	40.69	48.82	55.61	16,295
IBM4+VBH	8,668	**77.77**	**91.75**	**17.79**	**6,757**	40,543	40.31	48.96	55.78	16,343

explain the reason may be: although `Hieralign` cannot explore some alignments because of the limitation of BTG constraints, our force-align strategy improve the discontiguity problem caused in other asymmetric word alignment methods. Each source and target word are forced to align, it is prone to group the neighboring alignments aggressively into the same block. This strategy makes it to output more alignments. AER and precision are behind `fast_align`, even more than `GIZA++`.

When sampling the alignment results, we found that the output of the proposed method usually generates more alignments against the reference. From this table, we can also conclude that adding the distortion feature improves the alignment results.

Time

Table 3.4 shows the wall-clock time (minutes: seconds) required to obtain the symmetric alignments. Note that the total time also includes the time for symmetrizing asymmetric alignments for `GIZA++` and `fast_align` with the standard symmetrization heuristic: *grow-diag-final-and*. "online" means that Hieralign can run in the "online" mode (a trained model is given). This way of using provides an "online" service: when new sentences are input, it directly runs the second step to output word alignments.

Table 3.4: Wall-clock time (minutes:seconds) required to obtain the symmetric alignments.

	Hansard Corpus	**KFTT Corpus**
`pialign` + `pialign:itgstats.pl`	$\gg$240:00	199:46
`GIZA++` + `Moses:train-model.perl-3`	228:36	51:38
`fast_align` + `fast_align:atools`	6:58	1:40
`Hieralign` (beamsize=1)	6:43	1:33
`Hieralign` (beamsize=10)	8:36	2:03
`Hieralign` (online, beamsize=1)	1:36	0:22
`Hieralign` (online, beamsize=10)	3:21	1:13

Beam Size

Figure 3.8 shows that bigger beam size reduces AER until 20. Larger sizes are not helpful. In a real situation, a beam size of 10 should give a reasonable trade-off between time and accuracy (achieved comparable running time and accuracy with `fast_align`) as Table 3.5 and Figure 3.8 show.

3.4.4 Translation Evaluation

Recent research [Fraser and Marcu, 2007, Ganchev et al., 2008] questions the link between the word alignment quality metrics and translation results. Phrase-based SMT systems are built based on phrase pairs, not those word alignment directly.

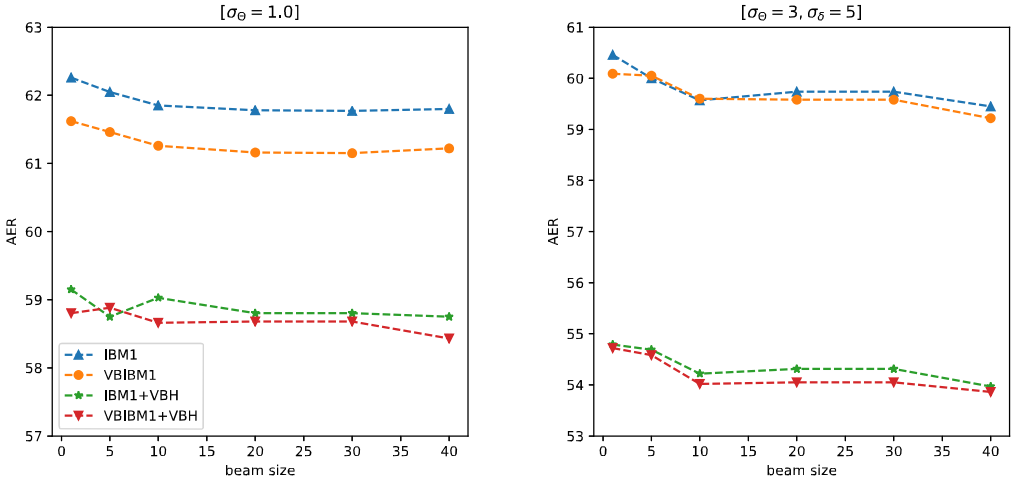


Figure 3.8: Determining the proper value for beam size according to AER on the KFTT corpus. Larger beam sizes have lower AER. The default beam size in `Hieralign` is setted as 10 given the trade-off between accuracy and speed.

There is no proof that improvements in alignment quality metrics lead to improvements in phrase-based machine translation performance. In other words, a lower AER does not imply a better translation accuracy, and we will discuss this conclusion in the following paragraphs.

Phrase Tables

For `Hieralign`, different initialization methods lead to the phrase tables with different sizes. Figure 3.9 gives the sizes of obtained models using different initialization methods. Figure 3.10 shows the comparison of phrase tables compared with other methods. Since we propose to solve the problem of discontiguity in phrase extraction, we force the word alignment at least 1-1, which should generate fewer entries in the translation tables. From Figure 3.10, we found that `pialign` (MOD) can significantly reduce the phrase table sizes while the traditional heuristic phrase extraction methods were not able to be observed. For our method `Hieralign`, we use (IBM 1+ VBH) to represent the remains, because tables obtained using `Hieralign` with differently initialized parameters have approximately the same size.

The findings are not unexpected but are relevant, the size of the phrase ta-

Table 3.5: Effect of beam size on alignment quality on Japanese-English data in terms of match number, precision, recall, AER and running time of HSSA.

	Test	**Matches**	**Prec**	**Rec**	**AER**	**CPU time** (sent./sec.)
Ref	33,377					
Best-1	43,017	15,879	36.91	47.57	58.43	3,389
Best-5	43,052	15,969	37.09	47.84	58.21	1,602
Best-10	43,150	16,030	37.15	48.03	58.11	993
Best-20	43,137	**16,036**	**37.17**	**48.05**	**58.08**	662
Best-40	43,137	16,007	37.11	47.96	58.16	336

IBM1+VBH with beam search ($\sigma_\theta = 1$).

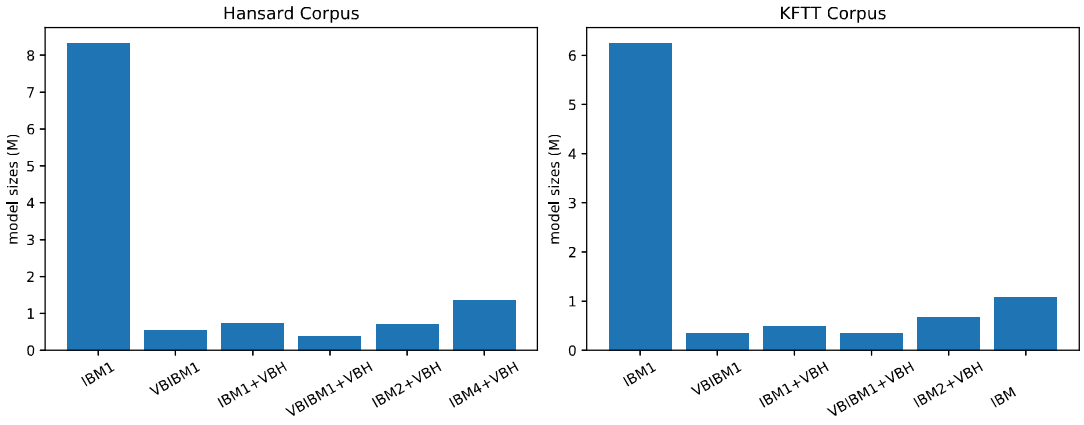


Figure 3.9: Comparison of model sizes (# of entries) with different initialization methods.

ble obtained from the alignments produced by `Hieralign` is smaller by a third in comparison to those of the baseline in (en–fr, en–de, es–pt) (see Figure 3.10, right). For `pialign`, we also give the size of the table using model-based extraction (MOD) [Neubig et al., 2011b]. In en–ja and ja–en, that is reduced by even two thirds (see Figure 3.10, left). Compare two symmetric methods `Hieralign` and `pialign`, we find that `pialign` achieves a higher reduction ratio using model-based phrase extraction approach (MOD) but remember the training time for `pialign` is much more than 4 hours while `Hieralign` is 2 minutes and no statistically significant

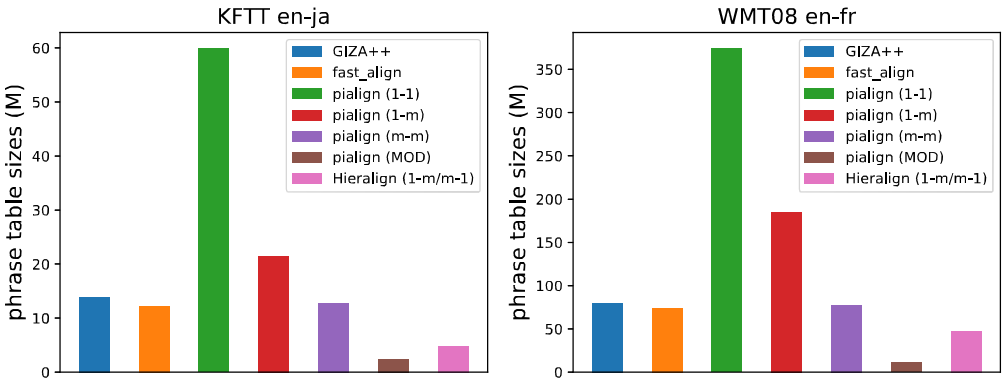


Figure 3.10: Phrase-table sizes (# of entries) using `GIZA++` (+GDFA), `fast_align` (+GDFA), `pialign` and `Hieralign` with heuristic of phrase extraction (Koehn et al. 2003).

difference in the translation quality.

Translation Results

We evaluate the final end-to-end translations produced by our symmetric method to those produced by Viterbi-based aligner `GIZA++` (+GDFA), `fast_align` (+GDFA) and BTG-based aligner `pialign` given the BLEU and RIBES scores.

For the evaluation of machine translation accuracy, some standard automatic evaluation metrics have been used: BLEU [Papineni et al., 2002] and RIBES [Isozaki et al., 2010]. To compare the performance between MT systems, we carry out the statistical significance testing. We apply the bootstrap re-sampling method as described in [Koehn, 2004].

Table 3.6 shows that the final translation scores are approximately the same. There is no significant difference on the final results of machine translation when using the alignments output by the proposed method and those output by `GIZA++` or `fast_align`. It can be seen `GIZA++` and `fast_align` have comparable BLEU score but `pialign` (MOD) and our methods slightly behind the baseline on a very large size corpus.

Table 3.6: BLEU and RIBES scores in translation experiments, † means significantly different with the `fast_align` baseline according to statistical significance tests ($p<0.05$).

	WMT08				**KFTT**				**Europarl v7**			
	en-fr		**en-de**		**en-ja**		**ja-en**		**en-fr**		**es-pt**	
	BLEU	**RIBES**	**BLEU**	**RIBES**	**BLEU**	**RIBES**	**BLEU**	**RIBES**	**BLEU**	**RIBES**	**BLEU**	**RIBES**
asymmetric												
`fast_align`	26.59	76.65	19.61	70.02	21.32	68.10	17.67	65.71	54.10	91.14	49.50	90.79
`GIZA++`	**26.79**	**77.17**	**19.82**	**70.48**	22.57†	**68.79**	18.20	65.25	54.40	91.22	49.34	90.62
`pialign`												
1-to-1	26.39	76.79	19.10†	70.25	21.92	68.61	18.06	66.03	**54.71**	91.29	49.51	89.51
1-to-many	26.57	76.95	19.45	69.98	22.07	68.68	**18.40**	65.55	53.79	91.13	49.48	89.49
symmetric												
`pialign`												
many-to-many	26.67	77.13	19.82	69.99	21.69	69.07	18.19	65.90	54.61	91.28	49.23	90.53
MOD	26.35	77.02	19.31	70.23	21.11	68.17	18.13	65.99	54.35	91.17	49.40	90.57
`Hieralign:` `1-to-many/many-to-1` ($\sigma_\theta = 1$)												
IBM1	26.05†	76.47	19.24	69.68	21.68	67.93	17.43	65.04	54.15	91.24	49.30	90.67
VBIBM1	26.23	76.71	19.58	70.12	22.10	68.36	17.42	64.67	54.24	91.25	49.59	90.79
IBM1+VBH	26.22	76.61	19.39	69.92	22.40†	68.56	17.70	64.93	53.83	91.14	49.15	90.65
VBIBM1+VBH	26.30	76.59	19.33	69.62	**22.69**†	67.94	17.60	66.06	53.86	91.26	49.35	91.21
IBM2+VBH	26.23	76.59	19.30	69.59	22.32†	68.11	17.44	**66.17**	53.52	91.06	49.54	90.76
IBM4+VBH	26.25	76.71	19.30	69.91	21.76	68.09	17.47	65.34	53.72	91.21	49.41	90.71
`Hieralign:` `1-to-many/many-to-1` ($\sigma_\theta = 3, \sigma_\delta = 5$)												
IBM1	26.13	76.70	19.18†	69.62	21.87	67.52	17.88	65.14	53.77	91.04	49.59	90.74
VBIBM1	26.30	76.97	19.43	69.92	21.31	67.35	17.58	64.83	54.32	91.24	49.61	90.75
IBM1+VBH	26.37	76.75	19.32	69.97	22.25	67.79	17.73	65.02	53.91	91.18	49.50	90.65
VBIBM1+VBH	26.55	76.86	19.55	69.90	22.57†	68.35	17.80	65.50	54.35	**91.31**	49.17	90.68
IBM2+VBH	26.25	76.54	19.56	70.00	22.44†	68.62	17.69	65.37	53.40	91.05	**49.80**	**90.80**
IBM4+VBH	26.22	76.58	19.65	70.03	22.34†	68.35	17.40	65.10	53.91	91.28	49.51	90.63

3.5 Summary of the Chapter

In this chapter, we investigate the bidirectional latent variable model of phrasal alignment, i.e., multiple word alignment, a fundamental parameter in SMT models, estimated from word alignments. Previous methods output bidirectional phrasal alignments based on asymmetric word alignments output by GIZA++ [Och and Ney, 2003] which relies on the EM algorithm. The computation of IBM models [Brown et al., 1993] of higher complexity is necessary to obtain accurate alignments. This results in large phrase tables and long training times. [Lardilleux et al., 2012] propose an alternative method, hierarchical sub-sentential alignment (HSSA), which outputs symmetric phrasal alignments in shorter times. However, it is sensitive to initial

word translation probabilities.

The chapter proposed a novel phrasal alignment method based on HSSA. Firstly, Variational Bayes [Riley and Gildea, 2012] is adopted into the EM algorithm for better estimation of initial translation probabilities. Then, it makes approximations of IBM models of lower complexity so that it is faster and does not need the computation of IBM models of higher complexity. Finally, it utilizes a beam search to control the searching choices in the HSSA method so as to increase the quality of the final phrasal alignments.

Our proposal has proven efficient in building phrase-based models for SMT. Those systems built using our methods achieved comparable results in common language pairs, like European languages, and even better results on distant language pairs, like English-Japanese. Compared to other methods, our proposed method is simple, fast and delivers small phrase tables. For instance, compared to state-of-the-art technique (`GIZA++` with IBM models), this novel phrasal alignment method (a) keeps the translation accuracy in various language pairs: English-Japanese, English-German, English-French and Spanish-Portuguese. But it is much more efficient because it outputs (c) much smaller translation tables (50% in average) (b) very fast (only 4% of the training time of the state-of-the-art method). Compared to `fast_align` [Dyer et al., 2013], another fast word aligner, it (a) significantly surpasses it (+1.37 BLEU points, p-value < 0.01) in English-Japanese and (c) delivers smaller translation tables (50% reduction) while (b) requiring the same training time.

Chapter 4

Exploiting Bidirectional Latent Variable Models in Syntax-based SMT: Syntactic Representation

In this chapter, we detail how to impose bilingual syntactic restrictions into statistical machine translation. We concentrate on bidirectional syntax representation for distant language pairs. A novel syntax-based SMT method using bidirectional latent BTG derivations is presented. The main content of this chapter is on the basis of the following papers:

- Wang, H. and Lepage, Y. (2018a). Improved BTG-based preordering for SMT via parallel parameter averaging: An empirical study. 自然言語処理 *(Journal of Natural Language Processing)*, 25(5):487–510
- Wang, H. and Lepage, Y. (2017a). BTG-based machine translation with simple reordering model using structured perceptron. In *Proceedings of the 31th Pacific Asia Conference on Language, Information and Computation (PACLIC 31)*, pages 114–123
- Zhang, Y., Wang, H., and Lepage, Y. (2016b). HSSA tree structures for BTG-based preordering in machine translation. In *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation (PACLIC 30)*, pages 123–132

In this chapter, we present from phrase-based models (the previous chapter) how SMT has been enriched with syntactical theory. Generally, phrase-based SMT suffers from the long-distance reordering problem. Recent research has shown that imposing some grammatical constraints improves the translation quality. Syntax-based machine translation differs from the phrase-based SMT model in that the translation model has access to the syntax of the sentences, i.e., parse trees, which provides an efficient way to handle long-distance reorderings. Therefore, the syntactic parse tree is an important latent variable in syntax-based translation models.

Most of the syntax-based approaches employ indenpendent syntactic parsers to obtain asymmetric syntactic parse trees. However, in some languages, there may not exist any asyntactic parser that is publicly available. Other methods, such as hierarchical phrase-based SMT [Chiang, 2005], rely on more complicated rule tables to reconstruct parse trees, thus dispensing with parsers.

The Bracketing Transduction Grammar (BTG) formalism [Wu, 1997] constitutes a simple bilingual parsing model to answer this problem. It is built upon the minimum case of synchronous context-free grammars. It avoids extracting a large number of rarely used translation rules. BTG parse trees allows that the source and target sentences share the same structure while having different word orders. Hence, it is a bidirectional model. It is insensitive to long-distance reordering. BTG parse trees can be used to reorder the words in sentences, either before the standard phrase-based SMT pipeline (preordering), or during translation (decoding). Consequently, we utilize the BTGs to develop our syntax-based SMT systems.

At first, we improve the top-down BTG-based preordering method [Nakagawa, 2015] with parallelization, then apply it to BTG-based decoding. When training on automatically aligned data sets, the top-down BTG-based preordering method is very sensitive to alignment errors in the training examples. For this reason, we modify the training algorithm using bootstrap aggregating with parameter mixing techniques. This allows training the reordering model via parallelization, so that the time of training is dramatically reduced. Then, the reordering models learned from the previous step are incorporated with other models, e.g., language models, phrasal translation models, to build the syntax-based SMT. In decoding, translation

hypotheses are extended under the constraint of BTGs, i.e., latent BTG parse trees.

1. **Reordering Model Training**

   Instead of the online passive-aggressive algorithm [Nakagawa, 2015], we adopt several techniques for parallel training (mini-batch, distributed, iterative distributed and k-best list), so that it is more insensitive to alignment errors in the training examples, i.e., the learning of reordering models becomes more efficient. As a result, (a) the latent BTG parse trees are more accurate according to the evaluations on reordering metrics, and more practical, (b) it is faster in training.

2. **Latent BTG-based Decoding**

   The traditional template-based approaches store reordering models as in-compressible rule tables. In our proposal, the reordering model is stored in a more efficient way, in which features are generated using a special hashing trick and the reordering scores are computed on-the-fly. This makes the reality of (c) lesser memory cost for the decoder initialization.

The structure of this chapter is as follows.

- Section 4.1 introduces the background of BTG-based preordering for phrase-based SMT and BTG-based decoding for syntax-based SMT.
- Section 4.2 reviews related works on syntax-based preordering in phrase-based SMT and syntax-based decoding in syntax-based SMT.
- Section 4.3 details our work on latent BTG-based reordering, including latent BTG-based preordering and latent BTG-based decoding. We first presents our solution to the problem of training data noise existing in the top-down BTG-based preordering method. Then, we show our work on BTG-based decoding using such a latent BTG-based reordering model.
- Section 4.4 deals with the experiments. We conduct two experiments: phrase-based SMT with preordering and BTG-based SMT using the improved BTG-based reordering method presented in this chapter.
- Finally, we summarize this chapter in Section 4.5.

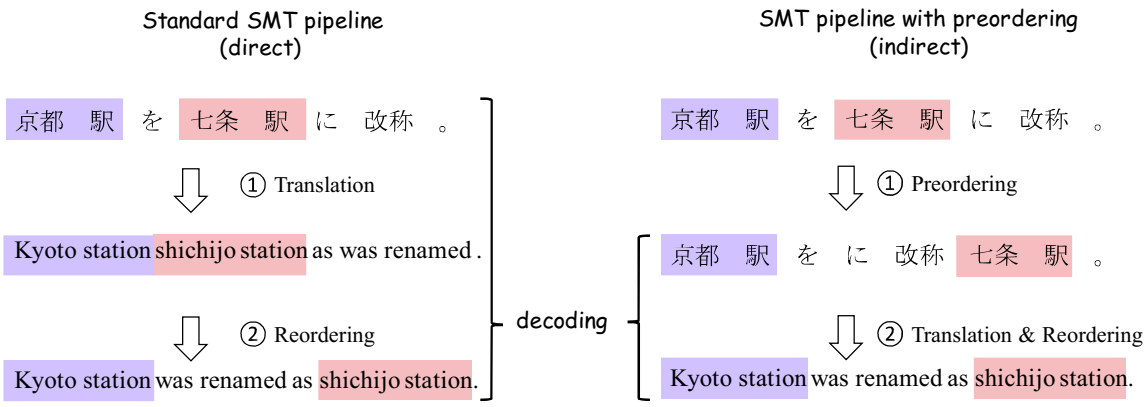


Figure 4.1: An example of the standard SMT pipeline w/o preordering.

4.1 Background

Structural divergence arises from machine translation for distant language pairs. For example, when translating from an SVO (Subject-Verb-Object) language, e.g. English, to an SOV language, e.g. Japanese, a verb often moves to the end of the target sentence/clause [Sudoh et al., 2010].

Although it has a long history since SMT researchers [Tillmann, 2004, Koehn et al., 2005, Galley and Manning, 2008] addressed the reordering problem, the reordering models used in phrase-based SMT systems are relatively weak. Phrase-based SMT systems usually exhibit lower BLEU scores on such a language pair, because the default lexical reordering model [Koehn et al., 2003] used is incapable of performing accurate long-distance reorderings.

To cope with the long-distance reordering problem, researchers have investigated syntax-based approaches to machine translation from two perspectives: *indirect* and *direct*. Figure 4.1 illustrates these two ways to syntax-based SMT.

4.1.1 Preordering in Phrase-based SMT (Indirect)

The indirect methods try to solve the reordering problem in the pre-processing phase of SMT, called *preordering*. The main method of preordering [Xia and McCord, 2004, Collins et al., 2005, Neubig et al., 2012a, Lerner and Petrov, 2013, Goto et al., 2015, Nakagawa, 2015] consists in introducing a reordering step on source sentences before

the standard PB-SMT pipeline [Koehn et al., 2003]. Source words are permuted according to word order of the target languages during preordering. By opposition to the standard view of SMT where a sentence $\mathbf{F}$ is translated into a target sentence $\mathbf{E}$ in one step, noted $\mathbf{F} \rightarrow \mathbf{E}$, preordering involves a two-phase translation:

$$\mathbf{F} \rightarrow \mathbf{F}' \rightarrow \mathbf{E} \tag{4.1.1}$$

1. the source sentence $\mathbf{F}$ is converted into $\mathbf{F}'$ by arranging the source words in the order of the target language, i.e., $\mathbf{F} \rightarrow \mathbf{F}'$, where $\mathbf{F}'$ is a latent variable.;

2. the reordered sentence $\mathbf{F}'$ is then translated into the target sentence $\mathbf{E}$ through the use of machine translation systems, i.e., $\mathbf{F}' \rightarrow \mathbf{E}$.

Directly mapping from $\mathbf{F} \rightarrow \mathbf{F}'$ is impractical. Recently, preordering has been proven effective to improve the translation for language pairs with different word orders. The majority of those works tend to employ a source language syntactic parser directly, while other methods learn a syntactic parser with the extension of reordering rules from the word-aligned parallel text.

There exists a considerable body of literature on making use of the underlying syntactic structures to improve phrase-based SMT. [DeNero and Uszkoreit, 2011] firstly show that it is possible to learn some reordering rules only based on source sentences, afterward [Neubig et al., 2012a] propose to learn a discriminative parser for preordering (also called *preorderer*) directly by treating the hidden synchronous parse trees as latent variables.

4.1.2 Decoding in Syntax-based SMT (Direct)

Phrase-based models can robustly perform translations because they are good for learning local word reorderings. The widely-used lexical reordering model [Koehn et al., 2003] can somehow handle swaps between adjacent phrases. However, it remains weak in capturing word order changes.

Previous works on syntax-based SMT mainly have involved the use of syntactic structures to handle long-distance reorderings during decoding, like syntax-based model [Yamada and Knight, 2001], syntax-augmented model [Galley et al., 2004]

and hierarchical phrase-based model [Chiang, 2007]. In these works [Yamada and Knight, 2001, Galley et al., 2004, Genzel, 2010], word reordering is implicitly addressed by translation rules extracted using either syntactic parsers or annotated treebanks. The performance of syntax-based SMT systems is subject to the errors in translation rules. Thus, training of a syntax-based reordering model is more difficult compared to the phrase-based model [Koehn et al., 2003].

Chiang's hierarchical phrase-based model [Chiang, 2007] arises from the phrase-based approach, called Hiero grammars. It is a non-labelled syntax-based model, which is much simpler than the previous syntax-based models. During decoding, it employs a rule table to score the probability of translation rules. Apparently, a definite weakness of Hiero grammars is that it may extract a large number of rarely used translation rules, resulting in more search errors in decoding. To reduce the mistakes, [Wang et al., 2007] apply the synchronous binarization technique to compress the rules extracted. This shows improvements over the baseline (Hiero grammars) systems. Since the binarized Hiero grammar indeed resembles the BTG framework, it is natural to exploit BTG-based translation model for syntax-based SMT.

