Syntax-based Preordering for Statistical Machine Translation

星野　翔

博士（情報学）

総合研究大学院大学
複合科学研究科
情報学専攻

平成３０（２０１８）年度

Syntax-based Preordering for Statistical Machine Translation

統計的機械翻訳のための統語に基づく事前並べ替え

Sho Hoshino

星野　翔

Department of Informatics

School of Multidisciplinary Sciences

The Graduate University for Advanced Studies

March 2019

Abstract

This thesis focuses on a major yet unsolved machine translation difficulty called a word ordering problem. Since every language has its own word order, machine translation systems have to translate one's word order into another, in addition to translation of words. However, practical machine translation systems do not translate most of word orders due to computational complexity of the word ordering problem. This makes machine translation between distant language pairs, such as English and Japanese, inaccurate, because they have exceptionally dissimilar word orders as their nature. It is therefore the final goal of this study to establish a machine translation system for distant language pairs, using a practical reordering model that can handle accurate word ordering.

Our challenge primarily involves two types of obstacles we need to overcome. One obstacle is the computational complexity of the word ordering problem that has given limits to practical machine translation systems. Another obstacle is the problem of exceptionally dissimilar word orders in distant language pairs, such as English and Japanese, which has reduced machine translation accuracy to date.

In order to tackle our challenges, we make use of a promising approach called syntax-based preordering for statistical machine translation. In this approach, we use syntactic parsers that automatically parse input texts and output parse trees. We then modify the parse trees so that they represent much similar word orders to translation output than before. After that, the modified parse trees are used as input to a statistical machine translation system, which automatically learns a machine translation model from large data sets as similar to real world applications.

Since this approach effectively divides the problem of complex machine translation into a separated reordering step and a translation step, we can fully focus on our challenges with two novel proposals.

The first proposal is a rule-based approach. We present two rule-based preordering methods named a two-stage method and a three-stage method. The two-stage method reorders a Japanese parse tree as similar to English using deep syntax information obtained with a predicate-argument structure analyzer. The three-stage method mimics the two-stage method by using little or no syntax. We eventually demonstrate the state-of-the-art performance in Japanese-to-English translation to date as a rule-based preordering approach.

The second proposal is a statistical approach. The statistical approach automatically learns reordering rules, unlike the rule-based approach. We present a simple yet effective statistical preordering method. In this method, we employ a greedy optimization strategy for modifying parse trees so that the modified parse trees maximize our objective function for reordering. We achieve the state-of-the-art accuracy in both English-to-Japanese and Japanese-to-English translation.

Acknowledgments

First and foremost, I would like to devote my best gratitude and wishes to my supervisor, Dr. Yusuke Miyao. I cannot find a way to credit all his patient guidance, constructive suggestions, and enthusiastic encouragement in his supervision of my doctorate for more than 8 years. A small list of what I learned from him includes syntactic and semantic parsing, other kinds of deep natural language processing in general, functional and logic programming, machine learning, academic life, and on and on. I hope people will discover the fun fact about natural language processing that almost everything leads to syntactic and semantic parsing, as it happened in this study.

I would like to express my deep gratitude to all members of my committee who guided and helped my research process as well as my research life. Dr. Akiko Aizawa, my chief supervisor at this moment, allowed me to use her supercomputers for tons of experiments. Dr. Ryutaro Ichise helped me at many intermediate committees held in my course. Dr. Noriko Kando, my deputy supervisor, offered me the data that I always use in this study. Dr. Mamoru Komachi pioneered preordering in Japanese which made this study possible. Dr. Ken Satoh invited me to a joint research project where we discussed relationship betwen inversion transduction grammars and permutation trees.

Most of this study was conducted as a joint research with Dr. Katsuhito Sudoh, who have hosted me three times as a visiting researcher. He kindly allowed me to interrupt him countless times and we discussed in front of a black board almost everyday. In the course of the joint research project, Dr. Katsuhiko Hayashi and Dr. Masaaki Nagata also helped me about nearly

everything, sometimes with special telephone meetings via telephone exchange units.

I deeply appreciate Dr. Yuka Tateisi, who introduced me to this exciting research field of natural language processing when I was an undergraduate. She guided me from teaching me in classes to assisting my career. She also allowed me to start my first machine translation project which ended up as my first paper on subjective evaluation. It was unusually lucky for me to be able to enjoy some joint research projects with her, even after starting this study.

This study would not exist today without the help from the people of the Kansai machine translation study group: Dr. Kevin Duh, Dr. Atsushi Fujita, Dr. Hideto Kazawa, Dr. Taku Kudo, Dr. Shinsuke Mori, Mr. Yuji Mori, Mr. Makoto Morishita, Dr. Toshiaki Nakazawa, Dr. Graham Neubig, Mr. Yusuke Oda, Ms. Miori Sagara, Dr. Daisuke Torii, and Dr. Taro Watanabe.

I would like to appreciate all the staffs of the National Institute of Informatics (NII), especially Ms. Yuki Amano, Ms. Noriko Katsu, Ms. Keiko Kigoshi, Ms. Kaori Omura, and Ms. Akiko Takenaka. Also, I would like to thank the core members of the secret NII table tennis club: Mr. Yoshinari Fujinuma, Mr. Xinjian Li, Mr. Shunsuke Ohashi, and Mr. Kyohei Tomita.

I cannot list everyone who helped this study, but I am particularly grateful to Dr. Akira Fujita, Dr. Alvin Grissom II, Dr. Dan Han, Dr. Tadayoshi Hara, Dr. Jun Hatori, Dr. Hideki Isozaki, Mr. Akihiro Kameda, Dr. Pascual Martinez-Gomez, Dr. Takuya Matsuzaki, Dr. Nghiem Quoc Minh, Dr. Nguyen Luu Thuy Ngan, Dr. Hiroshi Noji, Mr. Akihiro Osaki, Mr. Hubert Soyer, Dr. Ran Tian, Mr. Goran Topic, Dr. Tam Wai Lok, and Dr. Xianchao Wu. My special thanks to China Horizon Travel and Ayana who helped me recover from illness and keep talking to me in Beijing.

Last but not least, I would like to credit my family: my father Kenji, my mother Harumi, my brother Ken, my sister Hitomi, and above all my grandparents. They have supported me since I began seeking my research interests.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

At present, we live in a world with the largest human population over billions, where the Internet enabled us to communicate directly with these billions of people all over the world, from the Atlantic to the Pacific, even if you are in the Antarctic. At the same time, we live in a world with various languages, such as Arabic, Chinese, English, French, Russian, and Spanish, which are the official languages offered by the United Nations. Therefore, the majority of the population may not be able to communicate directly with other people via different languages. As a result, our broad communication suffers from the language barrier known as *lost in translation* that caused miscommunications, misunderstandings, and ultimate mistakes in our history of war and peace.

Breaking the language barrier has long been a hope of humanity, since the mysterious story of the Tower of Babel. Currently, human translators and interpreters are making great efforts and successes to provide high quality translation by translating one language into another on their hands, whereas their expensive and time-consuming approaches would not be suitable for all the translation needs that may involve everyday conversations in minor languages. In addition to human translators and interpreters, *machine translation* technology has become a new hope in reality. Machine translation systems automatically translate one language into another with moderate quality at a scale and speed impossible for humans. Nowadays, millions of text

chats and tweets of ordinary people are translated simultaneously by cloud machine translation services.

However, current machine translation technology still suffers from the language barrier due to a major yet unsolved difficulty called *word ordering* problem. The word ordering problem is particularly problematic in translating dissimilar languages called distant language pairs, such as English and Japanese. Therefore, people speaking these languages may not be able to communicate with each other, but sometimes they need machine translation the most.

Therefore, this study adventures the word ordering problem and establishes an accurate machine translation system for distant language pairs by solving it. We introduce the word ordering problem from what it is, why it matters, and how to solve it.

1.1 Every Language Has Its Own Word Order

People who speak similar languages, such as English and French, are already enjoying practical machine translation in various situations, from weather forecast manuscripts to parliament proceedings. On the other hand, current machine translation systems used in production essentially have problems in translating dissimilar languages, such as English and Japanese, primarily because we need to consider big word order differences between these languages. Just as "Mary killed John" is completely different from what "John killed Mary" means in English, every language has its own word order. Therefore, in addition to translating each word into another, we need to accurately translate the word order of one language into another.

Linguists categorize the word order of various languages so that they are mainly characterized by how a subject, a verb, and an object are placed in a sentence. For example, Chinese, English, and French have the order of subject-verb-object (SVO), while Japanese and Korean have the order of subject-object-verb (SOV). In that sense, although the vocabulary of Chinese and Korean might be more common than that of English, we see Chinese and English are similar but Chinese and Korean are not. There are other types of word order in the world, but many languages belong to either SVO or SOV.

1.2 When We Need to Translate Word Order

When we translate one language into another, the word ordering problem becomes our headache. If they contain the same word order, we can translate words in a sentence one by one, from left to right. Otherwise, we need to translate the word order of one language into another at the same time. Since every language has its own word order, it is more likely that we also need to translate the word order.

That is exactly what happened to current machine translation technology. Since machine translation of very similar language pairs, such as English-French and Japanese-Korean, is highly accurate, we can translate them without difficulty. Meanwhile, there are distant language pairs, such as English-Japanese, which suffer very low accuracy at present due to exceptionally dissimilar word orders.

1.3 Reordering Methods for Machine Translation

In machine translation research, long-standing efforts to generate sentences with accurate word ordering for any language pair are called reordering methods. The basic concept of reordering methods is that we learn word order differences between a language pair, then we transform one's word order into another for helping translation. For example, various syntax-based reordering approaches for statistical machine translation have been proposed over more than a decade: syntax-based statistical machine translation (Yamada and Knight, 2001), preordering (Xia and McCord, 2004; Collins et al., 2005), hierarchical phrase-based statistical machine translation (Chiang, 2007), and postordering (Sudoh et al., 2011).

Among such approaches, preordering does not calculate everything at once when translating a sentence, but performs reordering operations within a simple and independent preprocessing step. Therefore, we can take advantage of it as opportunities for optimization (Oda et al., 2016) and system change (Kawara et al., 2018). However, in practical statistical machine translation systems, non syntax-based approaches such as lexicalized reordering (Tillman, 2004) are still preferred because of simplicity.

Therefore, in this study, we aim to build a practical machine translation system that can translate distant language pairs accurately using simple yet effective preordering. In order to understand why machine translation has become so difficult and complicated, we need to look back on the history of machine translation.

1.4 Brief History of Machine Translation

In 1949, Warren Weaver wrote a famous memorandum to Norbert Wiener (Sergei Nirenburg and Wilks, 2003), which reads as follows:

> Also knowing nothing official about, but having guessed and inferred considerable about, powerful new mechanized methods in cryptography — methods which I believe succeed even when one does not know what language has been coded — one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say "This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.

Even today, the basic idea of machine translation is this way of thinking machine translation as a task of cryptography. Following this idea, we will examine a brief history of machine translation, from the initial implementation in the rule-based approach to the latest implementation in deep neural networks.

1.4.1 Rule-based Machine Translation

The Weaver's memorandum triggered the spread of machine translation research in the 1950s. An example of this era is a rule-based machine translation system that Georgetown University and IBM demonstrated in 1954, using "a very restricted vocabulary of 250 words and just six grammar rules" to translate Russian sentences into English (Brown, 2005). Such a rule-based machine translation system has the advantage of being perfectly understandable to details and can be completely controlled by humans. On the other hand, it turned out that it is impossible

for humans to describe all translation rules, and it is also difficult to maintain the consistency of translation rules.

1.4.2 Example-based Machine Translation

Therefore, instead of manually describing the translation rules, a data-driven approach called example-based machine translation (Nagao, 1984) was proposed and developed in the 1980s. The example based machine translation searches example sentences similar to input sentences from a corpus stored in advance, then obtains translated sentences by editing the retrieved example sentences. For example, if the input sentence is "I have a pen" and there is a sentence "I have an apple" in the corpus, we can output "I have a pen" by editing an apple to a pen. However, the lower the similarity between languages, the greater the need for editing. After all, especially in distant language pairs, we need to edit and reorder the entire input sentence instead of editing parts of example sentences.

1.4.3 Statistical Machine Translation

For this reason, another data-driven approach called statistical machine translation (Brown et al., 1990; Brown et al., 1993) was proposed in the 1990s. Returning to the idea of cryptology, they formulated machine translation as an instance of the noisy channel model (Shannon, 1948). In their formulation, the input sentence is considered to be encoded in a foreign language, and translation is done by decoding the input sentence into a native language. For example, when translating French into English, we can regard French as a foreign language and English as a native language. They succeeded in formulating all components in detail using Bayesian statistics as backbone. However, due to the detailed formulating, statistical machine translation has problems in computational complexity, especially when also considering the word order problem for distant language pairs. In addition, it is difficult to understand all the components used in statistical machine translation.

1.4.4 Neural Machine Translation

In recent years, yet another data-driven approach called neural machine translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015) that enables end-to-end translation using deep neural networks has also been proposed in the 2010s. With this end-to-end approach, machine translation is no longer a collection of components, but is regarded as a single black box that does not fall under the word ordering problem. The flip side of the coin is that neural machine translation suffers from the real word problems, such as poor performance in treating small training data and rare words, most of which did not become a serious problem in statistical machine translation (Koehn and Knowles, 2017). Therefore, neural machine translation is not yet realistic for critical applications, but it may be the closest to automatic machine translation in the future.

1.5 Contributions of This Study

In this way, the formulation of machine translation has become complicated to achieve fully automated machine translation, but gradually becomes uncontrollable and difficult to understand. Therefore, this study aims to establish a controllable and understandable yet accurate machine translation system for distant language pairs. In order to establish such a system, we need to address the following problems:

- Building a fully automatic machine translation system for distant language pairs
- Making the system controllable and easy to understand
- Improving accuracy while keeping the system controllable and understandable

In response to the problems listed above, we propose solutions using syntax-based preordering for statistical machine translation based on two hypotheses.

The first hypothesis is that the use of syntax-based preordering can enhance the controllability of statistical machine translation. Since preordering performs reordering operations as a preprocessing step, the use of preordering makes it possible to divide and conquer statistical

machine translation using a separated preordering process and a translation process. Then, we can concentrate on making accurate reordering for distant language pairs.

The second hypothesis is that the use of syntax-based preordering can reduce the difficulty of statistical machine translation. Since syntax-based preordering methods output intermediate representation in human-readable formats, we will be able to understand what is happening with a complex statistical machine translation system and modify the system as needed.

We examine these two hypotheses throughout this study, and each method proposed in these chapters corresponds to our contributions:

- In Chapter 2, we describe the overall picture of formulation in statistical machine translation.

- In Chapter 3, we present two types of fully controllable rule-based preordering methods, where one method uses syntax extensively and the other uses little or no syntax. Both methods address syntactic problems raised in previous studies on the task of rule-based preordering for Japanese-to-English translation. We then explore roles of syntactic information in rule-based preordering for distant language pairs by comparing the two methods in Japanese-to-English translation.

- In Chapter 4, we present a statistical preordering method that overcomes the limit on accuracy in rule-based approaches. The method also addresses optimization problems found in previous studies on statistical preordering. We then demonstrate effectiveness of syntactic information in statistical preordering in the task of English-to-Japanese and Japanese-to-English translations.

- Finally in Chapter 5, we draw our conclusions and future prospects.

Chapter 2

Background

Since this study involves various syntax-based methods for statistical machine translation, in this chapter, we give an overview of technologies behind our methodology, such as tokenization, syntax, statistical machine translation, and machine learning. We begin with tokenization, which is a basic standard of how we treat our data. We then move to syntax, a concept that determines the way how we look at the world of languages. Following syntax is the foundation of machine translation, from the statistical machine translation onwards, which is a framework of automatic translation from one language into another. In the meantime, we also stop by the field of machine learning, which is a technology that enabled machine translation to automatically learn from and make predictions on data, without being explicitly programmed, unlike the good old rule-based machine translation systems.

2.1 Tokenization

The first step we need to make automatic machine translation happen is *tokenization*, which transforms a sequence of characters into a sequence of words (one or more characters).

The tokenization becomes mandatory in preprocessing of multi-lingual machine translation, because some languages, such as Chinese and Japanese, do not have practice of inserting space between words in their writing system. For example, a Japanese sentence “吾輩は猫である” meaning “I am a cat” does not have any white space characters at all. Therefore, we tokenize

that sentence into “吾輩 は 猫 である”, which is a more readable form for both non Japanese speakers and machines.

Even in other languages such as English, it is usually better to have the tokenization step. For example, you want to split “they’re” into the two words “they” and “are” instead of keeping them as one word. Ultimately, one can define such a tokenization standard in a completely data-driven fashion, by using subword units based on byte pair encoding (Gage, 1994; Sennrich et al., 2016) and unigram language model (Kudo, 2018; Kudo and Richardson, 2018).

English tokenizer used in this study is the Moses tokenizer (Koehn et al., 2007). Japanese tokenizers used in this study include MeCab (Kudo et al., 2004) and JUMAN (Kurohashi and Kawahara, 2012).

2.2 Syntax

In this study, we borrow three types of syntactic structures from linguists: phrase structure, dependency structure, and predicate-argument structure. Although they are roughly interexchangeable, each of them has unique characteristics that cannot be directly and conveniently represented in another framework. Therefore, we stick to different frameworks for different purposes.

2.2.1 Phrase Structure

Let us think of some examples for linguistic analysis using syntactic structure. We say, “They drink coffee” and “They drink a bottle of coke” as well as “They drink several cups of tea a day”. Their meanings are all different, but we regard them as having a common meaning of “They drink something”. Figure 2.1 shows such a syntactic structure in a linguistic framework called *phrase structure* (also known as *constituency*).

This phrase structure indicates that the sentence has a hierarchical construction, which contains a noun and a verb followed by another noun and a punctuation mark. Since this phrase structure is a symbolic framework, we can easily modify a part of the sentence while keeping its basic structure, as we replace “something” with “a cup of coffee” as shown in Figure 2.2.

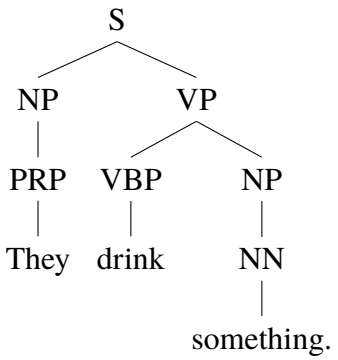

Figure 2.1: Example of phrase structure: the symbol S denotes a sentence, NP denotes a noun phrase, VP denotes a verb phrase, PRP denotes a personal pronoun, VBP denotes a present verb, and NN denotes a noun.

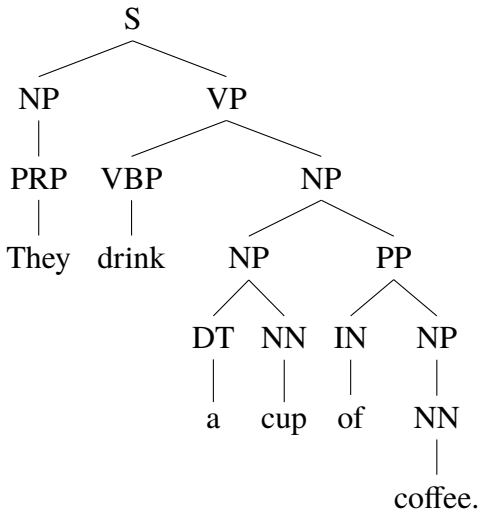

Figure 2.2: Another example of phrase structure: the symbol PP denotes a prepositional phrase, DT denotes a determiner, IN denotes a preposition.

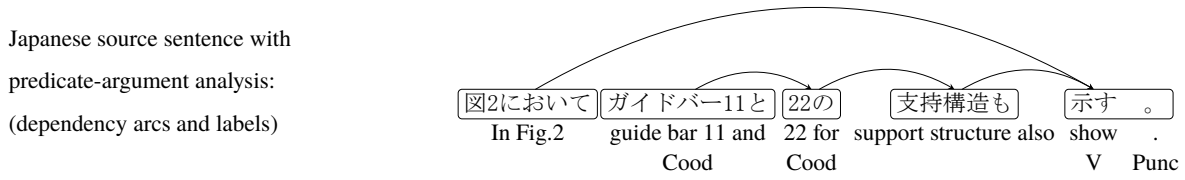

Figure 2.3: Example of predicate-argument structure in Japanese. Each box represents a chunk. The labels Cood, V, and Punc denote a chunk coordination, a head verb, and a Japanese punctuation mark, respectively.

Now this sentence represents a much more realistic object than just “something”, but, even after changing the words, we can manipulate the whole structure as a variant of “They drink something” just as before.

When we humans look at a sentence, we can automatically parse the sentence and get a syntactic structure similar to this. In the case of computational linguistic analysis on a computer like this study, we parse sentences automatically using a program called *syntactic parser*.

English constituency parsers used in this study include the Berkeley Parser (Petrov et al., 2006; Petrov and Klein, 2007) and Enju (Miyao and Tsujii, 2005; Miyao and Tsujii, 2008), which are trained on the Wall Street Journal section of the Penn Treebank (Marcus et al., 1993). The Berkeley Parser can also be used for parsing Japanese input using the toolkit called Haruniwa (Fang et al., 2014; Horn et al., 2017), which is trained on the Keyaki Treebank (Butler et al., 2012; Horn et al., 2017).

2.2.2 Dependency Structure

Unlike phrase structure, the framework of dependency structure is historically developed in the languages that have free word order, such as Japanesese. Nowadays, however, dependency is as popular in English as constituency.

The upper half of Figure 2.3 shows dependency arcs annotated to a Japanese sentence. Typical dependency structure in Japanese uses dependency on top of a chunk called *bunsetsu*, which is a

grammatical and phonological unit consisting of noun, verb, or adverb followed by dependents such as particles. The dependency arcs represent dependency relations to each head chunk from its dependent child chunks, ending with a root chunk indicating the head of a sentence.

English dependency parsers used in this study include the Stanford Parser (Marneffe et al., 2006) and MaltParser (Nivre et al., 2007), which are trained on the Penn Treebank converted from constituency to dependency beforehand (Marneffe et al., 2006). Japanese dependency parsers used in this study include CaboCha (Kudo and Matsumoto, 2002) and KNP (Kawahara and Kurohashi, 2006b; Sasano and Kurohashi, 2011), which are trained on the Kyoto University Text Corpus (Kawahara et al., 2002).