BTG-based SMT [Xiong et al., 2008] has many advantages:

- It is capable of long-distance and hierarchical reordering.
- It relies on the minimum case of synchronous context-free grammars.
- It provides a simple binary way to construct hierarchical bidirectional structures during translation.

4.2 Related work

4.2.1 Top-down BTG-based Preordering

BTG is a special case of synchronous context-free grammars. According to the definition of BTG, both the source and target sentences should share same underlying structures (i.e. BTG parse trees), while the order of constituents may be different (bidirecitioanl, but not symmetric). BTG provides a simple and effective way to

represent permutations and perform word reordering[1]. For example, the source and target sentences in Figure 4.1 can be represented as a BTG parse tree in Figure 4.2[2].

Following [DeNero and Uszkoreit, 2011], [Neubig et al., 2012a] propose to learn a discriminative parser for preordering based on the framework of Bracketing Transduction Grammar (BTG, [Wu, 1997]). However, if the training data contains too many long sentences, the training time of the parser substantially increases due to the high complexity of the bottom-up CYK algorithm ($O(n^5)$) used in parsing. For efficiency, [Nakagawa, 2015] adopts a top-down parsing algorithm. The time complexity for parsing a sentence is significantly reduced to $O(kn^2)$ (for a beam width of k and sentence length n), but training such a preorderer on large data sets is still time-consuming.

Consider the case of preordering, whereas the BTG parse tree becomes incomplete, lacking in target terminals. However, such a partial BTG parse tree is sufficient to accurate reorderings, which is interchangeable with the corresponding parser derivation in left-to-right top-down parsing. Let D denotes the parser derivation. In BTG-based preordering framework [Neubig et al., 2012a, Nakagawa, 2015], the inference procedure is defined as:

$$\tilde{\mathbf{F}}' = \operatorname*{argmax}_{\mathbf{F}' \in \hat{\mathbf{F}}} P(\mathbf{F}'|\mathbf{F}) \tag{4.2.2}$$

$$= \operatorname*{argmax}_{\mathbf{D} \in \hat{\mathbf{D}}} P(\mathbf{F}'|\mathbf{D}, \mathbf{F}) P(\mathbf{D}|\mathbf{F}) \tag{4.2.3}$$

where $\hat{\mathbf{F}}$ stand for all possible permutations and $\tilde{\mathbf{F}}'$ is the best one given the target word order. For a unique BTG tree, there exists only one deterministic derivation. Thus we can eliminate the first part in Formula 4.2.3. Therefore, the task of preordering can be solved as monolingual parsing with the extension of reordering operations. For a source sentence $\mathbf{F}$, finding the best reordering is equivalent to finding an underlying parser derivation $\mathbf{D}$ licensed by BTG. To assign a score for a

[1]There exist some permutations that BTG cannot represent, e.g. $X_2X_4X_1X_3$ to $X_1X_2X_3X_4$, see [Wu, 1997]. Multiple BTG parse trees may generate same word orders.

[2]In Figure 4.2, we omitted the non-terminal before each terminal for simplicity.

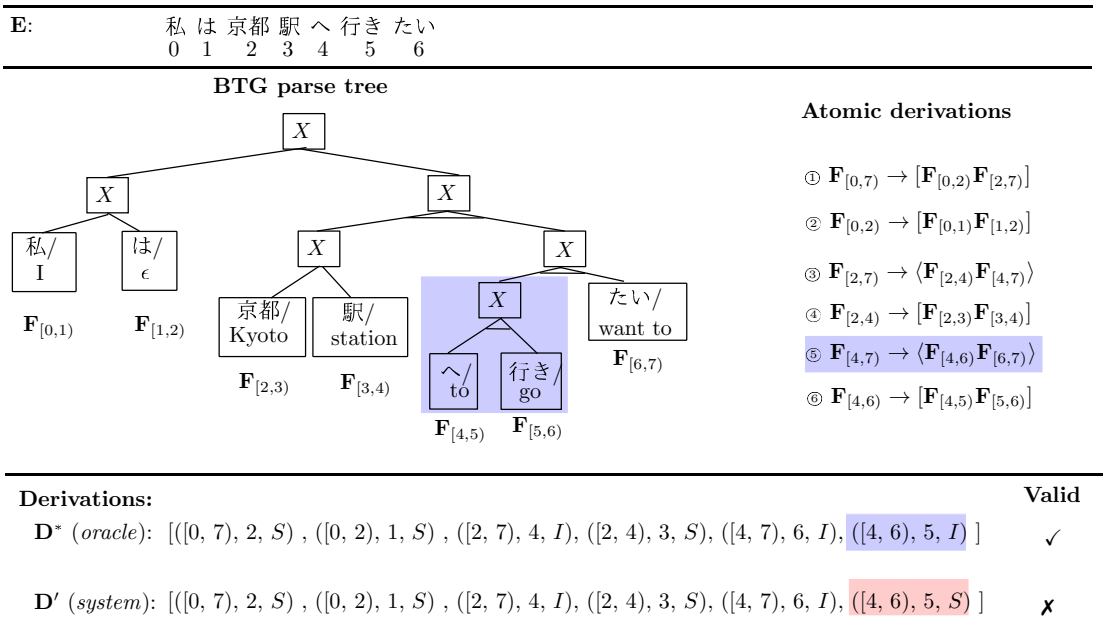


Figure 4.2: Example of BTG parsing and two parser derivations: $\mathbf{D}'$: system derivation with the highest model score; $\mathbf{D}^*$: oracle derivation with the highest model score from the valid subset.

particular derivation, [Nakagawa, 2015] defines the score function in a linear form:

$$P(\mathbf{D}|\mathbf{F},\mathbf{w}) = \text{Score}(\mathbf{D}|\mathbf{F}) \tag{4.2.4}$$

$$= \mathbf{w}^\top \cdot \Phi(\mathbf{D},\mathbf{F}) \tag{4.2.5}$$

$$= \sum_{d\in\mathbf{D}} \mathbf{w}^\top \cdot \Phi(d,\mathbf{F}) \tag{4.2.6}$$

where Φ is the feature function, and $\mathbf{w}$ is the weight vector. The parser (i.e., preorderer) makes greedy selections at each parsing step, so that $\mathbf{D}$ is a sequence of the atomic derivations. Each atomic derivation d is a triple $\langle \mathbf{F}_{[p,q)}, r, o\rangle$, where $\mathbf{F}_{[p,q)}$ is a parse span, e.g. $\mathbf{F}_{[0,7)}$, denoted by $[p,q)$ covers the source words $\mathbf{F}_p,\ldots,\mathbf{F}_{q-1}$; r is the splitting point; o is the type of non-terminal nodes (*straight* or *inverted*). To represent the atomic derivation, [Neubig et al., 2012a] factors d as a set of local features (e.g. POS tags/word classes of the first/last words in the former/latter parse spans) intersected with the node type label S or I. The score for d is the sum of the feature weights. The top-down parser starts with the whole source sentence ($\mathbf{F}_{[0,7)}$, i.e., the initial span), then splits the span dichotomously and recursively until it terminates. Finally, the derivation with the highest score is picked up to perform

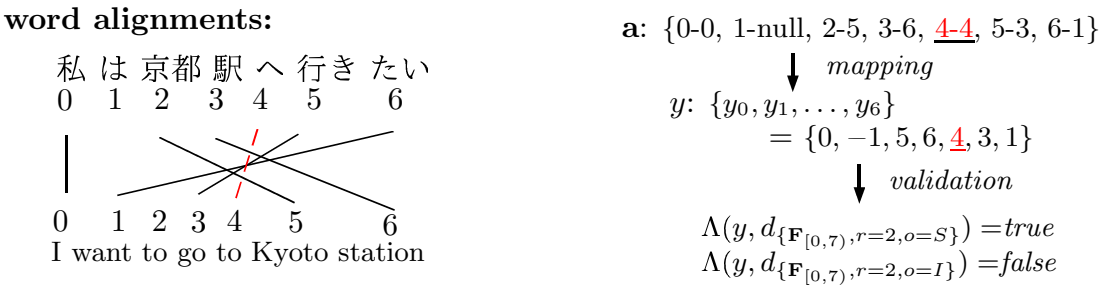


Figure 4.3: An example of derivation validation.

reordering.

4.2.2 Online Passive-Aggressive Algorithm

We briefly introduce the online learning algorithm (passive-aggressive-I algorithm, called PA-I [Crammer et al., 2006]) used in [Nakagawa, 2015] as shown in Algorithm 3 (see Appendix B). Given an example $(\mathbf{E}, \mathbf{a})$, the trainer runs as follows:

1. produce k-best parser derivations using beam search;

2. select a pair of derivations $\langle \mathbf{D}', \mathbf{D}^* \rangle$: one derivation with the highest model score from all candidates (called *system* derivation, noted $\mathbf{D}'$) and another with the highest model score from the valid subset; (called *oracle* derivation, noted $\mathbf{D}^*$)

3. if these two derivations diverge or the valid derivation falls off the beam, update weights.

The above procedure allows the oracle derivation to vary from one example to another during training. Syntactic parsers are usually trained on tree-annotated datasets, while there are no BTG-tree-annotated datasets available. Fortunately, the BTG-based preordering method offers a substantial advantage over the conventional methods is that it only needs word alignments to train the model. [Nakagawa, 2015] extend the atomic derivation d to the four tuple $\langle E_{[p,q)}, r, o, v \rangle$, in which the extra $v \in \{true, false\}$, records the validity of the current atomic derivation. $\Lambda(\cdot)$

(see Figure 4.3) is the validation function, which returns *true* if and only if the following property holds:

$$\max(\{y_i|i \in [p,r), y_i \neq -1\}) \leq \min(\{y_j|j \in [r,q), y_j \neq -1\}) \quad \text{if} \quad o = \textit{straight} \tag{4.2.7}$$

$$\max(\{y_j|j \in [r,q), y_j \neq -1\}) \leq \min(\{y_i|i \in [p,r), y_i \neq -1\}) \quad \text{if} \quad o = \textit{inverted} \tag{4.2.8}$$

where $\mathbf{y} = \{y_0, \ldots, y_{|x-1|}\}$ is the reordered ranks converted from $\mathbf{a}$. The above formula checks the crossings of the word alignment. In the beginning, the parser starts with an empty derivation D (valid), and during parsing, the following condition is checked for validation:

$$\forall d \in \mathbf{D},\ \Lambda(\mathbf{y}, d) = \textit{true} \tag{4.2.9}$$

Let $\mathbf{z}_0^n = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\} = \langle \mathbf{D}', \mathbf{D}^* \rangle_0^n$ denote the training examples. The trainer updates the feature weights $\mathbf{w}$ towards the solution to the following projection problem:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2}||\mathbf{w} - \mathbf{w}_t||^2 \quad \textbf{s.t.} \quad \ell(\mathbf{w}; \mathbf{z}_t) = 0 \tag{4.2.10}$$

where $\ell(\cdot)$ is a structured hinge-loss function, which separates $\mathbf{D}'$ from $\mathbf{D}^*$ by a margin being proportional to the square root of the errors. $\mathbf{w}_t/\mathbf{w}_{t+1}$ are the previous and new weight vectors respectively. According to the different case of *passive* or *aggressive*, the following update rules are derived for the optimization problem[3]:

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t & \text{if} \quad \ell(\mathbf{w}; \mathbf{z}_t) = 0, \quad \textit{passive} \\ \mathbf{w}_t + \eta_t \cdot \Phi(\mathbf{z}_t) & \text{if} \quad \ell(\mathbf{w}; \mathbf{z}_t) \neq 0, \quad \textit{aggressive} \end{cases} \tag{4.2.11}$$

Namely, $\ell(\cdot)$ is the errors, $\mathcal{E}$ is the error cost and γ is the margin:

$$\ell(\mathbf{w}; \mathbf{z}) = \begin{cases} 0 & \text{if} \quad \mathcal{E}(\mathbf{z}) = 0 \\ \mathcal{E}(\mathbf{z}) - \gamma(\mathbf{w}; \mathbf{z}) & \text{otherwise} \end{cases} \tag{4.2.12}$$

$$\mathcal{E}(\mathbf{z}) = ||\mathbf{D}^* - \mathbf{D}'||^{\frac{1}{2}} \tag{4.2.13}$$

[3]For details, see the mathematical proof with the Lagrange multipliers in [Crammer et al., 2006]

$$\gamma(\mathbf{w}; \mathbf{z}) = \mathbf{w}^\top \cdot \Phi(D^*) - \mathbf{w}^\top \cdot \Phi(D') \tag{4.2.14}$$

During training, η_t is the aggressiveness parameter controlled with an upper bound C, which changes with epochs and examples[4].

$$\eta_t = \min \left\{ C, \frac{\ell(\mathbf{w}_t; \mathbf{z}_t)}{||\Phi(\mathbf{z}_t)||^2} \right\} \tag{4.2.15}$$

4.2.3 Top-Down BTG Parser with Online Learning

Usually, we train syntactic parsers with tree-annotated datasets, while there do not exist annotated BTG trees for training the BTG parsers. Inspired by latent perceptron [Sun et al., 2009], [Nakagawa, 2015] trained the top-down BTG-based preorderer using the online passive-aggressive-I (PA-I) algorithm [Crammer et al., 2006] by regarding the parser derivations that license word reorderings as latent variables.

In [Nakagawa, 2015], a training example is a tuple $(\mathbf{F}, \mathbf{a})$ which contains a source sentence $\mathbf{F}$ and the corresponding word alignments $\mathbf{a}$. Given an example $(\mathbf{F}, \mathbf{a})$, the learning process runs as follows:

1. produces k-best parser derivations using beam search;

2. selects a single latent-BTG-tree ($\mathbf{D}'$, *system*) with the highest model score and another ($\mathbf{D}^*$, *oracle*) with the highest model score from the valid subset;

3. if these two derivations diverge or the valid derivation falls off the beam, updates different features.

This strategy allows the oracle derivation for each sentence to vary during training. F lidation, [Nakagawa, 2015] extended the atomic derivation d to the four tuple $\langle \mathbf{F}_{[p,q)}, r, o, v \rangle$, in which the extra $v \in \{true, false\}$ records the validity of the atomic derivations. The initial derivations are set *true*. $\Lambda(\cdot)$ (see Figure 4.4) is the function which returns the validity of atomic derivation. The return value is

[4] C equals 1.

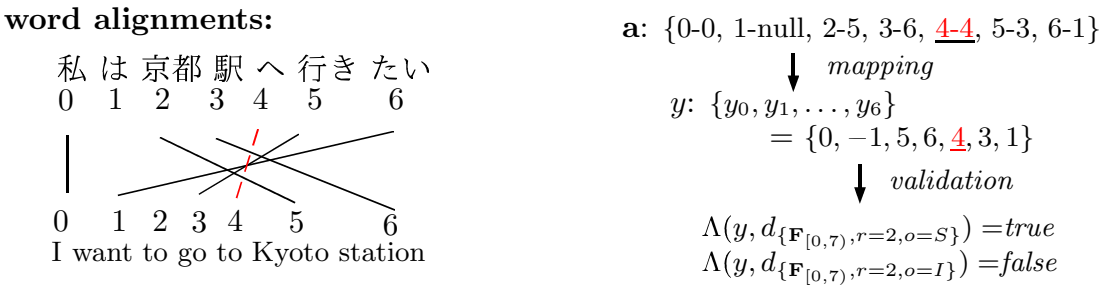


Figure 4.4: Validation of parser derivations.

true if and only if the following condition holds:

$$\max(\{y_i|i \in [p,r), y_i \neq -1\}) \leq \min(\{y_j|j \in [r,q), y_j \neq -1\}) \quad \text{if} \quad o = S \text{(4.2.16)}$$

$$\max(\{y_j|j \in [r,q), y_j \neq -1\}) \leq \min(\{y_i|i \in [p,r), y_i \neq -1\}) \quad \text{if} \quad o = I \text{(4.2.17)}$$

where $\mathbf{y} = \{y_0, \ldots, y_{|x-1|}\}$ is the reordered ranks converted from $\mathbf{a}$. **Figure 4.4** shows the examples of validation of atomic derivations. For a global parser derivation D, its validity is computed using the following property:

$$\forall d : \{d \in \mathbf{D}\}, \Lambda(\mathbf{y}, d) = true. \tag{4.2.18}$$

$\mathbf{D}$ is always *true* at the beginning of the parsing. During parsing, only the parser derivations licensed by $\mathbf{y}$ are labeled as *true*.

4.2.4 BTG-Based Decoding

In general, there are two kinds of reordering models in the Phrase-based SMT model: distortion model and lexical reordering model. The distortion model [Koehn et al., 2003, Zens et al., 2004] penalizes translations with long-distance jumps. It does not consider the context and even prevents long distance or clause-level reorderings. The lexical reordering model is simply learned from training data with maximum likelihood estimation (MLE), which is incapable of long-distance reordering.

Previous works focus on imposing BTG formalism into the decoding procedure, ranging from simple flat reordering model [Wu, 1997], maximum-entropy (MaxEnt) based model [Xiong et al., 2008] to tree kernel-based SVM model [Zhang and Li, 2009]. Those models commonly suffer from high computational complexity and

use bilingual features. Other approaches [Zens and Ney, 2006, Genzel, 2010, Lerner and Petrov, 2013] learn reordering rules from the source side but require annotated treebanks or syntactic parsers. Since the top-down BTG-based preordering method relies only on the features extracted from source sentences, a reordering model built using such a technique removes the restriction of target sentences. Given the truth that only source sentences are available, such a model is more suitable for decoding.

In the BTG formalism [Wu, 1995, Wu, 1997], a derivation $\mathbf{D}$ is defined as a sequence of independent operations $d_1, \ldots, d_K$, which apply a bracketing rules $X \rightarrow \gamma$ at each stage. The parser splits a source-target sentence pair $< \mathbf{F}, \mathbf{E} >$ on both sides dichotomously. In top-down left-right parsing, BTG trees are unique to each derivation. [Wu, 1997] treats BTG-based translation as two dependent monolingual parsing tasks. Similar to PSCFG, the probability of a BTG derivation (parse tree) is computed as:

$$P(\mathbf{D}) = \prod_{d \in \mathbf{D}} P(d : X \rightarrow \gamma) \tag{4.2.19}$$

where $d : X \rightarrow \gamma$ stands for the derivation that applys a billingual BTG rule. [Wu, 1997] also present a CYK-like parsing algorithm using a dynamic programming scheme with a time complexity $O(n^3 \times m^3 \times |R|)$.

4.3 Proposal: Latent BTG-based Reordering Model

Most of the previous research on syntax-based reordering has concentrated on preordering, but less attention has been paid to syntax-based decoding. In this chapter, we improve SMT from both the preordering (indirect, for phrase-based SMT) and decoding (direct, for syntax-based SMT) perspectives based on the top-down BTG-based preordering method.

4.3.1 Latent BTG-based Preordering

In top-down BTG-based preordering, the accuracy of parsers heavily relies on the accuracy of word aligners. Hence, it is necessary to find a proper method that can deal with alignment errors. Previous works [Breiman, 1996, Freund and Schapire, 1999, Džeroski and Ženko, 2004, Rokach, 2010, Li et al., 2014] on batch learning and

ensemble learning have proven helpful for many NLP tasks. These works mainly fall into two categories: inner methods and outer methods. The inner methods perform parameter mixing in small batches [Freund and Schapire, 1999, Li et al., 2014]. This is called *mini-batch* in machine learning. The outer methods often appear as the framework of *distributed training* [McDonald et al., 2010] in machine learning, or *ensemble learning* [Naftaly et al., 1997, Dietterich, 2000] in neural network research. Those methods have shown their capability of dealing with the noise in the training examples, which may benefit the top-down BTG-based preordering.

We propose to improve the top-down BTG-based preordering method via parallelization using the methods of either inner averaging or outer averaging. For the inner methods, we apply parameter averaging in each *mini-batch*. For the outer methods, we investigate two distributed averaging techniques, which do not need to change or modify the internal structure of the top-down BTG-based method: *distributed averaging* and *iterative distributed averaging*. We empirically compare these methods with the original online method of [Nakagawa, 2015].

Existing Problem: Gradient Noise

In [Nakagawa, 2015], pairwise parameter updating is performed if two parses (one: *system* derivation; another: *oracle* derivation) diverge. The performance of preordering models substantially relies on the quality of word alignments. [Nakagawa, 2015] suggest to use the symmetrized bidirectional word alignments with the INTERSECTION[5] heuristic using `GIZA++`. Another problem which deserves attention is the quality of the training data. The top-down preordering method [Neubig et al., 2012a] learns reordering models using the general framework of large-margin online structured prediction [Crammer et al., 2006, Watanabe et al., 2007]. To obtain accurate reordering models, the high quality of word alignments is essential. In a production environment, since the expensive cost for manual annotation, we often use some unsupervised word aligners to produce word alignments. Preorderers trained on those noisy alignments suffer from small mistakes in training examples.

[5]Other heuristics: *union*, *grow-diag-final*, and *grow-diag-final-and*.

Thus, preorders trained on large automatically aligned data sets usually exhibit lower accuracy than the one trained on small manual data sets.

The online pairwise updating strategy used in the top-down BTG-based preordering method is not suitable when training on automatically aligned datasets, given its inability to deal with the noise (i.e., a large number of alignment errors) into consideration. Therefore, it is necessary to find a robust method that can deal with the noise. A large body of research on batch learning and ensemble learning [Breiman, 1996, Freund and Schapire, 1999, Džeroski and Ženko, 2004, Rokach, 2010, Li et al., 2014] has been empirically observed to be helpful for in many NLP tasks, which is robust and efficiency against noise. These methods mainly fall into two categories: inner averaging and outer averaging. The inner averaging methods perform parameter mixing in small batches [Freund and Schapire, 1999, Li et al., 2014], called *mini-batch* in machine learning. The outer averaging methods appear in the framework of *distributed training* [McDonald et al., 2010] in machine learning, or *ensemble learning* [Naftaly et al., 1997, Dietterich, 2000] in neural network research. Such methods may improve the accuracy of the top-down BTG-based preorderers. Though these methods are capable of dealing with the noise in the training examples, there are few research studies online to parallel conversions.

However, using the intersected word alignment has two problems:

1. Alignment errors exist in word alignments obtained from `GIZA++`, which will heavily affect the performance of MT systems.

2. The intersected word alignment contains less alignment points, resulting in less constraint on determining the oracle derivation.

Both of the above problems lead to gradient noise.

- **Alignment Errors**

  The first one is easy to understand. For example, Figure 4.3 shows a mistake in the training example (the alignment point with underline). When using the online algorithm, frequent incorrect updates will cause a higher variance [Klasner and Simon, 1995], making the preorderer less accurate. A noisy

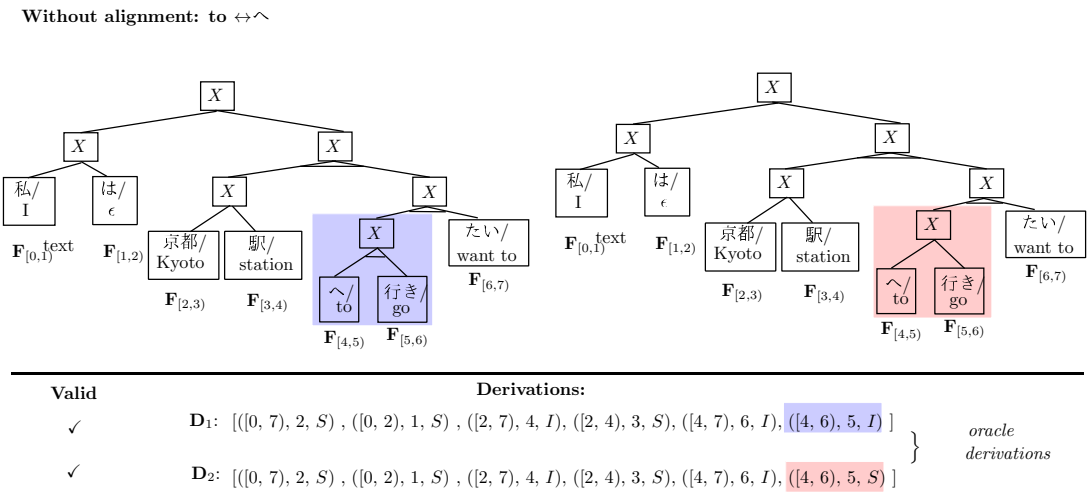


Figure 4.5: A particular case that two derivations tend to be valid, resulting from the weak constraint with fewer alignments. However, pairwise updating strategy considers only one derivation as the oracle.

gradient signal will be easily observed[6]. For this reason, we propose to adopt the parallel training approach to reduce gradient noise.

- **Missing Alignment**

  Figure 4.5 illustrates the second case. On the one hand, if we remove the underlined alignment point from the example, both derivations have a chance to be the oracle derivation. Thus an oracle derivation is probably not the best one for reordering. On the other hand, micro-reordering (colored in Figure 5.3) is meaningless for phrase-based SMT (see Figure 5.1). As a consequence, in the experiments of [Nakagawa, 2015], preorderers trained on the automatically aligned dataset underperform the one trained on the small manual datasets. To reduce such noise, we propose to use the derivations in the parser k-best list instead of training towards a single derivation.

[6]It does not mean that online PA-I algorithm is not good, on the contrary, it works very well on manual data sets.