2.2.3 Predicate-Argument Structure in Japanese

There is another linguistic framework commonly used in both Japanese syntactic and semantic parsing called *predicate-argument structure*. The bottom half of Figure 2.3 shows the predicate-argument structure annotated as labels. The predicate-argument structure introduces relationships between a predicate chunk and its argument chunks. Instead of introducing all the possible predicate-argument relations, we simply introduce three commonly used "S", "V", and "O" labels, which represent triples of subject, verb, and object:

- The "S" label represents a Japanese *ga* argument as a subject.
- The "V" label represents a Japanese predicate as a head verb.
- The "O" label represents Japanese *wo* or *ni* arguments as an object.

For instance, the chunk "示す show" from the same example is labeled "V", indicating that this chunk is a head verb.

There are some dependency relations that come up with dependency labels, such as a coordination label shown as "Cood" in this example. The coordination relation in Japanese dependency indicates a coordination structure between head and dependent chunks, which often implicitly

links two conjunct chunks without using a coordinating conjunction, unlike explicit use of coordinating conjunction in English, such as “and”, “or”, and “but”. For instance, the two chunks “ガイドバー11と guide bar 11 and” and “22の 22 for” in this example are coordinated, representing a construction similar to “guide bar 11 and 22” in English.

Japanese predicate-argument structure analyzer used in this study include SynCha (Iida and Poesio, 2011) and KNP (Kawahara and Kurohashi, 2006b; Sasano and Kurohashi, 2011), which are trained on the NAIST Text Corpus (Iida et al., 2007), the Kyoto University Case Frame (Kawahara and Kurohashi, 2006a), and the Kyoto University Noun Case Frame (Sasano and Kurohashi, 2009). We did not use any English predicate-argument structure analyzers, but Enju (Miyao and Tsujii, 2005; Miyao and Tsujii, 2008) can also output such information.

In this way, syntax provides an interpretable abstraction layer to natural language processing. This interpretability is very important when we want to track of errors on complex systems, such as statistical machine translation.

2.3 Formulation of Statistical Machine Translation

Statistical machine translation (Brown et al., 1990; Brown et al., 1993) has been used to automatically translate one language into another all over the world, from daily conversations to official documents of the United Nations. Statistical machine translation aims to formulate every single piece of machine translation modules as statistical models, following corpus-driven approaches in computational linguistics.

The basic assumption behind statistical machine translation is the noisy channel model (Shannon, 1948) used in cryptography. When we translate Russian into English, we treat our Russian input as noisy English; it was once English input but was mistakenly *encoded* in Russian, thus we will *decode* it into English. Therefore a complete statistical machine translation decodes output from input.

We will see how this noisy channel model works with statistics. We begin with one of the simplest yet most effective statistical modeling called *language model*.

2.3.1 Language Model

The language model (Shannon, 1951; Brown et al., 1992b) measures how likely it is that a sequence of words would be uttered by a native speaker in one language. For example in English, this model should prefer correct word order to incorrect word order as follows:

$$P(\text{the cat is small}) > P(\text{small the is cat}).$$

In general a language model is a function that takes a monolingual sentence and returns the probability that it was produced by a native speaker. It is more likely that an English speaker would utter the sentence "the cat is small" than the sentence "small the is cat". Therefore a good language model assigns a higher probability to the former sentence than the latter.

N-Gram Language Model

The n-gram language model (Shannon, 1951; Brown et al., 1992b) is the most widely used language modeling method that is based on statistics of how likely words are to follow each other. Recall the last example, if we analyze a large amount of text, we will observe that the word "cat" follows the word "the" more often than the word "is" does.

Formally we want to compute the probability of a sequence $W = w_1, w_2, \cdots, w_n$ where $n \in \mathbb{N}$. Intuitively $P(W)$ is the probability when we pick a random sequence of words from dictionaries and the sequence turns out to be W. Thus we only have to count how often W occurs naturally.

It is, however, commonly known that many words and most long sequences of words will not appear at all in the corpus we have. As a result we have to break down the computation of $P(W)$ into smaller steps, for which we can collect sufficient statistics and estimate probability distributions.

In that sense we introduce the Markov chain and predict one word at a time:

$$P(w_3|w_1, w_2) = \frac{count(w_1, w_2, w_3)}{\sum_w count(w_1, w_2, w)},$$

where in the sequence, w_1, w_2 is followed by the word w_3.

The sequence used here containing three words are called 3-grams (trigrams). Language models may also be estimated over 2-grams (bigrams), single words (unigrams), or any other order of n-grams.

Kneser-Ney Smoothing

In order to estimate the probability of W, which may contain the words that do not appear in the corpus but would appear in somewhere else, we employ smoothing methods for better counting.

The Kneser-Ney smoothing (Kneser and Ney, 1995; Chen and Goodman, 1996) is a smoothing method that takes the diversity of histories into account. We define the count of histories for a word w as:

$$N_{1+}(w) = |\{w_i | count(w_i, w) > 0\}|,$$

where $count(w_i, w)$ is the number of occurrences that a word w_i happened to be followed by the word w.

Recall that a unigram language model can be estimated as:

$$P(w) = \frac{count(w)}{\sum_i count(w_i)}.$$

In the Kneser-Ney smoothing, we simply replace the raw counts with the counts of histories for a word as follows:

$$\begin{aligned} P(w) &= \frac{count(w)}{\sum_i count(w_i)} \\ &= \frac{N_{1+}(w)}{\sum_{w_i} N_{1+}(w_i)}. \end{aligned}$$

2.3.2 Noisy Channel Model

From now on we step into the world of statistical machine translation from the language model, which still utilizes the language model as well as another component called translation model. Essentially statistical machine translation considers automatic translation process of one sentence at a time from the input called source language F to the output called target language E. These symbols are named after the early French-to-English translation attempts in the 1990's.

Formally statistical machine translation from a source sentence $\boldsymbol{f} \in F$ to a target language $\boldsymbol{e} \in E$ is formulated with the noisy channel model (Shannon, 1948):

$$\hat{\boldsymbol{e}} = \underset{\boldsymbol{e}}{\text{argmax}} P(\boldsymbol{e}|\boldsymbol{f}),$$

where obtaining $\hat{\boldsymbol{e}} \in E$ that maximizes $P(\boldsymbol{e}|\boldsymbol{f})$ is our final goal.

We apply the Bayes' theorem to this formula for transforming it into:

$$\begin{aligned}\hat{\boldsymbol{e}} &= \underset{\boldsymbol{e}}{\text{argmax}} P(\boldsymbol{e}|\boldsymbol{f}) \\ &= \underset{\boldsymbol{e}}{\text{argmax}} \frac{P(\boldsymbol{f}|\boldsymbol{e})P(\boldsymbol{e})}{P(\boldsymbol{f})} \\ &= \underset{\boldsymbol{e}}{\text{argmax}} P(\boldsymbol{f}|\boldsymbol{e})P(\boldsymbol{e}),\end{aligned}$$

where $P(\boldsymbol{f})$ is a constant with respect to $\underset{\boldsymbol{e}}{\text{argmax}}$.

With this transformation we no longer directly estimate $P(\boldsymbol{e}|\boldsymbol{f})$, but instead we can separately solve the statistical machine translation problem as a combination of the translation model $P(\boldsymbol{f}|\boldsymbol{e})$ and the language model $P(\boldsymbol{e})$ we already saw. We benefit from the last transformation because monolingual resources required for the language model are usually far more obtainable than bilingual or multilingual resources required for the translation model.

In order to generalize the variations of the statistical machine translation which we will introduce and discuss later, we also model intermediate representations on top of the translation model:

$$\begin{aligned}\hat{\boldsymbol{e}} &= \underset{\boldsymbol{e}}{\text{argmax}} P(\boldsymbol{f}|\boldsymbol{e})P(\boldsymbol{e}) \\ &= \underset{\boldsymbol{e}}{\text{argmax}} \sum_{d \in \mathcal{D}(\boldsymbol{f},\boldsymbol{e})} P(\boldsymbol{f}, \boldsymbol{d}|\boldsymbol{e})P(\boldsymbol{e}),\end{aligned}$$

where a latent variable $\boldsymbol{d}$ is the derivation that represents every possible intermediate representation, and $\mathcal{D}(\boldsymbol{f}, \boldsymbol{e})$ represents a set of such derivations.

We then train our translation model P_θ with the maximum likelihood estimation of parameter θ upon bilingual data $\langle F, E \rangle$:

$$\hat{\theta} = \underset{\theta}{\text{argmax}} \prod_{\langle \boldsymbol{f},\boldsymbol{e} \rangle \in \langle F,E \rangle} P_\theta(\boldsymbol{f}|\boldsymbol{e}),$$

where $\hat{\theta}$ is a trained parameter. Training of the latent variable $\boldsymbol{d}$ is done subsequently. The bilingual data is a clean set of parallel sentences created from bilingual or multilingual resources, which usually contains more than million sentences in parallel.

Eventually we decode the combined translation and language models according to the already trained parameter θ:

$$\hat{\boldsymbol{e}} = \underset{\boldsymbol{e}}{\operatorname{argmax}} \sum_{d} P_\theta(\boldsymbol{f}, \boldsymbol{d}|\boldsymbol{e}) P_\theta(\boldsymbol{e}).$$

In order to find feasible derivations under limited computation time and space, we apply the Viterbi approximation for changing our objective into:

$$\langle \hat{\boldsymbol{e}}, \hat{\boldsymbol{d}} \rangle = \underset{\langle \boldsymbol{e}, \boldsymbol{d} \rangle}{\operatorname{argmax}} P_\theta(\boldsymbol{f}, \boldsymbol{d}|\boldsymbol{e}) P_\theta(\boldsymbol{e}).$$

Practically we will find the Viterbi translation $\langle \hat{\boldsymbol{e}}, \hat{\boldsymbol{d}} \rangle$ with inexact search methods such as the beam search (Koehn et al., 2003; Koehn, 2004a; Koehn et al., 2007).

2.3.3 IBM Models

The IBM models (Brown et al., 1993) are the series of statistical machine translation modeling that introduced a concept called *word alignment* $\boldsymbol{a}$ as a derivation in the translation model:

$$\hat{\boldsymbol{e}} = \underset{\boldsymbol{e}}{\operatorname{argmax}} \sum_{\boldsymbol{a}} P_\theta(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{e}) P_\theta(\boldsymbol{e}).$$

The word alignment is a process and its output that finds bilingual relationships between words in a pair of source and target sentences. For example in German and English:

Ich trinke einen Kaffee

I drink a cup of coffee,

we can find bilingual pairs such as $\langle \text{I}, \text{Ich} \rangle$ and $\langle \text{coffee}, \text{Kaffee} \rangle$ in this bilingual sentence pair. In contrast words such as “of” are not aligned explicitly. The words that are considered to be not aligned and remained alone are called *null alignment* or *null words*. For instance, English articles and Japanese particles are usually null aligned.

In this example, however, one might see that the pair of words $\langle$a cup of coffee, einen Kaffee$\rangle$ is properly aligned in spite of the null word existence. Later we introduce *phrasal alignment* that also allows more than one word.

Formally word alignment $\boldsymbol{a}$ is defined as a set:

$$\boldsymbol{a} = \{\cdots, (j, i), \cdots\},$$

such that a bilingual relationship pair $\langle f_j, e_i \rangle$ appears in the sentence pair $\langle \boldsymbol{f}, \boldsymbol{e} \rangle$ where $f_j \in \boldsymbol{f}$, $e_i \in \boldsymbol{e}$, $i, j \in \mathbb{N}$, and $(j, i) \in \mathbb{N} \times \mathbb{N}$.

For their further modeling, some IBM models also made the *one-to-many alignment* assumption that a target word e_i is aligned to the only one source word f_j but source words can take multiple target words at the same time. This one-to-many alignment can be simply defined as a list:

$$\boldsymbol{a} = a_1, \cdots, a_j, \cdots a_J,$$

where $a_j = i$, $\boldsymbol{f} = f_1 \cdots f_j \cdots f_J$, $\boldsymbol{e} = e_1 \cdots e_i \cdots e_I$, and $I, J \in \mathbb{N}$.

In addition, in order to estimate many-to-many alignment from the one-to-many alignment estimated with the IBM models, we merge two instances of one-to-many alignment into a new many-to-many instance, which are obtained from bidirectional alignment between source and target languages:

$$\hat{\boldsymbol{a}}^{\triangleright} = \underset{\boldsymbol{a}^{\triangleright}}{\text{argmax}} P(\boldsymbol{f}, \boldsymbol{a}^{\triangleright} | \boldsymbol{e})$$

$$\hat{\boldsymbol{a}}^{\triangleleft} = \underset{\boldsymbol{a}^{\triangleleft}}{\text{argmax}} P(\boldsymbol{f}, \boldsymbol{a}^{\triangleleft} | \boldsymbol{e}),$$

where $\boldsymbol{a}^{\triangleright}$ and $\boldsymbol{a}^{\triangleleft}$ are one-to-many alignment of target-to-source and source-to-target directions, respectively.

We further calculate their intersection $\boldsymbol{a}^{\cap}$ and union $\boldsymbol{a}^{\cup}$ as:

$$\boldsymbol{a}^{\cap} = \{(j, i) | a_j^{\triangleright} = i \wedge a_i^{\triangleleft} = j\}$$

$$\boldsymbol{a}^{\cup} = \{(j, i) | a_j^{\triangleright} = i \vee a_i^{\triangleleft} = j\},$$

where $\boldsymbol{a}^{\triangleright} = a_1^{\triangleright}, \cdots, a_j^{\triangleright}, \cdots, a_J^{\triangleright}$ and $\boldsymbol{a}^{\triangleleft} = a_1^{\triangleleft}, \cdots, a_i^{\triangleleft}, \cdots, a_I^{\triangleleft}$.

Then we finally generate our many-to-many alignment $\boldsymbol{a}^{\triangleleft\triangleright}$ as a following set:

$$\begin{aligned}
\boldsymbol{a}^{\triangleleft\triangleright} = \{\cdots, (j,i), \cdots | (j,i) \in \boldsymbol{a}^{\cap} \vee \\
(j-1,i) \in \boldsymbol{a}^{\cap} \wedge (j,i-1) \in \boldsymbol{a}^{\cap} \wedge (j+1,i) \in \boldsymbol{a}^{\cap} \wedge (j,i+1) \in \boldsymbol{a}^{\cap} \wedge (j,i) \in \boldsymbol{a}^{\cup} \vee & \quad \text{grow} \\
(j-1,i-1) \in \boldsymbol{a}^{\cap} \wedge (j-1,i+1) \in \boldsymbol{a}^{\cap} \wedge (j+1,i-1) \in \boldsymbol{a}^{\cap} \wedge (j+1,i+1) \in \boldsymbol{a}^{\cap} \wedge (j,i) \in \boldsymbol{a}^{\cup} \vee & \quad \text{diag} \\
(j',i) \notin \boldsymbol{a} \wedge (j,i') \notin \boldsymbol{a} \wedge (j,i) \in \boldsymbol{a}^{\cup}\}, & \quad \text{final-and}
\end{aligned}$$

where $1 \leq j' \leq J$ and $1 \leq i' \leq I$. This is called "grow-diag-final-and" heuristic (Koehn et al., 2003) and shown empirically effective (Och and Ney, 2003).

2.3.4 Phrase-Based Machine Translation

The modeling we studied so far regarded a sentence as a composition of words and no more than that. It is, however, more realistic that we consider the multi word expressions that happen to have more than one word in both source and target languages, such as the previous $\langle$a cup of coffee, einen Kaffee$\rangle$ pair.

For this purpose, the phrase-based machine translation model (Koehn et al., 2003) extended word-based models with phrasal alignment $\boldsymbol{\alpha}$ and phrase pairs $\boldsymbol{\phi}$ defined as:

$$\begin{aligned}
\boldsymbol{\alpha} &= \{\alpha_1, \cdots, \alpha_L\} \\
\bar{\boldsymbol{f}} &= \bar{f}_1, \cdots, \bar{f}_L \\
\bar{\boldsymbol{e}} &= \bar{e}_1, \cdots, \bar{e}_L \\
\boldsymbol{\phi} &= \{\cdots, \langle \bar{f}_{\alpha_k}, \bar{e}_k \rangle, \cdots\},
\end{aligned}$$

where $\bar{f} \in F$, $\bar{e} \in E$, $1 \leq k \leq L$, $L \in \mathbb{N}$.

Then we replace the former word alignment $\boldsymbol{a}$ in the IBM models with this phrasal alignment $\boldsymbol{\alpha}$ and phrase pairs $\boldsymbol{\phi}$ as follows:

$$\begin{aligned}
\hat{\boldsymbol{e}} &= \underset{\boldsymbol{e}}{\operatorname{argmax}} P(\boldsymbol{f}|\boldsymbol{e}) P(\boldsymbol{e}) \\
&= \underset{\boldsymbol{e}}{\operatorname{argmax}} \sum_{\boldsymbol{\phi}, \boldsymbol{\alpha}} P(\boldsymbol{f}, \boldsymbol{\phi}, \boldsymbol{\alpha}|\boldsymbol{e}) P(\boldsymbol{e}),
\end{aligned}$$

where $P(\boldsymbol{f}, \boldsymbol{\phi}, \boldsymbol{\alpha}|\boldsymbol{e})$ represents the phrasal translation model.

2.4 Machine Learning for Statistical Machine Translation

After speculating various statistical machine translation models, we will study how to learn such a complex model from data automatically. That approach is called machine learning, where machines automatically learn to predict in a specific task given data. The machine learning tasks include *classification* and *regression*. In the classification task, we are given a fixed set of discrete labels called classes, and a learned model assigns one or more of these classes to its input. The regression task, in contrast to the classification, assigns a continuous value to its input.

When we have observed data $\boldsymbol{x}$ and classes y as a labeled data set $\mathcal{D} = \{(y, \boldsymbol{x})\}$, where $(y, \boldsymbol{x})$ represents a pair of a class and an observed instance, we want to know how to predict y from $\boldsymbol{x}$. This learning process can be defined as a conditional probability $P(y|\boldsymbol{x})$.

After finishing the learning process we predict unknown classes y from observed data $\boldsymbol{x}$. We define our prediction of classes $\hat{y}$ as:

$$\hat{y} = \underset{y}{\text{argmax}} P(y|\boldsymbol{x}).$$

We apply the Bayes' theorem to the conditional probability $P(y|\boldsymbol{x})$:

$$\begin{aligned}\hat{y} &= \underset{y}{\text{argmax}} P(y|\boldsymbol{x}) \\ &= \underset{y}{\text{argmax}} \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})} \\ &= \underset{y}{\text{argmax}} P(\boldsymbol{x}|y)P(y),\end{aligned}$$

where $P(\boldsymbol{x})$ is a constant with respect to $\underset{y}{\text{argmax}}$. With this formulation we can make use of the additional prior information $P(y)$.

In machine learning, the methods with the former formulation are called *discriminative models*. In contrast, the methods with the latter Bayesian formulation are called *generative models*.

Nevertheless the methods of both models that learn $\mathcal{D}$ are called *supervised methods*, otherwise we predict unknown data $\boldsymbol{x}$ from unlabeled data $\boldsymbol{x}$ alone as *unsupervised methods*. Super-

vised methods with an assumption that the inner product $\langle \boldsymbol{x}, \boldsymbol{w} \rangle$ of observed data $\boldsymbol{x}$ and a weight vector $\boldsymbol{w}$ is predictable of y are called *linear models*.

2.4.1 Linear Classifications

In this study we mainly focus on application of such linear models. For the data set $\mathcal{D}$ we define:

$$y \equiv \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, w \equiv \begin{bmatrix} w_1 \\ \vdots \\ w_K \end{bmatrix}, X \equiv \begin{bmatrix} x_1^\top \\ \vdots \\ x_N^\top \end{bmatrix} \equiv \begin{bmatrix} x_{11} & \cdots & x_{1K} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{NK} \end{bmatrix},$$

where $N \in \mathbb{N}$ is the data size and $K \in \mathbb{N}$ is the data rank.

For unknown data $\boldsymbol{x}$ we defined a classifier $y(\boldsymbol{x})$ as:

$$y(\boldsymbol{x}) = \langle \boldsymbol{x}, \hat{\boldsymbol{w}} \rangle,$$

where $\hat{\boldsymbol{w}}$ is a learned weight vector for this classifier. When we classify whether $y(\boldsymbol{x}) \geq 0$, this classifier returns only two types of classes and called binary classifications. Other types of the classifier are called multi class classifications.

In order to prevent our classifier from overfitting to our training data $\mathcal{D}$ alone, we learn the weight vector $\hat{\boldsymbol{w}}$ from the squared loss $\mathcal{L}$ and the regularization term $R(\boldsymbol{w})$ as:

$$\begin{aligned} \hat{\boldsymbol{w}} &= \underset{\boldsymbol{w}}{\operatorname{argmin}} \left(\mathcal{L}(\boldsymbol{w}, \mathcal{D}) + \lambda R(\boldsymbol{w}) \right) \\ &= \underset{\boldsymbol{w}}{\operatorname{argmin}} \left(\sum_{n=1}^{N} (y_n - \langle \boldsymbol{x}_n, \boldsymbol{w} \rangle)^2 + \lambda R(\boldsymbol{w}) \right), \end{aligned}$$

where λ is the parameter that weights the regularization term.

We then introduce two types of regularization as follows:

$$\begin{aligned} R(\boldsymbol{w}) &= \|\boldsymbol{w}\|_2^2 = \boldsymbol{w}^\top \boldsymbol{w} \qquad \text{L2 regularization} \\ R(\boldsymbol{w}) &= \|\boldsymbol{w}\|_1 = \sum_{k=1}^{K} |w_k|, \quad \text{L1 regularization} \end{aligned}$$

where either L2 or L1 regularizations are employed in actual task of regression and classification.

2.4.2 Expectation-Maximization Algorithm

When we consider various statistical machine translation models as a machine learning task, the task would be different from other tasks as it involves the derivation $\boldsymbol{d}$ as a latent variable in the course of the maximum likelihood estimation. In that case we apply the expectation-maximization algorithm (Dempster et al., 1977), EM algorithm in short, which is the iterative method consists of expectation and maximization steps.