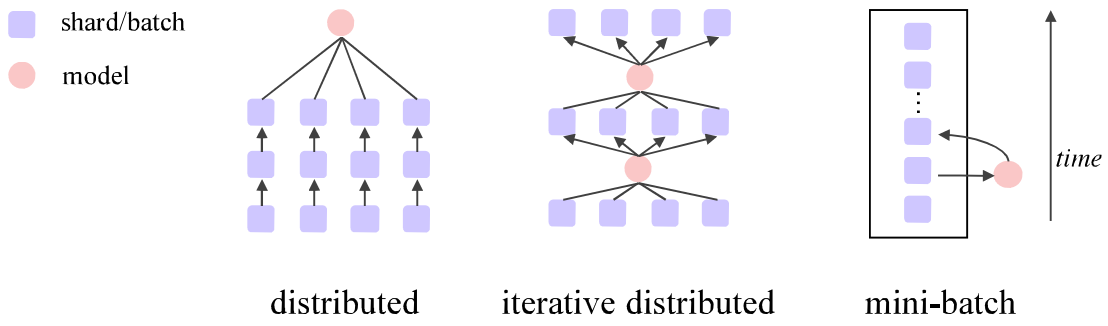


Figure 4.6: Characterization of three parallel averaging methods.

Parallel Averaging Strategies

The main focus of this section is parallel training algorithms for top-down BTG-based preordering. Previous works [Breiman, 1996, Freund and Schapire, 1999, Džeroski and Ženko, 2004, Rokach, 2010, McDonald et al., 2010, Broderick et al., 2013, Li et al., 2014] have shown that parameter averaging in model learning can dramatically reduce errors. The first idea is using the distributed training methods [McDonald et al., 2010, Broderick et al., 2013], in which parameter mixing is performed outside the course of data processing. These methods do not change the inside learning algorithm. We adopted two distributed parameter mixing strategies: *distributed averaging* and *iterative distributed averaging*. The second idea is to modify the inside algorithm to apply parameter averaging on mini-batches, called *mini-batch averaging*. Though these methods fall into different categories of the learning frameworks, the idea behind is same that computing the margins from the different levels or using the different sizes of the training examples to reduce the variance of the learned models. We use Figure 4.7 to give a visual comparison. We expect that the preorderer using parallel training algorithms to be more stable and robust against errors when trained on automatically-aligned data sets.

Distributed Averaging Distributed averaging may be the most straightforward one to apply, which performs a parameter mixing after training finished. It divides the training examples $\mathcal{T}$ into several disjoint shards as $\{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$ then trains

on each shard $\mathcal{T}_i$ in parallel as distributed systems[7]. [McDonald et al., 2010] show that the distributed perceptron algorithms with averaging parameter mixing return comparable or even better performance in the tasks of named entity recognization (NER) and dependency parsing compared to standard online perceptron[8]. Since weights are updated without commutations among the child processes during the whole period of training, [McDonald et al., 2010] points out that such a distributed strategy is also capable of other online algorithms. The final feature weights are calculated by taking the average of all sub-models. The distributed training algorithm used is presented in Algorithm 4 (see Appendix B).

Iterative Distributed Averaging Iterative distributed averaging also divides the data into several shards as distributed averaging. Differing from distributed averaging trains the model to convergence separately, iterative distributed averaging combines the weights of submodels after each epoch, i.e. 'iterative', then, resent the averaged weights to child processes and update the weights of sub-models. An obvious shortcoming of iterative distributed averaging is the extra memory cost to cached the sub-model weights and the extra computational cost for parameter mixing. [McDonald et al., 2010] showed that iterative distributed averaging outperforms distributed averaging and standard online method in the task of dependency parsing. Algorithm 5 shows the iterative distributed training algorithm used (see Appendix B).

Mini-batch Averaging Consider the online PA-I algorithm, in which some parameters are not linearly separable. For example, η is a parameter that requires to be computed dynamically during training. Hence, simple batch processing becomes impractical. Fortunately, [Crammer et al., 2006] points out that the PA online algorithm can also adapt batch learning. Theoretically, it is possible to estimate η approximately using a mini-batch instead of computing it example by example.

[7]Here, bootstrap aggregating or boosting may also be suitable, we only test the most straightforward way.

[8]The proof of the convergence can be found in [McDonald et al., 2010].

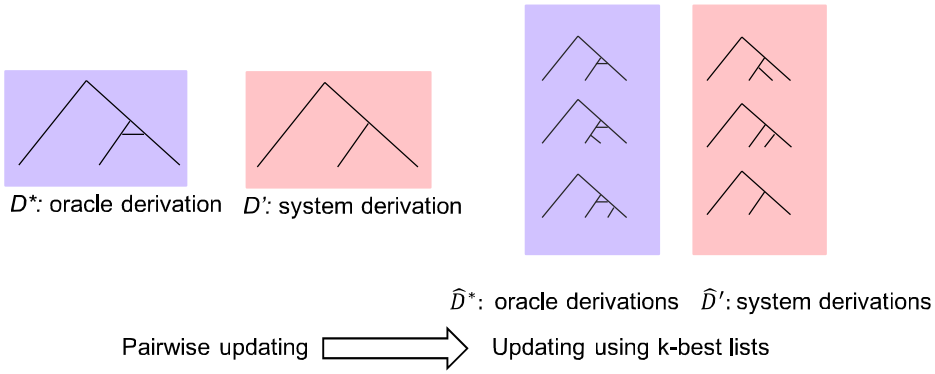


Figure 4.7: Characterization of the proposed method using k-best list.

Concretely, let $\beta \in \mathbb{R}$ be a small positive number; $\hat{\mathbf{z}} = \{\mathbf{z_1}, ..., \mathbf{z_m}\}$ be a fixed training set (i.e. one mini-batch); $\mathbf{w}_{t-1}$ denotes the initial weights before the current mini-batch; $\mathbf{w}_t$ denotes the weights after training. If we iterate over all examples in $\hat{\mathbf{z}}$ and updates weights using the one which $\ell(\mathbf{w}_1; \mathbf{z}) > \beta$, as a result, the loss for any $\mathbf{z} \in \hat{\mathbf{z}}$ at time t will be less than β, i.e., $\ell(\mathbf{w}_t; \mathbf{z}) \leq \beta$. Hence, the decrease in the loss function guarantees its convergence. PA-I algorithm is compatible with batch learning[9]. In our implementation, we collect all errors $\Delta\mathcal{E}(\hat{\mathbf{z}})$ in a mini-batch, then compute the aggressiveness parameter $\Delta\eta_t$ using the following formula:

$$\Delta\eta_t = \min\left\{C, \frac{\sum_{i=1}^{m}\ell(\mathbf{w}_t; \mathbf{z_i})}{||\Phi(\hat{\mathbf{z}})||^2}\right\} \tag{4.3.20}$$

The variance of gradient for the mini-batch is reduced by a factor of m (i.e. batch size) compared to the original online PA-I algorithm. Algorithm 6 presents the batch training algorithm used (see Appendix B).

Using *k*-best List Weight updating can also be carried out on multiple derivations (so-called *k-best-based*). For example, in SMT, batch MIRA tuning [Cherry and Foster, 2012] generates multiply translation hypotheses at a time. The decoder is tuned using two lists of k-best hypotheses (one: *hope*, another: *fear*) in parallel. Inspired by this idea, in this section, we propose to minimize the sum of hinge-losses for all derivations in the k-best list. Rather than pairwise updating used in

[9]The proof of the convergence can be found in [Crammer et al., 2006], p.6.

online-PA-I, we build two list using the top-k derivations.

$$\hat{\mathbf{D}}_{\text{hope}} = \{\mathbf{D}|\mathbf{D} \in \hat{\mathbf{D}}_{\text{top-}k} \ \wedge \ \forall d \in \mathbf{D}, \Lambda(d) = true\} \tag{4.3.21}$$

$$\hat{\mathbf{D}}_{\text{fear}} = \{\mathbf{D}|\mathbf{D} \in \hat{\mathbf{D}}_{\text{top-}k} \ \wedge \ \mathbf{D} \notin \hat{\mathbf{D}}_{\text{hope}}\} \tag{4.3.22}$$

The difference between pairwise updating and k-best–based updating is that the pairwise updating is replaced with the updating using two lists, where:

$$\mathcal{E}(\mathbf{z}) = ||\hat{\mathbf{D}}_{\text{hope}} - \hat{\mathbf{D}}_{\text{fear}}||^{\frac{1}{2}} \tag{4.3.23}$$

$$\gamma(\mathbf{w}; \mathbf{z}) = \mathbf{w}^{\top} \cdot \Phi(\hat{\mathbf{D}}_{\text{hope}}) - \mathbf{w}^{\top} \cdot \Phi(\hat{\mathbf{D}}_{\text{fear}}) \tag{4.3.24}$$

The advantage of using k-best is more effective and stable when training on large datasets. We observed that it is more fast to convergence compared to pairwise updating (also called greedy updating). The k-best-based training algorithm used is shown in Algorithm 7 (see Appendix B).

4.3.2 Latent BTG-Based Decoding

Furthermore, recent progress on BTG-based preordering [Nakagawa, 2015] also shows that obtaining the structure across language is not so difficult as before. An important by-production of BTG-based preordering is the BTG tree. In top-down BTG-based preordering, the parsing model is trained to maximize the conditional likelihood of BTG trees that license the reorderings implied by observed word alignments in a parallel corpus.

Since decoding in the opposite direction as parsing, we propose bottom-up BTG-based decoding, which incorporates the BTG-based reordering model into the decoding step of SMT directly. We apply the parser to obtain the bilingual hierarchical syntactic structures, which reduce the search space during decoding. We present experiments on two tasks, i.e., preordering and SMT translation, to highlight the efficiency of our proposed method.

Assuming $\mathbf{D}$ is a latent variable, to find the best translation $\tilde{\mathbf{E}}$, we have:

$$\tilde{\mathbf{E}} = \underset{\mathbf{E} \in \hat{\mathbf{E}}}{\operatorname{argmax}} P(\mathbf{E}|\mathbf{F}) \tag{4.3.25}$$

$$= \underset{\mathbf{E}}{\operatorname{argmax}} P(\mathbf{E}|\mathbf{D}, \mathbf{F}) P(\mathbf{D}|\mathbf{F}) \tag{4.3.26}$$

$$= \arg\max_{\mathbf{E}} P(\mathbf{D}|\mathbf{a}, \mathbf{F}, \mathbf{E}) \times P(\mathbf{a}|\mathbf{F}, \mathbf{E}) \times P(\mathbf{E}) \tag{4.3.27}$$

$$= \arg\max_{\mathbf{E}} P(\mathbf{D}|\mathbf{a}, \mathbf{F}) \times P(\mathbf{a}|\mathbf{F}, \mathbf{E}) \times P(\mathbf{E}) \tag{4.3.28}$$

Differing from [Xiong et al., 2008] in involving the use of both source and target sentences (see Formula 4.3.27), our reordering model only considers the source side (see Formula 4.3.28). This advantage greatly reduces the costs during decoding.

In Formula 4.3.28, there are three sub-models in Formula 4.3.28

- the language model $P(\mathbf{E})$
- alignment model $P(\mathbf{a}|\mathbf{F}, \mathbf{E})$
- reordering model $P(\mathbf{D}|\mathbf{a}, \mathbf{F})$

The generative story of Formula 4.3.28 is understood as follows: Once we found the hidden word alignment $\mathbf{a}$ with an alignment model $P(\mathbf{a}|\mathbf{F}, \mathbf{E})$ and the hidden derivation $\mathbf{D}$ using BTG-based reordering model $P(\mathbf{D}|\mathbf{a}, \mathbf{F}, \mathbf{E})$, we can translate the input source sentence $\mathbf{F}$ with the target translation $\tilde{\mathbf{E}}$. For the alignment model, we can easily obtain the intermediate variable $\mathbf{a}$ using unsupervised aligners that publicly available. The most difficult part is the reordering model.

Training

[Nakagawa, 2015] has shown that it is possible to obtain latent BTG parse trees (or parser derivations) for preordering. In this section, we shift our goal to reordering. Learning such a reordering model is relatively straightforward. As what we did in preordering, we maximize the conditional likelihood of trees that license the reorderings implied from observed word alignments in a parallel corpus. With the assumption that there exists an underlying derivation $\mathbf{D}$ that produced $\mathbf{E}$, where $\mathbf{E}$ is the target word orders under the constraints of BTGs. Translation is processed as follows:

$$\mathbf{F} \xrightarrow[\text{mapping via } \mathbf{a}]{\text{reordering with } \mathbf{D}} \mathbf{E} \tag{4.3.29}$$

where the derivations $\mathbf{D}$ is a latent variable. Since we have obtained the alignment model $\mathbf{a}$, given $(\mathbf{F}, \mathbf{D}) \rightarrow \mathbf{E}$, thus the objective function in our work can be represented as:

$$\begin{aligned} \tilde{\mathbf{E}} &= \arg\max_{\mathbf{E}} P(\mathbf{E}|\mathbf{F}) \\ &= \arg\max_{\mathbf{E}} P(\mathbf{E}|\mathbf{D}, \mathbf{F}) P(\mathbf{D}|\mathbf{F}) \end{aligned} \tag{4.3.30}$$

The task to translate an input source sentence can be solved by finding the latent derivation $\mathbf{D}$, which determines a target translation $\tilde{\mathbf{E}}$ with the maximal score in the above Equation.

It is reasonable to apply the score function $Score(\mathbf{D}|\mathbf{F})$ in BTG-based preordering to replace $P(\mathbf{D}|\mathbf{F})$. Following [Neubig et al., 2012a, Nakagawa, 2015], we assume that $Score(\mathbf{D}|\mathbf{F})$ is a linear combination of feature functions [Collins, 2002, Collins and Roark, 2004] defined over $\mathbf{D}$ and $\mathbf{F}$. We adopt BTG-based reordering model into SMT, which works as an inherent model in the decoder. In other words, the natural difference between their works and our work is the way of using the BTG-based reordering model, for preordering or decoding.

BTG-based Decoding

In our method, we train and use a BTG-based reordering model in three steps.

1. Training the BTG-based reordering model on the source side with shallow annotations. i.e., word alignments, POS-tags and word classes [Brown et al., 1992] on a bilingual corpus.

2. Estimating reordering scores use a large number of features, e.g., unigrams, bigrams and trigrams which can represent the current parser state[10].

3. During decoding, the reordering score works as an additional heuristic score which guides the decoder to pick out the best candidates among all hypotheses.

Assuming that the parser has the independent state for each step, we define the state at current state as a triple $\langle X, r, d \rangle$, where X is an unparsed span. For example,

[10]We make use of the same feature set described in [Nakagawa, 2015].

following the deductive proof system representations [Shieber et al., 1995,Goodman, 1999], $[X, p, q]$ covers $f_p, \ldots, f_q$. $d = \langle r, X \rightarrow \gamma \rangle$ is the derivation at current state with r is the splitting position between f_{r-1} and f_r and $X \rightarrow \gamma$ is the applied BTG rule. We employ our modified and boosted implementation of top-down BTG-based parser to obtain the reordering model[11].

The *-LM -RM* Decoder The integration of a standard n-gram-based language model into a CKY-style decoder is more complicated than into the standard phrase-based decoder [Koehn et al., 2003]. F llowing [Chiang, 2007], we first introduce the *-LM -RM* model in which the reordering and language model have been removed from the decoder:

$$w(\mathbf{D}) = \prod_{i \notin \{RM, LM\}} \Phi_i(\mathbf{D})^{\lambda_i} \tag{4.3.31}$$

If using the deductive proof system [Shieber et al., 1995,Goodman, 1999] to describe our *-LM -RM* decoder, the inference rules are as follows:

$$\frac{X \rightarrow Y_f / Y_e}{[X, p, q] : w} \tag{4.3.32}$$

$$\frac{X \rightarrow \langle X_1, X_2 \rangle : \quad [X_1, p, r] : w_1 \; [X_2, r+1, q] : w_2}{[X, p, q] : w_1 w_2} \tag{4.3.33}$$

$$\frac{X \rightarrow [X_1, X_2] : \quad [X_1, p, r] : w_1 \; [X_2, r+1, q] : w_2}{[X, p, q] : w_1 w_2} \tag{4.3.34}$$

where $X \rightarrow \gamma$ is the derivation rule, $[X, p, q]$ is the subtree rooted in a non-terminal X, w is the model score defined in Equation 4.3.31. When all terms on the top line are true, we derives the item on the bottom line. The final goal of the decoding is $[\mathbf{F}, 1, n]$, where $\mathbf{F}$ is the whole source sentence.

During decoding, the *-LM -RM* decoder flexibly explores the derivation without taking reordering into account. This strategy is a simple way to build a CYK-style decoder, but the decoder requires enormous beam size to find the best translation. Incorporating the *LM* and *RM* model directly into the translation construction will improve efficiency.

[11] We skip the sentences which cannot be parsed under the constraints of BTGs.

$$\frac{X \rightarrow Y_f/Y_e}{[X,p,q] : w[\Phi_{LM}(e)]^{\lambda_{\Phi_{LM}}}} \tag{4.3.35}$$

$$\frac{X \rightarrow \langle X_1, X_2\rangle : \quad [\exp P(X \rightarrow \langle X_1, X_2\rangle)]^{\lambda_{\Phi_{RM}}} \quad [X_1,p,r] : w_1 \; [X_2, r+1, q] : w_2}{[X,p,q] : w_1 w_2 [\exp \Phi_{RM}(X)]^{\lambda_{\Phi_{RM}}} [\Phi_{LM}(e_2 + e_1)]^{\lambda_{\Phi_{LM}}}} \tag{4.3.36}$$

$$\frac{X \rightarrow [X_1, X_2] : \quad [\exp P(X \rightarrow [X_1, X_2])]^{\lambda_{\Phi_{RM}}} \quad [X_1,p,r] : w_1 \; [X_2, r+1, q] : w_2}{[X,p,q] : w_1 w_2 [\exp \Phi_{RM}(X)]^{\lambda_{\Phi_{RM}}} [\Phi_{LM}(e_1 + e_2)]^{\lambda_{\Phi_{LM}}}} \tag{4.3.37}$$

$$X_1 \rightarrow f_1/e_1, X_2 \rightarrow f_2/e_2 \tag{4.3.38}$$

The *+LM +RM* Decoder The computational complexity of online strategy is reduced by using dynamic programming and incorporating the language model and the reordering model into decoding. A similar method has been described in [Chiang, 2007]. The decoder integrated with the n-gram language model is called: "*+LM* decoder". In our case, we also need to integrate the reordering model, so we call it "*+LM +RM* decoder". Given the inference rules described in Equations 4.3.32–4.3.34, we describe the *+LM +RM* decoding algorithm using Equations 4.3.35–4.3.38.

In our case, the reordering model affects computing the language model score if the derivation requires to swap the target sub-charts. We can calculate $\Phi_{RM}(X)$ by just taking the model score as the product of two sub-charts $\Phi_{RM}(X_1)$ and $\Phi_{RM}(X_2)$ with current reordering score $\Phi_{RM}(X \rightarrow \gamma)$. Since $\mathcal{R}$ is a log-linear expression, we compute the reordering score $\Phi_{RM}(X)$ for a given span $X : [X,p,q]$ that consists of $X_1 : [X_1,p,r]$ and $X_2 : [X_2, r+1, q]$ with a grammar rule $X \rightarrow \gamma$ as:

$$\Phi_{RM}(X) = \Phi_{RM}(X_1) + \Phi_{RM}(X_2) + P(X \rightarrow \gamma) \tag{4.3.39}$$

When we merge the chart $X_1 : [X_1,p,r]$ with $X_2 : [X_2, r+1, q]$ using the rule $X \rightarrow \gamma$, we update the total score for the composition model after applying each rule dynamically, we call this the *+RM* strategy. The BTG *terminal* rule ($T : X \rightarrow f/e$) is used to translate the source phrase f into the target phrase e while the *straight* and *inverted* rules ($S : X \rightarrow [X_1 X_2]$ and $I : X \rightarrow < X_1 X_2 >$) are used to concatenate

two neighboring phrases with a *straight* or *inverted* order as following:

$$e_x^y = \begin{cases} e_1 \cdot e_2, & X \rightarrow [X_1 X_2] \\ e_2 \cdot e_1, & X \rightarrow \langle X_1 X_2 \rangle \end{cases} \tag{4.3.40}$$

where $\cdot$ stands for concatenation between strings. After having decided the word order on the target side, we compute the score in the language model, noted $\Phi_{LM}(\cdot)$[12]. The language model score $P_{LM}(e_x^y)$ depends on the preceding $N-1$ words for any $e_x^y(|e_x^y| \geq N, 1 \leq x < y \leq m)$. It is computed as:

$$P_{LM}(e_x^y) = \prod_{x \leq z \leq y} p(\hat{e}_{z+N-1} | \hat{e}_z \dots \hat{e}_{z+N-2}) \tag{4.3.41}$$

The language model score function $\Phi_{LM}(e_x^y)$ depends on the rule type γ as follows:

$$\Phi_{LM}(e_x^y) = \begin{cases} P_{LM}(e_x^{y+1}), & |e_x^y| = |e_1^m| \\ 0, & |e_x^y| < N \\ P_{LM}(e_{x+N}^y), & \text{otherwise} \end{cases} \tag{4.3.42}$$

To determine whether we have the case $|e_x^y| = |e_1^m|$, we assume that, if the span of $X : [X, p, q]$ covers the entire source sentence f_1^n as $X : [X, 1, n]$, then the target translation e_x^y should also cover the entire target sentence. On the basis of *+RM* decoder, we add the *+LM* component into the decoder and build a *+LM+RM* decoder for CYK-style bottom-up decoding. Cube pruning [Chiang, 2007] is also applied to speedup the decoder.

We combine these models in a log-linear manner as shown in Equation 2.1. The feature functions employed are:

- Phrase-based translation models (TM): direct and inverse phrase translation probabilities, direct and inverse lexical translation probabilities.
- Language model (LM)
- Reordering models (RM): *straight* and *inverted* scores combined within the log-linear framework.

[12]For the case of start-of-the sentence and end of the sentence, we wrap the target sentence $\mathbf{e}$ (e_1^m) as $\hat{\mathbf{e}} = \hat{e}_1^{m+h} = \langle s \rangle^{N-1} e_1^m \langle \backslash s \rangle$.

- Penalties (PM): word penalty, phrase penalty, unknown word penalty.

The weights for each feature are tuned and estimated using the minimum error rate training (MERT) algorithm [Och, 2003].

4.4 Experiments

4.4.1 Preordering Experimental Setup

Table 4.1: Statistics of data used in preordering evaluation experiment and the training, development and testing sets for SMT experiments.

		Sentence Pairs	Tokens	
			English	Japanese
Preordering	Train (EM-100K)	100k	1.77M	1.82M
	Train (EM-10K)	10k	0.18M	0.18M
	Train (Manual)	653	17.1k	18.7k
	Test	582	12.5k	14.4k
SMT	Train	330k	6.09M	5.91M
	Dev	1,235	34.4k	30.8k
	Test	1,160	28.5k	26.7k

Data To evaluate the proposed methods, we conduct experiments on KFTT Corpus[13] (English–Japanese). The total training corpus is made of around 330,000 sentences. For preordering experiments, we prepare four training sets. Two small data sets: one, aligned by human annotators, which is initially provided in KFTT corpus, called Manual-653; another, the same data set aligned using `GIZA++`[14], called EM-653. Two large sets: one contains 10K sentence pairs, and another contains 100K sentence pairs with the corresponding word alignments obtained by using `GIZA++`[15], called EM-10K and EM-100K. For word alignments used for training the preorderers, we use the standard training regimen up to IBM Model 4, then symmetrized

[13]http://www.phontron.com/kftt/index-ja.html

[14]We run `GIZA++` on the whole training corpus.

[15]We randomly select the sentences from the training dataset.

bidirectional word alignments with the intersection heuristic. An additional test set with manual annotations is used for evaluation (initially provided in KFTT corpus). Translation experiments are measured on the official training, tuning and test sets.

Features For fair comparisons, the same features have been used as [Nakagawa, 2015], e.g., the head and tail of the current parser span (unigram, bigram, and trigram), the prefix and suffix of the splitting point, the parent BTG tree type of the current parser span, etc. Each word is factored as follows: lexical form, part-of-speech (POS) tag and word class [Brown et al., 1992].

Experiment Settings In our experiments, we compare parallel averaging methods (w/o the k-best list) with the online training method in [Nakagawa, 2015]. We use `TD-BTG-Preorderer`[16] for the baseline systems. For parallel implementation, we use `HieraParser` [17]. For SMT experiments, we utilize the `Moses` toolkit[18]. Standard phrase-based SMT systems [Koehn et al., 2007] are built with lexical reordering [Koehn et al., 2005], Minimum Error Rate Training [Och, 2003], and 5-gram `KenLM` language model [Heafield, 2011][19].

For POS taggers, we made use of Stanford Part-Of-Speech Tagger[20] [Toutanova et al., 2003] for English and KyTea[21] [Neubig et al., 2011a] for Japanese. To obtain word class tags, we used the implementation[22] of [Liang, 2005]. We fixed the class number to 256. We compared results in the both cases:

- no basic lexical reordering permitted (distortion limit=0);
- combining the default lexical reordering model with preordering (distortion limit=6).

[16]https://github.com/google/topdown-btg-preordering
[17]https://github.com/wang-h/HieraParser
[18]https://github.com/moses-smt/mosesdecoder
[19]h ld.com/code/kenlm/
[20]https://nlp.stanford.edu/software/tagger.shtml
[21]http://www.phontron.com/kytea/index-en.html
[22]https://github.com/percyliang/brown-cluster

For all experiments, we train the preorderers for 20 and 40 epochs. For distributed methods, we divided the training dataset into eight shards linearly. We run eight threads for multi-threading processing. For both `TD-BTG-Preorderer` and `HieraParser`, we employ the beam of size 20. For the k-best list, we selected the top 5 derivations. We perform preordering for SMT experiments using the models trained 20 epochs. The experiments reported in this section are measured on the same machine with an Intel Core i7-960 3.20GHz CPU (4 cores, eight threads) with 32GB RAM.

4.4.2 Preordering Evaluation

Intrinsic Reordering Evaluation

Intrinsic Metrics To evaluate preordering accuracy, we measure Fuzzy Reordering Score (FRS), Normalized Kendall's τ (NKT) and Complete Matching Score (CMS). We evaluate on the manual data set provided in KFTT corpus. Fuzzy Reordering Score [Talbot et al., 2011] is a widely-used evaluation metric which measures the quality of the reordering. This metric measures the continuity of the output order against the reference. Given the reference permutation $\mathbf{E}_{ref}$, FRS is calculated as the precision of word bi-grams:

$$\mathrm{FRS}(\mathbf{E}_{ref}, \mathbf{E}) = \frac{B(\mathbf{E}_{ref}, \mathbf{E})}{M(\mathbf{E}) + 1} \tag{4.4.43}$$

where B stands for the number of overlap bigrams appeared both in the reference source sentence and the reordered source sentence, M stands for the total number of words in the source sentence.

Kendall's τ [Kendall, 1938] is another simple metric:

$$\tau = 2 \times \frac{\text{number of increasing pairs}}{\text{the total number of pairs}} - 1 \tag{4.4.44}$$

Following [Isozaki et al., 2010], we justify this metric with normalization, because Kendall's τ may be a negative value. Normalized Kendall's τ is defined as following:

$$\mathrm{NKT} = 1 - \frac{C}{M \times (M-1)/2} \tag{4.4.45}$$

where C is the count of errors that pairs are not increasing.

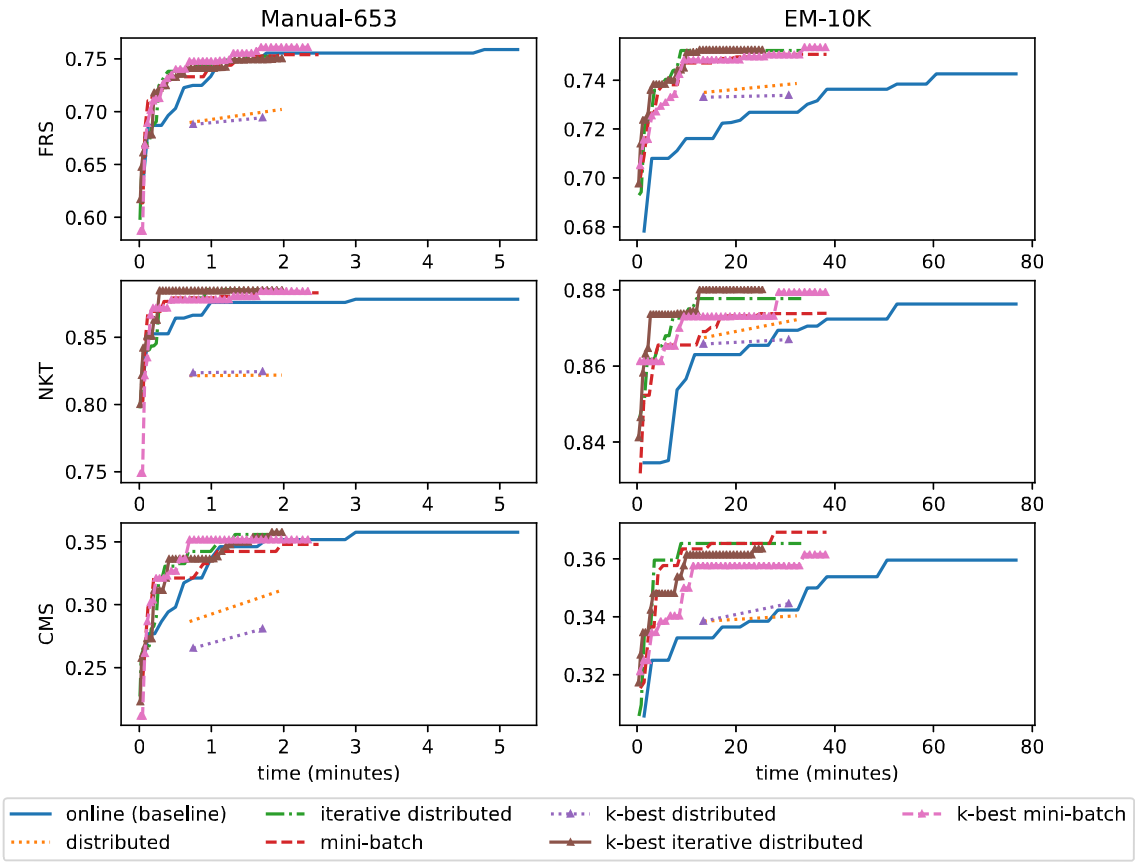


Figure 4.8: Best reordering scores changes over the training time (English–Japanese). Each curve stands for 40 epochs.

Complete Matching Score measures the percentage of the complete matches (the one that $\tau = 1.0$) against the total number of the testing examples. Assuming the test set contains N sentences, complete matching score is computed as:

$$\text{CMS} = \frac{\#\text{ of complete matches}}{N} \tag{4.4.46}$$

Reordering Results Reordering scores can be found in Table 4.2. Compared with the online training method [Nakagawa, 2015], all parallel averaging strategies for training the preorderer are sufficient to reduce the training time. In addition, these methods significantly improved the accuracy of reorderings in terms of CMS, FRS, and NKT when trained on automatically-aligned data sets, while underperformed on the small hand-aligned data set. Distributed averaging is not so effective on the small dataset, but it substantially upgrades with the size of the training dataset. Iterative distributed averaging and mini-batch averaging had the competi-

Table 4.2: Reordering scores and training times (in minutes) for English–Japanese. Bold numbers indicate significantly better than the baseline system (bootstrap re-sampling $p < 0.05$).

		epoch=20				epoch=40			
		FRS	**NKT**	**CMS**	**Times**	**FRS**	**NKT**	**CMS**	**Times**
					Manual-653				
online [Nakagawa, 2015]		75.54	87.57	35.19	2.30	75.86	87.83	35.77	5.25
greedy	distributed	68.96	83.13	26.73	0.70	70.19	82.18	31.15	1.97
	iterative distributed	74.38	87.47	34.23	0.73	75.24	87.72	35.19	1.87
	mini-batch	74.38	88.04	34.23	0.86	75.39	88.30	34.81	2.03
***k*-best**	distributed	68.78	83.31	28.08	0.74	69.41	82.44	28.08	1.71
	iterative distributed	74.13	88.47	33.65	0.77	75.03	88.50	35.87	1.97
	mini-batch	74.77	87.81	35.19	0.83	76.08	88.43	35.19	1.96
					EM-653				
online [Nakagawa, 2015]		69.67	85.20	32.12	3.77	69.62	85.32	30.57	8.39
greedy	distributed	64.10	80.85	25.19	1.07	66.76	82.00	27.12	2.55
	iterative distributed	68.94	84.57	30.00	1.41	69.52	84.93	30.38	3.47
	mini-batch	69.36	85.47	29.42	1.52	70.02	85.74	30.38	3.79
***k*-best**	distributed	61.42	79.16	23.65	0.98	62.46	79.73	24.81	2.34
	iterative distributed	67.87	85.17	29.62	1.25	68.74	85.54	30.77	3.16
	mini-batch	67.83	84.32	30.00	1.62	68.53	84.83	30.19	3.83
					EM-10K				
online [Nakagawa, 2015]		73.16	87.05	35.00	36.44	74.26	87.63	35.96	76.72
greedy	distributed	73.49	86.74	33.38	13.54	73.86	87.21	34.04	32.34
	iterative distributed	**74.52**	87.38	**36.15**	14.45	**75.22**	87.78	**36.54**	33.23
	mini-batch	**74.89**	87.32	**36.54**	19.05	75.05	87.39	**36.92**	38.33
***k*-best**	distributed	73.30	86.58	33.84	12.77	73.38	86.70	34.46	30.54
	iterative distributed	**75.15**	87.45	**36.15**	11.34	**75.25**	**88.01**	**36.35**	25.33
	mini-batch	**74.83**	87.31	**35.87**	17.85	**75.35**	87.35	36.15	38.08
					EM-100K				
online [Nakagawa, 2015]		74.66	87.58	36.35	394.23	76.62	88.63	37.68	811.17
greedy	distributed	**75.73**	87.48	35.77	129.34	75.80	87.72	36.15	266.21
	iterative distributed	**75.53**	87.68	36.15	151.25	**77.29**	88.40	37.97	334.26
	mini-batch	75.14	**88.02**	**36.97**	215.55	**76.99**	88.40	**38.45**	454.26
***k*-best**	distributed	**76.13**	87.74	36.15	184.92	76.33	87.93	36.35	406.75
	iterative distributed	**76.31**	87.79	**37.12**	154.11	**77.17**	88.45	38.07	353.05
	mini-batch	**76.66**	87.75	**38.07**	231.26	**77.27**	88.25	**38.65**	476.05

tive performance. Using the k-best list improves the results on EM-100K while no significant difference in EM-10K, resulting in slight changes in training time. *K-best* mini-batch averaging showed an outstanding effect on the small dataset (Manual-653). The results also exhibit that increasing the training data improves the overall reordering performance.

Extrinsic Reordering Evaluation

Extrinsic Metrics For translation evaluation, standard automatic evaluation metrics were used in all experiments, e.g. BLEU [Papineni et al., 2002] and RIBES [Isozaki et al., 2010]. We list BLEU because it is the most widely used metric. However, BLEU is insensitive to word order, especially for distant language pairs such as English–Japanese. Thus, we are also interested in RIBES, which is an automatic evaluation metric based on word order correlation coefficients between reference sentences and MT output.

Translation Evaluation Table 4.3 shows the translation scores. We observed that all parallel averaging methods had comparable performances in the SMT experiments either using EM-10K or EM-100K. It is worth noting that the automatically word-aligned datasets contain noise (alignment errors). Compared with the online training method [Nakagawa, 2015], our parallel averaging methods achieved comparable results on the larger automatically word-aligned data sets (EM-10K and EM-100K). In other words, training using parallel averaging methods are more stable near the convergence, and online training may not be necessary for good performance. Among these methods, though distributed averaging yielded significantly worse results on Manual-653 due to the unbalanced aggregation of sub-models, when using the larger training data set (EM-100k), it worked as well as other methods. K-best mini-batch averaging shows the best performance among all methods. Comparing Table 4.2 and Table 4.3, we found that the better quality of preordering does not necessarily lead to the better translation quality. There are some reasons can explain this phenomenon. On the one hand, for phrase-based SMT, it is well known that the performance of SMT systems is not strongly correlated with the quality

Table 4.3: Translation scores for each system. Bold numbers indicate no statistically significant difference with the best system. $^{\dagger}/^{\dagger\dagger}$: significantly better than the baseline system ($p < 0.05/p < 0.01$).

	en-ja				**ja-en**			
	DL=0		DL=6		DL=0		DL=6	
	BLEU	**RIBES**	**BLEU**	**RIBES**	**BLEU**	**RIBES**	**BLEU**	**RIBES**
No pre-reordering	19.79	65.91	21.24	68.01	16.21	65.29	18.45	65.66
				Manual-653				
baseline	**22.19**	**70.42**	**23.17**	**71.42**	**18.13**	66.24	**18.80**	66.80
greedy								
distributed	19.47	66.92	19.83	67.00	17.74	**67.18**†	**19.19**	**67.44**†
iterative distributed	20.76	68.31	21.72	69.36	17.35	66.06	18.50	65.66
mini-batch	**22.04**	69.57	**22.84**	70.61	17.12	66.35	**18.48**	66.77
***k*-best**								
distributed	20.78	67.79	21.73	69.27	17.21	**67.16**†	**18.66**	**67.58**†
iterative distributed	21.00	69.34	22.24	70.21	17.52	66.68	18.48	**67.24**
mini-batch	21.05	68.99	21.88	69.76	17.38	66.57	**19.04**	**67.37**
				EM-10k				
baseline	**22.77**	**71.37**	23.18	**71.81**	**18.53**	**67.55**	**19.16**	**68.07**
greedy								
distributed	22.17	70.07	22.61	71.15	**18.16**	**67.60**	**18.98**	67.74
iterative distributed	**22.56**	70.97	23.19	71.76	**18.63**	**67.99**	**19.36**	**68.15**
mini-batch	**22.99**	**71.45**	**23.60**†	**72.09**	**18.46**	**67.85**	**19.02**	**68.00**
***k*-best**								
distributed	22.06	70.12	22.83	71.27	17.95	66.91	**19.28**	67.28
iterative distributed	**22.83**	**71.61**	**23.34**	**72.23**	**18.28**	67.25	**19.26**	67.43
mini-batch	**22.85**	**71.15**	**23.64**†	71.94	**18.43**	66.43	**19.25**	**68.18**
				EM-100k				
baseline	23.15	**71.99**	23.57	72.08	18.81	67.73	19.47	68.08
greedy								
distributed	**23.28**	**71.55**	**23.98**†	**72.48**	18.61	67.91	19.28	67.81
iterative distributed	**23.24**	**71.63**	**23.81**	**72.56**†	18.40	**68.54**†	19.40	**68.75**†
mini-batch	**23.58**††	**72.05**	**24.03**††	**72.68**†	**19.48**††	**68.83**†	**19.83**	**68.78**†
***k*-best**								
distributed	22.92	**71.76**	**23.65**	72.12	18.84	67.73	19.55	**68.83**†
iterative distributed	23.01	**71.93**	23.54	**72.34**	18.78	**68.50**†	19.37	**68.65**†
mini-batch	**23.51**†	**71.89**	**23.81**†	**72.23**	**19.60**††	**68.87**††	**20.16**††	**68.51**†

of the latent variables, e.g., word alignment. For example, a lower alignment error rate (AER) does not necessarily entail better translation scores. On the other hand,

micro-reorderings between two leaves in a parse tree often lead to a higher reordering score. However, if the correspondences between the source and target words are captured in the phrase table, the translation accuracy may not changes much.

4.4.3 Decoding Experimental Setup

To evaluate our BTG-based SMT system, we conducted translation experiments on the KFTT Corpus (English–Japanese) and compared our system with baseline phrase-based (PB) and hierarchical phrase-based (HPB) SMT decoders provided in Moses[23] [Koehn et al., 2007]. For each language, the training corpus is around 330,000 sentences. The development set contains nearly 1,235 sentences. Nearly 1,160 sentences are used for testing. All models, e.g., translation models, target language models, and also traditional lexical [Koehn et al., 2005] reordering models or BTG-based reordering models are trained on the default training set. We use the default tuning set for tuning the parameters and the default test set for evaluation.

For word alignment, we first run `GIZA++` to obtain word alignments in both directions with the default settings, i.e., the standard bootstrap for IBM model 4 alignment in `GIZA++` ($1^5H^53^34^3$). Then, we symmetrize the word alignments using *grow-diag-final-and* (+*gdfa*) and the standard phrase extraction heuristic [Koehn et al., 2003] for all systems. For tuning, the optimal weights for each feature are estimated using the minimum error rate training (MERT) algorithm [Och, 2003] and parameter optimization with ZMERT[24] [Zaidan, 2009]. Other experiment settings are same to the previous section.

4.4.4 Decoding Evaluation

Experiment Results

For evaluation of machine translation quality, standard automatic evaluation metrics are used, like BLEU [Papineni et al., 2002] and RIBES [Isozaki et al., 2010] in all experiments. BLEU is used as the default standard metric, RIBES considers more

[23]http://www.statmt.org/moses/

[24]http://www.cs.jhu.edu/ ozaidan/zmert/

Table 4.4: Results of phrase-based baseline system, hierarchical phrase-based system and our BTG-based systems. Bold scores indicate no statistically significant difference at $p < 0.05$ from the best system [Koehn, 2004].

	BLEU	Model Size (GB)
`Moses` (PB-SMT)	20.81	> 2.0
`Moses` (HPB-SMT, beam=100 for rule, beam=30 for cell)	**21.67**	$\gg 5.0$
BTG-based SMT (beam=40)	**21.15**	**0.3**
BTG-based SMT (beam=100)	**21.24**	

word order. Table 4.4 shows the performance of MT systems on the KFTT test data, which are (1) `Moses`, trained using the phrase-based model (PB-SMT). (2) `Moses`, trained using the hierarchical phrase-based model (HPB-SMT) and last one (3) trained using the BTG-based model (BTG-SMT).

Analysis

Compared with the PB-SMT, BTG-based SMT makes use of weak linguistic annotations on the source side which provides additional information for reordering. We found that this strategy does help bilingual parse tree structure construction and finding final translations. However, our BTG-based method underperformed the HPB-SMT method. Increasing the beam size gains an improvement slightly.

There are two explanations for the result:

1. Final machine translation performance is also related to the used tools, which is sensitive to parse errors, alignment errors or annotation errors. Inaccurate labels harm the performance.

2. The constraints of BTGs may be too strict. This makes the decoder difficult to find some discontinuous phrases (translations).

4.5 Summary of the Chapter

This chapter provides an alternative to building syntax-based machine translation systems. We investigate the bidirectional latent variable of correspondence of syntactic structures across two languages. Syntactic representations are not directly observable for languages where no parser exists. Syntax-based SMT systems, such as hierarchical SMT [Chiang, 2005], allow to dispense with parsers, but they have more complex translation models. These methods heavily suffer from the long-distance reordering problem, typical of English-Japanese. The bracketing transduction grammar formalism (BTG) [Wu, 1997] provides a simple bidirectional syntactic model to answers this problem. The bilingual parse tree, i.e., BTG parse tree can be used to reorder words in the source or target sentences, either before the standard phrase-based SMT pipeline (preordering, for source), or during translation (decoding, for the target).

A method for latent BTG-based decoding is proposed in this chapter. We first improve the top-down BTG preordering method [Nakagawa, 2015], a state-of-the-art method, making less sensitive to errors in word alignments for initialization. Essentially, the training algorithm used bootstrap aggregating with several learning techniques (mini-batch, distributed, iterative distributed and k-best list) so as to select better reordering structures. As for the efficiency of preordering, intrinsic evaluation shows (a) a significant improvement of 2.00 FRS, 0.42 NKT and 1.65 CMS in reordering scores (b) 4 times faster in training time. When used for preordering in phrase-based SMT, the extrinsic evaluation shows that this method leads to (a) statistically significant gains in English–Japanese and Japanese–English translation accuracy (+0.5 and +0.8 BLEU point respectively, p-value < 0.01) (c) without substantial changes in the reordering model sizes. As for the efficiency of BTG-based decoding, the use of the reordering model learned in the previous step leads to (a) a statistically significant improvement (+0.43 BLEU point, p-value < 0.05) in English-Japanese translation accuracy over the standard `Moses` decoder (c) while reducing the reordering model size (20% of the state-of-the-art model size) in (b) a similar decoding time.

Chapter 5

Exploiting Bidirectional Latent Variable Models in Seq2seq NMT: Sub-word Segmentation

This chapter presents a bilingual word segmentation method for both SMT and NMT. We investigate the latent variable of vocabulary, which controls word segmentation. The quality of word segmentation affects the final translation quality. We propose to infer a limited size vocabulary using MDL and apply it to word segmentation. The main content of this chapter is on the basis of the following papers:

- Wang, H. and Lepage, Y. (2018b). Unsupervised word segmentation using minimum description length for neural machine translation. In *Proceedings of the 24th annual meetings of the Association for Natural Language Processing, Japan (言語処理学会第24回　次大会 発表論文集)*, pages 1080–1083

- Shan, B., Wang, H., and Lepage, Y. (2017). Unsupervised bilingual segmentation using MDL for machine translation. In *Proceedings of the 31th Pacific Asia Conference on Language, Information and Computation (PACLIC 31)*, pages 89–96

In this chapter, we draw our attention to word segmentation. We address the problem of word segmentation for East Asian Languages, e.g., Chinese or Japanese, which is a fundamental pre-processing step for MT as well as many other natural language processing tasks. This chapter includes our exploration of a bidirectional latent variable model, that is the vocabulary for word segmentation. Conventional supervised word segmenters, e.g., Juman and Stanford Segmenter, massively produce out-of-vocabulary (OOV) words that SMT cannot translate. This results in lower translation scores. Recent research has shown that making use of sub-strings, rather than words, is more efficient for SMT, because SMT somehow can solve data sparsity through learning the translation of OOV words via using such sub-word units.

The issue of OOV word is more critical for recent NMT. Compared to SMT, NMT is more conceptually straightforward in structure, where a single neural model is employed which is desired to capture all structural or phrasal phenomena. However, NMT cannot translate OOV words, neither make use of rare words. The reason is that NMT forces to use only a limited number of vocabularies to save the memory and keep the decoding efficient. This ignores the fact that translation is an open-vocabulary problem. For some languages, like Japanese and Chinese, they share some common lexicons, even subwords in their vocabularies. This chapter investigates how to a shared vocabulary for word segmentation for such a language pair. One of the many advantages of a bilingual word segmentation for NMT is that it allows sharing the word embedding layer for the encoder and the decoder. In NMT, the word embedding layer generally maps the words in a language to their semantic meanings using vector representations. A shared bilingual vocabulary makes the reality of more efficient NMT models by sharing a single word embedding layer. Recent research also shows that, for domain-specific translation tasks, machine translation systems prefer having longer common "words" in the vocabulary, e.g., entity names or terminologies. Only subword-based methods as byte pair encoding [Sennrich et al., 2016] decompose rare words into sub-words to deal with the OOV problem, which cannot satisfy this need.

Based on these considerations, we look for a more suitable solution to word

segmentation for both SMT and NMT. We propose a novel unsupervised method for bilingual word segmentation using the principle of minimum description length (MDL). Our contribution consists in imposing two additional restrictions: finite vocabulary (i.e., F) and minimum frequency (i.e., M) into the original MDL principle. This segmentation method based on a bidirectional latent variable model is advanced in the following aspects:

1. **Less Rare Words**

   This bilingual segmentation model tackles OOV words in the pre-processing step of machine translation. i.e., word segmentation. It eliminates the need for additional processes of word splitting during decoding. This is achieved by checking whether the frequency of new generated word is higher than the minimum frequency during the bottom-up lexicon inference, i.e., (a) less rare words. Such a bottom-up procedure also makes the vocabulary (b) train fast.

2. **Finite Vocabulary**

   A finite vocabulary of variable-length character sequences is sufficient to represent the current data, making it a very suitable word segmentation strategy for sequence-to-sequence neural network models, while (b) requiring little training time.

3. **Shared Vocabulary and Embedding**

   It exploits the overlap between source language lexicons and target language lexicons, bridging the encoder and the decoder via a shared embedding layer. This leads to (c) a dramatic reduction in memory cost.

The structure of this chapter is as follows.

- Section 5.1 introduces the background of word segmentation and the issue of the out-of-vocabulary word. We briefly discuss our motivation of bilingual word segmentation using MDL.
- Section 5.2 compares with related works in SMT and NMT. Substitution-based methods and sub-word–based methods are presented.
- We present our method, i.e., MDL-based bilingual word segmentation with finite vocabulary in Section 5.3.
- Section 5.4 deals with the experiments of SMT and NMT. The experimental results demonstrate the efficiency of the proposed bidirectional model in word segmentation for machine translation.
- Finally, we summarize this chapter in Section 5.5

5.1 Background

There are no distinct delimiters for the language which word boundary depends on context, like Chinese, Japanese and Korean (East Asian languages). For these languages, word segmentation is an essential pre-processing step in the standard MT pipelines. We show an example of the fact that may reflect how Japanese segmentation results vary over the Japanese segmentation tool used using Figure 5.1. Three segmenters: `Mecab`, `Juman` and `Kytea`, output quite different word segmentation results. The comparison actually discloses that difference in segmentation criteria

Segmenter	gu/li/i/n/pu/lo/da/ku/to/ma/i/ku
Juman	グリーン/ プロ/ダクト/ マーク
Mecab	グリーンプロダクト/マーク
Kytea	グリーン/プロダクト/マーク

Figure 5.1: An example of Japanese segmentation results.

leads to different segments. This brings up the question: whether the change of segmentation tool will make an influence on MT? (see Figure 5.2)

For SMT, it has been proven that the final end-to-end translation qualities vary with the segmenter used. Different segmenters produce different word segments as well as different numbers of OOV words, and consequently, leading to different translations. The quality of the MT system heavily relies on these segments. [Chang et al., 2008] show that segmentation consistency and granularity will affect the final SMT results. [Li et al., 2011,Zhao et al., 2013] suggest that different natural language processing tasks may have quite different requirements for the segmentation task, which is often beyond the issue of segmentation performance or standard. Similar conclusion for NMT also can be drawn from the results of the recent Workshop on Asian Translation (WAT) [Nakazawa et al., 2016a]

Compared to SMT, NMT expects not only less OOV words but also less rare words (low-frequency words), to ensure the translation quality, just because it cannot learn correct translations of the rare words. Since there is also an account for memory cost, NMT restricts vocabulary size to a limited number around 50k∼80k and only keeps the most frequent words to reduce the computational complexity. NMT simply

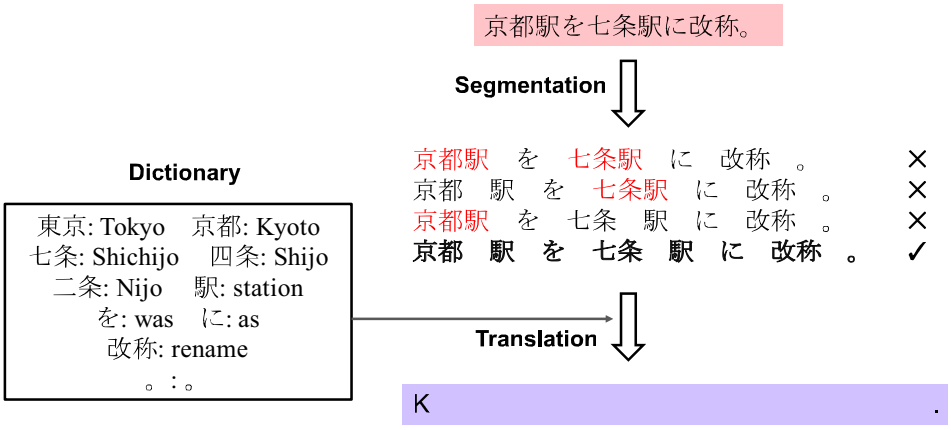


Figure 5.2: An example of how segmentation affects machine translation.

handles these out-of-vocabulary (OOV) words by converting these words into a single *unk* symbol. Rare words are probably removed during pre-processing.