Formally we have an observed data set $\mathcal{D} = \{\boldsymbol{x}_n | n = 1, \cdots, N\}$, and its latent variable $\boldsymbol{z}$ where $\boldsymbol{z}_n$ corresponds to $\boldsymbol{x}_n$. We estimate the likelihood L as:

$$\log L(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}) = \sum_{n=1}^{N} \log P(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta}),$$

where $\boldsymbol{\theta}$ represents a parameter vector. We then estimate the maximum likelihood as:

$$\boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log L(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}).$$

However, we cannot directly estimate $\boldsymbol{\theta}$ due to the latent variable $\boldsymbol{z}$. Instead we apply the iterative EM algorithm to the optimization of $P(\boldsymbol{x}|\boldsymbol{\theta})$ as we update $\boldsymbol{\theta}^{(t+1)}$ from $\boldsymbol{\theta}^{(t)}$ within the $t \in \mathbb{N}$-th iteration, considering the following facts:

$$\begin{aligned}
\frac{P(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})}{P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)})} &= \frac{P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}) P(\boldsymbol{x}|\boldsymbol{\theta})}{P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)})} \\
\log \frac{P(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})}{P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)})} &= \log \frac{P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta})}{P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)})} + \log P(\boldsymbol{x}|\boldsymbol{\theta}) \\
\log P(\boldsymbol{x}|\boldsymbol{\theta}) &= \log \frac{P(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})}{P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)})} - \log \frac{P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta})}{P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)})} \\
\log P(\boldsymbol{x}|\boldsymbol{\theta}) - \log P(\boldsymbol{x}|\boldsymbol{\theta}^{(t)}) &= \log P(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}) - \log P(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}^{(t)}) - \log \frac{P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta})}{P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)})},
\end{aligned}$$

where $\boldsymbol{\theta}^{(t)}$ represents the current parameter vector and $\boldsymbol{\theta}^{(t+1)}$ represents the next parameter vector.

We now maximize $P(\boldsymbol{x}|\boldsymbol{\theta})$ by maximizing $\log P(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})$ iteratively as follows:

$$\boldsymbol{\theta}^{(t+1)} = \underset{\theta}{\operatorname{argmax}} \sum_{\boldsymbol{z} \in Z} P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)}) \log P(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}).$$

In an actual iteration for the EM algorithm, we estimate $P(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})$ in the expectation step then update $\boldsymbol{\theta}^{(t+1)}$ in the maximization step. We randomly initialize the value of $\boldsymbol{\theta}^{(1)}$ beforehand.

2.4.3 Tuning

After training our translation model with the EM algorithm, we would like to combine it with the language model and keep balance between them. Therefore we apply the log linear model for tuning our system (Och, 2003):

$$\begin{aligned}\langle \hat{\boldsymbol{e}}, \hat{\boldsymbol{d}} \rangle &= \underset{\langle \boldsymbol{e}, \boldsymbol{d} \rangle}{\operatorname{argmax}} P(\boldsymbol{e}, \boldsymbol{d}|\boldsymbol{f}) \\ &= \underset{\langle \boldsymbol{e}, \boldsymbol{d} \rangle}{\operatorname{argmax}} \boldsymbol{w}^\top \boldsymbol{h}(\boldsymbol{e}, \boldsymbol{d}, \boldsymbol{f}),\end{aligned}$$

where $\boldsymbol{h}(\cdot)$ is a feature vector and $\boldsymbol{w}$ is its weight vector. We usually optimize this model with evaluation criteria such as BLEU, as we discuss in the following section.

2.5 Evaluation of Statistical Machine Translation

After developing a machine translation system, we would like to evaluate its output so that we can get to know relative improvement from a previous version and overall quality when compared to other systems including human translators. Evaluation on machine translation is conducted by human or machine. The former is called *manual evaluation* and the latter is called *automatic evaluation*. We also conduct meta evaluation between manual and automatic evaluations as there is a huge gap between them. In general manual evaluation is more acceptable, interpretable, and reasonable while automatic evaluation offers better cost-performance, reproducibility, and speed.

2.5.1 Manual Evaluation

Even manual evaluation of translation quality is far from a trivial problem, as there are many ways to translate a part of a sentence or just one word. For example, the English word “I” has various counter parts in Japanese such as “私”, “俺”, “吾輩”, and so on.

In widely used *subjective evaluation*, human evaluators who are native or very fluent speakers of the target language are asked to rate a translated sentence called *hypothesis*. Rating is given on a fixed measure like a scale of one to ten in terms of evaluation criterion such as *adequacy* and *fluency*. Adequacy focuses on expressiveness and faithfulness of translation. Fluency, in contrast, focuses on grammatical correctness and natural soundness of translated sentence.

However, even the criteria of adequacy and fluency may not be sufficient as human evaluators tend to be unreliable when they met ambiguous cases where evaluation criteria are indistinguishable to them. Therefore subjective evaluation usually involves two or more human evaluators who try to seek agreement on ambiguous cases as much as possible and normalize overall score afterward. In addition to the normalization process, *reference translation* made by professional human translators beforehand would be given as a reference, which may cause *reference bias* where human evaluators prefer a reference to a translated sentence for no reason.

When we need to analyze what kind of errors impacted adequacy and fluency, more complex criteria such as HTER (Snover et al., 2006) and a detailed analysis framework (Vilar et al., 2006) are used. While this human-in-the-loop evaluation process is regarded as the most reliable evaluation method to date, it is time consuming, extremely expensive, and usually unstable as the process involves human evaluators.

2.5.2 Automatic Evaluation

Therefore we instead utilize fully automatic evaluation methods for reducing overall turn around time especially during development. Automatic evaluation makes an assumption that translated sentences that are similar to reference translation are better translation. In order to reduce reference bias in that situation, some methods utilize more than one reference translation as multiple references.

Automatic evaluation methods differ in how to define such a similarity measure. We will see two widely used yet different measures: BLEU and RIBES.

BLEU

BLEU (Papineni et al., 2002) is the most popular measure that utilizes word n-gram based precision matches between hypothesis and reference, as hypothesis that matches to reference is regarded as similar.

Formally the BLEU measure is defined as:

$$\text{BLEU-}n = \exp \sum_{i=1}^{n} \frac{\log \text{precision}_i}{n} \times \text{BP}$$

$$\text{precision}_i = \frac{\text{matched } i\text{-gram length}}{\text{hypothesis } i\text{-gram length}}$$

$$\text{BP} = \min\left(1, \exp\left(1 - \frac{\text{reference length}}{\text{hypothesis length}}\right)\right),$$

where the brevity penalty (BP) reduces the score if hypothesis is too short. The maximum order $n \in \mathbb{N}$ for n-grams to be matched is typically set to 4. This setting is called BLEU-4.

We show how to calculate the score of BLEU-4 using the following example, which consists of an input sentence, its reference translation, and a hypothesis generated by a machine translation system:

Input: あなたと私は昨日ピザを食べました。

Reference: You and I ate a pizza yesterday .

Hypothesis: I ate a pizza today .

In this example, our hypothesis mistranslated the word "yesterday" in the reference as "today", and, at the same time, it lacks the translation of "You and", which appeared in the input as "あなたと". In other words, the hypothesis correctly translated 5 words out of the 8 words in the reference, including a comma counted as a word. Therefore, we calculate our unigram precision (precision_1) as:

$$\text{precision}_1 = \frac{\text{matched unigram length}}{\text{hypothesis unigram length}} = \frac{5}{6}.$$

Please note that we do not use the number of words in the reference explicitly, because we do calculate precision instead of recall.

We further calculate our bigram precision (precision_2), the matching between the bigram of the reference ("You and", "and I", ...) and that of the hypothesis ("I ate", "ate a", ...), as follow:

$$\text{precision}_2 = \frac{\text{matched bigram length}}{\text{hypothesis bigram length}} = \frac{3}{5}.$$

Furthermore, we calculate our trigram precision (precision_3) and 4-gram precision (precision_4) as:

$$\text{precision}_3 = \frac{\text{matched trigram length}}{\text{hypothesis trigram length}} = \frac{2}{4}$$
$$\text{precision}_4 = \frac{\text{matched 4-gram length}}{\text{hypothesis 4-gram length}} = \frac{1}{3}.$$

In order to calculate the score of BLEU, we need to calculate BP, in addition to the calculation of precision, as follow:

$$\begin{aligned}\text{BP} &= \min\left(1, \exp\left(1 - \frac{\text{reference length}}{\text{hypothesis length}}\right)\right) \\ &= \min\left(1, \exp\left(1 - \frac{8}{6}\right)\right) \\ &= \exp\left(-\frac{1}{3}\right) \approx 0.717\end{aligned}$$

After that, we can finally calculate the score of BLEU-4:

$$\begin{aligned}\text{BLEU-4} &= \exp\sum_{i=1}^{4}\frac{\log \text{precision}_i}{4} \times \text{BP} \\ &= \exp\left(\frac{1}{4}\log\frac{5}{6} + \frac{1}{4}\log\frac{3}{5} + \frac{1}{4}\log\frac{2}{4} + \frac{1}{4}\log\frac{1}{3}\right) \times \exp\left(-\frac{1}{3}\right) \\ &\approx 0.3850 = 38.50\%.\end{aligned}$$

RIBES

Of many variations and extensions that have been proposed to the BLEU measure, the RIBES measure (Isozaki et al., 2010a) focuses more on word order differences between hypothesis and reference. Thus hypothesis that matches to reference's word order is regarded as similar even when only a few words matches to reference, as opposed to BLEU. As a result this measure

correlates more to human subjective evaluation than the BLEU measure (Goto et al., 2011; Goto et al., 2013) on distant language pairs where word order extremely differs.

Given a word order list of the n words output $list$, this method utilizes the Kendall's rank correlation coefficient (Kendall, 1938), Kendall's τ, as follows:

$$\begin{aligned}
\text{RIBES} &= \frac{\text{Kendall's } \tau + 1}{2} \times \text{precision}_1^{\alpha} \times \text{BP}^{\beta} \\
\text{Kendall's } \tau &= \frac{count(\text{ascending pairs})}{count(\text{all pairs})} \times 2 - 1 \\
count(\text{ascending pairs}) &= \sum_{i,j \in [1,n], i<j} \delta(\text{list}_i < \text{list}_j) \\
count(\text{all pairs}) &= \binom{n}{2},
\end{aligned}$$

where $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$, and $\delta(\text{list}_i < \text{list}_j)$ is the Kronecker's delta function that returns 1 if $\text{list}_i < \text{list}_j$ and 0 otherwise. The hyperparameters α and β are typically set to $\alpha = 0.25$ and $\beta = 0.10$.

We show how to calculate RIBES by recalling the "pizza" example as:

Reference: You[1] and[2] I[3] ate[4] a[5] pizza[6] yesterday[7] .[8]

Hypothesis: I[3] ate[4] a[5] pizza[6] .[8]

where we iteratively assigned integer to each word in the reference and the hypothesis, while eliminating the word "today" that did not appear in the reference.

In order to calculate the score of RIBES, we calculate Kendall's τ, by counting the number of ascending pairs in the hypothesis, as follow:

$$\begin{aligned}
\text{Kendall's } \tau &= \frac{count(\text{ascending pairs})}{count(\text{all pairs})} \times 2 - 1 \\
&= \frac{20}{20} \times 2 - 1 = 1.0.
\end{aligned}$$

After that, we calculate the score of RIBES using Kendall's τ, our unigram precision, and BP:

$$\begin{aligned}\text{RIBES} &= \frac{\text{Kendall's } \tau + 1}{2} \times \text{precision}_1^{\alpha} \times \text{BP}^{\beta} \\ &= 1.0 \times \left(\frac{5}{6}\right)^{0.25} \times \left(-\frac{1}{3}\right)^{0.10} \\ &\approx 0.9241 = 92.41\%.\end{aligned}$$

Chapter 3

Exploring Roles of Syntactic Information in Rule-based Preordering

In this chapter, we explore roles of syntactic information in rule-based preordering, by presenting two rule-based preordering methods for Japanese-to-English statistical machine translation, where the two-stage method (Hoshino et al., 2013) described in Section 3.2 uses syntax extensively and the three-stage method (Hoshino et al., 2014) described in Section 3.5 uses little or no syntax. Our methods addressed syntactic problems raised in previous studies (Komachi et al., 2006; Katz-Brown and Collins, 2008) on the task of rule-based preordering for Japanese-to-English translation. These problems have prevented rule-based methods from achieving accurate reordering and translation. By solving them, both methods outperformed previous methods, and as a result, we achieved the state-of-the-art performance in Japanese-to-English translation to date using a rule-based preordering method.

3.1 Motivation of Rule-based Preordering

The task of preordering, and natural language processing in general, has been divided into two major paradigms: a rule-based approach and a statistical approach. The rule-based approach (Collins et al., 2005) has many advantages, such as being easy to understand from a traditional point of view and not requiring training data. Yet the effort required for its development makes

it less appealing than the statistical approach (Xia and McCord, 2004) in terms of cost performance. The latter is easy to maintain, robust, and, more importantly, ready to scale with big data. That is why statistical methods dominate natural language processing today.

However, our data size is not always large enough. Or even worse, the data we want for a particular task may not be obtainable due to copyright issues. In that case, we have to come up with the old-school rule-based approach that does not require training data. Nonetheless, it is fully possible to achieve high accuracy comparable with statistical methods using rule-based methods.

As such, we demonstrate various techniques to improve rule-based preordering methods for Japanese-to-English statistical machine translation, which is one of the most difficult translation direction in the world in terms of word ordering. We first address problems raised in the previous studies (Komachi et al., 2006; Katz-Brown and Collins, 2008) on the task of rule-based predering for the Japanese-to-English translation with the syntax-based two-stage method (Hoshino et al., 2013). We then address the robustness of the two-stage method with the three-stage method that uses little or no syntax (Hoshino et al., 2014).

3.2 Two-Stage Preordering with Predicate-Argument Structure

We improve previous rule-based preordering methods (Komachi et al., 2006; Katz-Brown and Collins, 2008) by combining dependency structure with predicate-argument structure. This can be easily done as most predicate-argument analyzers take dependency structure as input and annotate predicate-argument structure on top of the dependency as output, as shown in our example in Figure 3.1. Then, by using our combined predicate-argument structure, we reorganized preordering rules and apply them in two-stages: an inter-chunk stage and an intra-chunk stage. Operations in the inter-chunk stage are responsible for the orders of chunks, whereas operations in the intra-chunk stage focus on the word orders inside chunks.

Let us explain the notations used in our explanations before we offer detailed explanation of the rules. The "$\rightarrow$" symbol in ($L \rightarrow R$) represents a rewrite operation over a head and its

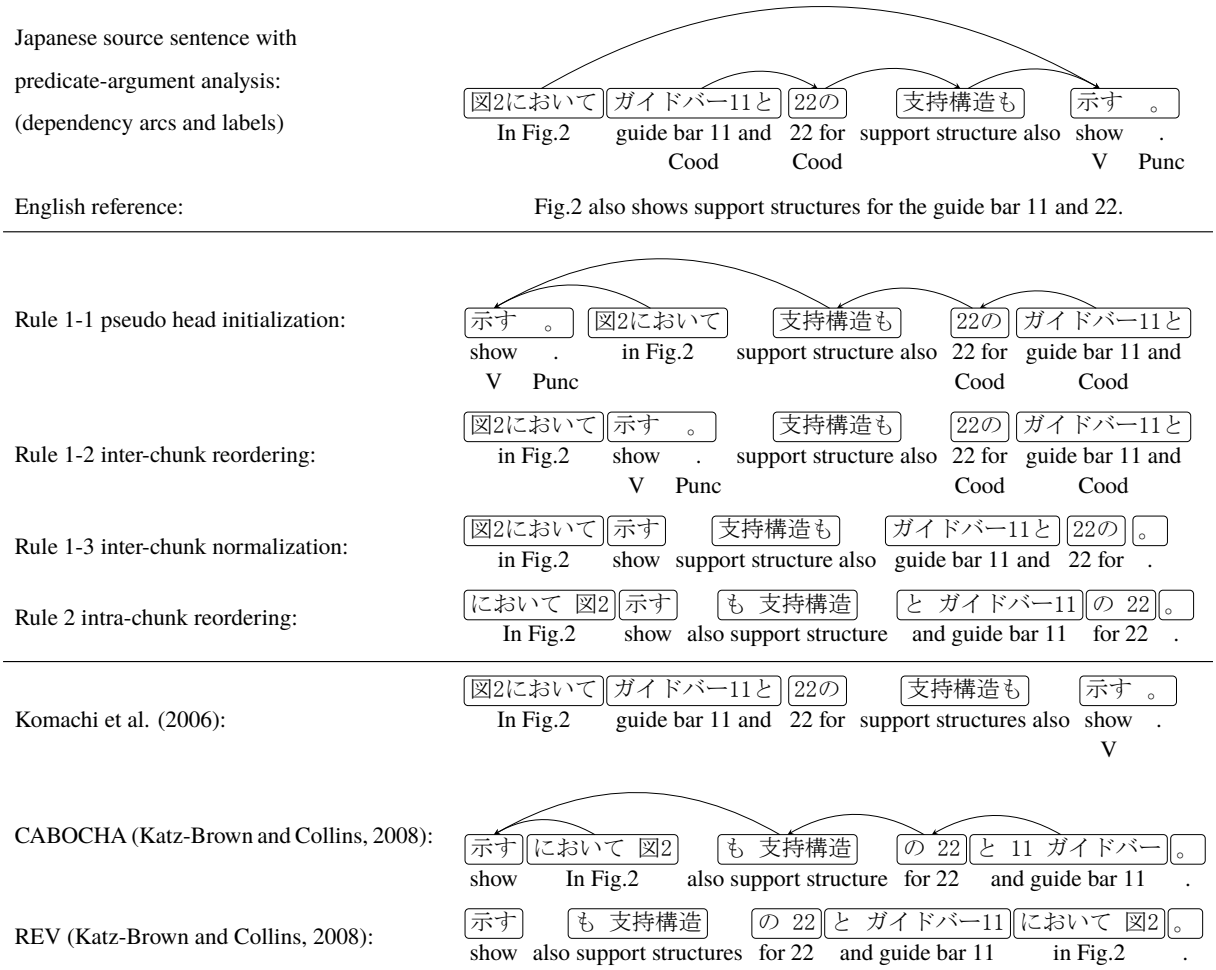

Figure 3.1: Preordering example along with predicate-argument structure in Japanese, comparing the two-stage method with the previous methods. Each box represents a chunk. The labels Cood, V, and Punc denote a chunk coordination, a head verb, and a Japanese punctuation mark, respectively.

dependency, indicating symbols on the left hand side (L) are converted to symbols on the right hand side (R). We apply pattern matching to both left and right hand sides. Each symbol in the rules including x, y, and z represents a chunk. The Kleene star such as $x^\star$ represents zero or more chunks. Below are the main preordering rules in our system:

3.2.1 Rule 1-1 Pseudo Head Initialization

In order to convert the SOV word order in Japanese into the head-initial word order similar to English, this rule applies pseudo head-initialization sorting to the hierarchical dependency structure in the predicate-argument structure, resulting in yielding the VSO word order. After parsing an input sentence with a predicate-argument analyzer, the rule modifies the order of chunks as each head chunk is always followed by its dependent child chunks. That reordering operation is specifically done by traversing the dependency tree inside the predicate-argument structure analyzed for the input sentence in pre-order[1].

For instance, the head verb "示す show(s)" in our example is relocated to the leftmost position after applying this rule.

3.2.2 Rule 1-2 Inter-chunk Reordering

If there is a head verb represented with the V label in a sentence, we then apply this rule after Rule 1-1. This rule coverts the SOV word order into the SVO word order in English, even when no explicit subject or object is placed in a sentence. That happens as a result of a parsing error, or a correct analysis of a pro-drop construction that is often the case in the pro-drop Japanese language.

Basically we move the head verb V instead of the subject S or the object O, as we already have the head-initial VSO word order generated by Rule 1-1. Specifically we relocate V immediately after S ($\mathrm{V}x^\star\mathrm{S}y^\star \rightarrow x^\star\mathrm{SV}y^\star$) or immediately before O in case no subject is found ($\mathrm{V}x^\star\mathrm{O}y^\star \rightarrow x^\star\mathrm{VO}y^\star$).

Instead, when we only have the head verb V without the subject S and the object O, we relocate

[1]Do not confuse with preordering. This is a way to traverse a tree structure.

V immediately before the rightmost chunk (V$x^\star y \rightarrow x^\star$V$y$) to avoid yielding the VSO word order directly.

For instance, the head verb "示す show(s)" in our example is relocated to next to the leftmost position after applying this rule, generating a pseudo SVO construction regardless of no subject chunk in the sentence. In contrast, when tested within the previous methods (Komachi et al., 2006; Katz-Brown and Collins, 2008), the V is incorrectly relocated to the leftmost or the rightmost, resulted in yielding incorrect VSO or SOV word orders.

3.2.3 Rule 1-3 Inter-chunk Normalization

If there are coordinated chunks or punctuation in a sentence, we apply this rule after Rule 1-1 and Rule 1-2. This rule fixes incorrect shuffling of coordinated chunks or punctuation often generated by pseudo head-initializing sorting in Rule 1-1. Basically we keep these chunks unchanged from their original positions, specifically by relocating the coordinated chunks to the leftmost position ($x^\star$Cood$^\star y^\star \rightarrow$ Cood$^\star x^\star y^\star$) where Cood in the right hand side represents the coordinated chunks sorted in the original order, and the punctuation to the rightmost position ($x^\star$Punc$^\star y^\star \rightarrow x^\star y^\star$Punc*) where Punc represents one of the Japanese punctuation "、" and "。". An exceptional case is when we have the comma "、" followed by the period "。" after the above. In that case we remove all commas immediately before the period.

For instance, the period "。" in our example sentence is relocated to the rightmost position, unlike Komachi et al. (2006). And the coordinated chunks "ガイドバー11と22の the guide bar 11 and 22" restored their original order during the reordering operations. Neither Komachi et al. (2006) or Katz-Brown and Collins (2008) comes with such a rule for handling coordinated chunks.

3.2.4 Rule 2 Intra-chunk Reordering

We apply this rule after Rule 1-1, Rule 1-2, and Rule 1-3. This rule converts postpositional phrases in Japanese into prepositional phrases in English. Specifically we swap function words and content words in each chunk (Content Function $\rightarrow$ Function Content) where both Content

and Function labels are given to every word by the predicate-argument analyzer, indicating a content word and a function word, respectively.

For instance, the chunk "ガイドバー11と the guide bar 11 AND" in our example has three words: the two content words "ガイドバー11 the guide bar 11" and the function word "と AND". Thus the words in the chunk are reordered as "と ガイドバー11 AND the guide bar 11". In contrast, Katz-Brown and Collins (2008) naively reversed all the three words, yielding "と 11 ガイドバー AND 11 the guide bar".