A natural question arises:

- Can we develop a word segmenter for East Asian Languages, which is capable of reducing the number of OOV words in SMT or rare words in NMT?

5.2 Related Works

5.2.1 Translation through Substrings in SMT

There is a rich literature in the field of MT try to make use of substrings. Many recent works address the challenge of OOV word in SMT to improve translation quality. [Vilar et al., 2007] first examine the possibility of character-based translation method. [Neubig et al., 2012b] propose to build MT systems using character strings, i.e., substring has shown promising results in some European languages, e.g., English and Finish. This character-based translation method achieves comparable results as word-based systems while effectively translating unknown and uncommon words. Other works using substrings like [Noeman, 2009, Cherry and Suzuki, 2009] are related to transliteration, which deals with named entities such as person names, organization names and location names, given these names are crucial for machine translation.

5.2.2 Addressing Rare Words in NMT

Compared to SMT, not only OOV words, even rare words become more critical for NMT. NMT often operate with fixed word vocabularies, and the target word inference heavily depends on the vocabulary size. Hence, all out-of-vocabulary (OOV) words are converted to a single unk symbol during pre-processing. A significant weakness in conventional NMT systems is their inability to translate very rare words correctly. Many researchers [Sutskever et al., 2014a, Bahdanau et al., 2015] have observed that rare words harm the translation quality. Most of the recent works are focusing on effective techniques to address the rare word problem in NMT. To categorize, we classify these works into two classes:

Substitution-based Methods

[Luong et al., 2015b] describe a replacement approach that adding a post-processing step to translate the OOV word using a dictionary. During the post-processing, for each target OOV word in the output, they perform a word dictionary lookup to replace the OOV word $\langle unk \rangle$ by the corresponding translation on the basis of the extracted dictionary using an unsupervised aligner.

Another kind of methods use the indirect translation model, [Li et al., 2016] propose a substitution-translation-restoration method. In the substitution step, based on a word-similarity model, rare words are replaced with their most similar in-vocabulary words. In restoration steps, the translation of those rare words is substituted back using word alignments and an n-gram–based language model.

Subword-based Methods

The most recent works in improving translation quality by addressing the OOV problem for NMT are subwords-based models. These works are on the basis of a strong assumption that the segmentation of rare words into appropriate subword units is sufficient to allow for the neural translation network to learn transparent translations.

For example, to compress the data, [Chitnis and DeNero, 2015] propose to replace the rare words with two pseudo-word symbols using the Huffman encoding scheme.

Other methods attempt the encoding way using a finite vocabulary without adding any new symbols into the vocabulary, e.g., [Sennrich et al., 2016] propose the *Byte Pair Encoding* (BPE) scheme for NMT by split the low-frequency words into the higher-frequency subwords. [Wu et al., 2016] employ the wordpieces model [Schuster and Nakajima, 2012] to build the Google's NMT systems.

A common characteristic of all these methods is that to reduce OOV words, all these methods encode the plaintext to the sub-word sequences before training the NMT models. The sub-word–based model provides a good balance between the flexibility of single characters and the efficiency of full words, they have exhibited impressive results in morphologically rich languages.

5.2.3 Unsupervised Word Segmentation Using MDL

Unsupervised word segmentation (UWS) can provide domain-adaptive segmentation for statistical machine translation (SMT). Recent UWS works mainly fall into the use of minimum description length. The minimum description length principle [Rissanen, 1986, Grünwald, 1995, Barron et al., 1998] is an information theory criterion that provides a versatile solution for the selection of models. The philosophy of the MDL principle assumes that the selected model with a shorter code length should be able to describe the observed data better. Thus the principle of MDL is theoretically fit to the tasks like data compression and model inference. As a result, MDL has been widely used in various NLP tasks, for example, grammar induction [Grünwald, 1995, Jonyer et al., 2004, Saers and Wu, 2013], morphology analysis [Argamon et al., 2004], word segmentation [Zhikov et al., 2013, Magistry and Sagot, 2013] and translation model compression [González-Rubio and Casacuberta, 2014].

In general, the MDL principle attempts to find the minimal code or least complicated model that can describe the data observed. [Rissanen, 1986] devise the formula of MDL based on two-part codes, which includes the code[1] for encoding a model (noted Φ) over data (noted $\mathcal{D}$) and the code for the data using the model Φ. For

[1]The development of MDL borrowed heavily from Shannon's work on coding theory, hence we use the terms "code length" and "description length" interchangeably.

a given set of training data, following [Barron et al., 1998], the MDL inference is formalized as an objective function of two components: the model description length $\mathrm{DL}(\Phi)$ and data description length $\mathrm{DL}(\mathcal{D}|\Phi)$:

$$\hat{\Phi} = \arg\min_{\Phi} \mathrm{DL}(\mathcal{D}, \Phi) \tag{5.2.1}$$

$$= \arg\min_{\Phi} \mathrm{DL}(\mathcal{D}|\Phi) + \mathrm{DL}(\Phi) \tag{5.2.2}$$

The code length and the optional model in MDL are respectively similar to prior probability and marginal likelihood in the Bayesian framework. [Zhikov et al., 2013] conclude that, because of the close ties, the scheme of using MDL is equivalent to Bayesian inference. [Hansen and Yu, 2001] challenge this view that and give an alternative explanation. The MDL paradigm serves as an objective platform for distinguishing Bayesian procedures from non-Bayesian procedures. Only some forms of MDL would coincide with Bayesian schemes. However, this discussion is beyond the scope of this chapter. Throughout this chapter, we consider that the commonly used two-stage coding scheme in MDL is the same as the maximum a posteriori (MAP) problem in Bayesian analysis.

Many previous works are focusing on using MDL for prefix-based segmentation for either morphological analysis or word segmentation. [Grünwald, 1995] has shown that, although fewer symbols are used, it is possible to describe the whole data. [Argamon et al., 2004] propose to greedily construct a morph dictionary using MDL by re-segmenting affixes from the corpus. According to a given codebook Φ, we thus segment the text by looking up the codebook as a dictionary. Different strategies [Argamon et al., 2004, Hewlett and Cohen, 2011, Zhikov et al., 2013] have been proposed in the literature to compute description lengths for word segmentation. We follow [Argamon et al., 2004], defining the model description length as the product of the total length in characters:

$$\mathrm{DL}(\Phi) = b \sum_{w \in \Phi} len(w) \tag{5.2.3}$$

where b is the number of bits needed to encode a character, and $len(w)$ is the length of word w in characters. In the case of NMT, each word vector in the word embedding layer has the same code length. For the sake of simplicity, we assume

that b equals $\log |\Phi_c|$. $|\Phi_c|$ represents the initial character code book size. The total data description length is calculated as:

$$\mathrm{DL}(\mathcal{D}|\Phi) = -\sum_{w \in \Phi} C(w) \times \log P(w) \tag{5.2.4}$$

$$= -\sum_{w \in \Phi} C(w) \Big(\log C(w) - \log N \Big) \tag{5.2.5}$$

where $C(w)$ is the count number of word w (code book entry), N represents the count number of all tokens, and $P(w)$ stands for the probability of word w in the corpus.

In the framework of word segmentation, which is comparable to that of data compression, we minimize the description length $\mathrm{DL}(\mathcal{D}, \Phi)$ to find the minimal model Φ. Given that $\mathcal{D}$ is an unsegmented training corpus, in this case, word segmentation via MDL is an unsupervised learning task. The system is expected to learn a segmentation model from the raw data. The inferred model should be able to give a minimal code length both for the model itself and the corpus. Based solely on compression, MDL provides a robust foundation for refining the vocabulary into a more compact representation while identifying the 'words' in the text.

Previous MDL-based word segmentation methods for NMT lack a clear goal. Most previous methods treat MDL as a word segmentation approach that attempts to produce good 'words' in comparison either with human-annotated data or with other conventional segmenters trained in a supervised manner. However, [Li et al., 2011] and [Zhao et al., 2013] suggest that segmentation tasks should not be separated from their applications. Therefore, in this chapter, we investigate the application of MDL-based segmentation for bilingual tasks like SMT and NMT taking only the final end-to-end translation scores to measure the segmentation quality.

5.3 Proposal: Bilingual FM-MDL–based Word Segmentation

Conventional monolingual supervised segmenters often massively produce rare words that MT systems cannot make use of, which directly harms the final translation

results. And they produce segments in each language separately. Hence, it is natural to find a proper segmentation standard that can reduce the rare words.

We find that the key to the OOV word issue, in fact, is the vocabulary behind the process of segmentation. To exploit the role of the vocabulary in word segmentation, we investigate different scales of sizes in the experiments. We assume that a well-learned[2] finite vocabulary is sufficient to perform rough segmentation. This should also benefit the learning of translation models. This assumption has two restrictions: first, a productive vocabulary can be obtained by restricting the minimum frequency of each word, and second, a finite and compressed vocabulary is capable of representing the original vocabulary.

The above discussion also suits the bilingual case where lexicon inference is performed for both the source and target languages. We propose a novel FM–MDL-based segmentation method based on the principle of minimum description length [Rissanen, 1978, Grünwald, 2007] imposing two additional restrictions :

- minimal frequency
- finite vocabulary

Though the finite vocabulary inferred cannot completely solve the OOV problem, neither cover all possible plain words in each language, the inference starting from the smallest writing units (i.e., CJK characters) should achieve the best trade-off between representation effectiveness and word productivity. Our bilingual word segmentation method is a two-phase process:

- Firstly, starting with character vocabulary and a parallel corpus, we attempt to infer a codebook with a limited number of entries. Based on the principle of MDL, the number of entries in codebook will continue to grow in an iterative procedure.
- Secondly, the inferred codebook allows reusing for word segmentation. Our goal is to tackle the OOV problem during the process of word segmentation

[2]'Well-learned' means that the vocabulary of words learned can describe most words in a language, though it is much smaller than the vocabulary of all existing words.

for these languages.

5.3.1 Minimum Frequency and Finite Vocabulary Restrictions

Since NMT cannot make use of OOV words, it is reasonable to only focus on high-frequency words rather than low-frequency words. The minimum frequency restriction requires entries in the vocabulary have higher frequencies than the minimum frequency. Preliminary experiments [Sennrich et al., 2016] show that such a filtering strategy contributes to NMT in learning the translation of OOV words through high-frequency morphemes or phonemes. Another restriction is fixing the vocabulary to a limited size. [Wu et al., 2016] obverse that a fine-tuned number of vocabularies lead to a drastic reduction in both estimation time and the number of OOV words.

Our situation also differs from the standard MDL in that the vocabulary size is fixed and the minimum frequency of word need to be restricted. [Hansen and Yu, 2001] has shown that the MDL paradigm is also suitable for the problem of finitely coding. The assumption is that the learned finite vocabulary is sufficient to perform rough prefix-based segmentation, and further, it will benefit the NMT to learn transparent translations.

Following [Sennrich et al., 2016] and [Wu et al., 2016], we only consider two variations of MDL to word segmentation for NMT:

- one with the minimum frequency restriction only (i.e., M), called M-MDL.
- another with the minimum frequency restriction (i.e., M) and the additional restriction on vocabulary size (i.e., F), to distinguish, we call it FM-MDL[3].

5.3.2 Inference Procedure

We now describe how to perform the maximization in Eq. 5.2.2. [Schuster and Nakajima, 2012] indicate that, for CJK languages, in general, the size of the common characters is further smaller than the fixed vocabulary size desired in NMT. Con-

[3]F stands for "finite vocabulary", and "M" stands for "minimum frequency restriction".

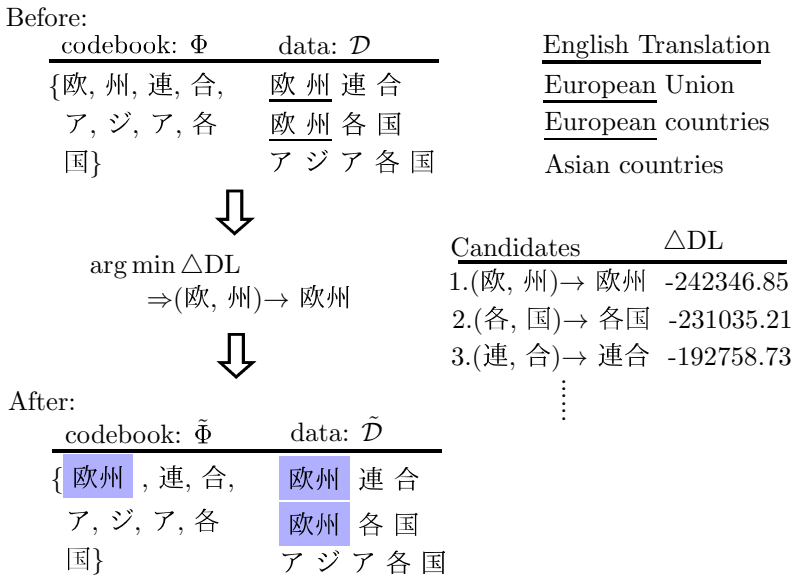


Figure 5.3: A Japanese example of bottom-up code book inference using MDL. The greedy heuristic yields a candidate with the minimal △DL for updating at each time. This method is also suitable for bilingual segmentation, though we only show a monolingual examples here.

sider the unsegmented case (see Figure 5.3), in the initial code book, each character is an entry in the code book. To reduce the description length, given two adjacent characters w_1, w_2, we try to insert a new entry into code book which is a bigram w_1w_2 by merging w_1 with w_2. A greedy method of minimizing DL is: each time finding a new longer entry to insert that has the minimal △DL and updating the code book recursively. Assume $\tilde{\Phi}$ and $\tilde{\mathcal{D}}$ are the new model and the new data after insertion of a new entry, the total change in coding cost for the code book and data encoding is:

$$\triangle \text{DL} = \text{DL}(\tilde{\mathcal{D}}, \tilde{\Phi}) - \text{DL}(\mathcal{D}, \Phi) \tag{5.3.6}$$

$$= \triangle \text{DL}(\tilde{\Phi}, \Phi) + \triangle \text{DL}(\tilde{\mathcal{D}}, \mathcal{D}) \tag{5.3.7}$$

To reduce the description length, taking two adjacent characters w_1, w_2 and given the previous and next characters, we try to insert the pair w_1w_2 as a new entry into the code book by merging w_1 with w_2. Each update of the code book, i.e. inserting a longer entry or deleting shorter unused entries should reduce the description length.

For each pair candidate, to estimate the variation in data description length,

we compute the change of data description length ($\triangle$DL) before and after inserting this entry into the code book iteratively. The length difference between the old code book Φ and the new code book $\tilde{\Phi}$ is trivial. Given the deletions/insertion in code book, we compute the model description length difference $\triangle\text{DL}(\tilde{\Phi}, \Phi)$ according to $\tilde{\Phi}$ and Φ by classifying them into four cases:

$$\triangle \text{DL}(\tilde{\Phi}, \Phi) = b \times \begin{cases} len(w_1 w_2) & , \{w_1, w_2\} \subset \tilde{\Phi} \\ len(w_1 w_2) - len(w_1) - len(w_2) & , \{w_1, w_2\} \not\subset \tilde{\Phi} \\ len(w_1 w_2) - len(w_1) & , w_1 \notin \tilde{\Phi} \\ len(w_1 w_2) - len(w_2) & , w_2 \notin \tilde{\Phi} \end{cases} \tag{5.3.8}$$

Algorithm 8 and Algorithm 9 (see Appendix B) indicate the details of M-MDL and FM-MDL. The key component in both Algorithm 8 and Algorithm 9 is the function $\triangle$DL. However, we cannot afford to perform segmentation for each candidate, which is an extremely expensive process. In our proposal, we first collect the potential combinations of the current entries, then we sort the candidates with their approximate value of $\triangle$DL. Finally, we commit to the best candidate with the minimal $\triangle$DL in queue recursively. Specifically, we apply the codebook only if there are no valid candidates in the queue. After several iterations, it may collect enough entries for a fixed number.

Note that the variation in data length exists after inserting a new entry $w_1 w_2$. The data change contains both the decreases in coding length of w_1 and w_2 and the increases in coding length of $w_1 w_2$. Besides, there are changes in the coding cost for the remaining entries in the corpus (i.e, other tokens in the corpus), which affects the frequency of those entries in code book. We use $O(1)$ to represent the bias:

$$\triangle \text{DL}(\tilde{\mathcal{D}}, \mathcal{D}) = \triangle\text{DL}(w_1) + \triangle\text{DL}(w_2) + \text{DL}(\tilde{\mathcal{D}}|w_1 w_2) + O(1) \tag{5.3.9}$$

The estimation of the cost for w_2 is analogous to w_1. Thus, from N to $\tilde{N}$, the term of $\triangle\text{DL}(w_1)$ in Eq. 5.3.9 can be rewritten as:

$$\triangle \mathrm{DL}(w_1) = C(w_1) \times \log\Big(\frac{C(w_1)}{N}\Big) - \Big(C(w_1) - C(w_1 w_2)\Big) \times \log\Big(\frac{C(w_1) - C(w_1 w_2)}{\tilde{N}}\Big) \tag{5.3.10}$$

where $\tilde{N}$ is the new data length, equals to $N - C(w_1 w_2)$. The bias $O(1)$ is computed as following:

$$O(1) = \sum_{w}^{|\bar{\Phi}|} C(w) \times \log\Big(\frac{C(w)}{N}\Big) - \sum_{w}^{|\bar{\Phi}|} C(w) \times \log\Big(\frac{C(w)}{\tilde{N}}\Big) \tag{5.3.11}$$

$$= \sum_{w}^{|\bar{\Phi}|} C(w) \times \Big(\log \tilde{N} - \log N\Big) \tag{5.3.12}$$

$$= \Big(N - C(w_1) - C(w_2)\Big) \times \log\Big(\frac{N - C(w_1 w_2)}{N}\Big) \tag{5.3.13}$$

[Zhikov et al., 2013] point out that the operation of replacing on the entire corpus is prohibitive and retrieval of indices is a challenging task. [Argamon et al., 2004] employ the *main suffix trie* structure for fast retrieval. In our proposal, we use a *suffix array* [Kärkkäinen and Sanders, 2003] to remember the indices for updating the pair statistic on-the-fly. Algorithm 10 shows how to update the statistics.

5.3.3 Prefix-based Segmentation

There is a sizable literature on word segmentation: recent reviews include [Sproat and Emerson, 2003] and [Huang and Zhao, 2007]. Previous works, which attempt to extract features to identify word boundaries from annotated training datasets, are mainly based on supervised learning. These methods are divided into two major categories, namely lexical rule-based approaches [Sproat et al., 1996], and tagging-based approaches [Xue et al., 2003].

A few studies [Mochihashi et al., 2009, Hewlett and Cohen, 2011, Zhikov et al., 2013] also focus on unsupervised learning approach, which reports that it is possible for an unsupervised learning approach with a well-defined model to outperform the widely-used supervised approach.

In this paper, we apply the forward maximum matching (FMM) algorithm for deterministic segmentation. During segmenting, the segmenter looks up each possible

candidate as a sequence in the codebook, and only outputs the longest one. Although FMM is the simplest method, it can output reasonably good word segments for SMT and NMT on account of the number of OOV words and the translation scores.

5.4 Experiments

5.4.1 Datasets

We evaluate our method on the Asian Scientific Paper Excerpt Corpus (ASPEC) [Nakazawa et al., 2016b].

For Chinese–Japanese and Japanese–Chinese translation tasks, we use the default training, development, and test sets to measure the performance of our MDL-based segmenters. The final training corpus contains 672,315 sentence pairs. We follow the official development/test split of ASPEC-JC in WAT[4]: 2,090 sentence pairs for development, and 2,017 sentence pairs for testing.

For English-Japanese translation tasks, we use the top 1.5M sentence pairs in the training set of ASPEC Corpus (ASPEC-JE), because this corpus contains some unreliable sentence pairs (their translation probabilities rank sentence pairs). We use the official development/test split: 1,790 sentence pairs for development, and 1,812 sentence pairs for testing. We also filter out the sentences longer than 50 words.

5.4.2 Experiment Settings

For all experiments, we followed the basic parameter setting of the baseline system in WAT translation task[5]. We describe the details as follows.

[4] `http://lotus.kuee.kyoto-u.ac.jp/ASPEC/`

[5] `http://lotus.kuee.kyoto-u.ac.jp/WAT/WAT2017/baseline/baselineSystems.html`

Tokenizers/Segmenters

For English, we used the `Moses` script[6] as the default tokenizer. For Chinese and Japanese word segmentation, Stanford Segmenter[7] and Juman[8] were used.

SMT Settings

For the SMT experiments, we utilize the `Moses` toolkit[9] to build a standard phrase-based SMT system for translation experiments. The systems [Koehn et al., 2007] are built with `Moses` with lexical reordering [Koehn et al., 2005], Minimum Error Rate Training [Och, 2003], and the 5-gram `KenLM`[10] language model [Heafield, 2011]. We set the distortion limit as 20. For bilingual setup, we learn a joint vocabulary for both the source and target languages.

NMT Settings

We utilize an NMT system that follows an encoder-decoder architecture with attention mechanism [Luong et al., 2015a] as our baseline system, which has been shown in Figure 2.6. All NMT systems are built with an open-sourced NMT implementation: Open-NMT.[11] We use a minibatch stochastic gradient descent (SGD) method to train NMT models with the following arguments: 2 hidden layers for both encoder and decoder, word embedding of size 500, mini-batch size 64, hidden la .3 for all inter-layers to prevent over-fitting, 13 epochs for training, single model (no ensemble), and decoder beam with a size of 5. The vocabulary size of input and output are set to 50K. Training sentences are randomly selected. For bilingual setup, we share the word embedding layer between the encoder and decoder.

[6] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl`

[7] `https://nlp.stanford.edu/software/segmenter.shtml`

[8] `http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN`

[9] `https://github.com/moses-smt/mosesdecoder`

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

[11] `https://github.com/OpenNMT/OpenNMT-py`

Subwords Models

To measure the performance of our FM-MDL segmenter, we compare our proposal with other state-of-the-art subword methods. We choose `SentencePiece`[12] (*model_type=bpe*) for BPE without pre-segmentation as our baseline systems, notes *BPE w/o pre-seg*[13]. To apply BPE after segmentation, notes *BPE w pre-seg*, we first segment the raw texts using Stanford Segmenter/Juman, then apply the implementation of `subword-nmt`[14]. We also show the result of recent sentence piece model `SentencePiece` use (*model_type=unigram*) which allow to perform sub-word segmentation without using any pre-tokenizer/segmenter.

To compare the effectiveness of the additional restrictions, we consider two variations of MDL: M-MDL (without the limitation of vocabulary size) and FM-MDL. M-MDL serves to prove the importance of vocabulary size restriction. We set the minimum frequency of `subword-nmt`, M-MDL or FM-MDL to 5. We limit BPE, SPM, and FM-MDL to a limited vocabulary size as 50K to ensure that the generated vocabulary can be used by all NMT systems having built. In particular, we process the English sentences using `Moses` tokenizer with `subword-nmt` for all experiments involving English.

We evaluate two methods of applying MDL:

1. learning two independent vocabularies, one for the source, one for the target, which we call monolingual MDL;

2. or learning a shared vocabulary on the union of the two vocabularies, which we call bilingual MDL.

Evaluation Metrics

To evaluate the translation quality, we use BLEU [Papineni et al., 2002] and RIBES [Isozaki et al., 2010] as the evaluation metrics. BLEU is the usual standard metric for MT. RIBES assesses word order as well as N-gram–based matches in BLEU. It

[12] `https://github.com/google/sentencepiece`

[13] The original implementation of [Sennrich et al., 2016] does not support unsegmented text.

[14] `https://github.com/rsennrich/subword-nmt`

penalizes word order mistakes. We also separately study the numbers of unknown words in the development/test set.

5.4.3 Experimental Results

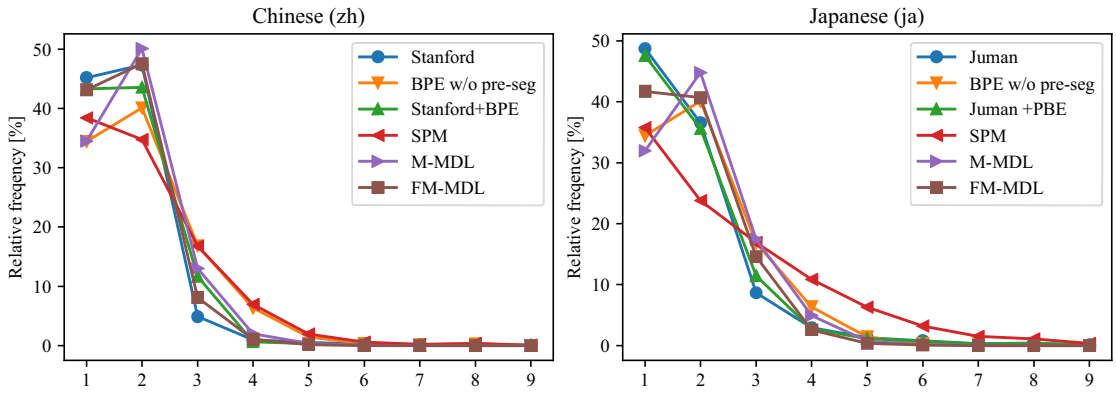


Figure 5.4: Statistics for word lengths in CJK characters (from 1 to 9) using various segmentation tools for monolingual setups.