3.3 Experimental Settings of Two-Stage Preordering

In order to test the two-stage method by comparing it with previous methods in the same settings, we conduct several Japanese-to-English statistical machine translation experiments. For all these experiments, we set up a standard statistical machine translation system that consists of SRILM 1.7.0 (Stolcke, 2002) for 6-gram language modeling, MGIZA 0.7.3 (Gao and Vogel, 2008) with the grow-diag-final-and heuristic for obtaining many-to-many word alignment, and Moses 0.91 (Koehn et al., 2007) with the minimum error rate training (Och, 2003), and the lexicalized reordering model (Tillman, 2004) for tuning and decoding.

We train a standard statistical machine translation system with two different corpora: one in the patent domain and another in the news domain. It is the first one that we used mainly. For the patent domain we utilize several English-Japanese portions of NTCIR patent corpora. They contain in total more than 3.2 million parallel sentences in English and Japanese, making a reasonably large parallel corpus for our experiments. Basically following the NTCIR-9 (Goto et al., 2011) workshop configuration, we employed the NTCIR-7 and NTCIR-8 training sets, the NTCIR-8 development set, and the NTCIR-9 test set (*test9*) in our experiments. We utilized all sentences from the training and test data, while we only used the first 500 sentences from the development set for faster development. For the news domain we utilize the relatively small JENAAD corpus (Utiyama and Isahara, 2003), which consists of exactly 150,000 parallel sentences in English and Japanese. We split the corpus into three parts: the first 1,000 sentences

for testing, the following 1,500 sentences for development, and the remaining 148,500 sentences for training, respectively.

To evaluate our system outputs, we judge the quality of global word ordering in RIBES (Isozaki et al., 2010a) and local word ordering in BLEU (Papineni et al., 2002).[2] Both metrics represent the similarity between system outputs and human reference translations as a score range between $[0, 100]\%$ where a high score indicates similarity and a low score indicates dissimilarity.

We explored two parser configurations for parsing Japanese sentences:

1. The CaboCha+SynCha configuration employed MeCab 0.994 (Kudo et al., 2004) within the default IPA dictionary for tokenization, CaboCha 0.65 (Kudo and Matsumoto, 2002) for dependency parsing, and SynCha 0.3 (Iida and Poesio, 2011) for predicate-argument structure analysis on top of the CaboCha output.

2. The KNP configuration employed JUMAN 7.0 within the default JUMAN dictionary for tokenization and KNP 4.01 (Kawahara and Kurohashi, 2006b; Sasano and Kurohashi, 2011) for combined dependency parsing and predicate-argument structure analysis. The tokenization standard for this configuration is slightly different from that of the CaboCha+SynCha configuration.

In a preprocessing step prior to statistical machine translation training or decoding, we tokenized and parsed all the Japanese sentences with both parser configurations. Then we ran preordering methods by feeding the parsed sentences. We employed our rule-based method, an implementation of the statistical method proposed by Neubig et al. (2012) called lader 0.1.3[3], and faithful implementations of previous rule-based methods including Komachi et al. (2006), CABOCHA (Katz-Brown and Collins, 2008), and REV (Katz-Brown and Collins, 2008), except that we relocated V instead of S and O in the case of Komachi et al. (2006). We basically applied the CaboCha+SynCha configuration for all the preordering methods and our baseline

[2]We ran the Travatar toolkit (Neubig, 2013) in default settings for both evaluation metrics and their statistical testings.

[3]Only 10,000 lines sampled from a training data set are used for training when applying this statistical method due to its computational complexity that consumed 120 GB of memory space for almost an entire month.

Table 3.1: Results in the Japanese-to-English patent translation. Bold text denotes either the highest score or the insignificance from the highest ($p < 0.01$) in bootstrap resampling (Koehn, 2004b).

		test9	
Method	DL	RIBES	BLEU
Moses (our baseline without preordering)	20	68.48	29.19
CABOCHA (Katz-Brown and Collins, 2008)	20	66.15	27.74
Komachi et al. (2006)	10	69.10	29.58
lader (Neubig et al., 2012) with 10k data	10	70.15	29.93
REV (Katz-Brown and Collins, 2008)	10	**72.34**	30.39
Two-stage method with CaboCha+SynCha	10	**72.35**	30.01
Two-stage method with KNP	10	**72.26**	30.65
Two-stage method with KNP and refinements	10	**72.05**	**31.49**

without applying any preordering methods. We also applied the KNP configuration only for the two-stage method.

3.4 Experimental Results of Two-Stage Preordering

Table 3.1 shows a comparison of translation accuracy in the patent domain obtained before and after applying the following preordering methods: Komachi et al. (2006), CABOCHA and REV proposed by Katz-Brown and Collins (2008), Neubig et al. (2012), and our two-stage method Hoshino et al. (2013). Moses indicates our baseline results attained without applying any preordering methods. Refinements for the two-stage method indicate the results obtained after fixing a bug in detecting predicate-argument structure.

The two-stage method (Hoshino et al., 2013) with refinements outperformed all the other preordering methods in terms of local word ordering in BLEU, whereas it outperformed all the previous methods except the REV (Katz-Brown and Collins, 2008) in terms of global word or-

Table 3.2: Ablation tests of the two-stage method (Hoshino et al., 2013) with the KNP configuration. Bold text denotes either the highest score or the insignificance from the highest ($p < 0.01$) in bootstrap resampling (Koehn, 2004b).

Rule 1-2	Rule 1-3	Rule 2	RIBES	BLEU
			68.48	29.19
√			71.00	29.76
	√		69.50	27.71
		√	65.61	28.29
	√	√	70.40	28.84
√		√	71.34	**30.94**
√	√		71.74	**30.41**
√	√	√	**72.26**	**30.65**

dering in RIBES. These very strong results suggest that our proposed method achieved the most accurate word ordering along with syntactic structure than the previous methods with syntax (Komachi et al., 2006; Katz-Brown and Collins, 2008) and little or no syntax (Katz-Brown and Collins, 2008).

As shown in 3.2, we also conduct ablation tests of our preordering rules with the KNP configuration to check contributions of each rule, which compared all the possible combinations of Rule 1-2, Rule 1-3, and Rule 2 in terms of translation accuracy for the same Japanese-to-English patent translation. From these tests we observed incremental performance gain when adding any rules or a combination of rules, combining all the rules performed the best. An exceptional case is only when adding Rule 2 to the barebone baseline, which is intuitively a straightforward phenomenon as swapping content words and function words without any chunk reordering makes no sense.

In addition to the translation accuracy, we conducted an intrinsic evaluation of the two-stage method with the Kendall's τ distribution (Isozaki et al., 2010a), which represents the mono-

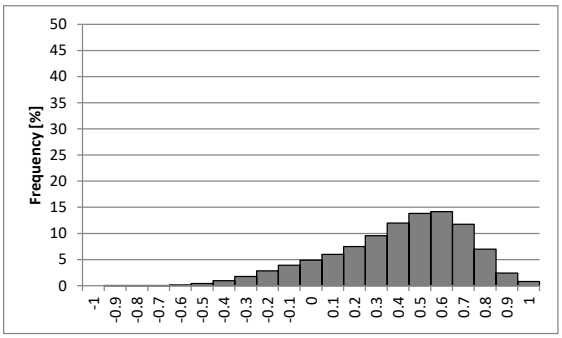

Figure 3.2: The development data in the patent domain before applying preordering: average $\tau = 0.391$

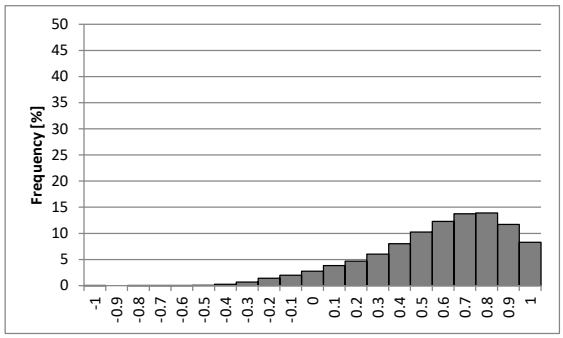

Figure 3.3: The development data after applying the two-stage method with KNP: average $\tau = 0.575$

Table 3.3: Experiment results in the Japanese-to-English news translation. Bold text denotes either the highest score or the insignificance from the highest ($p < 0.01$) in bootstrap resampling (Koehn, 2004b).

Method	DL	RIBES	BLEU
Moses	20	62.71	15.03
Two-stage method with KNP	10	**69.30**	**16.12**

tonicity of sentences as a graph that has values of the Kendall's τ measure in the x-axis and percent ratios of every sentence in the y-axis, with the averaged Kendall's τ of all sentences, shown in Figure 3.2 and 3.3. This evaluation suggests that our method generated much more monotonic sentences (Figure 3.3) than the baseline without preordering (Figure 3.2), and most of our reordering operations resulted in positive improvement of word ordering.

Table 3.3 shows experiment results in the news domain that compare translation accuracy obtained before and after applying the two-stage method (Hoshino et al., 2013) with the KNP configuration. Similar to the results in the patent domain, our method slightly improved translation accuracy in both RIBES and BLEU in the news domain, indicating that our rule-based method works robustly in any domains regardless of the relatively small data used for training the

statistical machine translation system. We can largely benefit from this robustness as practical statistical machine translation systems often involve small data in various domains.

3.5 Three-Stage Preordering without Predicate-Argument Structure

Since previous rule-based preordering methods for Japanese-to-English statistical machine translation (Komachi et al., 2006; Hoshino et al., 2013) heavily depend on the accuracy of complex predicate-argument structure analysis, parsing errors often become the direct cause of reordering errors.

To address this, Katz-Brown and Collins (2008) proposed the REV preordering method that naively reverses parts of Japanese sentences by using little or no syntax information, making it immune to parsing errors. This method, however, mistakenly reverses word orders of compound nouns and coordinations as the method lacks any clues about these syntactic constructions. Isozaki (2013) slightly improved the REV's shortcoming on compound nouns by labeling them with part-of-speech tags, yet the problem with coordination still remains.

We therefore propose a new set of preordering rules for Japanese-to-English statistical machine translation. Our idea is that we mimic the fully syntax-based two-stage method (Hoshino et al., 2013) by using little or no syntax information as Katz-Brown and Collins (2008) did, while we keep track of problematic coordination structures. As shown in Figure 3.4, we apply our rules in three-stages: In the first stage, we split a sentence into several chunks while keeping track of coordination; In the second stage we reorder the chunks; and in the final third stage we reorder the words inside the chunks. We make our system immune to parsing errors by utilizing syntactic information only at the first stage.

Before we offer detailed explanation of the rules, let us explain the notations used in our explanations. Our input sentence, called $input$, consists of $l \in \mathbb{N}$ chunks ($input = c_1...c_l$); each chunk c_x $(1 \leq x \leq l)$ consists of $q \in \mathbb{N}$ words ($c = w_1...w_q$); and the "$\rightarrow$" symbol in $(L \rightarrow R)$ represents a rewrite operation, indicating symbols on the left hand side (L) are converted to symbols on the right hand side (R).

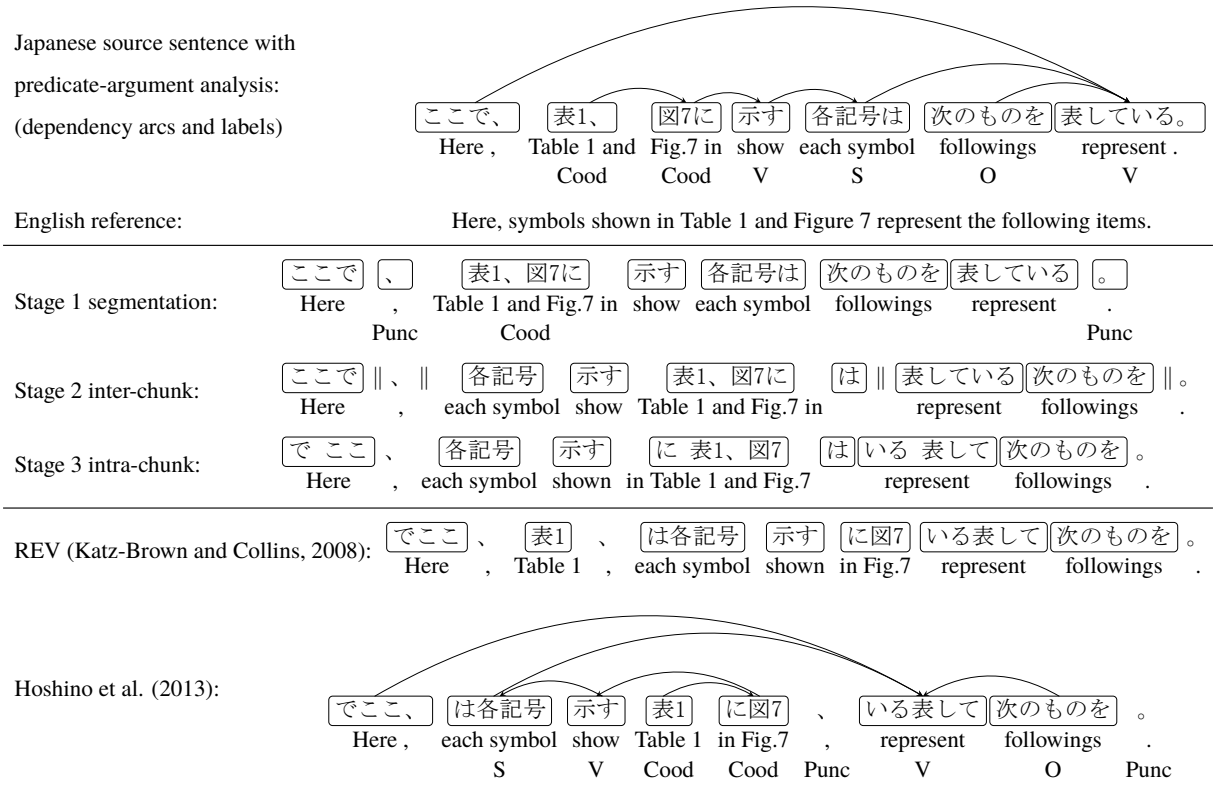

Figure 3.4: Another preordering example along with predicate argument structure, comparing the three-stage method and the previous methods including the two-stage method. The labels Cood, V, S, O, and Punc denote a chunk coordination, a head verb, a subject, an object, and a Japanese punctuation mark, respectively. Each symbol ‖ represents a boundary between two segments of chunks.

3.5.1 Stage 1 Segmentation

This rule splits a Japanese sentence into several chunks using syntactic information. To keep track of coordination in latter stages, we regard all the sequential chunks in $input$ that are linked together within coordination as one chunk ($input = c_1...c_l \rightarrow c_1...c_m$). Thus number of chunks in $input$ is reduced from l to m where $1 \leq m \leq l$.

After that we separate punctuation from each chunk as an individual chunk, preventing it from being shuffled with other chunks in the next inter-chunk reordering stage. Specifically we separate $u \in \mathbb{N}$ punctuation from these chunks, making each punctuation a new chunk ($input = c_1...c_m \rightarrow c_1...c_n$). The number of chunks in $input$ is increased from m to n where $n = m + u$ as long as any punctuation exist.

For instance, the two coordinated chunks "表1、 Table 1 ," and "図7に Fig.7 in" in our example are merged into the one chunk "表1、図7に Table 1 and Fig.7 in". Also two punctuation chunks "、 ," and "。 ." are separated from other chunks. These operations make reordering of chunks much easier and more accurate than previous works (Komachi et al., 2006; Katz-Brown and Collins, 2008; Hoshino et al., 2013).

Nonetheless, in the case of any coordinated chunks c_x where $1 \leq x \leq m$, we apply the following three exceptional rules for balancing the segmentation between coordination and punctuation:

1. If a noun, the word that predicate-argument analysis tagged as the part-of-speech "NN", is followed by punctuation in c_x, we do not split this chunk at that punctuation. For instance, we do not split the merged coordinated chunk "表1、図7に Table 1 and Fig.7 in" at punctuation following this exception.

2. If the first exception is applied and the chunk c_x ends with one of the Japanese topic words "は" or "が", we regard the rightmost punctuation in the chunk as a new chunk, splitting the chunk c_x into three chunks at that punctuation, regardless of the first exception. For instance, when we happened to have a coordinated chunk like "表1、図7は Table 1 and Fig.7", we split this chunk at punctuation, making the last fragment more similar to an individual subject chunk

than a coordinated noun clause.

3. If this chunk c_x ends with one of the Japanese punctuation "、" and "。", we split the chunk only at that punctuation, making it into two chunks regardless of the first and second exceptions. For instance, when we happened to have a coordinated chunk like "表1、図7、 Table 1 and Fig.7 ,", we split this chunk only at the last punctuation, leaving the former part including a punctuation mark as a merged coordinated chunk.

3.5.2 Stage 2 Inter-Chunk Reordering

This rule reorders each chunk segment surrounded by punctuation from the SOV word order to the SVO word order similar to English with a hard assumption that a subject always ends with a Japanese topic word, keeping punctuation left untouched. From the chunks we have $c_1...c_n$, we scan a chunk segment surrounded by punctuation $c_i...c_j$ where $1 \leq i \leq j \leq n$, $c_{i-1} = \text{Punc} \iff 1 < i$, and $c_{j+1} = \text{Punc} \iff j < n$. We find a chunk $c_t = w_1...w_q$ ($i \leq t \leq j$) that ends with one of the Japanese topic words "は" or "が" from each segment $c_i...c_j$.

After that we reorder each segment $c_i...c_j$, specifically by reversing $c_i...c_t$ and $c_{t+1}...c_j$ separately ($c_i...c_j \rightarrow c_t...c_ic_j...c_{t+1}$). We then separate the topic word w_q from the chunk c_t, placing it between the two reversed fragments ($c_t...c_ic_j...c_{t+1} \rightarrow w_1...w_{q-1}c_{t-1}...c_iw_qc_j...c_{t+1}$).

For instance, we regard the three chunks "表1、図7に Table 1 and Fig.7 in", "示す show", and "各記号は each symbol" in our example as one segment, because this segment ends with the topic word "は". We then reverse the order of these chunks while leaving the topic word in the end, yielding the chunk sequence of "各記号 each symbol", "示す show", "表1、図7に Table 1 and Fig.7 in", and "は".

3.5.3 Stage 3 Intra-Chunk Reordering

This rule converts postpositional phrases in Japanese into prepositional phrases similar to English. We split each chunk $c_k = w_1...w_q$ from all the segments $c_i...c_k...c_j$ into content words $w_1...w_p$ and function words $w_{p+1}...w_q$ where $0 \leq p \leq q$. We then swap the two fragments and

concatenate the two fragments while reversing the function words ($c = w_1...w_q \rightarrow w_q...w_{p+1}w_1...w_p$).

For instance, the chunk "ここで" in our example is split into the content word "ここ" and the function word "で". We therefore swap the two words, yielding the sequence "で ここ". If there are more than one function word such as a chunk "ここではね" containing two function words "では" and "ね", we also reverse the order of function words during the swap operation, yielding the final sequence of "ね では ここ".

3.6 Experimental Settings of Three-Stage Preordering

In a similar way as the previous experimental settings for the two-stage preordering method, we conduct Japanese-to-English statistical machine translation experiments for comparing the three-stage method with previous methods. We basically follow the same settings used in the previous patent domain experiments. Some of the tools we use are updated (Moses 1.0 and KNP 4.1 beta). We also do some training data filtering this time. We set up a standard statistical machine translation system that consists of SRILM 1.7.0 (Stolcke, 2002) for 6-gram language modeling, MGIZA 0.7.3 (Gao and Vogel, 2008) with the grow-diag-final-and heuristic for obtaining many-to-many word alignment, and Moses 1.0 (Koehn et al., 2007) with the minimum error rate training (Och, 2003), and the lexicalized reordering model (Tillman, 2004) for tuning and decoding.

We trained the standard statistical machine translation system only with several English-Japanese portions of NTCIR patent corpora. They contain in total more than 3.2 million parallel sentences in English and Japanese. We employed the NTCIR-7 and NTCIR-8 training sets, the NTCIR-8 development set, and the NTCIR-9 test set (*test9*). We utilized all sentences from the training and test data, while we only used the first 500 sentences from the development set for faster development. We also filtered out training sentences that are longer than 64 words.

To evaluate our system outputs, we judge the quality of global word ordering in RIBES (Isozaki et al., 2010a) and local word ordering in BLEU (Papineni et al., 2002).[4] Both met-

[4]We ran the Travatar toolkit (Neubig, 2013) in default settings for both evaluation metrics and their statistical testings.

Table 3.4: Results in the Japanese-to-English patent translation. Bold text denotes either the highest score or the insignificance from the highest ($p < 0.01$) in bootstrap resampling (Koehn, 2004b).

		test9		
Method	DL	RIBES	BLEU	τ
Moses	10	68.08	27.57	0.3935
REV (Katz-Brown and Collins, 2008)	10	73.10	29.87	0.5186
Two-stage method	10	72.37	30.56	0.5829
Three-stage method	10	**74.14**	**31.14**	0.6091

rics represent the similarity between system outputs and human reference translations as a score range between $[0, 100]\%$ where a high score indicates similarity and a low score indicates dissimilarity.

In all these experiments, we stick to the same parser configuration: JUMAN 7.0 with KNP 4.1 beta (Kawahara and Kurohashi, 2006b; Sasano and Kurohashi, 2011) for combined dependency parsing and predicate-argument structure analysis.

3.7 Experimental Results of Three-Stage Preordering

Table 3.4 shows a comparison of translation accuracy obtained before and after applying the following preordering methods: the REV method (Katz-Brown and Collins, 2008), the two-stage method (Hoshino et al., 2013), and the three-stage method (Hoshino et al., 2014). Moses indicates our baseline results attained without applying any preordering methods.

The three-stage method outperformed all the other methods including the two-stage method (Hoshino et al., 2013) in both RIBES and BLEU as well as the averaged Kendall's τ, indicating that we achieved the most accurate word ordering in terms of both global and local word orders. These substantial results suggest that we achieved the state-of-the-art translation accuracy with a rule-based preordering method to date.

Japanese input	··· | エネルギーが | 240keV、 | ドーズ量が | 4×1012/cm2 | ···
Ideal reordering	··· | エネルギーが240 keV | 、 | ドーズ量が4×1012/cm2 | ···
English reference	··· the energy of 240 keV, the dose of 4×1012/cm2 ···
Hoshino et al. (2014)	
Stage 1	··· | エネルギーが | 240 keV、ドーズ量が | 4×1012/cm2 | ···
Stage 2	··· | 4×1012/cm2 | 240keV、ドーズ量が | エネルギーが | ···
Stage 3	··· | 4×1012/cm2 | が 240 keV 、 ドーズ 量 | が エネルギー | ···

Figure 3.5: Typical reordering errors generated from applying the three-stage method. Each symbol | represents a boundary between chunks.

Nevertheless, we are still far from achieving perfect reordering in the Japanese-to-English preordering task, as our averaged Kendall's τ shows the value around 0.6 instead of 1.0. Therefore we conducted detailed analysis of reordered sentences for finding what is the cause of typical reordering errors from our actual mistakes.

3.8 Analysis of Typical Reordering Errors in Three-Stage Preordering

Figure 3.5 shows a typical cause of reordering errors that was found harmful to all the three-stages we have. This example sentence is full of coordinated chunks containing one punctuation mark in the middle, which corresponds to the English conjunct "and". Our segmentation rules in the first stage, however, failed to split these coordinated chunks into the two chunks bridged by the punctuation as similar to English. Instead our rules split these chunk at wrong points following the first exception rule, turning them into the three chunks. Both the second and third stages suffered from that mistake in the aftermath, by mistakenly shuffling the chunks in the second inter-chunk reordering stage and the words in the third intra-chunk reordering stage, respectively.

Now we face a serious problem in handling complex reordering errors of this kind within the rule-based approach. We have to add tons of rules to fix all these reordering errors, but adding

more and more rules rapidly increases complexity of our method. As history shows, we will eventually lose this fight long before we can achieve the perfect accuracy.

Alternatively one can actually attribute the cause of this problem to parsing errors, because the problem we have is how to obtain the ideal structure for reordering than just how to manipulate such a structure. To this end, we pose a following question: *Can we parse a sentence in a way suitable for reordering?* We will study such a research question within the statistical approach to overcome the difficulty of the rule-based approach in the next chapter.

3.9 Related Work

The first set of rules for the Japanese-to-English preordering task is proposed by Komachi et al. (2006). They employed predicate-argument structure on top of chunks to determine subject, verb, and object in the first place for effectively reordering the subject-object-verb (SOV) word order in Japanese into the subject-verb-object (SVO) word order in English in their method, specifically by relocating the subject before the verb and the object after the verb. While their linguistic typology motivated method works pretty well for simple matrix sentences where the triples of subject, verb, and object appear sequentially, it loses its charms when we happened to have a more complex construction such as coordination or subordinate clause as the method heavily relies on the relocations around the single verb. For instance, their method does not relocate chunks in our example sentence at all, because the sentence has no subject or object. That is often the case as Japanese is a pro-drop language.

After that Katz-Brown and Collins (2008) proposed another set of rules for the Japanese-to-English preordering task called CABOCHA. These rules reorder sentences along with hierarchical dependency structure on top of chunks. Basically they relocate head verbs instead of subjects and objects unlike Komachi et al. (2006), specifically by relocating a head verb after the rightmost subject or before the leftmost object, making the whole structure similar to the subject-verb-object construction. Otherwise they relocate a head before its all dependent child chunks for yielding head-initial sequences. In addition to reordering of chunks, they reverse the word

order within each chunk. Punctuation is left untouched from its original position. For instance, their rules successfully relocated the verb chunk from the rightmost position to the leftmost position in our example sentence. That leftmost position is still not perfect as the sentence lacks an explicit subject to the method. Another shortcoming of their method is that they determine which word is the head verb, the subject, and the object based on surface forms, instead of using more reliable predicate-argument analyzer as Komachi et al. (2006) did.

In addition to CABOCHA, Katz-Brown and Collins (2008) proposed yet another set of rules called REV. They took an unique approach that utilizes little or no syntactic information during reordering operations. Basically this method converts the SOV sequence into the SVO sequence by reversing the former S part and the latter OV part separately. Specifically this method takes three steps: we split a Japanese sentence into segments at punctuation; again we split each segment into new two segments at a Japanese topic marker "は"; and we naively reverse the word orders of each segment separately, regardless of what is inside. For instance, their rules naively reversed the word order of our example sentence including the coordinated chunk except the last punctuation, as the sentence lacks any topic markers.

3.10 Summary

We presented two rule-based preordering methods (Hoshino et al., 2013; Hoshino et al., 2014) for the Japanese-to-English statistical machine translation. We addressed major syntactic problems raised in previous studies, which involved accurate reordering of coordinated chunks along with predicate-argument structure in Japanese. We changed our strategy in the course of our development, from completely relying on reordering operations over syntactic input in retrospect, to minimal syntactic operations that abstract the information we needed for our reordering task. We eventually developed a rule-based preordering method that demonstrated the state-of-the-art performance for the Japanese-to-English preordering task to date.

Our finding is that syntactic information plays an important role in effectively generalizing our rules, as one of our methods showed robustness in domain difference while handling relatively

small data for training a statistical machine translation system. We are still curious how far we can go forward within this syntax-based direction beyond the rule-based approach. We will see what happens with the statistical approach in the following chapter.

Chapter 4

Statistical Preordering with Inversion Transduction Grammar

In this chapter, we propose a simple yet effective syntax-based preordering method, which improves a previously proposed preordering method (Li et al., 2007) that uses binary parse trees as a source-side syntactic structure. That is, we parse an input sentence into a binary constituent tree using an off-the-shelf source-side constituent parser. Each node in the binary parse tree is then binary classified, i.e., either monotone or reversed, using a linear support vector machine as a binary classifier. The tree reordered according to the classified labels is used to yield a reordered source sentence, which is fed to a standard statistical machine translation system to generate translation.

Our proposal improves the previous method (Li et al., 2007) in two ways: the use of theoretically grounded oracle reordering labels and effective features. Specifically, we introduce a novel procedure to obtain oracle reordering labels so as to maximize Kendall's τ (Kendall, 1938), as an alternative to a heuristic procedure used in the previous method. In particular, we show that oracle reordering labels obtained with the proposed procedure will lead to optimal sentence-wise reordering judged by Kendall's τ, owing to its compositional property previously discussed in Yang et al. (2012) and Neubig et al. (2012). Another improvement is a novel set of features that directly captures the syntactic relation in each node, which is obtained as a result of our feature

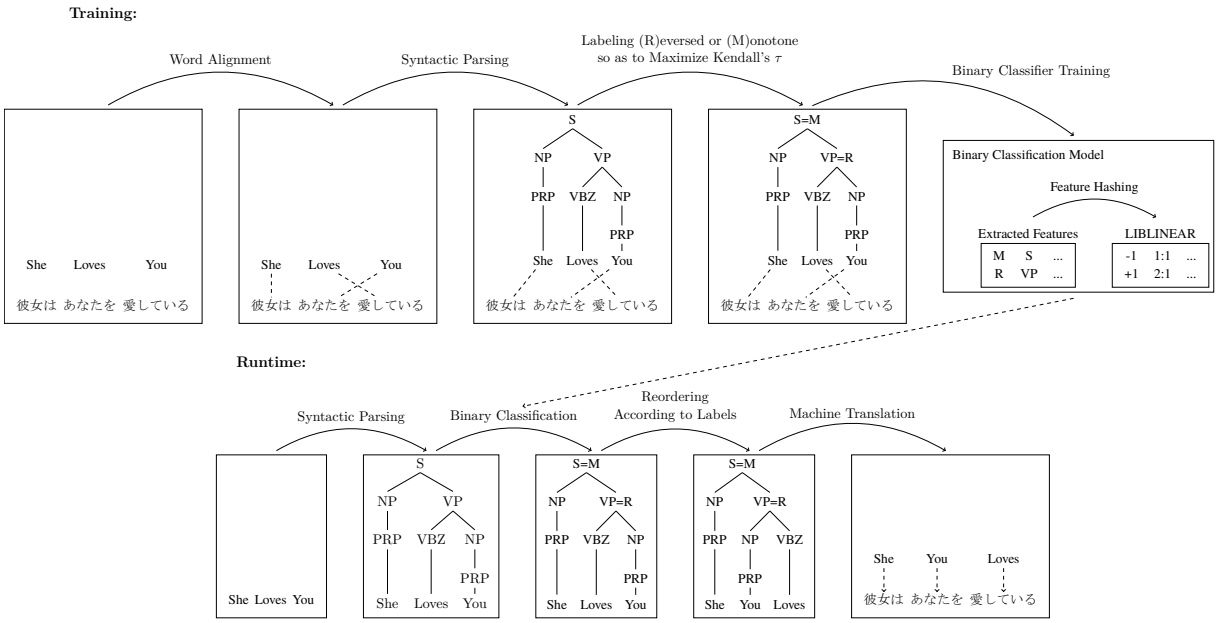

Figure 4.1: Overview of the proposed method at training and runtime.

engineering focused on improving binary classification accuracy.

In English-to-Japanese and Japanese-to-English patent translation experiments, our method leads substantial improvements in translation accuracy measured by RIBES (Isozaki et al., 2010a) and BLEU (Papineni et al., 2002), although we only need a simple preprocessing implementation using a binary classifier and a source-side constituent parser. In particular, when compared with the baseline preordering method of Li et al. (2007), the proposed method substantially outperforms the baseline for both English-to-Japanese and Japanese-to-English translations. We also show that our simple preordering method can obtain performance superior to, or at least close to state-of-the-art syntax-based reordering methods in terms of translation accuracy.

4.1 Syntax-based Preordering Method

Figure 4.1 shows an overview of our syntax-based preordering method using a binary classifier, which describes essentially the same steps as the previous method of Li et al. (2007) and only differs in how to train a binary classifier. This framework has two different pipelines for training

and runtime.

At training, we first run a word aligner and obtain word alignment between source and target sentences from the aligner. We then parse and binarize the source sentence using a source-side constituent parser and obtain a source-side binary parse tree. After that, we assign each node an oracle reordering label, either monotone or reversed, using word alignment information. Finally, we extract features from the source-side tree and train a binary classifier using the extracted features and assigned oracle reordering labels.

At runtime, we parse and binarize an input sentence again using a source-side constituent parser and obtain a source-side binary parse tree. Each node in the binary parse tree is then binary classified, i.e. either monotone or reversed, using the trained binary classifier. The tree reordered according to the classified labels is used to yield a reordered source sentence, which is fed to a standard statistical machine translation system.

4.1.1 Syntax-based Preordering using Binary Classifier

Followings describe the binary tree version[1] of the previous method (Li et al., 2007), as we fit it as basis of our proposal using alternative notation.

For a binary tree that corresponds to a sentence with n words, a binary node in the tree is expressed as:

$$v(i, p, j),$$

where $1 \leq i \leq p < p + 1 \leq j \leq n$. This formula indicates that the binary node takes a span (i, j) from the i-th to j-th words, and the entire span can be divided into the left span (i, p) from the i-th to p-th words and the right span $(p + 1, j)$ from the $(p + 1)$-th to j-th words.

We then make a reordering decision as to whether a node should be reversed as:

$$P(x \mid \theta(v(i, p, j))),$$

[1]Their original paper also discussed an extension to the 3-ary node reordering by replacing binary classifications with multi-class classifications of six candidates. For the sake of simplicity, however, our proposal is rather based on the binary tree version.

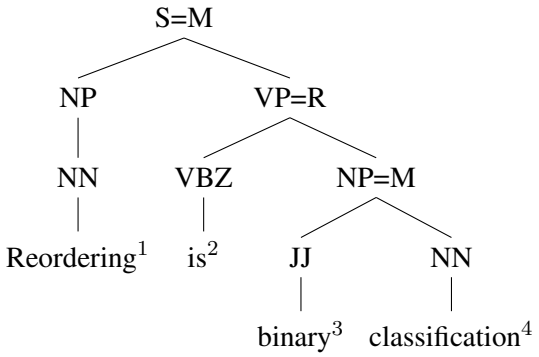

Figure 4.2: Each node is assigned a binary reordering label. The label R indicates reversed and the label M indicates monotone.

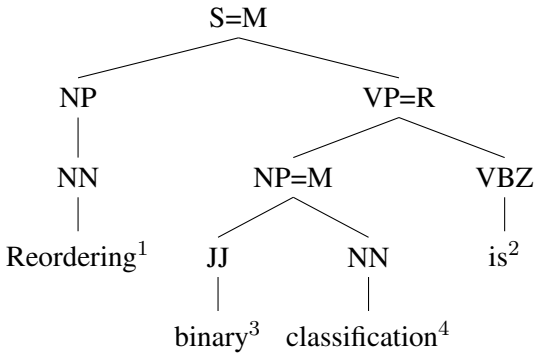

Figure 4.3: The tree reordered according to the assigned labels, which resembles the reordered sentence: *Reordering binary classification is*

where $x \in \{R, M\}$.[2] The label R indicates reversed (reverse the order of its child nodes), the label M indicates monotone (do not reverse the order of its child nodes), and θ is a feature function that takes the node as input. We describe the feature templates used in the function in Section 4.1.4.

For instance, Figure 4.2 shows a sentence ($n = 4$) that has three binary nodes, S, VP, and NP, which are our reordering candidates. We examine the NP node $v(3, 3, 4)$, which has a left span

[2]We actually used a linear support vector machine as a binary classifier as mentioned in Section 4.2.1, because we need to run it on a large amount of training data made out of millions of sentences. The proposed method is, however, not limited to a specific classifier. For example, Li et al. (2007) used a maximum entropy classifier.

$(3, 3)$ that covers the word $binary$[3] and a right span $(4, 4)$ that covers the word $classification$[4]. The reordering decision of the NP node is determined by $P(x \mid \theta(v(3, 3, 4)))$, and is classified as $x = M$ in this example. The actions for the VP node $v(2, 2, 4)$ and the S root node $v(1, 1, 4)$ are determined in a similar fashion.

Once all reordering decisions along a binary constituent tree are made, only the children of the nodes that are labeled R are reversed. For example, this reordering process produces a new tree shown in Figure 4.3 from the tree shown in Figure 4.2. The new tree represents a reordered sentence: *Reordering binary classification is*, which has word order that is similar to Japanese word order. The reordered sentence is fed to a statistical machine translation system.

4.1.2 Obtaining Oracle Reordering Labels so as to Maximize Kendall's Tau

In order to get the most from this framework, we need to train a binary classifier using oracle reordering labels, which represent the best combination of reordering labels for an entire sentence. Since there exists no such label data and manually annotating all training data is expensive and may cause inconsistency, we instead introduce a systematic procedure to obtain oracle reordering labels using word alignment information, similar to those of previous studies (Yang et al., 2012; Neubig et al., 2012).[3] The obtained reordering labels are used for training a binary classifier as well as evaluating the trained classifier.

The principal concept of our procedure is to assign each binary node $v(i, p, j)$ an oracle reordering label so that Kendall's τ (Kendall, 1938) is maximized for the node's span (i, j). Given the task of preordering is to generate a source word sequence whose word ordering correlates with its translation, we employed Kendall's τ as our objective, since that metric is extensively exploited in the context of statistical machine translation research, such as evaluation metric (Birch and Osborne, 2010; Isozaki et al., 2010a; Talbot et al., 2011) and optimization criterion (Yang et al., 2012; Neubig et al., 2012).

[3]We explore the degree to which the actual word alignment accuracy affects the reordering accuracy in Section 4.2.2. In general, the word alignment accuracy and reordering accuracy depend on each other. For instance, Ding et al. (2015) improved word alignment accuracy for distant language pairs by applying a reordering method.

Given a number list $\mathbf{x} = x_1, \ldots, x_n$, Kendall's τ measures the similarity between $\mathbf{x}$ and sorted $\mathbf{x}$ as:

$$\tau(\mathbf{x}) = \frac{4c(\mathbf{x})}{n(n-1)} - 1,$$

where $c(\mathbf{x})$ is the number of concordant pairs between $\mathbf{x}$ and sorted $\mathbf{x}$, which is defined as:

$$c(\mathbf{x}) = \sum_{i,j \in [1,n], i<j} \delta(x_i < x_j),$$

where $\delta(x_i < x_j)$ is the Kronecker's delta function that returns 1 if $x_i < x_j$ and 0 otherwise. The τ function indicates that $\mathbf{x}$ is completely monotonic when $\tau(\mathbf{x}) = 1$, and in contrast, $\mathbf{x}$ is completely reversed when $\tau(\mathbf{x}) = -1$.

For the task of reordering, we instead use word alignment $\mathbf{a}$ as the number list $\mathbf{x}$ as in the form $\mathbf{a} = a_1, \ldots, a_n$, where $a_x = y$ indicates that the x-th word in a source sentence corresponds to the y-th word in a target sentence.[4] Moreover, we define the limited word alignment for a span (i, j) as $\mathbf{a}(i, j) = \{a_k \mid i \leq k \leq j\}$, which corresponds to each binary node in a given binary constituent tree.

We then determine a reordering decision for each binary node $v(i, p, j)$ according to a *local* score for the node's span (i, j), which is defined as:

$$\begin{aligned} s(v(i,p,j)) = \tau(\mathbf{a}(i,p) \cdot \mathbf{a}(p+1,j)) \\ -\tau(\mathbf{a}(p+1,j) \cdot \mathbf{a}(i,p)), \end{aligned}$$

where $\cdot$ indicates a concatenation of word alignment. This formula compares the number of concordant pairs in the left-to-right perspective $\tau(\mathbf{a}(i,p) \cdot \mathbf{a}(p+1,j))$ and the number of concordant pairs in the right-to-left perspective $\tau(\mathbf{a}(p+1,j) \cdot \mathbf{a}(i,p))$. We then assign the oracle label R to nodes for which $s(v(i,p,j)) < 0$ and the oracle label M to nodes for which $s(v(i,p,j)) > 0$. All other nodes for which $s = 0$ are excluded from the binary classifier's training data, because

[4] Since word alignment is usually given as many-to-many representation using the grow-diag-final-and heuristic, we convert many-to-many word alignment into the $a_x = y$ indices by selecting median values in advance.

they are noisy and ambiguous samples in terms of the binary classification task.[5]

When there exists a node that has two words and the two words happened to share the same word alignment $a_x = y$, the node always has the score of zero and is excluded from training data. This is because we are not sure whether the node should be reversed, and the reordered result would not be affected by the reordering decision of the node anyway.

This procedure determines an oracle reordering label for each node locally. Now, the following question arises: *Can oracle reordering labels determined by making local decisions achieve the best overall reordering for a sentence?*

4.1.3 Proof: Decomposition of Kendall's Tau Computation

This hypothesis is shown to be true, because Kendall's τ at the sentence level can be decomposed in a recursive manner over a hierarchical syntactic structure (Yang et al., 2012; Neubig et al., 2012). As a proof, we present a similar decomposition in the same recursive manner. Please note that the idea of the decomposition of Kendall's τ is not new (Yang et al., 2012; Neubig et al., 2012), while we apply this observation to obtaining of the oracle reordering labels by making local decisions.

Let $c(\mathbf{a}(i,j))$ be decomposed as:

$$c(\mathbf{a}(i,j)) = c(\mathbf{a}(i,p)) + c(\mathbf{a}(p+1,j)) + \sum_{k\in[i,p],l\in[p+1,j]} \delta(a_k < a_l).$$

The three terms in this formula are mutually independent. That is, any reordering of $\mathbf{a}(i,p)$ changes only the first term, and all other terms remain unchanged. The same goes for the second and third terms.

We can maximize $c(\mathbf{a}(i,j))$ by maximizing each term. Since the first and second terms can be maximized in a recursive manner, we only need to maximize the third term, which corresponds

[5]The number of each labeled node varies in data. In Section 4.2.2 we obtained the following numbers out of the 8,000 sampled sentences that are used to train binary classifiers. The *Giza* data: 69,178 nodes were labeled R, 102,916 nodes were labeled M, and 43,896 nodes were excluded. The *Nile* data: 90,724 nodes were labeled R, 70,583 nodes were labeled M, and 54,683 nodes are excluded.

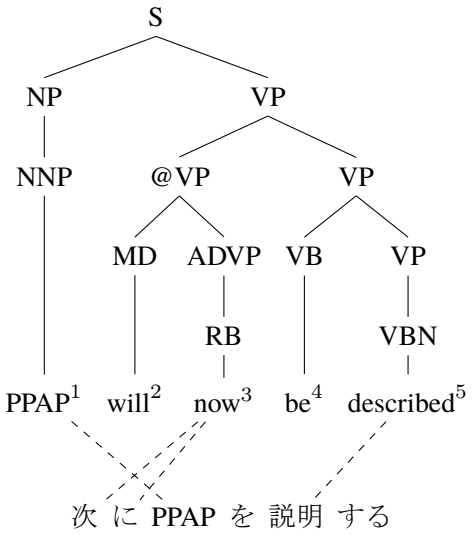

Figure 4.4: Rare case of $\tau(\mathbf{a}) < 1$ under tree structure: Dashed lines represent word alignment between the English parse tree and its translation in Japanese.

to the local score $s(v(i, p, j))$ of the proposed procedure. Therefore, our attempt to maximizing the local score of the proposed procedure leads to the maximization of $c(\mathbf{a}(i, j))$. Since $\tau(\mathbf{x})$ is proportional to $c(\mathbf{x})$, the maximization of $\tau(\mathbf{x})$ and hence the global maximization of Kendall's τ are achieved in the course of maximizing $c(\mathbf{x})$.

Please note that oracle reordering labels guarantee $\tau(\mathbf{a}) \geq 0$, but $\tau(\mathbf{a}) = 1$ is not guaranteed under syntactic constituent trees. For example, Figure 4.4 shows such a rare case, where we cannot make $\tau(\mathbf{a}) = 1$ from reversing any binary nodes, as discussed in Section 4.3.

4.1.4 Features

We introduce a new feature function that is used for training a binary classifier. Previous studies explored such a feature function, which is designed specifically for the task of reordering in a similar but different fashion. For the task of preordering using BTG, Neubig et al. (2012) used syntactic information such as a node's span length and its balance in subnodes, word surface forms, non-terminals, word classes obtained with Giza, and a phrase table as a dictionary.

Nakagawa (2015) extended the BTG prediction feature function with more feature combinations, using word classes obtained with the Brown clustering (Brown et al., 1992a). For the task of pre-ordering using binary classifier, Li et al. (2007) proposed the use of head words, non-terminals, and words that surround a node's span, which is derived from the literature on conventional parsing.

Based on these previous studies, we propose a new set of features that improves the binary classification accuracy of the previous method of Li et al. (2007). Our feature function consists of word surface forms and non-terminals, including pre-terminals, as in the form of string features similar to previous studies. In contrast to previous studies, our feature function captures syntactic contiguous sequences that contribute to our reordering decisions without using a phrase table dictionary in order to speed up binary classifications.