Segmentation Results

Our major objective is to tackle the OOV problem in segmentation for NMT. At first, we exploit the distribution of the subword length in terms of the subword frequency for each language independently. Figure 5.4 gives the statistics on the training set. Stanford and Juman are baseline systems in Chinese and Japanese respectively. We analyze these two graphs separately. In the first graph (Chinese), the length of most words is less than 5. However, as shown in the second graph, there exist more longer words in Japanese. It reflects the great differences that exist between Chinese and Japanese. As shown in Figure 5.4, in both languages, high-frequency words are centered on the single-character and two-character word. As the comparison, WPM produces more words that are longer than others, on the contrary, FM-MDL is prone to generate somewhat shorter sub-words but there still exist more longer words than other methods. According to this figure, M-MDL always generates the least number of single-character words. It is apparent from the graphs by comparison across languages that the average length of Chinese words

Table 5.1: Corpus statistics for Chinese with different word segmentation techniques, e.g., byte pair encoding (BPE), SentencePiece, MDL with minimum frequency restriction (M-MDL) and MDL with minimum frequency and finite vocabulary restrictions (FM-MDL).

		#tokens	**#types**	**avg len±std.**	**#unk tokens**	**#unk types**
	train	18.21 m	307,139	27.08±14.62		
Stanford	dev	59,279	7,915	28.36±15.58	650	541
	test	58,318	7,724	27.68±14.13	588	505
	train	19.71 m	53,076	29.32±17.38		
Stanford+BPE	dev	63,178	7,825	30.23±17.08	9	8
	test	62,115	7,636	29.48±16.05	1	1
	train	15.40	52594	22.90±13.20		
BPE	dev	63430	7045	30.35±16.79	5	5
	test	48334	12486	22.94±12.28	1	1
	train	15.68 m	52,672	23.33±13.33		
SentencePiece	dev	49,955	12,069	23.90±13.50	5	5
	test	49,249	12,028	23.37±12.40	1	1
	train	17.07 m	72,714	25.39±14.30		
M-MDL	dev	55,338	9,757	26.48±14.65	3	3
	test	54,589	9,584	25.91±13.68	0	0
	train	18.16 m	35,839	27.01±15.33		
FM-MDL	dev	58,542	8,059	28.01±15.58	3	3
	test	57,648	8,017	27.36±14.49	0	0

is longer than the Japanese. This phenomenon may be explained by the fact that MDL considers the coding cost in the model, while WPM not.

For more details, Table 5.1 and Table 5.2 show the number of OOV word in the training, development and test sets. Different methods output the different amounts of tokens. The sub-word models significantly reduce OOV words. The numbers of types decrease by up to 80% for Chinese and 50% for Japanese. For the training sets of both languages, FM-MDL produces the minimal identical word types, followed by SPM or BPE. For the test and development set, by contrast, WPM has the most types. Although FM-MDL generates the smallest number of types among all these methods against the training set, BPE has the minimum tokens in the development and test sets. WPM has almost over four-fifth average length as the baselines, which

T le 5.2: Corpus statistics for Japanese with different word segmentation techniques.

		#tokens	#types	avg len±std.	#unk tokens	#unk types
	train	22.21 m	150,779	33.04±17.57		
Juman	dev	70,035	7,057	33.51±17.94	291	214
	test	69,015	7,036	32.76±16.67	261	217
	train	22.92 m	48,088	34.09±18.66		
Juman + BPE	dev	72463	7,047	34.67±18.81	0	0
	test	71716	7,027	34.04±17.61	0	0
	train	23.28	22800	34.62±18.69		
BPE	dev	73320	6835	35.08±18.87	1	1
	test	72548	6854	34.43±17.54	1	1
	train	16.00 m	52,022	23.80±13.65		
SentencePiece	dev	49,867	13,358	23.86±13.52	0	0
	test	49,715	13,237	23.60±12.50	0	0
	train	20.13 m	59,130	29.94±16.54		
M-MDL	dev	64,072	9,345	30.66±16.82	0	0
	test	64,354	9,190	30.54±15.91	0	0
	train	20.71 m	33,848	30.80±17.07		
FM-MDL	dev	65,391	8,383	31.29±17.26	0	0
	test	65,614	8,318	31.14±16.19	0	0

is the shortest one at the same time. We also investigate the number of OOV word in the training, development and test sets. SentencePiece (WPM) shows slightly better text compression ratio than other methods, but no significant differences in BLEU score.

The more direct-viewing results, i.e., some segmentation results, can be found in Figure 5.5. For Japanese, different methods output significantly different segmentation. The Japanese segmentation results that how this method deals with the problem of Japanese *Katagana* transliteration using the first example. FM-MDL broke the high-frequency Japanese word (blue, "conbiniensu") into pieces as same as SPM, but produce a different segmented sequence. For the low-frequency Japanese word (red, "i-rudo"), BPE splits it into single characters. SPM and FM-MDL output different solutions. F lty exists in Chinese word segmentation is that to handle the segmentation disambiguation problem. For ex-

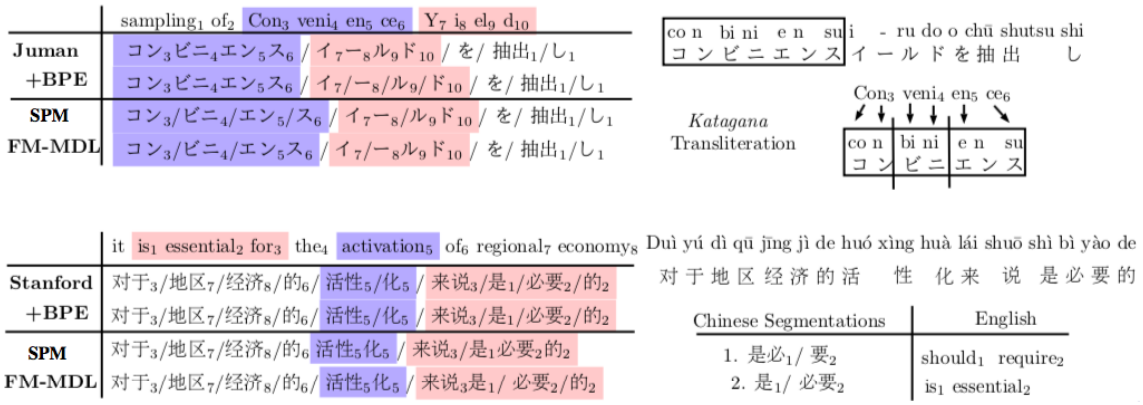


Figure 5.5: Samples of segmentation results (top: a Japanese transliteration example, bottom: a Chinese word segmentation disambiguation example).

ample, "*shì bì yào*" is ambiguous in Chinese, which has two segmentation ways. *shìbì yào* means "should require" while *shì bìyào* means "is essential". The segmentation results indicate that all these methods can deal with the segmentation ambiguity somehow, but vary with the segmentation schemata. FM-MDL concatenated *shì* to the previous word. WPM glued characters into one word *shìbìyào*.

Translation Results

Table 5.3, Table 5.4 and Table 5.5 show the automatic evaluation results on ASPEC tasks. NMT systems achieved better results than SMT systems (4.61 BLEU points for zh–ja and 4.22 BLEU points for zh–ja). With the restriction of limited vocabulary improved the translation quality significantly by up to (0.5∼0.9) BLEU point relatively. All these models (BPE, WPM, M-MDL, and FM-MDL) surpassed the baseline system with a significant difference in both BLEU and RIBES. However, the values of perplexity and accuracy vary with the different methods. WPM outperformed word-based methods BPE on the Chinese-Japanese direction but worked not robust on the Japanese-Chinese direction. Pre-tokenization can slightly improve the BLEU scores in English to Japanese. In Japanese to English translation, the conclusion that BPE might not be effective for NMT can be drawn from the final results. We observed that the proposed M-MDL and FM-MDL methods could significantly improve translation quality, but FM-MDL is more strong and robust than M-MDL. We also did the experiments of shared vocabulary and shared embedding, and we

Figure 5.6: Samples of segmentation and translation results.

	Source (Chinese)	**Target** (Japanese)
Stanford/Juman	食品 添加剂 联合 委员会 （ JECFA ）	食品 添加 物 合同 委員 会 （ < *unk* > ）
+ BPE	食品 添加剂 联合 委员会 （ J E C F A ）	食品 添加 物 連携 委員 会 （ Ｊ Ｅ Ｃ Ｆ Ａ ）
BPE w/o seg	食品 添加剂 联合 委员会 （ J EC FA ）	食品 添加 剤 連携 委員 会 （ Ｊ Ｅ Ｃ Ｆ Ａ ）
SentencePiece	食品添加剂 联合 委员会 （ J EC FA ）	食品添加物 合同 委員会 （ Ｊ Ｅ Ｃ Ｆ Ａ ）
FM-MDL	食品 添加剂 联合 委员会 （ J EC FA ）	食品 添加 物 合同 委員 会 （ Ｊ Ｅ Ｃ Ｆ Ａ ）
English	The Joint Committee on Food Additives (JECFA)	
Chinese	联合 委员会 食品 添加剂 (JECFA)	
Japanese	合同 委員 会 食品 添加物 （ Ｊ Ｅ Ｃ Ｆ Ａ ）	

	Source (Chinese)	**Target** (Japanese)
Stanford/Juman	使用 杉木炭 净化 地下 水 质 的 研究	< *unk* > を 用いた 地 下水 浄化 の 研究
+ BPE	使用 杉 木炭 净化 地下 水 质 的 研究	スギ 炭 を 用いた 地 下水 の 浄化 に 関する 研究
BPE w/o seg	使用 杉 木炭 净化 地下水 质 的研究	スギ 木炭 を 用いた 地 下水 質 浄化 の 研究
SentencePiece	使用 杉 木炭 净化 地下水 质 的研究	スギ 炭 を用いた 地下水 質 浄化 に関する研究
FM-MDL	使用 杉木 炭 净化 地下水 质 的 研究	スギ 炭 を 用いた 地 下水 浄化 の 研究
English	research on water purification using the charcoal of cedar	
Chinese	研究 地下水 净化 使用 炭 杉木	
Japanese	研究 地下水 浄化 用いた 炭 スギ	

	Source (Chinese)	**Target** (Japanese)
Stanford/Juman	油及 Ｈ Ｎ Ｓ 泄漏 事故 预防 处理 术	< *unk* > Ｈ Ｎ Ｓ 流出 事故 予防 処理 技術
+ BPE	油 及 Ｈ Ｎ Ｓ 泄漏 事故 预防 处理 术	油 及び Ｈ Ｎ Ｓ 流出 事故 防止 処理 技術
BPE w/o seg	油 及 Ｈ Ｎ Ｓ 泄漏 事故 预防 处理 术	油 および Ｈ Ｎ Ｓ 漏洩 事故 防止 処理 技術
SentencePiece	油 及 Ｈ Ｎ Ｓ 泄漏 事故 预防 处理 术	油 及び Ｈ Ｎ Ｓ 漏洩 事故 予防 処理技術
FM-MDL	油 及 Ｈ Ｎ Ｓ 泄漏 事故 预防 处理 术	油 および Ｈ Ｎ Ｓ 漏洩 事故 防止 処理 技術
English	Oil and HNS leak prevention and treatment technology	
Chinese	油 及 Ｈ Ｎ Ｓ 泄漏 预防 处理 术	
Japanese	油 及び Ｈ Ｎ Ｓ 漏洩 予防 処理 技術	

found our methods have no statistically significant difference with the monolingual case.

Table 5.6 gives the training time. Although all these methods are capable of multi-threading in a production environment, to provide a fair comparison, we ran within a single threaded mode when computing training times. The CPU time was measured on an Intel Core i7-960 3.20GHz with 32GB RAM. Compared to BPE and SPM, FM-MDL significantly reduced training times[15].

5.4.4 Comparison

In this section, we characterize our proposed model by comparing with other subword-based models (e.g. BPE [Sennrich et al., 2016], WPM [Schuster and Nakajima, 2012]

[15]The segmentation time may be ignored.

Table 5.3: English–Japanese NMT experiment results on ASPCE-JE Corpus (50K).

		en → ja		**ja → en**	
		BLEU	**RIBES**	**BLEU**	**RIBES**
SMT	[Koehn et al., 2003]	27.50	68.41	19.52	64.31
NMT					
Juman + Stanford	[Luong et al., 2015a]	34.24	81.45	26.45	71.61
BPE w/o pre-seg (*baseline*)	[Sennrich et al., 2016]	34.56	82.15	27.13	71.71
BPE w pre-seg	[Sennrich et al., 2016]	34.78	81.79	**28.03**	**71.62**
SPM	[Wu et al., 2016]	**35.57**	**82.40**	27.12	71.17
Proposed					
M-MDL		35.01	82.03	27.22	71.31
FM-MDL		**35.35**†	**82.43**†	**28.15**†	**72.01**†

Table 5.4: Chinese–Japanese NMT experiment results on ASPCE-JC Corpus (50K).

		zh → ja		**ja → zh**	
		BLEU	**RIBES**	**BLEU**	**RIBES**
SMT	[Koehn et al., 2003]				
Juman + Stanford		35.00	78.65	26.90	79.36
FM-MDL		36.12	81.69	28.46	81.67
FM-MDL (shared vocab, shared embed)		36.22	82.05	28.56	81.99
NMT					
Juman + Stanford	[Luong et al., 2015a]	39.61	86.34	31.12	83.51
BPE w/o pre-seg (*baseline*)	[Sennrich et al., 2016]	41.14	86.57	32.14	83.99
BPE w pre-seg	[Sennrich et al., 2016]	**41.62**	**86.70**	**32.60**	**84.33**
SPM	[Wu et al., 2016]	**41.87**	**86.84**	32.28	84.28
Proposed					
M-MDL		41.17	86.38	32.21	84.31
FM-MDL		**42.02**†	**86.88**†	**32.84**†	**84.75**†
FM-MDL (shared vocab, shared embed)		**41.52**†	**86.64**†	**32.48**†	**84.55**†

and SPM [Wu et al., 2016]) using Table 5.7. All of these models belong to the unsupervised method. They all train a segmentation/tokenization model. However, there are sharp differences between these methods.

All of the above methods treat the NMT system as a 'black box' with the extension of direct training from raw sentences. Hence these methods fit all kinds of state-of-the-art NMT models. A common feature for BPE, WPM, and FM-MDL is that the size of the vocabulary is always predetermined, i.e., they all use a finite vocabulary. For East Asian languages, in fact, BPE is a simplified version of MDL without careful computation of the coding cost. WPM differs from MDL methods, and BPE in that WPM is unable to guarantee that a new generated word has a

Table 5.5: Chinese-Japanese NMT experiment results using bilingual segmentation on ASPEC-JC Corpus (20K).

	zh→ja		ja → zh		Time-zh	Time-ja
	BLEU	RIBES	BLEU	RIBES	(min)	(min)
`Juman/Stanford`	39.54	85.91	31.09	83.76	–	–
Monolingual						
BPE w/o pre-seg (*baseline*)	40.48	86.12	31.63	83.37	17.70	17.33
BPE w pre-seg	40.63	86.24	**32.24**	**84.13**	–	–
SPM	**41.21**	**86.66**	**32.23**	83.51	5.61	6.91
FM-MDL	**40.93**	**86.32**	**32.14**	**84.24**	**1.78**	**2.10**
Bilingual (shared vocab and shared embed)						
BPE w/o pre-seg (*baseline*)	40.59	**86.27**	31.86	83.50	31.86	
BPE w pre-seg	40.53	**86.26**	**31.97**	**83.94**	–	
SPM	**41.01**	**86.35**	**32.01**	83.61	11.42	
FM-MDL	**40.76**	**86.25**	**32.30**	**84.37**	**4.62**	

Table 5.6: Training times on ASPEC-JC Corpus.

Time (mins)	**zh**	**ja**
BPE w/o	39.96	41.95
SPM	13.51	19.47
FM-MDL	4.56	6.54

higher frequency than required. In fact, WPM is quite similar to FM-MDL, but it does not take the coding cost of the codebook (model) into consideration (see Column 3). SPM and WPM use the same objective function, but training/segmentation procedures are different. Both WPM and SPM aim to maximize the language model likelihood, which is the same as $\mathrm{DL}(\mathcal{D}|\Phi)$, while M-MDL and FM-MDL pursue the minimum $\mathrm{DL}(\mathcal{D}|\Phi)$ on account of the coding cost. In addition, BPE updates the codebook only according to sorted pair frequencies. BPE w/o pre-segmentation and FM-MDL start with the characters in the vocabulary, while SPM[16] uses Unicode lattice to find the most common pieces in a top-down way.

[16]SPM treats white space as an essential symbol.

Table 5.7: Comparison of segment methods for subword-based NMT.

	Training	**Segmentation**	**Objective function**
BPE	Greedy	Binary Merge	$\lvert\Phi\rvert$
WPM	Greedy	Binary Merge	$\mathrm{DL}(\mathcal{D}\lvert\Phi)$
SPM	Top-down EM	Viterbi	$\mathrm{DL}(\mathcal{D}\lvert\Phi)$
FM-MDL	Bottom-up EM	FMM	$\mathrm{DL}(\mathcal{D}\lvert\Phi) + \mathrm{DL}(\Phi)$

5.5 Summary of the Chapter

This chapter investigates the bidirectional latent variable of vocabulary for word segmentation. Word segmentation is used for the computation of the vocabulary of East Asian languages without word separators (Chinese or Japanese). Conventional supervised word segmenters massively produce rare words which MT systems cannot translate in different quantity for different segmenters. This results in lower and different translation scores in SMT or NMT. To tackle the OOV/rare word problem in European languages, it has been proposed to go beyond words and decompose into sub-words. To translate rare words in East Asian languages and reduce sensitivity to segmentation, this chapter offers a novel bilingual unsupervised sub-word segmentation method based on the principle of minimum description length (MDL). Both the vocabulary and the word embedding layer are shared between the encoder and decoder. Essentially, the contribution consists in the resulting bilingual segmenter. It learns a finite vocabulary of words common to the two languages, each with a minimum frequency.

To tackle the rare word problem in East Asian languages and reduce sensitivity to segmentation, this dissertation proposes a novel bilingual unsupervised sub-word segmentation method based on the principle of minimum description length (MDL). This method learns a single (F) finite-size vocabulary, where sub-words are common to the two languages, each with (M) minimal frequency. Besides, this enables to share word embedding layer between the encoder and the decoder in seq2seq NMT.

When compared with state-of-the-art word segmenters, Juman for Japanese and Stanford Segmenter for Chinese, the proposed method (a) leads to statistically significant improvements in translation accuracy (+1.0 BLEU, p-value < 0.01) (b)

Times cannot be compared as Juman and Stanford Segmenter are frozen models, but (c) the vocabulary size is largely reduced (20 times smaller) as well as the size of the word embedding layer (50% smaller than in a monolingual setting) in both SMT and NMT experiments in Japanese-Chinese and Chinese-Japanese. When comparing with two other sub-word models for NMT sentence piece model [Wu et al., 2016] and byte pair encoding [Sennrich et al., 2016], the proposed method (a) leads to comparable or above translation accuracy in both monolingual and bilingual cases while (b) being 3 to 8 times as fast, (c) for the same vocabulary sizes.

Chapter 6

Contributions and Conclusions

The following key ideas are essential in the thesis:

1. **Translation divergences** should be given more attention.

2. Sharing **cross-linguistic equivalents** is an efficient way to accelerate the search for accurate translations while reducing memory cost.

3. **Bidirectional latent variable models** are naturally suitable for modeling machine translation.

This final chapter reviews the accomplishments of the dissertation. As discussed in Chapter 1, directly modeling translation in statistical or neural models is impractical due to the existence of translation divergences. To deal with these translation divergences, we enhanced the model architecture with bidirectional latent variables. Several crucial problems in machine translation such as discontiguity of word alignments, long-distance reordering and out-of-vocabulary words have been investigated by using these bidirectional latent variable models. Specifically, these models explicitly solve translation divergences at different levels, while utilizing shareable vocabulary, shareable word co-occurrences, or shareable syntactic structures. We evaluated our bidirectional models only considering the final machine translation quality. Experimental results demonstrate that bidirectional latent variables are of great benefit to the learning step in machine translation while being capable of reducing the model size. Bidirectional models not only yield state-of-the-art perfor-

mance but also are consistently more efficient for the translation of distant language pairs, like English–Japanese or Chinese–Japanese.

For the reader to have a general idea of machine translation, Chapter 1 walked the reader through the literature in machine translation and provided an overlook of latent variables in machine translation. We highlighted the drawbacks of the existing approaches for distant language pairs. The background of the research, e.g., basic notions and concepts in statistical machine translation and neural machine translation together with the details of some advanced MT systems, e.g., phrase-based SMT, syntax-based SMT and seq2seq NMT, were given in Chapter 2.

Our work focused on improving current MT approaches. They were presented in the following chapters. We investigated the use of bidirectional latent variables: symmetric phrasal alignments, BTG derivations which are bilingual by construction, and shared vocabulary in word segmentation sequentially.

In Chapter 3, we explained why phrasal alignment should be a bidirectional latent variable in phrase-based SMT. Our approach differs from conventional approaches relying on higher IBM models (3∼5). Those methods capture different types of alignments either using higher IBM models (IBM 3∼5), like `GIZA++` [Och and Ney, 2003] which are difficult to compute and require long training times, or lower IBM models (IBM 1∼2), like `fast_align` [Dyer et al., 2013] which are insufficient for long-distance alignments of distant language pairs. Our contributions constitute better estimation of translation probabilities using Variational Bayes [Riley and Gildea, 2012] for hierarchical sub-sentential alignment (HSSA) and improved HSSA using beam search, and consequently more accurate word alignments. The resulting aligner enabled us to create symmetrized alignments and to produce contiguous word alignments directly. In the future, better ways to build soft matrices using neural-based attention models may be explored. The possibility of using unsupervised bilingual word embedding to initialize those soft matrices is also a possible future research path.

In Chapter 4, we utilized BTG parse trees, another bidirectional latent variable, to build syntax-based SMT systems. The proposed method allows dispensing with parsers, resulting in a method which is insensitive to long-distance reorderings.

BTG parse trees can be used to reorder words in sentences, either before the standard phrase-based SMT pipeline (preordering for phrase-based SMT) [Neubig et al., 2012a, Nakagawa, 2015], or during translation (BTG-based decoding) [Xiong et al., 2008]. We improved the training algorithm of [Nakagawa, 2015] using bootstrap aggregating with several learning techniques (mini-batch, distributed, iterative distributed and k-best list). Our parallel training methods are capable of dealing with alignment errors that exist in the training examples. The conducted experiments demonstrated that our proposal is more effective and efficient when trained on automatically aligned data sets. To our best knowledge, this is the first research on parameter mixing techniques for latent structured inference. We also presented a latent BTG-based SMT method. This shows the possibility of reordering dispensing the target features. The main contribution in the chapter results in more efficient BTG-based SMT methods.

Chapter 5 presented the exploration of bidirectional latent variable of the vocabulary in word segmentation. Word segmentation is a necessary pre-processing step for many tasks of NLP for East Asian languages. For both SMT and NMT models, the quality of translations heavily relies on the quality of segments. This dissertation tackled the rare word problem in conventional approaches by proposing a novel unsupervised bilingual segmentation method based on the principle of MDL. It allows that not only the vocabulary is shared in SMT translation models but also the word embedding layer is shared between the encoder and decoder in NMT. Essentially, the contribution consists in better domain-adaptable word segmentation for machine translation. The resulting segmenter learns a finite vocabulary (F) of sub-words, common to the two languages, each with a minimal frequency (M). The use of the FM-MDL segmenter leads to statistically significant improvements on Chinese-Japanese when compared with Juman and Stanford Segmenter in both SMT and NMT experiments. Thus, this method is more efficient on account of time and memory cost. As for future work, we expect more experiments on other languages that use the Latin alphabet besides East Asian languages to verify the impact of MDL-based segmentation. We will also explore the performance of subword models using other NMT models, e.g. the CNN [Gehring et al., 2017] and

Transformer [Vaswani et al., 2017] models.

The work reported here is an important step towards more advanced machine translation systems. It constitutes a complete framework of bidirectional latent variables for machine translation. The dissertation presents a systematic solution to the translation divergence problem. More future work on recurrent latent variable model [Zhang et al., 2016a,Su et al., 2018] for NMT are expected. The joint learning of syntactic structures and translations using multi-task learning techniques are also attractive and desirable.