One might argue that structured learning with dynamic features, such as the use of parent node type feature (Nakagawa, 2015), will provide better classification accuracy than the static features we propose. However, we could not find any effective dynamic features that provide significant improvement of classification accuracy. Our interpretation of the observation is that the discrimination we need might be less dynamic than that of BTG methods, because, unlike the proposed method, oracle BTG trees have many ambiguities, including the differences in left-branching and right-branching.

Table 4.1 lists two types of feature templates used in our feature function θ given the binary node $v(i, p, j)$ in Section 4.1.1, namely *span features* and *tree features*. In order to identify the differences between a left span (i, p) and a right span $(p + 1, j)$, such as whether the head word of the node is on the left or the right, we introduce the *span features* for individual indices *x:y* that denote a span from the x-th to y-th words, where $t_{x:y}$ represents a pre-terminal, $w_{x:y}$ represents a word surface form, and $\circ$ represents a feature combination. In addition, we also introduce features that use the iteratively generated left and right sub-spans, because we intend to exhaustively cover all the possible lengths of spans. We call them a left sub-span (l, p) and a right sub-span $(p + 1, r)$, where integers d, l, and r satisfy $d \geq 0 \wedge l = \max(i, p - d) \wedge r =$

Table 4.1: Feature templates for the node $v(i, p, j)$, where integers d, l, and r satisfy $d \geq 0 \wedge l = \max(i, p-d) \wedge r = \min(p+1+d, j)$.

Span Features	Tree Features
$t_{i:p}, t_{p+1:j}, w_{i:p}, w_{p+1:j},$	$\sigma(v(i,p,j)),$
$t_{i:p} \circ t_{p+1:j}, w_{i:p} \circ w_{p+1:j},$	$\sigma_r(v(i,p,j)),$
$t_{i:p} \circ t_{p+1:j} \circ w_{i:p} \circ w_{p+1:j},$	$\sigma_t(v(i,p,j)),$
$t_{l:p}, t_{p+1:r}, w_{l:p}, w_{p+1:r},$	$\sigma_w(v(i,p,j))$
$t_{l:p} \circ t_{p+1:r}, w_{l:p} \circ w_{p+1:r},$	
$t_{l:p} \circ t_{p+1:r} \circ w_{l:p} \circ w_{p+1:r},$	

$\min(p+1+d, j)$. The left sub-span is a subset of the entire left span but always ending with the same rightmost index p. Similarly, the right sub-span is a subset of the entire right span but always starting with the same leftmost index $p+1$. The integer d is generated sequentially from 0 up to the maximum length of sub-spans in order to generate all the possible sub-spans. We found that these sub-spans are also helpful to learn the differences between the left and right spans.

The *tree features* directly represent the shape of trees by combining the S-expression notation of trees and the indexed non-terminals, where $\sigma(v(i,p,j))$ represents a tree structure under node $v(i,p,j)$, $\sigma_r(v(i,p,j))$ represents all the non-terminals of the node and their combinations between parent and child non-terminals, each annotated with an integer prefix that denotes a depth from the span's root node, where each combination separated by a white space boundary become a feature, $\sigma_t(v(i,p,j))$ represents the tree structure including only the non-terminals, and $\sigma_w(v(i,p,j))$ represents the tree structure including only the word surface forms.

Table 4.2 shows instances of features for the VP node $v(2,2,4)$ in Figure 4.2, which has left (is^2) and right ($binary^3 classification^4$) spans. Individual templates represent string features, such as the template $t_{2:2}$ instantiates the pre-terminal VBZ. In addition to these individual instances, many feature combinations that consist of pre-terminals and word surface forms are used

Table 4.2: Feature instance examples for the VP node $v(2, 2, 4)$ in Figure 4.2.

Feature Template	Instance
$t_{2:2}$	VBZ
$t_{3:4}$	JJ_NN
$t_{3:3}$	JJ
$w_{2:2}$	is
$w_{3:4}$	binary_classification
$w_{3:3}$	binary
$\sigma(v(2, 2, 4))$	(VP(VBZis)(NP(JJbinary)(NNclassification)))
$\sigma_r(v(2, 2, 4))$	0VP 1VBZ 1NP 2JJ 2NN 0VP_VBZ 0VP_NP 1NP_JJ 1NP_NN
$\sigma_t(v(2, 2, 4))$	(VP(VBZ)(NP(JJ)(NN)))
$\sigma_w(v(2, 2, 4))$	((is)((binary)(classification)))

in actual classifications, such as the combination of VBZ and JJ_NN (the instance of $t_{2:2} \circ t_{3:4}$) as shown in each row in the span features.

4.2 Experiment

In order to assess various aspects of the proposed method, we conduct a series of experiments on the English-Japanese language pair, which is known to be a challenging task for translation, especially in terms of reordering, in the following order:

- We first investigate the amount and quality of word alignment data needed for training a binary classifier that assigns reordering labels, by comparing unsupervised and supervised word aligners (Section 4.2.2).
- We then study feature ablation for the purpose of evaluating the contribution of each feature template used in the proposed method (Section 4.2.3).
- After that, we validate how much does the use of binarized syntactic trees, instead of n-ary

Table 4.3: Data Splitting: Both the 8,000 manually annotated sentences and one million unannotated sentences are used to train binary classifier.

Source	Training NTCIR-7/8				Development NTCIR-8		Test NTCIR-9/10	
Label	training			unused	tuning	dev	test9	test10
	manually annotated	unannotated random sample				manually annotated		
#sentences	8,000	1,000,000	1,908,615		1,000	1,000		
Total	2,916,615			269,669	2,000		2,000	2,300

trees, affect to the performance of the proposed method (Section 4.2.4).

- Finally, we compare the proposed method with state-of-the-art reordering methods (Section 4.2.5).

4.2.1 Experimental Settings

We conducted the experiments using the several English-Japanese portions of NTCIR patent corpora, which in total consist of more than three million parallel sentences in English and Japanese. Following the NTCIR-9 (Goto et al., 2011) and NTCIR-10 (Goto et al., 2013) workshop configurations, we used the NTCIR-7 and NTCIR-8 training sets, the NTCIR-8 development set, and the NTCIR-9 and NTCIR-10 test sets (*test9* and *test10*). Table 4.3 summarizes our data splitting. We removed 269,669 sentences from the training data due to parsing errors. We then separately sampled 8,000 sentences and one million sentences from the training data. We divided the 2,000 sentences from the development data into two parts: the first 1,000 sentences were used for tuning the machine translation system, and the second 1,000 sentences (*dev*) were used for evaluating the system during development. Both the second set of sentences (*dev*) and the 8,000 sentences sampled from the training data were manually annotated with word alignment.

We used a standard phrase-based statistical machine translation system built on top of Moses

3 (Koehn et al., 2007). In all experiments, they share the following settings: 6-gram language model is trained by SRILM 1.7.0 (Stolcke, 2002). Word alignment is automatically annotated by MGIZA (Gao and Vogel, 2008) using the grow-diag-final-and heuristic. The system is tuned using the minimum error rate training method (Och, 2003), the lexicalized reordering model (Tillman, 2004), and distortion limit hyperparameter. Preordering methods are deployed as a preprocessing step to the statistical machine translation system.

In order to evaluate the system output, RIBES (Isozaki et al., 2010a) and BLEU (Papineni et al., 2002) in the default settings were used to judge the quality of the global word ordering in RIBES and the local word ordering in BLEU.[6] Both metrics represent the similarity between the system output and human reference translations as a score range in $[0, 100]\%$, where higher the score the higher the similarity.

For preordering data preprocessing, we generated binary constituent trees of the NTCIR parallel sentences in English and Japanese by applying the Berkeley Parser 1.7 (Petrov et al., 2006; Petrov and Klein, 2007). In order to parse Japanese sentences using the Berkeley Parser, we setup virtually a Japanese version of the parser using the binary branching model of Haruniwa (Fang et al., 2014; Horn et al., 2017), Comainu 0.7.0 (Kozawa et al., 2014), MeCab 0.996 (Kudo et al., 2004), and UniDic 2.1.2 (Den et al., 2008). In their parsing accuracy studies, the Berkeley Parser achieved a score of F1 90.2 in English, whereas Haruniwa reported a relatively lower score of F1 80.29 in Japanese.

As a binary classifier, we used LIBLINEAR 1.96 (Fan et al., 2008) to perform binary classifications within a linear support vector machine, where the parameter C is fixed to 1.0 and no bias term is added. In addition, the string features we introduced in Section 4.1.4 are converted into unsigned 30-bit integers using the feature hashing technique (Shi et al., 2009) in order to achieve faster learning and smaller memory consumption for the linear support vector machine.

[6] We ran the Travatar toolkit (Neubig, 2013) for both evaluation metrics and their statistical testings.

4.2.2 Comparison of Word Aligners

We explore two types of word alignment data for training a binary classifier used in the proposed method. The first data set (*Giza*) is created by running the unsupervised aligner Giza (Och and Ney, 2003) on the training data (three million sentences). The second data set (*Nile*) is generated by training the supervised aligner Nile (Riesa et al., 2011) on the 8,000 manually annotated sentences sampled from the training data. The trained Nile aligner was then used to annotate remaining training data. We also evaluated the word alignment accuracy of the two aligners using the manually annotated development data (*dev*). In the evaluation Giza achieved a score of F1 50.1, whereas Nile achieved a score of F1 86.9.

We then trained two binary classifiers, i.e., Giza and Nile classifiers, which were named after the two respective data types, using the obtained training data, which consist of the fixed number (8,000) of manually annotated sentences and one million additional unannotated sentences that were randomly sampled from the remaining training data. We conducted an intrinsic evaluation of the two classifiers using Kendall's τ distribution (Isozaki et al., 2010a), which is a graphical representation of the distribution of the monotonicity of sentences in terms of Kendall's τ measure on the x-axis and the percent ratios of sentence on the y-axis, with the averaged Kendall's τ of all sentences. Again both the Giza and Nile classifiers are evaluated using manually annotated development data (*dev*).

Figure 4.5 shows the Kendall's τ distribution of the development data (*dev*) before applying the preordering step. The value of 0.4 in the averaged Kendall's τ, which indicates a weak correlation to monotone word order, suggests the difficulty of the task of reordering for the English-Japanese language pair. In contrast, Figure 4.6 shows the development data after applying the proposed procedure to obtain oracle reordering labels so as to maximize Kendall's τ. The value of 0.9 indicates a very strong correlation to the monotone word order. The differences between Figures 4.5 and 4.6 clearly shows the effects of the proposed procedure to obtain oracle reordering labels.

Figures 4.7 and 4.8 show the Kendall's τ distributions created by applying the trained Giza and

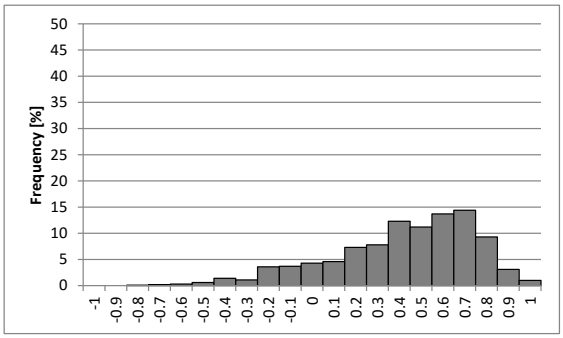

Figure 4.5: Japanese-to-English development data (dev): average $\tau = 0.4172$

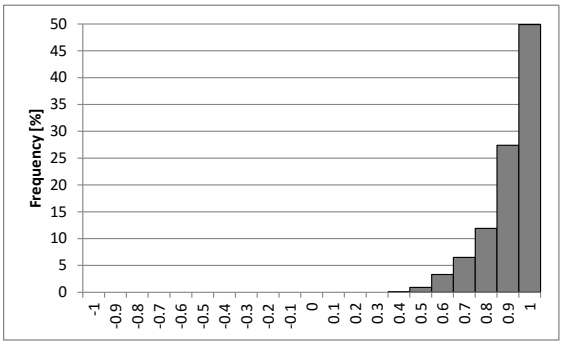

Figure 4.6: Development data (dev) after applying the proposed procedure to obtain oracle reordering labels so as to maximize Kendall's τ: average $\tau = 0.9091$

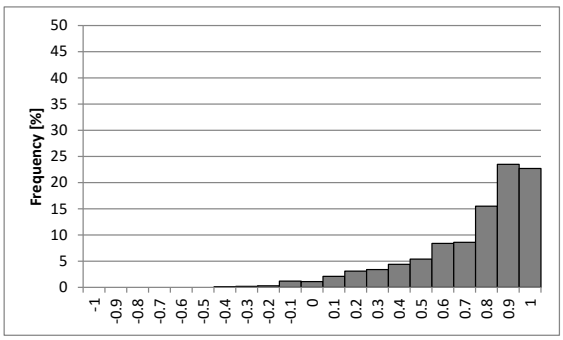

Figure 4.7: Development data (dev) reordered by the Giza classifier: average $\tau = 0.7320$

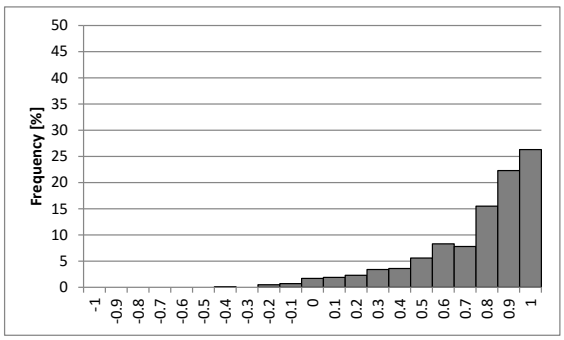

Figure 4.8: Development data (dev) reordered by the Nile classifier: average $\tau = 0.7478$

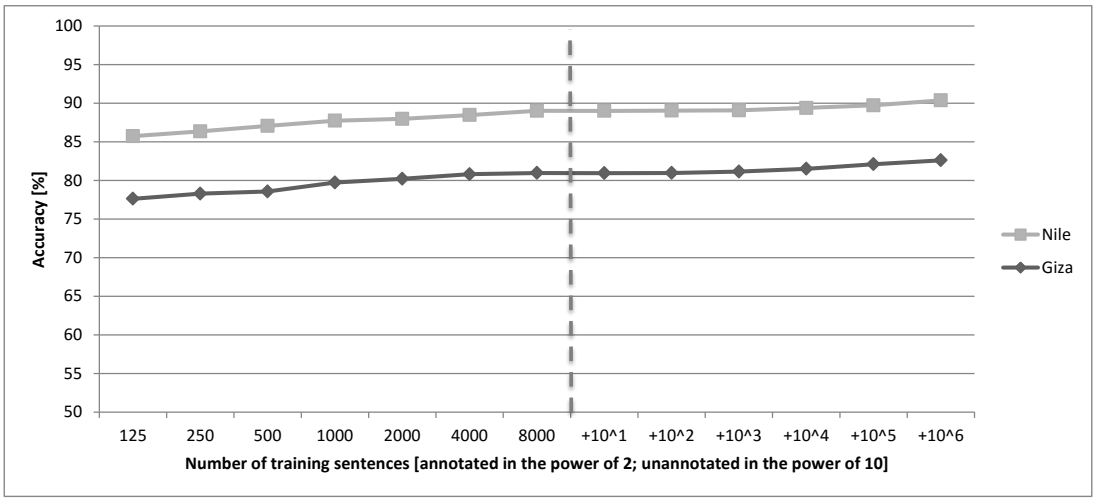

Figure 4.9: Binary classification accuracy of the learning curves of the Giza and Nile classifiers for the development data (*dev*)

Nile classifiers, respectively, to the development data (*dev*). Comparison of these distributions revealed that the Nile classifier provided a greater number of completely monotonic sentences, where $\tau(\mathbf{x}) = 1$, as compared to the Giza classifier.

Figure 4.9 shows the learning curves of the Giza and Nile binary classifiers in terms of accuracy for the development data (*dev*). Since the training data set consists of the 8,000 manually annotated sentences and one million unannotated sentences, we varied the number of annotated sentences from 125 to 8,000 using the power of 2. Then, in addition to the 8,000 annotated sentences, we also varied the additional amount of unannotated sentences from 10 to one million using the power of 10. As indicated by the figure, the Nile classifier has higher accuracy than the Giza classifier.

Table 4.4 shows extrinsic evaluation of the proposed method in terms of the translation accuracy for the development data (*dev*), where we compared the differences between the use of the Giza and Nile classifiers, as well as 3 types of distortion limit settings (*DL*). We enlisted 0, 10, and 20 as our distortion limit candidates because they are usually set to a value from a range of 0 to 20. These results suggest that the Nile classifier with a distortion limit of 10 was the best

Table 4.4: Results of the Giza and Nile classifiers for the Japanese-to-English translation of the development data (*dev*). Bold text denotes either the highest score or the insignificance from the highest ($p < 0.01$) obtained within bootstrap resampling (Koehn, 2004b).

			dev	
Classifier	DL	Accuracy	RIBES	BLEU
Giza	0	82.60	**76.72**	**31.38**
Giza	10	82.60	**76.71**	**31.64**
Giza	20	82.60	**76.22**	31.11
Nile	0	90.36	**76.80**	**31.45**
Nile	10	90.36	**76.89**	**31.81**
Nile	20	90.36	76.32	**31.77**

configuration. As such, we choose the same settings for later experiments in the development process.

4.2.3 Feature Ablation Tests

Table 4.5 shows a comparison of binary classification accuracy and translation accuracy for the development data (*dev*) between the proposed feature templates with ablation and the previous feature template used by Li et al. (2007). The results of the comparison suggest that the full use of the proposed feature templates is the best setting. They also suggest that the span features contributed more than any of the tree features. It is likely that combinations of word surface forms and non-terminals are strong enough signals as our non-terminals include syntactic phrase labels. The differences from the previous feature template demonstrate the importance of the empirical contributions of the proposed feature templates.

4.2.4 Comparisons of Predicted and Oracle Preordering Output

In order to evaluate the validity of the proposed method based on binary classifications, we conducted another set of experiments that show the effects of binary classification errors and

Table 4.5: Ablation tests in the Japanese-to-English translation of the development data (*dev*). Bold text denotes either the highest score or the insignificance from the highest ($p < 0.01$) obtained within bootstrap resampling (Koehn, 2004b).

			dev	
Feature Template	DL	Accuracy	RIBES	BLEU
Full features	10	90.36	**76.89**	**31.81**
Without the span features	10	86.97	73.02	30.26
Without $\sigma(v(i,p,j))$	10	90.18	**76.83**	**31.71**
Without $\sigma_r(v(i,p,j))$	10	90.29	**76.28**	31.67
Without $\sigma_t(v(i,p,j))$	10	90.33	**76.73**	**31.91**
Without $\sigma_w(v(i,p,j))$	10	90.31	**76.94**	**31.62**
Li et al. (2007)	10	84.38	66.59	27.39

binary tree constraints in terms of translation accuracy for the development data (*dev*).

For the binary classification errors, we compare the predicted preordering output (*Nile*) and oracle preordering output (*Oracle with binary tree constraints*) obtained so as to maximize Kendall's τ in Section 4.1.2, in order to show the distance from the current prediction level to the upper bound of the proposed preordering method. For the binary tree constraints, we compare the oracle preordering output (*Oracle with binary tree constraints*) and the oracle preordering output without binary tree constraints (*Oracle without binary tree constraints*) obtained by applying a stable sort to word alignment as described by Tromble and Eisner (2009). We then directly reordered sentences according to the sorted word alignment.

Table 4.6 shows the results of the predicted preordering, the oracle preordering with binary tree constraints, and the oracle preordering without binary tree constraints in terms of translation accuracy for the development data (*dev*). The translation accuracy differences between the predicted and oracle classifiers suggest that there are still numerous binary classification errors that caused a significant degradation in translation accuracy. The results also indicate that the use

Table 4.6: Results of predicted and oracle classifiers for the Japanese-to-English translation of the development data (*dev*). Bold text either denotes the highest score or the insignificance from the highest ($p < 0.01$) obtained within bootstrap resampling (Koehn, 2004b).

			dev	
Classifier	DL	Accuracy	RIBES	BLEU
Nile	10	90.36	76.89	31.81
Oracle with binary tree constraints	10	100	82.36	33.58
Oracle without binary tree constraints	10		**84.65**	**36.62**

of binary tree constraints caused additional degradation in translation accuracy. Therefore, it is possible to further improve the proposed method by improving either the binary classification accuracy or the use of the binary tree constraints in the future.

4.2.5 Comparison with State-of-the-Art Methods

Tables 4.7 and 4.8 show the final performance of the proposed method for the test data sets for English-to-Japanese and Japanese-to-English translations, respectively. These results indicate that the proposed preordering method significantly improved translation accuracy in terms of both RIBES and BLEU scores, as compared to the baseline results attained by Moses without preordering. In particular, the proposed preordering method trained using the Giza data revealed a substantial improvement, whereas the use of the Nile data further improved translation accuracy. This suggests that the proposed method is particularly effective when high-accuracy word alignment is obtained. In addition, we achieved modest improvements even when no distortions are allowed ($DL = 0$), which indicates the monotonicity of the output of the proposed system.

Table 4.9 and the upper half of Table 4.10 show a comparison of the proposed method with the previous method of Li et al. (2007) and the rule-based preordering method of Isozaki et al. (2010b). The proposed method outperformed the previous methods for both English-to-Japanese and Japanese-to-English translations, especially when the previous method of Li et al. (2007)

Table 4.7: Final testing results for the English-to-Japanese translation. Bold text denotes either the highest score or an insignificant difference ($p < 0.01$) from the highest obtained within bootstrap resampling (Koehn, 2004b).

		test9		test10	
Classifier	DL	RIBES	BLEU	RIBES	BLEU
Baseline without preordering					
Moses	0	64.91	25.83	65.16	27.64
Moses	10	69.40	31.34	69.33	32.57
Moses	20	70.74	32.05	70.82	32.91
Proposed preordering					
Giza	0	**78.68**	34.62	78.75	35.61
Giza	10	**78.72**	34.82	78.88	**36.28**
Nile	0	**79.05**	35.03	**79.11**	**36.25**
Nile	10	**79.04**	**35.42**	**79.29**	**36.59**

Table 4.8: Final testing results for the Japanese-to-English translation. Bold text denotes either the highest score or the insignificant difference ($p < 0.01$) from the highest obtained within bootstrap resampling (Koehn, 2004b).

		test9		test10	
Method	DL	RIBES	BLEU	RIBES	BLEU
Baseline without preordering					
Moses	0	66.95	26.36	67.50	27.17
Moses	10	68.95	29.41	69.64	30.20
Moses	20	69.88	30.12	70.22	30.51
Proposed preordering					
Giza	0	77.49	33.08	77.49	33.65
Giza	10	77.44	**33.28**	77.42	33.77
Nile	0	77.74	32.97	77.89	**33.91**
Nile	10	**77.97**	**33.55**	**78.07**	**34.13**

Table 4.9: Results of the proposed method and previous methods for the English-to-Japanese translation. Bold text denotes either the highest score or the insignificance from the highest ($p < 0.01$) obtained within bootstrap resampling (Koehn, 2004b).

		test9		test10		
Method	DL	RIBES	BLEU	RIBES	BLEU	Training Time
Moses without preordering	20	70.74	32.05	70.82	32.91	
Isozaki et al. (2010b)	10	75.39	34.08	75.68	34.69	
Li et al. (2007)	10	77.04	34.29	77.21	35.48	1 day
Proposed procedure	10	76.88	34.13	77.20	35.64	1 day
Proposed features	10	**78.82**	**35.29**	78.94	**36.58**	1 day
Proposed procedure and features	10	**79.04**	**35.42**	**79.29**	**36.59**	1 day

Table 4.10: Results of the proposed and previous methods for the Japanese-to-English translation. Bold text denotes either the highest score or the insignificance from the highest ($p < 0.01$) obtained within bootstrap resampling (Koehn, 2004b).

		test9		test10		
Method	DL	RIBES	BLEU	RIBES	BLEU	Training Time
Moses without preordering	20	69.88	30.12	70.22	30.51	
Li et al. (2007)	10	68.37	29.13	69.34	30.09	1 day
Proposed procedure	10	66.24	28.46	65.88	29.00	1 day
Proposed features	10	77.49	**33.25**	**77.77**	**34.31**	1 day
Proposed procedure and features	10	**77.97**	**33.55**	**78.07**	**34.13**	1 day
Preordering methods trained with the 8,000 manually annotated sentences						
Yang et al. (2012)'s ranking	10	72.96	30.63	73.19	31.17	8 minutes
lader + chunk fragmentation	10	73.13	32.00	72.29	32.37	15 days
lader + Kendall's τ	10	75.16	32.45	75.31	32.47	8 days
Proposed procedure and features	10	76.96	33.15	77.10	33.70	8 minutes

exhibited poor performance in the more difficult Japanese-to-English translation.

In addition to this comparison, we conducted another comparison to order to clarify the differences between the proposed procedure based on Kendall's τ and previous optimization strategies (Yang et al., 2012; Neubig et al., 2012). When using Kendall's τ, these previous studies apply the CKY algorithm to n-ary or binary tree structures in the $O(n^3)$ fashion, whereas the proposed procedure makes local reordering decisions so as to maximize Kendall's τ over binary tree structures in the $O(n)$ fashion, rather than n-ary tree structures. This difference in computational complexity can be observed in practice by comparing training time of the proposed method with the previous methods. In this comparison, we reduced the size of the preordering training data set from three million sentences to the 8,000 manually annotated sentences due to the expected high cost for training those models. The experimental settings were left unchanged unless otherwise noted, and we used Moses without preordering as our baseline. The proposed method with the 8,000 training data was based on the Nile classifier in Section 4.2.2. We then compared the results of the proposed method with those two previous methods (Yang et al., 2012; Neubig et al., 2012), under the following settings:

- While Yang et al. (2012) applied the CKY algorithm to n-ary dependency with their original feature set, we could not follow their methodology because only a source-side constituent parser and its binary parse trees were available in the present study. Instead, we used an extended version of LIBLINEAR (Lee and Lin, 2014) as a linear ranking support vector machine and defined a combination of parent and child non-terminals (denoted as $h_t \circ c_t$ in their original feature template) as a feature given a binary constituent tree (similar to the proposed method). While this modification may greatly reduce accuracy as well as computational complexity, we focus on reproducing their ranking support vector machine settings.

- For the method of Neubig et al. (2012), we used lader 1.6 with the cube growing extension (Na and Lee, 2013) in order to speed up computation. The objective function is set to either

chunk fragmentation or Kendall's τ. We reduced the number of iterations from the default value of 500 to 200, because the former setting took 19 days in total, even in the case of the faster Kendall's τ objective function, although there were no significant updates between the 200th and 500th iterations. Moreover, due to this required classifier training time, the default configuration of the use of basic features were applied instead of fully syntactic features.

The bottom half of Table 4.10 shows the performance of the compared preordering methods, including training time, for the Japanese-to-English translation. We did not compare runtime, because we can distribute the computation of reordering as well as parsing at runtime, in parallel, but we cannot easily distribute classifier training. The proposed method exhibited the best performance, whereas the proposed method using Yang et al. (2012)'s ranking and Neubig et al. (2012)'s lader showed rather modest improvements. In terms of training time, no difference was caused by the use of the Yang et al. (2012)'s ranking. The two lader setups, however, took much longer than others due to their computational complexity. Although we did not reproduce Yang et al. (2012)'s method using dependency and the settings we used for Neubig et al. (2012)'s lader have room for improvement[7], the results showed the effectiveness of the proposed method from a practical standpoint.

Table 4.11 lists the scores obtained using the proposed system and those obtained using state-of-the-art systems published in the literature (Hoshino et al., 2014; Hayashi et al., 2013; Goto et al., 2015), which include a rule-based preordering method (Hoshino et al., 2013), two post-ordering methods (Goto et al., 2012; Hayashi et al., 2013), conventional syntax-based statistical machine translation and preordering methods (Chiang, 2007; Hoang et al., 2009; Genzel, 2010; Neubig et al., 2012), and a syntax-based preordering method that projects a target-side constituent tree into a source-side tree (Goto et al., 2015).[8] One complication of these concrete

[7] Also, Nakagawa (2015) proposed a much faster method using different formulation, which has a computational complexity of $O(n^2)$

[8] Although Neubig and Duh (2014) reported that RIBES +6.19 (75.94) and BLEU +2.93 (33.70) improvements were obtained with their forest-to-string system on the same corpus, we cannot include their results because of the unfortunate test data

Table 4.11: Published state-of-the-art system scores for the Japanese-to-English translation. The symbol Δ denotes the difference in the score from its baseline Moses without preordering. Bold text indicates the highest score and difference.

		test9				test10			
Method	DL	RIBES	Δ	BLEU	Δ	RIBES	Δ	BLEU	Δ
Moses	20	69.88		30.12		70.22		30.51	
Li et al. (2007)	10	68.37	-1.51	29.13	-0.99	69.34	-0.88	30.09	-0.42
Proposed method	10	**77.97**	**+8.09**	33.55	**+3.43**	**78.07**	+7.85	34.13	+3.62
Published by Hoshino et al. (2014)									
Moses	10	68.08		27.57					
Hoshino et al. (2013)	10	72.37	+4.29	30.56	+2.99				
Published by Goto et al. (2012)									
Moses	20	68.28		30.20					
Goto et al. (2012)		75.48	+7.20	33.04	+2.84				
Published by Hayashi et al. (2013)									
Moses	20	69.31		29.43		68.90		29.99	
Hayashi et al. (2013)	0	76.46	+7.15	32.59	+3.16	76.76	**+7.86**	33.14	+3.15
Published by Goto et al. (2015)									
Moses	20	68.79		30.92		68.30		31.07	
Chiang (2007)		70.11	+1.32	30.29	-0.63	69.69	+1.39	30.77	-0.30
Hoang et al. (2009)		72.54	+3.75	31.94	+1.02	71.32	+3.02	32.40	+1.33
Genzel (2010)	6	71.88	+3.09	29.23	-1.69	71.20	+2.90	29.40	-1.67
Neubig et al. (2012)	6	74.31	+5.52	32.98	+2.06	73.98	+5.68	33.90	+2.83
Goto et al. (2015)	6	76.35	+7.56	**33.83**	+2.91	75.81	+7.51	**34.90**	**+3.83**

comparisons is that each study reports a different baseline accuracy, although Moses is shared as a baseline, because these systems differ in various settings in data preprocessing, tokenization criteria, etc. Since this makes a fair direct comparison difficult, we include the score difference (Δ) for each system from its own baseline.

The proposed method showed the high performance that is comparable to, or superior to, the compared state-of-the-art methods, including the latest preordering method (Goto et al., 2015). More precisely, the proposed method exhibited the best or the second best performance for all evaluation criteria, even though the proposed method is kept simpler than other syntax-based methods while achieving state-of-the-art performance. This highlights the effectiveness and importance of the proposed method using binary classifications.

In contrast to the substantial gain for RIBES, however, we attained somewhat comparable gain for BLEU. The investigation of the output of the proposed system for the Japanese-to-English translation suggests that insufficient generations of English articles caused significant degradation of the local word ordering quality in BLEU. The postordering systems listed in Table 4.11 incorporated article generation techniques and demonstrated their positive effects (Goto et al., 2012; Hayashi et al., 2013). While we achieved state-of-the-art performance without using such a language-specific technique, integration of the proposed preordering method with language-specific article generation and null subject generation (Kudo et al., 2014) is a promising future direction.

4.3 Analysis of Typical Reordering Errors

In search of the major cause of reordering errors in the proposed method, Table 4.12 lists 10 typical pre-terminals with their binary classification accuracy for the development data (*dev*) in English and Japanese. These pre-terminals used in English and Japanese are mostly different, where QP in English is similar to NUMCLP in Japanese, but S, SBAR, and NX in English and CP-QUE, IP-EMB and IP-MAT in Japanese have no counterpart. Still we can compare the

mismatching between our test9/10 and their test7.

Table 4.12: Binary classification accuracy for the node $v(i, p, j)$ in the development data (*dev*): the English symbol QP denotes a quantifier phrase, ADJP denotes an adjective phrase, SBAR denotes a subordinating conjunction, ADVP denotes an adverb phrase, PRN denotes a parenthetical, and NX denotes a head of NP, where the Japanese symbol CP-QUE denotes a question, IP-EMB denotes a gapless noun-modifying clause, PP denotes a particle phrase (in contrast to English prepositional phrase), IP-MAT denotes a matrix clause, and NUMCLP denotes a numeral-classifier phrase.

English		Japanese	
Pre-terminal	Accuracy	Pre-terminal	Accuracy
QP	76.9231% (10/13)	CP-QUE	63.6364% (7/11)
VP	89.9674% (2206/2452)	IP-EMB	78.2796% (364/465)
NP	91.9967% (4414/4798)	VP	81.9434% (2260/2758)
ADJP	92.0455% (243/264)	ADVP	90.0000% (36/40)
PP	93.2560% (2572/2758)	NP	91.5311% (4669/5101)
S	93.5727% (1616/1727)	ADJP	92.3077% (12/13)
SBAR	96.7320% (296/306)	PP	94.2336% (2582/2740)
ADVP	96.8750% (31/32)	IP-MAT	95.6961% (2179/2277)
PRN	99.0991% (220/222)	PRN	96.4912% (550/570)
NX	100.0000% (18/18)	NUMCLP	97.9167% (282/288)
Overall	92.9142% (16391/17641)	*Overall*	90.3627% (15096/16706)

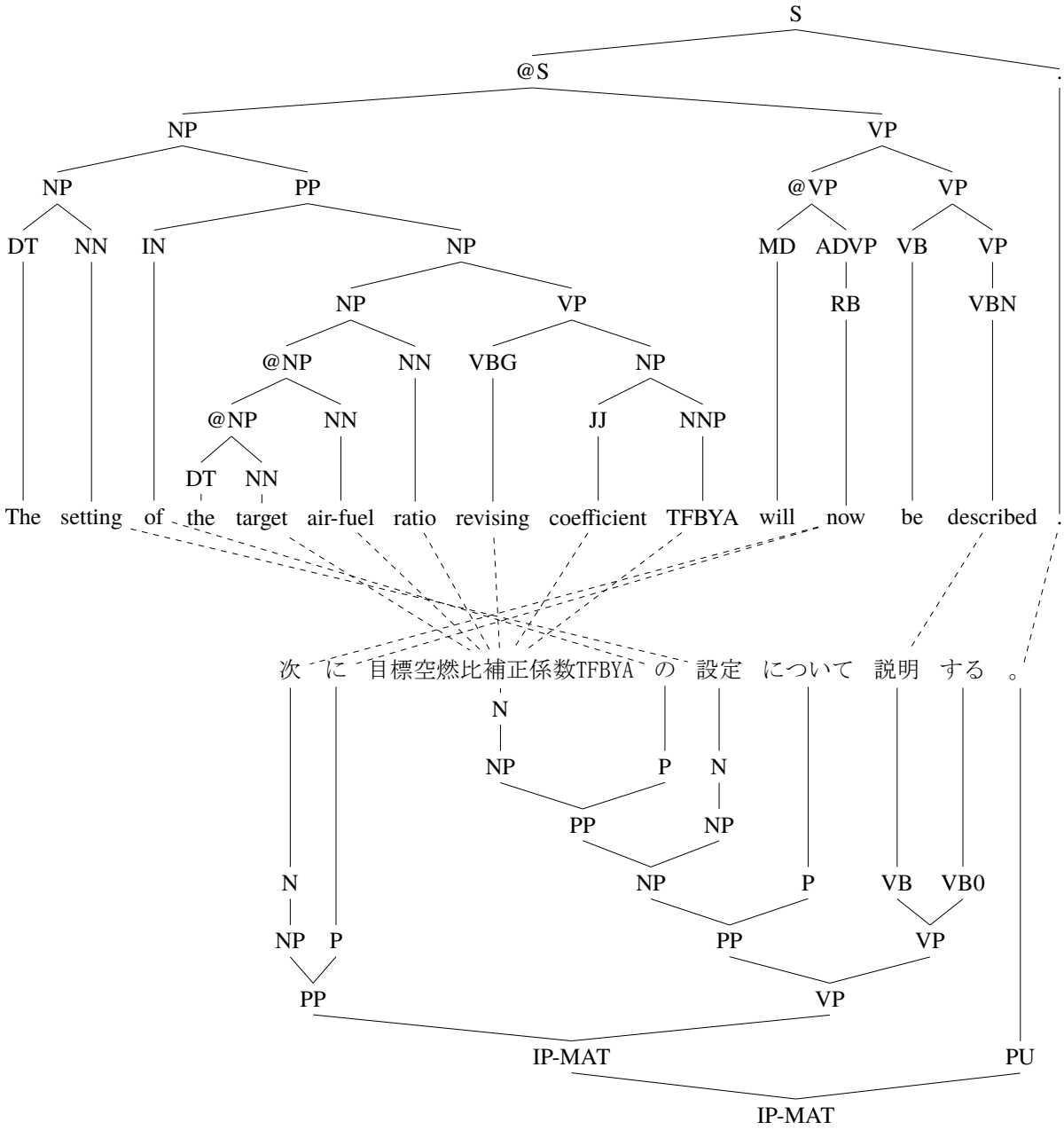

Figure 4.10: Example of the inside-out matching

shared per-terminals such as NP and VP.

We can see from the table that the *overall* binary classification accuracy in English is much better than that of Japanese, which explains why the English-to-Japanese translation was more accurate than the Japanese-to-English translation. Measured by the number of occurrences, we found that VP was the major cause of the difference: the accuracy in English is roughly 90% but the accuracy in Japanese is only 82%. Since VP plays one of the most important roles in syntax, we attribute the major cause of our reordering errors to the moderate classification accuracy of VP.

In addition to searching the major cause in binary classification errors, we figured that another type of errors is caused by the use of binary parse trees. Although most of the errors can be attributed to parsing errors, what we found was a problematic example shown in Figure 4.10. To our contrary, this complex yet realistic example has no parsing errors at all, but we cannot yield a correctly reordered tree.

Specifically, in this example, the Japanese phrase "次 に (*next*)" is aligned with the English word "now" and is supposed to be inserted between the long noun phrase "the target ... TFBYA" and the verb "described", generating "the target ... TFBYA will **now** be described". That insertion is, however, impossible using either the English or Japanese trees.

This type of errors can be regarded as an instance of *inside-out* matching (Wu, 1997) and a similar instance has been reported for English-to-English alignment (Wellington et al., 2006).

4.4 Related Work

4.4.1 Preordering based on Constituency Parsing

Li et al. (2007) proposed a simple preordering method along the constituent tree described in Section 4.1.1. Since their method was lacking oracle reordering labels for an entire sentence at that time, they instead provided a heuristic procedure for obtaining reordering labels to train their method, although its effectiveness was unknown. Their procedure used Giza to align their training sentences, and source words were then sorted to resemble target word indices. The sorted

source sentences without the overlap of word alignment were then used to train binary classifier, as either reversed or monotone reordering labels, according to the given word alignment. They gained a BLEU +1.54 improvement in the Chinese-to-English translation evaluation.

In terms of the formulation of the proposed method using Kendall's τ, their heuristic selects training instances, where $\tau(\mathbf{a}) = 1$ or $\tau(\mathbf{a}) = -1$, in order to compensate for the loss of oracle reordering labels. Thus, their procedure is effective but is not likely to be optimal for an entire sentence. The proposed proposed follows their framework, whereas we do not rely on their procedure for preparing its training data. Instead, we introduced the novel procedure to obtain oracle reordering labels so as to maximize Kendall's τ in Section 4.1.2.

4.4.2 Preordering based on Dependency Parsing

Yang et al. (2012) proposed a learning-to-rank method to produce oracle reordering decisions in another preordering method along a dependency tree. Their idea is to minimize the alignment crossing-link number recursively, as discussed in Section 4.4.6. Since they targeted the complex n-ary node dependency tree instead of a simple binary tree, their method only calculates approximated oracle reordering decisions in practice by the ranking principle. We did not take the n-ary dependency tree into consideration in order to follow Li et al. (2007)'s preordering method along a constituent tree, whereas the use of the binary tree enabled us to produce the strict oracle reordering decisions as a secondary effect.

Lerner and Petrov (2013) proposed yet another preordering method along a dependency tree, which classifies whether the parent of each node should be the head in a target language. They reported a BLEU +3.7 improvement in English-to-Japanese translation. Hoshino et al. (2013) proposed a similar but rule-based approach for the Japanese-to-English dependency preordering.

4.4.3 Preordering based on Child Swapping

Jehl et al. (2014) proposed a preordering method that implemented n-ary child swapping (Tromble and Eisner, 2009) along a dependency tree with a logistic regression. Similar to the Yang et al. (2012)'s method, they converted a pseudo constituent tree from the dependency tree and then

reordered the converted tree by minimizing the alignment crossing-link number. Their method provided a BLEU +1.47 improvement for English-to-Japanese translation. Its extension to neural networks (de Gispert et al., 2015), and similar approaches for hierarchical phrase-based statistical machine translation (Hayashi et al., 2010; Feng and Cohn, 2013; Zhang et al., 2015) are also available.

4.4.4 Head Finalization for HPSG Preordering

Isozaki et al. (2010b) proposed another rule-based preordering approach called head finalization that uses a head-driven phrase structure grammar (HPSG) (Pollard and Sag, 1994; Sag and Wasow, 1999), which has head and non-head annotations for every binary node in a syntactic constituent tree. Such a HPSG derivation can be obtained by using the Enju parser in English (Miyao and Tsujii, 2005; Miyao and Tsujii, 2008) or in Chinese (Yu et al., 2010).

With a linguistic observation that the Japanese language has exceptionally head-final word order, their method improved English-to-Japanese translation by reversing the non head-final nodes in an English HPSG tree into head-final word order. They also introduced two language-specific rules that delete English articles (a, an, and the) and add virtual Japanese particles. Han et al. (2012) applied this method into Chinese-to-Japanese translation by adding more language-specific rules.

4.4.5 BTG and ITG Preordering

There have been several syntax-based preordering approaches that use inversion transduction grammars (ITGs) (Wu, 1997) for reordering. An ITG that expresses simple word reordering in one language is defined as follows:

$$
\begin{aligned}
A &\rightarrow \langle BC \rangle \\
A &\rightarrow [BC] \\
A &\rightarrow w,
\end{aligned}
$$

where A, B, and C belong to syntactic non-terminals such as phrase labels and pre-terminals; w denotes a word surface form; and the binary rules $\langle BC \rangle$, and $[BC]$ denote the same actions as the reversed and monotone reordering labels defined in Section 4.1.1, respectively.

In contrast to the ITG, its minimal set, which only has one non-terminal symbol X, is called a bracketing transduction grammar (BTG) (Wu, 1997):

$$\begin{aligned} X &\rightarrow \langle XX \rangle \\ X &\rightarrow [XX] \\ X &\rightarrow w, \end{aligned}$$

where the symbol X is no longer a syntactic non-terminal unlike the ITG. Therefore a BTG derivation tree is not equal to a syntactic constituent tree, and it can be made out of source words and word alignment in either bottom-up (Huang et al., 2009) or top-down (Nakagawa, 2015).

In addition, it is not always possible to form both ITG and BTG trees given word alignment, because of inside-out matching (Wu, 1997). In other words, an ITG or BTG tree is said to be satisfying ITG constraint if one can successfully form a tree from word alignment (it indicates that $\tau(\mathbf{a}) = 1$). Na and Lee (2014) showed that the English-Japanese language pair has an exceptionally high percentage (10%) of inside-out sentences, which is a slightly different situation from other language pairs (Wellington et al., 2006) including English-Chinese and English-Korean.

DeNero and Uszkoreit (2011) proposed a BTG preordering method with a three-step approach in order to induce the BTG derivation. Neubig et al. (2012) improved the BTG induction steps as a joint step considering a latent variable BTG derivation. Nakagawa (2015) further improved the joint step with an incremental top-down BTG parsing approach, which uses the latent variable structured perceptron (Sun et al., 2009) in combination with the early update technique (Collins and Roark, 2004) and the inexact beam search (Collins and Roark, 2004; Huang et al., 2012) for the perceptron. As a result, the top-down BTG approach achieved a very fast parsing speed, which enabled the further use of variable features. Since these BTG preordering methods are applicable to any language pair, they have an advantage over the ITG preordering approach,

which requires a source-side or target-side constituent parser.