List of Publications

- **Journals**

  1. Wang, H. and Lepage, Y. (2018a). Improved BTG-based preordering for SMT via parallel parameter averaging: An empirical study. 自然言語処理 *(Journal of Natural Language Processing)*, 25(5):487–510

  2. Wang, H. and Lepage, Y. (2017b). Hierarchical sub-sentential alignment with IBM models for statistical phrase-based machine translation. 自然言語処理 *(Journal of Natural Language Processing)*, 24(4):619–646

  3. Yang, W., Wang, H., and Lepage, Y. (2014). Deduction of translation relations between new short sentences in Chinese and Japanese using analogical associations. *International Journal of Advanced Intelligence (IJAI)*, 6(1):13–34

- **International Conferences with Reviewing Committee**

  1. Wang, H. and Lepage, Y. (2017a). BTG-based machine translation with simple reordering model using structured perceptron. In *Proceedings of the 31th Pacific Asia Conference on Language, Information and Computation (PACLIC 31)*, pages 114–123

  2. Shan, B., Wang, H., and Lepage, Y. (2017). Unsupervised bilingual segmentation using MDL for machine translation. In *Proceedings of the 31th Pacific Asia Conference on Language, Information and Computation (PACLIC 31)*, pages 89–96

  3. Chen, K., Wang, H., and Lepage, Y. (2017). A study of analogical relationships in n-gram embedding models. In *Proceedings of the 15th*

International Conference of the Pacific Association for Computational Linguistics (PACLING 2017), pages 56–65

4. Wang, H. and Lepage, Y. (2016b). Yet another symmetrical and real-time word alignment method: Hierarchical sub-sentential alignment using F-measure. In *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation (PACLIC 30)*, pages 143–152. (Best Paper Award)

5. Wang, H. and Lepage, Y. (2016a). Combining fast_align with hierarchical sub-sentential alignment for better word alignments. In *Proceedings of the 6th Workshop on Hybrid Approaches to Translation (Hytra 6), the 26th International Conference on Computational Linguistics (COLING 2016)*, pages 1–7

6. Zhang, Y., Wang, H., and Lepage, Y. (2016b). HSSA tree structures for BTG-based preordering in machine translation. In *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation (PACLIC 30)*, pages 123–132

7. Wang, H. and Lepage, Y. (2015). Analogy-based on-line reordering approach for machine translation. In *7th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC'15)*, pages 165–169

8. Wang, H., Lyu, L., and Lepage, Y. (2015). Translation of unseen bigrams by analogy using an SVM classifier. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation (PACLIC 29)*, pages 16–25

9. Wang, H., Yang, W., and Lepage, Y. (2014a). Improved Chinese-Japanese phrase-based MT quality using an extended quasi-parallel corpus. In *2014 International Conference on Progress in Informatics and Computing (PIC 2014)*, pages 6–10. IEEE

10. Yang, W., Wang, H., and Lepage, Y. (2013c). Using analogical associations to acquire chinese-japanese quasi-parallel sentences. In *Proceedings*

of the tenth symposium on natural language processing (SNLP 2013), pages 86–93

11. Yang, W., Wang, H., and Lepage, Y. (2015b). Automatic acquisition of rewriting models for the generation of quasi-parallel corpus. In *6th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC'13)*, pages 409–413

- **Conferences without Reviewing Committee**

1. Wang, H. and Lepage, Y. (2018b). Unsupervised word segmentation using minimum description length for neural machine translation. In *Proceedings of the 24th annual meetings of the Association for Natural Language Processing, Japan (*言語処理学会第*24*回　次大会 発表論文集*)*, pages 1080–1083

2. Yang, B., Wang, H., and Lepage, Y. (2015a). Faster development of statistical machine translation systems with sampling-based alignment and hierarchical sub-sentential alignment. In *Proceedings of the 21th annual meetings of the Association for Natural Language Processing, Japan (*言語処理学会第*21*回　次大会 発表論文集*)*, pages 732–735

3. Wang, H., Yang, W., and Lepage, Y. (2014b). Sentence generation by analogy: Towards the construction of a quasi-parallel corpus for chinese-japanese. In *Proceedings of the 20th annual meetings of the Association for Natural Language Processing, Japan (*言語処理学会第*20*回　次大会 発表論文集*)*, pages 900–903

4. Lepage, Y., Yang, W., and Wang, H. (2014). Generating quasi-parallel training data to improve Chinese-Japanese machine translation. *The 14th China-Japan Natural Language Processing Collaboration Promotion Conference (CJNLP2014)*

5. Yang, W., Wang, H., and Lepage, Y. (2013a). Using analogical association to acquire Chinese-Japanese quasi-parallel sentences. *7th International collaboration Symposium on Information, Production and Systems*

6. Yang, W., Wang, H., and Lepage, Y. (2013b). Using analogical association to acquire Chinese-Japanese quasi-parallel sentences. *International Workshop on Machine Vision for Industrial Innovation*

Bibliography

[Argamon et al., 2004] Argamon, S., Akiva, N., Amir, A., and Kapah, O. (2004). Efficient unsupervised recursive word segmentation using minimum description length. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 1058. Association for Computational Linguistics.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bahdanau et al., 2015] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *6th International Conference on Learning Representations*, abs/1409.0473.

[Barron et al., 1998] Barron, A., Rissanen, J., and Yu, B. (1998). The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760.

[Bentivogli et al., 2016] Bentivogli, L., Bisazza, A., Cettolo, M., and Federico, M. (2016). Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267, Austin, Texas. Association for Computational Linguistics.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

[Broderick et al., 2013] Broderick, T., Boyd, N., Wibisono, A., Wilson, A. C., and Jordan, M. I. (2013). Streaming variational bayes. In *Advances in Neural Information Processing Systems*, pages 1727–1735.

[Brown et al., 1988] Brown, P., Cocke, J., Pietra, S. D., Pietra, V. D., Jelinek, F., Mercer, R., and Roossin, P. (1988). A statistical approach to language translation. In *Proceedings of the 12th Conference on Computational Linguistics*, volume 1, pages 71–76. Association for Computational Linguistics.

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

[Brown et al., 1992] Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

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

[Chang et al., 2008] Chang, P.-C., Galley, M., and Manning, C. D. (2008). Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the third Workshop on Statistical Machine Translation*, pages 224–232. Association for Computational Linguistics.

[Charniak, 1997] Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. *AAAI/IAAI*, 2005(598-603):18.

[Chen et al., 2017] Chen, K., Wang, H., and Lepage, Y. (2017). A study of analogical relationships in n-gram embedding models. In *Proceedings of the 15th International Conference of the Pacific Association for Computational Linguistics (PACLING 2017)*, pages 56–65.

[Chen and Goodman, 1996] Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th*

annual meeting on Association for Computational Linguistics, pages 310–318. Association for Computational Linguistics.

[Cherry and Foster, 2012] Cherry, C. and Foster, G. (2012). Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436. Association for Computational Linguistics.

[Cherry and Lin, 2007] Cherry, C. and Lin, D. (2007). Inversion transduction grammar for joint phrasal translation modeling. In *Proceedings of the NAACL-HLT 2007/AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 17–24. Association for Computational Linguistics.

[Cherry and Suzuki, 2009] Cherry, C. and Suzuki, H. (2009). Discriminative substring decoding for transliteration. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1066–1075. Association for Computational Linguistics.

[Chiang, 2005] Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics.

[Chiang, 2007] Chiang, D. (2007). Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

[Chiang et al., 2009] Chiang, D., Knight, K., and Wang, W. (2009). 11,001 new features for statistical machine translation. In *Proceedings of human language technologies: The 2009 annual conference of the north american chapter of the association for computational linguistics*, pages 218–226. Association for Computational Linguistics.

[Chitnis and DeNero, 2015] Chitnis, R. and DeNero, J. (2015). Variable-length word encodings for neural translation models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2088–2093.

[Cho et al., 2014] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *The 2014 Conference on Empirical Methods on Natural Language Processing*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

[Collins, 2002] Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

[Collins et al., 2005] Collins, M., Koehn, P., and Kučerová, I. (2005). Clause restructuring for statistical machine translation. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 531–540. Association for Computational Linguistics.

[Collins and Roark, 2004] Collins, M. and Roark, B. (2004). Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111. Association for Computational Linguistics.

[Crammer et al., 2006] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. In *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

[DeNero and Uszkoreit, 2011] DeNero, J. and Uszkoreit, J. (2011). Inducing sentence structure from parallel corpora for reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 193–203. Association for Computational Linguistics.

[Dietterich, 2000] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, pages 1–15. Springer.

[Ding et al., 2015] Ding, C., Utiyama, M., and Sumita, E. (2015). Improving fast_align by reordering. In *Proceedings of the on Empirical Methods on Natural Language Processing*. Association for Computational Linguistics.

[Dorr, 1994] Dorr, B. J. (1994). Machine translation divergences: A formal description and proposed solution. *Computational Linguistics*, 20(4):597–633.

[Dyer et al., 2013] Dyer, C., Chahuneau, V., and Smith, N. A. (2013). A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings for the NAACL*. Association for Computational Linguistics.

[Džeroski and Ženko, 2004] Džeroski, S. and Ženko, B. (2004). Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54(3):255–273.

[Eddy and Durbin, 1994] Eddy, S. R. and Durbin, R. (1994). Rna sequence analysis using covariance models. *Nucleic acids research*, 22(11):2079–2088.

[Fraser and Marcu, 2007] Fraser, A. and Marcu, D. (2007). Measuring word alignment quality for statistical machine translation. *Computational Linguistics*, 33(3):293–303.

[Freund and Schapire, 1999] Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296.

[Galley et al., 2004] Galley, M., Hopkins, M., Knight, K., and Marcu, D. (2004). What's in a translation rule? In *Proceedings of the HLT/NAACL-04*.

[Galley and Manning, 2008] Galley, M. and Manning, C. D. (2008). A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 848–856. Association for Computational Linguistics.

[Ganchev et al., 2008] Ganchev, K., Graca, J. V., and Taskar, B. (2008). Better alignments= better translations? In *ACL-08: HLT*, page 986. Association for Computational Linguistics.

[Gehring et al., 2017] Gehring, J., Auli, M., Grangier, D., and Dauphin, Y. (2017). A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 123–135, Vancouver, Canada. Association for Computational Linguistics.

[Genzel, 2010] Genzel, D. (2010). Automatically learning source-side reordering rules for large scale machine translation. In *Proceedings of the 23rd international conference on computational linguistics*, pages 376–384. Association for Computational Linguistics.

[González-Rubio and Casacuberta, 2014] González-Rubio, J. and Casacuberta, F. (2014). Inference of phrase-based translation models via minimum description length. In *EACL, volume 2: Short Papers*, pages 90–94.

[Goodman, 1999] Goodman, J. (1999). Semiring parsing. *Computational Linguistics*, 25(4):573–605.

[Goto et al., 2015] Goto, I., Utiyama, M., Sumita, E., and Kurohashi, S. (2015). Preordering using a target-language parser via cross-language syntactic projection for statistical machine translation. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 14(3):13.

[Grünwald, 1995] Grünwald, P. (1995). A minimum description length approach to grammar inference. In *International Joint Conference on Artificial Intelligence*, pages 203–216. Springer.

[Grünwald, 2007] Grünwald, P. D. (2007). *The minimum description length principle*. MIT press.

[Haghighi et al., 2009] Haghighi, A., Blitzer, J., DeNero, J., and Klein, D. (2009). Better word alignments with supervised itg models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 923–931. Association for Computational Linguistics.

[Hansen and Yu, 2001] Hansen, M. H. and Yu, B. (2001). Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774.

[Heafield, 2011] Heafield, K. (2011). Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation (WMT 2011)*, pages 187–197. Association for Computational Linguistics.

[Hewlett and Cohen, 2011] Hewlett, D. and Cohen, P. (2011). Fully unsupervised word segmentation with bve and mdl. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 540–545. Association for Computational Linguistics.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Huang and Zhao, 2007] Huang, C. and Zhao, H. (2007). Chinese word segmentation: a decade review. *Journal of Chinese Information Processing*, 21(3):8–20.

[Isozaki et al., 2010] Isozaki, H., Hirao, T., Duh, K., Sudoh, K., and Tsukada, H. (2010). Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952. Association for Computational Linguistics.

[Jonyer et al., 2004] Jonyer, I., Holder, L. B., and Cook, D. J. (2004). MDL-based context-free graph grammar induction and applications. *International Journal on Artificial Intelligence Tools*, 13(01):65–79.

[Kärkkäinen and Sanders, 2003] Kärkkäinen, J. and Sanders, P. (2003). Simple linear work suffix array construction. In *ICALP*, volume 2719, pages 943–955. Springer.

[Kendall, 1938] Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

[Klasner and Simon, 1995] Klasner, N. and Simon, H. U. (1995). From noise-free to noise-tolerant and from on-line to batch learning. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 250–257. ACM.

[Kneser and Ney, 1995] Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

[Koehn, 2004] Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395. Citeseer.

[Koehn et al., 2005] Koehn, P., Axelrod, A., Birch, A., Callison-Burch, C., Osborne, M., Talbot, D., and White, M. (2005). Edinburgh system description for the 2005 iwslt speech translation evaluation. In *IWSLT*, pages 68–75.

[Koehn et al., 2007] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

[Koehn et al., 2003] Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

[Lardilleux et al., 2012] Lardilleux, A., Yvon, F., and Lepage, Y. (2012). Hierarchical sub-sentential alignment with anymalign. In *Proceedings of the 16th Annual Conference of the European Association for Machine Translation*, pages 279–286.

[Lepage et al., 2014] Lepage, Y., Yang, W., and Wang, H. (2014). Generating quasi-parallel training data to improve Chinese-Japanese machine translation. *The 14th China-Japan Natural Language Processing Collaboration Promotion Conference (CJNLP2014).*

[Lerner and Petrov, 2013] Lerner, U. and Petrov, S. (2013). Source-side classifier preordering for machine translation. In *EMNLP*, pages 513–523.

[Li et al., 2014] Li, M., Zhang, T., Chen, Y., and Smola, A. J. (2014). Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670. ACM.

[Li et al., 2011] Li, M., Zong, C., and Ng, H. T. (2011). Automatic evaluation of chinese translation output: word-level or character-level? In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 159–164. Association for Computational Linguistics.

[Li et al., 2012] Li, P., Yang, L., and Sun, M. (2012). A beam search algorithm for itg word alignment. In *Proceedings of the 24th International Conference on Computational Linguistics*, page 673.

[Li et al., 2016] Li, X., Zhang, J., and Zong, C. (2016). Towards zero unknown word in neural machine translation. In *International Joint Conference on Artificial Intelligence*, pages 2852–2858.

[Liang, 2005] Liang, P. (2005). *Semi-supervised learning for natural language*. PhD thesis, Citeseer.

[Liang et al., 2006] Liang, P., Taskar, B., and Klein, D. (2006). Alignment by agreement. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

[Liu et al., 2006] Liu, Y., Liu, Q., and Lin, S. (2006). 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*, pages 609–616. Association for Computational Linguistics.

[Liu et al., 2010] Liu, Y., Liu, Q., and Lin, S. (2010). Discriminative word alignment by linear modeling. *Computational Linguistics*, 36(3):303–339.

[Liu et al., 2009] Liu, Y., Xia, T., Xiao, X., and Liu, Q. (2009). Weighted alignment matrices for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, volume 2, pages 1017–1026. Association for Computational Linguistics.

[Luong et al., 2015a] Luong, T., Pham, H., and Manning, C. D. (2015a). Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

[Luong et al., 2015b] Luong, T., Sutskever, I., Le, Q., Vinyals, O., and Zaremba, W. (2015b). Addressing the rare word problem in neural machine translation. In *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

[Magistry and Sagot, 2013] Magistry, P. and Sagot, B. (2013). Can mdl improve unsupervised chinese word segmentation? In *Sixth International Joint Conference on Natural Language Processing: Sighan workshop*, page 2.

[Makhoul et al., 1999] Makhoul, J., Kubala, F., Schwartz, R., Weischedel, R., et al. (1999). Performance measures for information extraction. In *Proceedings of DARPA broadcast news workshop*, pages 249–252.

[Matusov et al., 2004] Matusov, E., Zens, R., and Ney, H. (2004). Symmetric word alignments for statistical machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 219. Association for Computational Linguistics.

[McDonald et al., 2010] McDonald, R., Hall, K., and Mann, G. (2010). Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Associa-*

tion for Computational Linguistics, pages 456–464. Association for Computational Linguistics.

[Mermer and Saraclar, 2011] Mermer, C. and Saraclar, M. (2011). Bayesian word alignment for statistical machine translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers*, volume 2, pages 182–187. Association for Computational Linguistics.

[Mihalcea and Pedersen, 2003] Mihalcea, R. and Pedersen, T. (2003). An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and using Parallel Texts: Data Driven Machine Translation and Beyond*, volume 3, pages 1–10. Association for Computational Linguistics.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *ICLR Workshop*.

[Mikolov et al., 2010] Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

[Mochihashi et al., 2009] Mochihashi, D., Yamada, T., and Ueda, N. (2009). Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 100–108. Association for Computational Linguistics.

[Moore, 2005] Moore, R. C. (2005). Association-based bilingual word alignment. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 1–8. Association for Computational Linguistics.

[Naftaly et al., 1997] Naftaly, U., Intrator, N., and Horn, D. (1997). Optimal ensemble averaging of neural networks. *Network: Computation in Neural Systems*, 8(3):283–296.

[Nakagawa, 2015] Nakagawa, T. (2015). Efficient top-down btg parsing for machine translation preordering. In *ACL 2015*, pages 208–218. Association for Computational Linguistics.

[Nakazawa et al., 2016a] Nakazawa, T., Mino, H., Ding, C., Goto, I., Neubig, G., Kurohashi, S., and Sumita, E. (2016a). Overview of the 3rd Workshop on Asian Translation. *Proc. WAT*.

[Nakazawa et al., 2016b] Nakazawa, T., Yaguchi, M., Uchimoto, K., Utiyama, M., Sumita, E., Kurohashi, S., and Isahara, H. (2016b). Aspec: Asian scientific paper excerpt corpus. In Chair), N. C. C., Choukri, K., Declerck, T., Grobelnik, M., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the 2016 International Conference on Language Resources and Evaluation*, pages 2204–2208, Portorož, Slovenia. European Language Resources Association (ELRA).

[Neubig et al., 2011a] Neubig, G., Nakata, Y., and Mori, S. (2011a). Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 529–533. Association for Computational Linguistics.

[Neubig et al., 2012a] Neubig, G., Watanabe, T., and Mori, S. (2012a). Inducing a discriminative parser to optimize machine translation reordering. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 843–853. Association for Computational Linguistics.

[Neubig et al., 2012b] Neubig, G., Watanabe, T., Mori, S., and Kawahara, T. (2012b). Machine translation without words through substring alignment. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 165–174. Association for Computational Linguistics.

[Neubig et al., 2011b] Neubig, G., Watanabe, T., Sumita, E., Mori, S., and Kawahara, T. (2011b). An unsupervised model for joint phrase alignment and extrac-

tion. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 632–641, Portland, Oregon, USA. Association for Computational Linguistics.

[Noeman, 2009] Noeman, S. (2009). Language independent transliteration system using phrase based smt approach on substrings. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration*, pages 112–115. Association for Computational Linguistics.

[Och, 2003] Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL 2003)*, volume 1, pages 160–167. Association for Computational Linguistics.

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

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

[Pham et al., 2013] Pham, V., Kermorvant, C., and Louradour, J. (2013). Dropout improves recurrent neural networks for handwriting recognition. *CoRR*, abs/1312.4569.

[Riesa and Marcu, 2010] Riesa, J. and Marcu, D. (2010). Hierarchical search for word alignment. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 157–166. Association for Computational Linguistics.

[Riley and Gildea, 2012] Riley, D. and Gildea, D. (2012). Improving the ibm alignment models using variational bayes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers*, volume 2, pages 306–310. Association for Computational Linguistics.

[Rissanen, 1978] Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5):465–471.

[Rissanen, 1986] Rissanen, J. (1986). Stochastic complexity and modeling. *The annals of statistics*, pages 1080–1100.

[Rokach, 2010] Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39.

[Saers and Wu, 2013] Saers, M. and Wu, D. (2013). Bayesian induction of bracketing inversion transduction grammars. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1158–1166.

[Savitch, 1982] Savitch, W. J. (1982). *Abstract machines and grammars*. Little Brown.

[Schuster and Nakajima, 2012] Schuster, M. and Nakajima, K. (2012). Japanese and korean voice search. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 5149–5152. IEEE.

[Sennrich et al., 2016] Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. pages 1715–1725.

[Shan et al., 2017] Shan, B., Wang, H., and Lepage, Y. (2017). Unsupervised bilingual segmentation using MDL for machine translation. In *Proceedings of the 31th Pacific Asia Conference on Language, Information and Computation (PACLIC 31)*, pages 89–96.

[Sheridan, 1955] Sheridan, P. (1955). Research in language translation on the ibm type 701. *IBM Technical Newsletter*, 9:5–24.

[Shi and Malik, 2000] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine intelligence*, 22(8):888–905.

[Shieber et al., 1995] Shieber, S. M., Schabes, Y., and Pereira, F. C. (1995). Principles and implementation of deductive parsing. *The Journal of logic programming*, 24(1):3–36.

[Smith and Johnson, 2007] Smith, N. A. and Johnson, M. (2007). Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics*, 33(4):477–491.

[Sproat and Emerson, 2003] Sproat, R. and Emerson, T. (2003). The first international Chinese word segmentation bakeoff. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*, pages 133–143. Association for Computational Linguistics.

[Sproat et al., 1996] Sproat, R., Gale, W., Shih, C., and Chang, N. (1996). A stochastic finite-state word-segmentation algorithm for chinese. *Computational linguistics*, 22(3):377–404.

[Su et al., 2018] Su, J., Wu, S., Xiong, D., Lu, Y., Han, X., and Zhang, B. (2018). Variational recurrent neural machine translation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5488–5495.

[Sudoh et al., 2010] Sudoh, K., Duh, K., Tsukada, H., Hirao, T., and Nagata, M. (2010). Divide and translate: improving long distance reordering in statistical machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 418–427. Association for Computational Linguistics.

[Sun et al., 2009] Sun, X., Matsuzaki, T., Okanohara, D., and Tsujii, J. (2009). Latent variable perceptron algorithm for structured classification. In *IJCAI*, volume 9, pages 1236–1242.

[Sutskever et al., 2014a] Sutskever, I., Vinyals, O., and Le, Q. V. (2014a). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[Sutskever et al., 2014b] Sutskever, I., Vinyals, O., and Le, Q. V. (2014b). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

[Talbot et al., 2011] Talbot, D., Kazawa, H., Ichikawa, H., Katz-Brown, J., Seno, M., and Och, F. J. (2011). A lightweight evaluation framework for machine translation reordering. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 12–21. Association for Computational Linguistics.

[Tillmann, 2004] Tillmann, C. (2004). A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 101–104. Association for Computational Linguistics.

[Tillmann and Ney, 2003] Tillmann, C. and Ney, H. (2003). Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational linguistics*, 29(1):97–133.

[Toutanova et al., 2003] Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

[Tyers et al., 2010] Tyers, R., Anderson, K., et al. (2010). *Disarray in world food markets: a quantitative assessment.* Cambridge University Press.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.

[Vilar et al., 2007] Vilar, D., Peter, J.-T., and Ney, H. (2007). Can we translate letters? In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 33–39. Association for Computational Linguistics.

[Vogel et al., 1996] Vogel, S., Ney, H., and Tillmann, C. (1996). Hmm-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics*, volume 2, pages 836–841. Association for Computational Linguistics.

[Wang and Lepage, 2015] Wang, H. and Lepage, Y. (2015). Analogy-based on-line reordering approach for machine translation. In *7th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC'15)*, pages 165–169.

[Wang and Lepage, 2016a] Wang, H. and Lepage, Y. (2016a). Combining fast_align with hierarchical sub-sentential alignment for better word alignments. In *Proceedings of the 6th Workshop on Hybrid Approaches to Translation (Hytra 6), the 26th International Conference on Computational Linguistics (COLING 2016)*, pages 1–7.

[Wang and Lepage, 2016b] Wang, H. and Lepage, Y. (2016b). Yet another symmetrical and real-time word alignment method: Hierarchical sub-sentential alignment using F-measure. In *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation (PACLIC 30)*, pages 143–152. (Best Paper Award).

[Wang and Lepage, 2016c] Wang, H. and Lepage, Y. (2016c). Yet another symmetrical and real-time word alignment method: Hierarchical sub-sentential alignment using f-measure. In *The 30th Pacific Asia Conference on Language, Information and Computation*, pages 143–152. PALCIC.

[Wang and Lepage, 2017a] Wang, H. and Lepage, Y. (2017a). BTG-based machine translation with simple reordering model using structured perceptron. In *Proceedings of the 31th Pacific Asia Conference on Language, Information and Computation (PACLIC 31)*, pages 114–123.

[Wang and Lepage, 2017b] Wang, H. and Lepage, Y. (2017b). Hierarchical sub-sentential alignment with IBM models for statistical phrase-based machine translation. 自然言語処理 *(Journal of Natural Language Processing)*, 24(4):619–646.

[Wang and Lepage, 2018a] Wang, H. and Lepage, Y. (2018a). Improved BTG-based preordering for SMT via parallel parameter averaging: An empirical study. 自然言語処理 *(Journal of Natural Language Processing)*, 25(5):487–510.

[Wang and Lepage, 2018b] Wang, H. and Lepage, Y. (2018b). Unsupervised word segmentation using minimum description length for neural machine translation. In *Proceedings of the 24th annual meetings of the Association for Natural Language Processing, Japan (*言語処理学会第*24*回　次大会 発表論文集*)*, pages 1080–1083.

[Wang et al., 2015] Wang, H., Lyu, L., and Lepage, Y. (2015). Translation of unseen bigrams by analogy using an SVM classifier. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation (PACLIC 29)*, pages 16–25.

[Wang et al., 2014a] Wang, H., Yang, W., and Lepage, Y. (2014a). Improved Chinese-Japanese phrase-based MT quality using an extended quasi-parallel corpus. In *2014 International Conference on Progress in Informatics and Computing (PIC 2014)*, pages 6–10. IEEE.

[Wang et al., 2014b] Wang, H., Yang, W., and Lepage, Y. (2014b). Sentence generation by analogy: Towards the construction of a quasi-parallel corpus for chinese-japanese. In *Proceedings of the 20th annual meetings of the Association for Natural Language Processing, Japan (*言語処理学会第*20*回　次大会 発表論文集*)*, pages 900–903.

[Wang et al., 2016] Wang, R., Zhao, H., Lu, B.-L., Utiyama, M., and Sumita, E. (2016). Connecting phrase based statistical machine translation adaptation. In *COLING*.

[Wang et al., 2007] Wang, W., Knight, K., and Marcu, D. (2007). Binarizing syntax trees to improve syntax-based machine translation accuracy. In *EMNLP-CoNLL*, pages 746–754. Citeseer.

[Watanabe et al., 2007] Watanabe, T., Suzuki, J., Tsukada, H., and Isozaki, H. (2007). Online large-margin training for statistical machine translation. In *In Proc. of EMNLP*. Citeseer.

[Wu, 1995] Wu, D. (1995). Stochastic inversion transduction grammars, with application to segmentation, bracketing, and alignment of parallel corpora. In *IJCAI*, volume 95, pages 1328–1335. Citeseer.