The formulation of Nakagawa (2015) introduced the ITG constraint as criterion called ITG violation constraint, unlike other BTG methods. The formulation defined a BTG tree *invali*d if it violates the ITG constraint and *valid* otherwise. The training algorithm of Nakagawa (2015) then updates the perceptron based on whether a BTG hypothesis matches to a *valid* BTG. Therefore, in contrast to the proposed method that maximizes Kendall's τ (hence $\tau(\mathbf{a}) \geq 0$), Nakagawa (2015) used *valid* BTG trees (hence $\tau(\mathbf{a}) = 1$) as gold standards.

Goto et al. (2015) took a unique ITG preordering approach that projects a syntactic constituent tree in the target English language obtained by the Enju parser into the source Japanese language. They induced a special ITG parser from the projected tree, which is annotated with oracle reordering labels after the projection and then trained with the Berkeley Parser. The induced ITG parser for the source language is then used to reorder the source input in statistical machine translation. Although their method was proposed earlier than the proposal of Hoshino et al. (2015), a part of their procedure is sharing essentially the same process for obtaining oracle reordering labels as in Section 4.1.2.

4.4.6 Kendall's τ Optimization for Preordering

Kendall's rank correlation coefficient (Kendall, 1938), which is referred to as Kendall's τ and is defined in Section 4.1.2, has been widely used as an automatic evaluation metric in statistical machine translation research (Birch and Osborne, 2010; Isozaki et al., 2010a; Talbot et al., 2011) because this metric is suitable for measuring the word order similarity. Moreover, another metric called the alignment crossing-link number (Genzel, 2010) is equivalent to Kendall's τ when normalized (Nakagawa, 2015). Talbot et al. (2011) showed that Kendall's τ correlates to subjective translation evaluation even at the sentence level (0.371 in Pearson's r and 0.450 in Spearman's ρ). Therefore, optimization of these metrics will lead to improved translation accuracy.

Since Kendall's τ requires combinatorial optimization, efforts have been made to enable op-

timization of Kendall's τ at the sentence level by incorporating tree constraints in the following reordering studies. Yang et al. (2012) showed that the alignment crossing-link number for an n-ary node dependency tree can be expressed in a recursive formula, although they targeted permutation of n-ary node subtrees instead of binary subtrees. In addition, Neubig et al. (2012) formulated Kendall's τ as a sentence-level loss function for a BTG derivation subtree, which consists of a loss for its left subtree, a loss for its right subtree, and an additional loss between both subtrees. We introduced the novel procedure to obtain oracle reordering labels in Section 4.1.2, based on these previous studies, which showed that Kendall's τ at the sentence level can be computed in a recursive manner, as described in Section 4.1.3.

Other metrics, such as the fuzzy reordering score (Talbot et al., 2011), its modified version called chunk fragmentation (Neubig et al., 2012), and ITG violation constraint (Nakagawa, 2015), have also been used in previous studies. However, we found that, unlike Kendall's τ, these alternative metrics cannot be applied to the proposed procedure, because these alternative metrics are not decomposable in a similar way to the proposed formulation. The two metrics based on the fuzzy reordering score are formulated as precision of word bigrams, hence they are not directly decomposable. In addition, the ITG violation constraint makes an assumption that every valid hypothesis satisfies the ITG constraint ($\tau(\mathbf{a}) = 1$), but it is not guaranteed in the proposed formulation ($\tau(\mathbf{a}) \geq 0$). Thus, we need another optimization strategy for these metrics, such as the cube pruning (Neubig et al., 2012) and the beam search (Nakagawa, 2015).

A different step we can take is to use a more relaxed framework than ITGs, such as permutations trees (Stanojević and Sima'an, 2015). Permutation trees can also be used as evaluation metric and is as effective as the fuzzy reordering score and Kendall's τ (Stanojević and Sima'an, 2016).

4.4.7 Postordering

Another technique called postordering (Sudoh et al., 2011; Goto et al., 2012; Hayashi et al., 2013) is developed for Japanese-to-English translation by applying a preordering method devel-

oped for opposite English-to-Japanese translation. Given a reordered English constituent tree as input, which is obtained by applying the head finalization preordering (Isozaki et al., 2010b) to an English tree, they conducted a roughly monotonic translation from the Japanese source input into reordered English using an statistical machine translation system. The translated output in English are then reordered to the original English word order using another statistical machine translation system. Goto et al. (2012) improved the latter reordering step by replacing the statistical machine translation system with ITG preordering using the Berkeley Parser. Hayashi et al. (2013) further improved translation accuracy by developing a syntactic ITG parser specifically trained for the latter step.

4.5 Summary

We proposed a simple yet effective syntax-based preordering method for statistical machine translation based on the method of Li et al. (2007). We introduce a new procedure to obtain reordering labels that are used to train a binary classifier of the previous method, which determines our binary reordering decisions, either monotone or reversed. We showed that oracle reordering labels obtained with the proposed procedure, so as to maximize Kendall's τ, will lead to optimal sentence-wise reordering judged by Kendall's τ owing to its compositional property, as shown in previous studies. We also explored a suitable feature function for the proposed method in order to improve binary classification accuracy. The proposed method is applicable to any language pair that has a source-side constituent parser.

The proposed method is kept simple, but outperformed significantly the previous method in terms of translation accuracy in English-to-Japanese and Japanese-to-English machine translation experiments with a large amount of parallel sentences in the patent domain. In comparison with state-of-the-art reordering methods, the proposed simple method exhibited the best or the second best performance for all evaluation criteria. Such a high performance obtained without language-specific techniques for the distant English-Japanese language pair indicates the importance and effectiveness of the proposed simple method.

Chapter 5

Conclusion

In this thesis, we explored the fundamental problem of machine translation called word ordering, which originates in word order differences between our languages. We empirically studied the problem through experiments with real world large-scale data sets and practical statistical machine translation settings. Two types of solutions were provided in our study: the rule-based preordering approach and the statistical preordering approach.

In the rule-based preordering approach, we proposed a two-stage preordering method and a three-stage preordering method. These methods reorder our input to resemble the order of our target language, by taking manually designed rules into account. Therefore, the rule-based approach is especially useful when we have little or no training data. Nonetheless, using this kind of approach, we showed that it is possible to achieve the state-of-the-art performance in the task of Japanese-to-English translation to date.

In the statistical preordering approach, we proposed a syntax-based preordering method that uses a power of machine learning as a binary classifier of the reordering labels, either monotone or reversed, assigned to each node of binary parse trees. By providing the combination of a novel procedure to obtain oracle reordering labels and a novel set of features to train the binary classifier, the proposed method achieved performance superior to, or at least close to state-of-the-art syntax-based preordering methods to date in the task of English-to-Japanese and Japanese-

to-English patent translation.

The following sections summarize the contributions of the each of approaches and then describe our future work.

5.1 Rule-based Preordering Approach

Both the two-stage method and three-stage method we proposed rely on the linguistic categorization that English has the order of subject-verb-object (SVO), while Japanese has the order of subject-object-verb (SOV). After identifying each of these labels, we can easily construct a set of rules that converts the SOV order of Japanese input into the SVO order of English. Our hypothesis in this rules-based preordering approach was that syntax information plays an important role in generalizing our rules.

With that hypothesis in mind, the two-stage method uses deep syntax information obtained with a predicate-argument structure analyzer. Specifically, we parse an input sentence into predicate-argument structure, using a predicate-argument structure analyzer that can label subject (S), verb (V), and object (O) in the sentence. The labeled sentence is then reordered from the SOV word order to the SVO word order, while the order of coordinated chunks and punctuation was left unchanged. In addition, in order to mitigate the problem of converting postpositional phrases in Japanese into prepositional phrases in English, we swap the order of function words and content words. After that, the reordered Japanese input is fed to a statistical machine translation system to generate its English translation.

In contrast to the two-stage method that heavily depends on predicate-argument structure, the three-stage method uses little or no syntax, making it immune to parsing errors caused by predicate-argument structure analyzer. The three-stage method is designed to mimic the two-stage method by using little or no syntax information, while keeping track of problematic coordination structure in our input as processed in a newly added preliminary stage. After finishing conversion of the SOV order into the SVO order and swapping function words and content words, similar to the two-stage method, again the reordered Japanese is fed to a statistical machine trans-

lation system to generate translation.

By carefuly addressing the problems in existing rule-based preordering methods used in the task of Japanese-to-English translation, such as the handling of coordination structure, the three-stage method achieved the state-of-the-art performance to date in the task using the rule-based preordering approach.

5.2 Statistical Preordering Approach

Knowing the limit of the rule-based approach, we proposed a statistical preordering method based on a syntax framework called inversion transduction grammar (ITG). Specifically, we parse an input sentence into a binary constituent tree using a source-side constituent parser. Each node in the binary parse tree is then binary classified, i.e., either monotone or reversed, using a binary classifier. The tree reordered according to the classified labels is used to yield a reordered source sentence, which is fed to a standard statistical machine translation system to generate translation.

In the course of replacing an exising heuristic procedure to obtain oracle reordering labels with theoretically grounded labels, we introduced a novel procedure to obtain oracle reordering labels so as to maximize Kendall's τ. We showed that our procedure can achieve the global optimization of Kendall's τ, unlike the heuristic procedure. We also introduced novel set of features that directly captures the syntactic relation in each node, which is obtained as a result of our feature engineering focused on improving binary classification accuracy.

5.3 Future Work

Future work of this study can be easily extended to cover more syntactic structures such as a bracketing transduction grammar, with various parsing techniques including advances in structural learning. Since parsing and reordering improvements always make machine translation accuracy better, the best of the both worlds would contribute well in the future.

To conclude, our approaches employed syntax-based preordering for statistical machine translation that utilizes syntactic information obtained with syntactic parsers. We showed that syntax-

based preordering approaches can maintain the simplicity of our entire system with tree structures, while we effectively manipulate word orders by simply modifying the same tree structures. Such simpleness and effectiveness of syntax-based preordering enabled us to overcome the main difficulties of the word ordering problem.

As a result, we eventually achieved translation accuracy comparable, or superior to, state-of-the-art methods with a simple preordering method without language-specific techniques in the tasks of English-to-Japanese and Japanese-to-English translations, while they are regarded as one of the most difficult language pairs in the world in terms of word ordering. Our success prospects possibilities of much better practical machine translation systems in reality.

Bibliography

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Alexandra Birch and Miles Osborne. 2010. LRscore for evaluating lexical and reordering quality in MT. In *Proceedings of the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR*, pages 327–332.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Peter F. Brown, Peter V. de Souza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992a. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Peter F. Brown, Vincent J. Della Pietra, Robert L. Mercer, Stephen A. Della Pietra, and Jenifer C. Lai. 1992b. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1).

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

Keith Brown, editor. 2005. *Encyclopedia of Language and Linguistics*. Elsevier, second edition.

Alastair Butler, Tomoko Hotta, Ruriko Otomo, Kei Yoshimoto, Zhen Zhou, and Hong Zhu. 2012. Keyaki Treebank: Phrase structure with functional information for Japanese. In *Proceedings of Text Annotation Workshop*.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*.

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

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 111–118.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 531–540.

Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. 2015. Fast and accurate preordering for SMT using neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics and the Human Language Technologies*, pages 1012–1017.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–38.

Yasuharu Den, Junpei Nakamura, Toshinobu Ogiso, and Hideki Ogura. 2008. A proper approach to Japanese morphological analysis: Dictionary, model, and evaluation. In *Proceedings of the The 6th International Conference on Language Resources and Evaluation*, pages 1019–1024.

John DeNero and Jakob Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Chenchen Ding, Masao Utiyama, and Eiichiro Sumita. 2015. Improving fast_align by reordering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1034–1039.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Tsaiwei Fang, Alastair Butler, and Kei Yoshimoto. 2014. Parsing Japanese with a PCFG treebank grammar. In *Proceedings of the 20th Meeting of the Association for Natural Language Processing*, pages 432–435.

Yang Feng and Trevor Cohn. 2013. A Markov model of machine translation using non-parametric Bayesian inference. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 333–342.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57.

Dmitriy Genzel. 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.

Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K. Tsou. 2011. Overview of the patent machine translation task at the NTCIR-9 workshop. In *Proceedings of the 9th NTCIR Workshop Meeting*, pages 559–578.

Isao Goto, Masao Utiyama, and Eiichiro Sumita. 2012. Post-ordering by parsing for Japanese-English statistical machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 311–316.

Isao Goto, Ka Po Chow, Bin Lu, Eiichiro Sumita, and Benjamin K. Tsou. 2013. Overview of the patent machine translation task at the NTCIR-10 workshop. In *Proceedings of the 10th NTCIR Workshop Meeting*, pages 260–286.

Isao Goto, Masao Utiyama, Eiichiro Sumita, and Sadao Kurohashi. 2015. Preordering using a target-language parser via cross-language syntactic projection for statistical machine translation. *Transactions on Asian and Low-Resource Language Information Processing*, 14(3):13:1–13:23.

Dan Han, Katsuhito Sudoh, Xianchao Wu, Kevin Duh, Hajime Tsukada, and Masaaki Nagata. 2012. Head finalization reordering for Chinese-to-Japanese machine translation. In *Proceedings of the 6th Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 57–66.

Katsuhiko Hayashi, Hajime Tsukada, Katsuhito Sudoh, Kevin Duh, and Seiichi Yamamoto. 2010. Hierarchical phrase-based machine translation with word-based reordering model. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 439–446.

Katsuhiko Hayashi, Katsuhito Sudoh, Hajime Tsukada, Jun Suzuki, and Masaaki Nagata. 2013. Shift-reduce word reordering for machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1382–1386.

Hieu Hoang, Philipp Koehn, and Adam Lopez. 2009. A unified framework for phrase-based, hierarchical, and syntax-based statistical machine translation. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 152–159.

Stephen Wright Horn, Alastair Butler, and Kei Yoshimoto. 2017. Keyaki Treebank segmentation and part-of-speech labelling. In *Proceedings of the 23th Meeting of the Association for Natural Language Processing*, pages 414–417.

Sho Hoshino, Yusuke Miyao, Katsuhito Sudoh, and Masaaki Nagata. 2013. Two-stage pre-ordering for Japanese-to-English statistical machine translation. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 1062–1066.

Sho Hoshino, Yusuke Miyao, Katsuhito Sudoh, and Masaaki Nagata. 2014. A preordering method robust to parsing errors for Japanese-to-English statistical machine translation. In *Proceedings of the 20th Meeting of the Association for Natural Language Processing*, pages 432–435 (in Japanese).

Sho Hoshino, Yusuke Miyao, Katsuhito Sudoh, Katsuhiko Hayashi, and Masaaki Nagata. 2015. Discriminative preordering meets Kendall's τ maximization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 139–144.

Liang Huang, Hao Zhang, Daniel Gildea, and Kevin Knight. 2009. Binarization of synchronous context-free grammars. *Computational Linguistics*, 25(4):559–595.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics and the Human Language Technologies*, pages 142–151.

Ryu Iida and Massimo Poesio. 2011. A cross-lingual ILP solution to zero anaphora resolution. In *Proceedings of the 2011 Conference of the 49th Annual Meeting of the Association for Computational Linguistics and the Human Language Technologies*, pages 804–813.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proceedings of the Linguistic Annotation Workshop*, pages 132–139.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010a. 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.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010b. Head finalization: A simple reordering rule for SOV languages. In *Proceedings of the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR*, pages 244–251.

Hideki Isozaki. 2013. OkaPU's Japanese-to-English translator for NTCIR-10 PatentMT. In *Proceedings of the 10th NTCIR Workshop Meeting*, pages 348–349.

Laura Jehl, Adrià de Gispert, Mark Hopkins, and Bill Byrne. 2014. Source-side preordering for translation using logistic regression and depth-first branch-and-bound search. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 239–248.

Jason Katz-Brown and Michael Collins. 2008. Syntactic reordering in preprocessing for Japanese→English translation: MIT system description for NTCIR-7 patent translation task. In *Proceedings of the 7th NTCIR Workshop Meeting*.

Daisuke Kawahara and Sadao Kurohashi. 2006a. Case frame compilation from the Web using high-performance computing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 1344–1347.

Daisuke Kawahara and Sadao Kurohashi. 2006b. A fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. In *Proceedings of the 2006 Conference of the Human Language Technologies and the North American Chapter of the Association for Computational Linguistics*, pages 176–183.

Daisuke Kawahara, Sadao Kurohashi, and Kôiti Hasida. 2002. Construction of a Japanese relevance-tagged corpus. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 2008–2013.

Yuki Kawara, Chenhui Chu, and Yuki Arase. 2018. Recursive neural network based preordering for English-to-Japanese machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 21–27.

Maurice George Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing*, pages 181–184.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the Human Language Technologies and the North American Chapter of the Association for Computational Linguistics*, pages 48–54.

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

Philipp Koehn. 2004a. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the 2004 Conference of the Association for Machine Translation in the Americas*, pages 115–124.

Philipp Koehn. 2004b. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

Mamoru Komachi, Yuji Matsumoto, and Masaaki Nagata. 2006. Phrase reordering for statistical machine translation based on predicate-argument structure. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 77–82.

Shunsuke Kozawa, Kiyotaka Uchimoto, and Yasuharu Den. 2014. Adaptation of long-unit-word analysis system to different part-of-speech tagset. *Journal of Natural Language Processing*, 21(2):379–401 (in Japanese).

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of the 6th Conference on Natural Language Learning*, pages 63–69.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237.

Taku Kudo, Hiroshi Ichikawa, and Hideto Kazawa. 2014. A joint inference of deep case analysis and zero subject generation for Japanese-to-English statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 557–562.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 66–75.

Sadao Kurohashi and Daisuke Kawahara, 2012. *Japanese Morphological Analysis System JUMAN 7.0 Users Manual*.

Ching-Pei Lee and Chih-Jen Lin. 2014. Large-scale linear rankSVM. *Neural Computation*, 26(4):781–817.

Uri Lerner and Slav Petrov. 2013. Source-side classifier preordering for machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 513–523.

Chi-Ho Li, Minghui Li, Dongdong Zhang, Mu Li, Ming Zhou, and Yi Guan. 2007. A probabilistic approach to syntax-based reordering for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 720–727.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. Marneffe, B. Maccartney, and C. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 449–454.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 83–90.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80.

Hwidong Na and Jong-Hyeok Lee. 2013. A discriminative reordering parser for IWSLT 2013. In *Proceedings of the 10th International Workshop on Spoken Language Translation*, pages 83–86.

Hwidong Na and Jong-Hyeok Lee. 2014. Linguistic analysis of non-ITG word reordering between language pairs with different word order typologies. *Transactions on Asian and Low-Resource Language Information Processing*, 13(3):11:1–11:12.

Makoto Nagao. 1984. A framework of a mechanical translation between Japanese and English by analogy principle. In *Proceedings of the International NATO Symposium on Artificial and Human Intelligence*, pages 173–180.

Tetsuji Nakagawa. 2015. Efficient top-down BTG parsing for machine translation preordering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 208–218.

Graham Neubig and Kevin Duh. 2014. On the elements of an accurate tree-to-string machine translation system. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 143–149.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. 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.

Graham Neubig. 2013. Travatar: A forest-to-string machine translation engine based on tree transducers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 91–96.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

Yusuke Oda, Taku Kudo, Tetsuji Nakagawa, and Taro Watanabe. 2016. Phrase-based machine translation using multiple preordering candidates. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 1419–1428.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of the 2007 Conference of Human Language Technologies and the North American Chapter of the Association for Computational Linguistics*, pages 404–411.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440.

Carl Pollard and Ivan Andrew Sag. 1994. *Head-driven Phrase Structure Grammar*. University of Chicago Press.

Jason Riesa, Ann Irvine, and Daniel Marcu. 2011. Feature-rich language-independent syntax-based alignment for statistical machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 497–507.

Ivan Andrew Sag and Thomas Wasow. 1999. *Syntactic Theory: A Formal Introduction*. Center for the Study of Language and Information Publications, Stanford University.

Ryohei Sasano and Sadao Kurohashi. 2009. A probabilistic model for associative anaphora resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1455–1464.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to Japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 758–766.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Harold Somers Sergei Nirenburg and Yorick Wilks, editors. 2003. *Readings in Machine Translation*. MIT Press.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656.

Claude Elwood Shannon. 1951. Prediction and entropy of printed English. *Bell System Technical Journal*, 30(1):50–64.

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, and S.V.N. Vishwanathan. 2009. Hash kernels for structured data. *Journal of Machine Learning Research*, 10:2615–2637.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 2006 Conference of the Association for Machine Translation in the Americas*, pages 223–231.

Miloš Stanojević and Khalil Sima'an. 2015. Reordering grammar induction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 44–54.

Miloš Stanojević and Khalil Sima'an. 2016. Hierarchical permutation complexity for word order evaluation. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 2164–2173.

Andreas Stolcke. 2002. SRILM: An extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 901–904.

Katsuhito Sudoh, Xianchao Wu, Kevin Duh, Hajime Tsukada, and Masaaki Nagata. 2011. Post-ordering in statistical machine translation. In *Proceedings of the Machine Translation Summit XIII*, pages 316–323.

Xu Sun, Takuya Matsuzaki, Daisuke Okanohara, and Jun'ichi Tsujii. 2009. Latent variable perceptron algorithm for structured classification. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 1236–1242.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 3104–3112.

David Talbot, Hideto Kazawa, Hiroshi Ichikawa, Jason Katz-Brown, Masakazu Seno, and Franz Josef Och. 2011. A lightweight evaluation framework for machine translation reordering. In *Proceedings of the 6th Workshop on Statistical Machine Translation*, pages 12–21.

Christoph Tillman. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of the 2004 Conference of Human Language Technologies and the North American Chapter of the Association for Computational Linguistics*, pages 101–104.

Roy Tromble and Jason Eisner. 2009. Learning linear ordering problems for better translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1007–1016.

Masao Utiyama and Hitoshi Isahara. 2003. Reliable measures for aligning Japanese-English news articles and sentences. In *Proceedings of the 41st Annual Meeting of the Association for Computatieonal Linguistics*, pages 72–79.

David Vilar, Jia Xu, D'Haro Luis Fernando, and Hermann Ney. 2006. Error analysis of statistical machine translation output. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 697–702.

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 977–984.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 508–514.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530.

Nan Yang, Mu Li, Dongdong Zhang, and Nenghai Yu. 2012. A ranking-based approach to word reordering for statistical machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 912–920.

Kun Yu, Yusuke Miyao, Xiangli Wang, Takuya Matsuzaki, and Junichi Tsujii. 2010. Semi-automatically developing Chinese HPSG grammar from the Penn Chinese treebank for deep parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1417–1425.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, and Hai Zhao. 2015. Learning word reorderings for hierarchical phrase-based statistical machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 542–548.