[Wu, 1997] Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

[Wu and Wang, 2007] Wu, H. and Wang, H. (2007). Comparative study of word alignment heuristics and phrase-based smt. In *Proceedings of the MT Summit XI*.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

[Xia and McCord, 2004] Xia, F. and McCord, M. (2004). Improving a statistical mt system with automatically learned rewrite patterns. In *Proceedings of the 20th international conference on Computational Linguistics*, page 508. Association for Computational Linguistics.

[Xiong et al., 2008] Xiong, D., Zhang, M., Aw, A., and Li, H. (2008). A linguistically annotated reordering model for btg-based statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 149–152. Association for Computational Linguistics.

[Xiong et al., 2010] Xiong, D., Zhang, M., and Li, H. (2010). Learning translation boundaries for phrase-based decoding. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 136–144. Association for Computational Linguistics.

[Xue et al., 2003] Xue, N. et al. (2003). Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

[Yamada and Knight, 2001] Yamada, K. and Knight, K. (2001). A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics.

[Yang et al., 2015a] Yang, B., Wang, H., and Lepage, Y. (2015a). Faster development of statistical machine translation systems with sampling-based alignment and hierarchical sub-sentential alignment. In *Proceedings of the 21th annual meetings of the Association for Natural Language Processing, Japan (言語処理学会第21回　次大会 発表論文集)*, pages 732–735.

[Yang et al., 2013a] Yang, W., Wang, H., and Lepage, Y. (2013a). Using analogical association to acquire Chinese-Japanese quasi-parallel sentences. *7th International collaboration Symposium on Information, Production and Systems*.

[Yang et al., 2013b] Yang, W., Wang, H., and Lepage, Y. (2013b). Using analogical association to acquire Chinese-Japanese quasi-parallel sentences. *International Workshop on Machine Vision for Industrial Innovation*.

[Yang et al., 2013c] Yang, W., Wang, H., and Lepage, Y. (2013c). Using analogical associations to acquire chinese-japanese quasi-parallel sentences. In *Proceedings of the tenth symposium on natural language processing (SNLP 2013)*, pages 86–93.

[Yang et al., 2014] Yang, W., Wang, H., and Lepage, Y. (2014). Deduction of translation relations between new short sentences in Chinese and Japanese using analogical associations. *International Journal of Advanced Intelligence (IJAI)*, 6(1):13–34.

[Yang et al., 2015b] Yang, W., Wang, H., and Lepage, Y. (2015b). Automatic acquisition of rewriting models for the generation of quasi-parallel corpus. In *6th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC'13)*, pages 409–413.

[Zaidan, 2009] Zaidan, O. (2009). Z-mert: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

[Zens and Ney, 2006] Zens, R. and Ney, H. (2006). Discriminative reordering models for statistical machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 55–63. Association for Computational Linguistics.

[Zens et al., 2004] Zens, R., Ney, H., Watanabe, T., and Sumita, E. (2004). Reordering constraints for phrase-based statistical machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 205. Association for Computational Linguistics.

[Zha et al., 2001] Zha, H., He, X., Ding, C., Simon, H., and Gu, M. (2001). Bipartite graph partitioning and data clustering. In *Proceedings of the 10th International Conference on Information and knowledge Management*, pages 25–32. ACM.

[Zhang et al., 2016a] Zhang, B., Xiong, D., su, j., Duan, H., and Zhang, M. (2016a). Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530, Austin, Texas. Association for Computational Linguistics.

[Zhang and Gildea, 2005] Zhang, H. and Gildea, D. (2005). Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 475–482. Association for Computational Linguistics.

[Zhang et al., 2007] Zhang, M., Jiang, H., Aw, A., Sun, J., Li, S., and Tan, C. L. (2007). A tree-to-tree alignment-based model for statistical machine translation. In *MT-Summit-07*, pages 535–542.

[Zhang and Li, 2009] Zhang, M. and Li, H. (2009). Tree kernel-based svm with structured syntactic knowledge for btg-based phrase reordering. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 698–707. Association for Computational Linguistics.

[Zhang et al., 2016b] Zhang, Y., Wang, H., and Lepage, Y. (2016b). HSSA tree structures for BTG-based preordering in machine translation. In *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation (PACLIC 30)*, pages 123–132.

[Zhao et al., 2013] Zhao, H., Utiyama, M., Sumita, E., and Lu, B.-L. (2013). *An Empirical Study on Word Segmentation for Chinese Machine Translation*, pages 248–263. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Zhikov et al., 2013] Zhikov, V., Takamura, H., and Okumura, M. (2013). An efficient algorithm for unsupervised word segmentation with branching entropy and mdl. *Information and Media Technologies*, 8(2):514–527.

[Zhou et al., 2016] Zhou, J., Cao, Y., Wang, X., Li, P., and Xu, W. (2016). Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4:371–383.

Appendices

Appendix A

Basic and Auxiliary Formulation

A.1 Reformulation of Ncut to F-measure

When splitting into two subparts a sentence pair, also called matrix of soft alignments, the hierarchical sub-sentential alignment method tries to minimize the following criterion $Ncut$:

$$Ncut(XY) = \frac{cut(XY)}{cut(XY) + 2 \times W(X,Y)} + \frac{cut(\bar{X}\bar{Y})}{cut(\bar{X}\bar{Y}) + 2 \times W(\bar{X},\bar{Y})}$$

where cut is defined as:

$$cut(XY) = W(X,\overline{Y}) + W(\overline{X},Y) \tag{A.1.1}$$

A sentence pair matrix can be interpreted as a contingency matrix for X relatively to Y. Actually, minimizing $Ncut(XY)$ is equivalent to maximizing the arithmetic mean of the F-measures of X relatively to Y and $\bar{X}$ relatively to $\bar{Y}$. The proof is given below.

In a contingency matrix, precision (P) captures substitution and insertion errors while recall (R) captures substitution and deletion errors. These two measures have been used regularly to assess the performance of information retrieval and information extraction systems. The F-measure is defined as the harmonic mean of precision and recall [Makhoul et al., 1999].

$$\frac{1}{F(X,Y)} = \frac{1}{2} \times (\frac{1}{P(X,Y)} + \frac{1}{R(X,Y)}) \quad \text{(A.1.2)}$$

To interpret the sentence pair matrix as contingency matrices, it suffices to read translation strengths as reflecting the contribution of a source word to a target word and reciprocally (see Figure 3.2). With this interpretation, the precision (P) and the recall (R) for two sub-parts of the source and the target sentences can easily be expressed using the sum of all the translation strengths inside a block (Equations A.1.3 and A.1.4, also see Figure 3.2). These two measures can thus be defined by Equations A.1.3 and A.1.4.

$$P(X,Y) = \frac{W(X,Y)}{W(X,Y) + W(X,\bar{Y})} \quad \text{(A.1.3)}$$

$$R(X,Y) = \frac{W(X,Y)}{W(X,Y) + W(\bar{X},Y)} \quad \text{(A.1.4)}$$

Now, it suffices to replace precision and recall by their values in terms of *cut* to derive the following formula.

$$\frac{1}{F(X,Y)} = \frac{1}{2}\left(\frac{1}{R(X,Y)} + \frac{1}{P(X,Y)}\right) \quad \text{(A.1.5)}$$

$$= \frac{1}{2}\left(\frac{W(X,Y) + W(\bar{X},Y)}{W(X,Y)} + \frac{W(X,Y) + W(X,\bar{Y})}{W(X,Y)}\right) \quad \text{(A.1.6)}$$

$$= \frac{2 \times W(X,Y) + W(\bar{X},Y) + W(X,\bar{Y})}{2 \times W(X,Y)} \quad \text{(A.1.7)}$$

$$= \frac{2 \times W(X,Y) + cut(X,Y)}{2 \times W(X,Y)} = \frac{cut(X,Y) + 2 \times W(X,Y)}{2 \times W(X,Y)} \quad \text{(A.1.8)}$$

By taking the reciprocal:

$$F(X,Y) = \frac{2 \times W(X,Y)}{cut(X,Y) + 2 \times W(X,Y)} = 1 - \frac{cut(X,Y)}{cut(X,Y) + 2 \times W(X,Y)} \quad \text{(A.1.9)}$$

Hence we have:

$$\frac{cut(X,Y)}{cut(X,Y) + 2 \times W(X,Y)} = 1 - F(X,Y) \quad \text{(A.1.10)}$$

By using Equation A.1.10 twice in Equation A.1.1, for (X, Y) and $(\bar{X}, \bar{Y})$, we obtain a new definition of Ncut:

$$\begin{aligned} \text{Ncut}(X,Y) &= 1 - F(X,Y) + 1 - F(\bar{X},\bar{Y}) && \text{(A.1.11)} \\ &= 2 - \big(F(X,Y) + F(\bar{X},\bar{Y}) \big) && \text{(A.1.12)} \\ &= 2 \times \left(1 - \frac{F(X,Y) + F(\bar{X},\bar{Y})}{2} \right) && \text{(A.1.13)} \\ & && \text{(A.1.14)} \end{aligned}$$

An equivalent way of writing is:

$$\frac{Ncut(XY)}{2} = 1 - \frac{F(X,Y) + F(\bar{X},\bar{Y})}{2} \tag{A.1.15}$$

As there is a negative sign before the sum of F-measures, the above formula obviously shows that minimizing $\text{Ncut}(X, Y)$ is equivalent to maximizing the arithmetic mean of the F-measures of X relatively to Y, and $\bar{X}$ relatively to $\bar{Y}$. This in fact makes sense intuitively if we look for the best possible way for parts of the source and target sentences to correspond. These parts should cover one another in both directions as much as possible, that is to say, they should exhibit the best recall and precision at the same time.

To summarize, the interpretation of sentence pair matrix as contingency matrices between a source and a target sentences through the translation strengths between the words they contain, and the use of the definition of *cut* given in Equation A.1.1 leads to the relationship between F-measure and *cut*, expressed in Equation A.1.16,

$$F(X,Y) = 1 - \frac{cut(X,Y)}{cut(X,Y) + 2 \times W(X,Y)} \tag{A.1.16}$$

which in turn leads to an alternative definition of $Ncut$ (Equation A.1.15) that shows that trying to minimize it, is equivalent to trying to maximize the arithmetic mean of the F-measures of the two blocks under consideration.

Appendix B

Algorithms

Algorithm 1: HIERACHICAL SUB-SENTENTIAL ALIGNMENT USING BEAM SEACH

Input : sentence pairs $\mathcal{T} = \langle \mathbf{F}, \mathbf{E} \rangle_{i=0}^{n}$
translation probability model η

Output : final derivation $\tilde{\mathbf{D}}$

$\mathcal{M} \leftarrow$ INITIALIZESOFTMATRIX(F, E, η); `// build soft alignment matrix`

$S_0 \leftarrow \{$INITIALIZEHYPO$(0, |\mathbf{F}|, 0, |\mathbf{E}|)$ $\}$;

for $i = 0$ *to* MIN$(\mathbf{F}, \mathbf{E})$ **do**

  if $S_i = \{\}$ *;* `// no new hypothesis`

  then

    continue;

  end

  forall *hypo* $\in$ EXPANDHYPOS$(S_i, \mathcal{M})$*;* `// expand current hypothesis`

  do

    $S_{i+1} \leftarrow S_{i+1} \cup hypo$;

    if COVERAGE$(hypo) = true$*;* `// check whether all words aligned`

    then

      $S_{final} \leftarrow S_{final} \cup hypo$;

    end

    $S_{i+1} \leftarrow top_k(S)$; `// prune the beam with top-k`

  end

  $\tilde{D} = \underset{D \in S_{\text{final}}}{\text{argmax}}\ Score(D_{F_{avg}} | \mathcal{M})$; `// select the best derivation`

end

return $\tilde{D}$

Algorithm 2: EXPANDING HYPOTHESES

Input : beam S_i, $\mathcal{M}$

Output : next beam S

$S \leftarrow \{\}$;
while S_i **do**
  $hypo \leftarrow S_i.pop()$; `// pop the item from the previous stack`
  forall $block \in hypo.blocks$ **do**
    $S' \leftarrow \{\}$;
    $\{i_1, j_1, i_3, j_3\} \leftarrow block$; `// four corner indices in sub-matrix`
    forall $\{i_2, j_2\} \in \mathcal{M}_{(i_1,j_1,i_3,j_3)}$; `// search for all possible partitions`
    do
      for $\gamma \in [0, 1]$ **do**
        $score = \text{ComputeScore}(\gamma, i_2, j_2, \mathcal{M}_{(i_1,j_1,i_3,j_3)})$;
        if $\gamma = 0$; `// straight case in BTG`
        then
          $block_1, block_2 \leftarrow$DIAGONALMATRICES$(\mathcal{M}_{(i_1,j_1,i_3,j_3)}, i_2, j_2)$
        end
        else
          $block_1, block_2 \leftarrow$ANTIDIAGONALMATRICES$(\mathcal{M}_{(i_1,j_1,i_3,j_3)}, i_2, j_2)$
        end
        $S \leftarrow S \cup \{i_2, j_2, score, \gamma, block_1, block_2\}$;
      end
    end
  end
  forall $\{i_2, j_2, score, \gamma\, block_0, block_1\} \in top_k(S')$ **do**
    $new\_hypo = hypo.update([block_1, block_2], i_2, j_2, \gamma, score)$; `// expand current hypothesis`
  end
end
$S \leftarrow S \cup new\_hypo$; `// add new hypothsis into new stack`
return S

Algorithm 3: Online Training Algorithm for Top-Down BTG-based Preordering (OnlineTrainParser)

Input : training examples $\mathcal{T} = \langle F, \mathbf{a} \rangle_{i=0}^{n}$

initial feature weights $\mathbf{w}_0^\top$

Output : final feature weights $\mathbf{w}^\top$

foreach *epoch* $t \in (0, T)$ **do**

  $\mathbf{w}_{\mathbf{t+1}}^\top, \mathbf{w'}_{\mathbf{t+1}}^\top$=TrainOneEpoch($\mathcal{T}_t, \mathbf{w}_{\mathbf{t}}^\top, \boldsymbol{w'}_{\mathbf{t}}^\top$);

end

$\mathbf{w}^\top = \mathbf{w'}_{\mathbf{t+1}}^\top$;

return $\mathbf{w}^\top$;

Function TrainOneEpoch($\mathcal{T}, \mathbf{w}_{\mathbf{t}}^\top, \mathbf{w'}_{\mathbf{t}}^\top$)

  foreach *example* $\langle \mathbf{F}, \mathbf{a} \rangle$ **do**

    $\mathbf{D'} = \underset{\mathbf{D}}{\operatorname{argmax}}\ \text{Score}(\mathbf{D}|\mathbf{F}; \mathbf{w}_{\mathbf{t}}^\top)$; // System

    $\mathbf{D}^* = \underset{\mathbf{D} \wedge \text{Constraint}(\mathbf{D}, \mathbf{a})}{\operatorname{argmax}}\ \text{Score}(\mathbf{D}|\mathbf{F}; \mathbf{w}_{\mathbf{t}}^\top)$; // Oracle

    if $D^* \notin \mathit{Top}(\mathbf{D}, k)$ **then**

      break ; // early update

    end

    if $\mathbf{D'} \neq D^*$ **then**

      $\mathbf{w}_{\mathbf{t+1}}^\top = \mathbf{w}_{\mathbf{t}}^\top + \eta_{\mathbf{t}} \Big(\mathbf{\Phi}(\mathbf{D}^*) - \mathbf{\Phi}(\mathbf{D'}) \Big)$; // updating

      $\mathbf{w'}_{t+1}^\top = \mathbf{w'}_{t}^\top + \beta_{t,i} \times \eta_t \Big(\Phi(D^*) - \Phi(\mathbf{D'}) \Big)$; // averaged

    end

  end

return $\mathbf{w}_{\mathbf{t+1}}^\top$, $\mathbf{w'}_{t+1}^\top$

Algorithm 4: Distributed Training Algorithm for Top-Down BTG-based Preordering

Input : Training examples $\mathcal{T} = \langle \mathbf{F}, \mathbf{a} \rangle_{i=0}^{n}$

initial feature weights $\mathbf{w_0^\top}$

Output : Final feature weights $\mathbf{w}^\top$

Initialization: Shard T into $\{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$

foreach $\mathcal{T}_j \in \{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$ **do**

  $\mathbf{w_j^\top}$ =OnlineTrainParser($\mathcal{T}_j, \mathbf{w_0^\top}$);

end

$\mathbf{w}^\top = \frac{1}{m} \sum_{j=1}^{m} \mathbf{w_j^\top}$; // take the average

return $\mathbf{w_{t+1}^\top}$;

Algorithm 5: Iterative Distributed Training Algorithm for Top-Down BTG-based Preordering

Input : training examples $\mathcal{T} = \langle \mathbf{F}, \mathbf{a} \rangle_{i=0}^{n}$

initial feature weights $\mathbf{w_0^\top}$

Output : final feature weights $\mathbf{w_t^\top}$

Initialization: Shard T into $\{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$

foreach *epoch* t **do**

  foreach $\mathcal{T}_j \in \{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$ **do**

    $\mathbf{w_{t+1,j}^\top}, \mathbf{w'^\top_{t+1,j}}$ =TrainOneEpoch($\mathcal{T}_j, \mathbf{w_{t,j}^\top}, \mathbf{w'^\top_{t,j}}$);

  end

  $\mathbf{w_{t+1}^\top} = \frac{1}{m} \sum_{j=1}^{m} \mathbf{w_{t+1,j}^\top}$; // take the average

  $\mathbf{w'}^\top_{t+1}$ += $\frac{1}{m} \sum_{j=1}^{m} \mathbf{w'}^\top_{t+1,j}$;

  for $j \in (1, m)$ **do**

    $\mathbf{w_{t+1,j}^\top} = \mathbf{w_{t+1}^\top}$;

  end

end

$\mathbf{w}^\top = \mathbf{w'^\top_{t+1}}$;

return $\mathbf{w}^\top$;

Algorithm 6: Batch Training Algorithm for Top-Down BTG-based Preordering

Input : Training examples $T = \langle F, \mathbf{a} \rangle_{i=0}^{n}$

initial feature weights $\mathbf{w_0}^\top$

Output : Final feature weights $\mathbf{w}^\top$

Initialization: shard $\mathcal{T}$ into K batches $\{\mathcal{T}_1, \ldots, \mathcal{T}_K\}$

foreach *epoch* t **do**
  for *k-th batch* $\mathcal{T}_k \in \mathcal{T}$ **do**
    $S \leftarrow \textsc{CollectDerivations}(\mathcal{T}_k, \mathbf{w_t}^\top)$;
    $\Delta\ell, \Delta\Phi^*, \Delta\Phi, \Delta\eta_t, \Delta\mathbf{w}^\top \leftarrow \textsc{ComputeDeltas}(\mathbf{S})$;
    $\mathbf{w_{t+1}}^\top = \mathbf{w_t}^\top + \boldsymbol{\Delta}\eta_\mathbf{t}(\boldsymbol{\Delta}\boldsymbol{\Phi}^*_\mathbf{k} - \boldsymbol{\Delta}\boldsymbol{\Phi}_\mathbf{k})$;
    $\boldsymbol{w'}_{t+1}^\top = \boldsymbol{w'}_t^\top + \Delta\beta_{t,k} \times \Delta\eta_t(\Delta\Phi^*_k - \Delta\Phi_k)$;
  end
end
$\mathbf{w}^\top = \boldsymbol{w'}_{\mathbf{t+1}}^\top$;
return $\mathbf{w}^\top$;
Function $\textsc{CollectDerivations}(\mathcal{T}_k, \mathbf{w_t}^\top)$
  $S \leftarrow \{\}$;
  foreach *example* $\langle \mathbf{F}, \mathbf{a} \rangle \in \mathcal{T}_k$ **do**
    $\mathbf{D}' = \underset{\mathbf{D}}{\text{argmax}}\ \text{Score}(\mathbf{D}|\mathbf{F}; \mathbf{w_t}^\top)$; `// System`
    $\mathbf{D}^* = \underset{\mathbf{D} \wedge \text{Constraint}(\mathbf{D},\mathbf{a})}{\text{argmax}}\ \text{Score}(\mathbf{D}|\mathbf{F}; \mathbf{w_t}^\top)$; `// Oracle`
    $S \leftarrow S \cup \{\langle \mathbf{D}', \mathbf{D}^* \rangle\}$;
  end
return S;

Algorithm 7: CollectKBestDerivations

Function CollecttKBestDerivations($\mathcal{T}_k, \mathbf{w_t}^\top$)
  $S \leftarrow \{\}$;
  foreach $\mathbf{D} \in \mathit{Score}_{\mathit{top\text{-}k}}(\mathbf{D}|\mathbf{F}; \mathbf{w_t}^\top)$ **do**
    if $\mathbf{D} \wedge \mathit{Constraint}(\mathbf{D}, \mathbf{a})$ **then**
      $\hat{\mathbf{D}}* \leftarrow \hat{\mathbf{D}}* \cup \mathbf{D}$;
    else
      $\hat{\mathbf{D}}' \leftarrow \hat{\mathbf{D}}' \cup \mathbf{D}$;
    end
  end
  $S \leftarrow S \cup \{\langle \hat{\mathbf{D}}', \hat{\mathbf{D}}* \rangle\}$;
return S;

Algorithm 8: IterativeCodebookInference (M-MDL).

Input : data $\mathcal{D}$

Output : codebook Φ

Parameters : vocabulary size **m**

min_count **n**

while $|\Phi| < m$ **do**
  $\Phi \leftarrow$ GenerateCodebook $(\mathcal{D})$; `// scan the corpus, initialize the codebook.`
  $\mathcal{P} \leftarrow$ CollectPairStatistics$(\mathcal{D})$; `// scan the corpus, initialize the pair statistic.`
  $\mathcal{P}' \leftarrow$ SortedBy$\triangle$DL$(\mathcal{P})$; `// compute` $\triangle$`DL for each operation` $(w_1, w_2) \rightarrow w_1 w_2$.
  foreach $(w_1, w_2) \in \mathcal{P}'$ **do**
    if $\triangle$DL ≤ 0 & $\mathcal{P}[w_1 w_2] \geq$ **n then**
      UpdateCodebookAndPairStatistics$(\Phi, \mathcal{P}, w_1, w_2)$; `// commit, see Algorithm 10.`
    end
  end
  $\mathcal{D} \leftarrow$ ApplySegmentation$(\mathcal{D}, \Phi)$; `// FMM segmentation with the learned codebook.`
  if $\triangle$DL > 0 **then**
    Break ; `// stop condition.`
  end
end
return Φ;

Algorithm 9: FiniteIterativeCodebookInference (FM-MDL).

Input : Data $\mathcal{D}$

Output : Codebook Φ

Parameters : Vocabulary size **m**, min_count **n**

while $|\Phi| < m$ **do**
  $\Phi \leftarrow$ generate_codebook($\mathcal{D}$);
  $\mathcal{P} \leftarrow$ CollectPairStatistics($\mathcal{D}$);
  $\mathcal{P}' \leftarrow$ SortedBy$\triangle$DL($\mathcal{P}$);
  foreach $(w_1, w_2) \in \mathcal{P}'$ **do**
    if $|\Phi| \geq$ **m then**
      return Φ; `// stop condition.`
    end
    if $\triangle$DL ≤ 0 & $\mathcal{P}[w_1 w_2] \geq$ **n then**
      UpdateCodebookAndPairStatistics($\Phi, \mathcal{P}, w_1, w_2$);
      `// commit, see Algorithm 10.`
    end
  end
  $\mathcal{D} \leftarrow$ ApplySegmentation($\mathcal{D}, \Phi$); `// FMM segmentation with the learned codebook.`
end

Algorithm 10: UPDATECODEBOOKANDPAIRSTATISTICS.

Input : Pair (w_1, w_2), codebook Φ, pair statistic $\mathcal{P}$, suffixarray sa, min_count n

Output : New codebook $\tilde{\Phi}$, new pair_statistic $\tilde{\mathcal{P}}$,

Parameters : Vocabulary size m, min_count n

$\mathbb{R} \leftarrow$ get_indices(sa, w_1w_2); `// find the indices in suffixarray.`
$C \leftarrow \mathcal{P}[w_1w_2]$;
for $i \in \mathbb{R}$ **do**
  previous_word $\leftarrow sa[i-1]$;
  next_word $\leftarrow sa[i+2]$;
  if *(*previous_word*,* w_1*)* $\in \Phi$ **then**
    $C--$; `// invalid candidate.`
  end
  else if *(*w_2*,* next_word*)* $\in \Phi$ **then**
    $C--$; `// invalid candidate.`
  end
  if $\triangle\texttt{DL}(w_1, w_2, C)) \leq 0$ **then**
    $\Phi[w_1w_2] = \Phi[w_1w_2] + C$; `// insert new entry` w_1w_2 `into codebook.`
    $\Phi[w_1] = \Phi[w_1] - C$;
    $\Phi[w_2] = \Phi[w_2] - C$;
    if $\Phi[w_1] < 1$ **then**
      `Delete` $\Phi[w_1]$;
    else if $\Phi[w_2] < 1$ **then**
      `Delete` $\Phi[w_2]$;
    end
  end
end
`Delete` $\mathcal{P}[w_1, w_2]$; `// remove the candidate` w_1w_2.
$\tilde{\Phi}, \tilde{\mathcal{P}} \leftarrow \Phi, \mathcal{P}$;
return $\tilde{\Phi}, \tilde{\mathcal{P}}$;

Appendix C

Released Tools

The open-source software described in this dissertation are publicly available via Github:

- `Hieralign` (C++)
  `https://github.com/wang-h/Hieralign`
  a simple, fast unsupervised word aligner which delivers smaller translation tables without loss in translation quality (Chapter 3);

- `HieraParser (C++)`
  `https://github.com/wang-h/HieraParser`
  a fast top-down BTG parser for preordering and BTG-based decoding which supports multi-threading and performs a more accurate reordering (Chapter 4);

- `FMDL (Python/C++)`
  `https://github.com/wang-h/FMDL`
  a bilingual unsupervised subword segmenter for East Asian languages (Chapter 5).