Karlsruhe Institute of Technology

Rule-Based Preordering on Multiple Syntactic Levels in Statistical Machine Translation

Master Thesis of

Ge Wu

At the Department of Informatics
Institute for Anthropomatics and Robotics (IAR)

Advisor: Alex Waibel
Second Advisor: Yuqi Zhang

Duration: 1st March 2014 – 31st August 2014

Abstract

We propose a novel data-driven rule-based preordering approach, which uses the information of syntax tree and word alignment to reorder the words in source sentences before decoding in a phrase-based SMT system between English and Chinese. The preorering algorithm extracts reordering patterns from multiple levels of the syntax trees of the training data and applies the rules on the source sentences in a similar manner.

We've conducted experiments in English-to-Chinese and Chinese-to-English translation directions. Our results show that the approach has led to improved translation quality both when it was applied separately on the baseline or when it was combined with some other reordering approaches. We report an improvement of 0.43 in BLEU score when our preordering approach was used in addition to the short rule (Rottmann and Vogel, 2007), long rule(Niehues and Kolss, 2009) and tree rule (Herrmann et al., 2013b) based preordering approaches in the English-to-Chinese translation direction. There's also an improvement of 0.3 in BLEU score when our preordering approach was used in addition to the aforementioned preordering approaches in the Chinese-to-English translation direction. Through analyzing the translations, we've also found improvements in syntactic structure with our preordering approach.

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

31st August 2014

Ge Wu

Contents

1. Introduction

1.1. Motivation

Word order is a general issue when we want to translate text from one language to the other. Different languages normally have different word orders and the difference could be tremendous, when two languages come from different language families, such as English and Chinese. Depend on the languages, different word orders could have very distinguish features. For example, 45% of the languages in the world are Subject-Object-Verb (SOV) languages. Unlike in English, verbs are put after object in these languages. Japanese is a popular language among them. Instead of saying *The black cat climbed to the tree top*, people would say *The black cat the tree top to climbed* in Japanese. Another example is Spanish, in which people often put the adjective after the modified nouns. Following example shows how people would order the words differently (Lerner and Petrov, 2013).

English	The black cat climbed to the tree top.
Japanese	The black cat the tree top to climbed.
Spanish	The cat black climbed to the top tree.

Among all the languages, Chinese is one language which is very different from English, because they belong to different language families and have long period of separately development. Both languages have a SOV order, but they also have a lot of differences in word order. Especially sentences in both languages can sometimes have completely different syntactic structures. The differences may involve long-distance or unstructured position changes.

Most state-of-the-art phrase-based SMT systems use language model, phrase table or decoder to adjust the word order. Or an additional reordering model is used in the log-linear model for word reordering. However, these methods may have some disadvantages, such as some don't handle long-distance reordering, some don't handle unstructured reordering and some are rather time consuming.

Encouraged by the results from the paper Rottmann and Vogel (2007), Niehues and Kolss (2009) and Herrmann et al. (2013b), we further propose a new data-driven, rule-based preordering method, which extracts and applies reordering rules based on syntax tree. The method is called MLT reordering, which orders the constituents on multiple levels of the syntax tree all together. This preordering method rearranges the words in source language into a similar order as they are supposed to be in the target language before translation. With the appropriate word order, better translation quality can be achieved. Especially, our preordering method is very suitable for translation between language pairs like English and Chinese, which have very different word orders. Besides, the method can also be combined with the above mentioned rule-based reordering methods to achieve better translation quality.

To be more accurate, Chinese is referred to Mandarin Chinese throughout this thesis, which is the official language in People's Republic of China and standardized by its government.

1.2. Objective and Contribution

So the objective of this thesis can be defined as follows:

> We establish a new data-driven rule-based preordering method for translation between English and Chinese, which is based on multi-level syntactic information. The method reorders the source text of a SMT system before translation by using the information from word alignments, Part-of-Speech (POS) tags and syntax trees of the training data. Also we evaluate this method by checking the resulted translation quality and by comparing it with some other rule-based preordering method.

The ground of this thesis are three papers about rule-based preordering: Rottmann and Vogel (2007), Niehues and Kolss (2009) and Herrmann et al. (2013b). While their reordering methods are primarily designed and optimized for German or other languages with similar characteristics, they are not necessarily suitable for Chinese translation, which is a language that belongs to a completely different language family and has some very distinct features. Or at least, there may be still much space for improvement in Chinese translation.

In this context, we further explore the possibilities for a more suitable reordering method for Chinese. And we propose the MLT reordering method, which extracts the rules by detecting position change of constituents on multiple levels of subtrees in syntax trees from parallel training data. And guided by the rules, we can reorder the new text by examining subtrees of the same structure.

We will also evaluate our reordering method and compare it with some other these methods. Through the evaluation and comparison, we'll have a thorough understanding of what our approach can achieve.

1.3. Related Work

Word reordering is an important problem for statistical machine translation, which has long been addressed.

In a phrase-based SMT system, there are several possibilities to change the word orders. Words can be reordered during the decoding phase by setting a window, which allows the decoder to choose the next word for translation. Reordering could also be influenced by the language model, because the language model give probability of how a certain word is likely to follow. Different language model may give different probability, which further influences the decision made by log-linear model. Other ways to change the word orders include using distance based reordering models or lexicalized reordering models (Tillmann, 2004; Koehn et al., 2005). The lexicalized reordering model reorders the phrases by using information of how the neighboring phrases change orientations.

The hierarchical phrase-based translation model (Chiang, 2007) is especially suitable for Chinese translation, and provide very good translation results. It extracts hierarchical rules by using information of the syntactic structure. Phrases from different hierarchies, or so-called phrases of phrases, are reordered during the decoding.

The idea of phrases on different hierarchies has inspired us to create this preordering method based on multiple levels of the syntax tree. Besides, we also hope to detach the reordering from decoding phase and do it separately in a pre-process before decoding, in order to reduce the time for translation. This kind of preordering approaches use linguistic information to modify the word orders.

Reordering approaches can also be rule-based, which extracts different types of reordering rules by observing reordering patterns from the training data and apply the rules to the sentences to be translated. Depends on how the rules are defined, different information may be used such as word alignments, POS tags, syntax trees, etc.

Some early approaches use manually defined reordering rules based on the linguistic information for particular languages (Collins et al., 2005; Popovic and Ney, 2006; Habash, 2007). Especially Wang et al. (2007) is based on Chinese and has analyzed the reordering cases between English and Chinese based on syntactic structure. Later come the data-driven methods (Zhang et al., 2007; Crego and Habash, 2008), which learn the reordering rules automatically.

Rottmann and Vogel (2007) introduced the idea of extracting reordering rules from the POS tag sequences of training data and use them for reordering. Niehues and Kolss (2009) went further, and developed a method for long-distance word reordering, which works good on German-English translation task due to the long-distance shift of verbs. The method extracts discontinuous reordering rules in addition to the continuous ones, which contains a placeholder to match several words and enables the word to shift cross long distance.

Afterwards, Herrmann et al. (2013b) introduced a novel approach to reorder the words based on syntax tree, which leads to further improvements on translation quality. The algorithm takes syntactic structure of the sentences into account and extract the rules from the syntax tree by detecting the reordering of child sequences. It also has the variant based only on part of the child sequences which is suitable for language with flat syntactic structure such as German.

Recently, Lerner and Petrov (2013) introduced a novel classified preordering approach. Unlike existing preordering models, it trains feature-rich discriminative classifiers that directly predict the target side word order. It reorders the children in subtrees in a similar manner as the syntax tree based method, but trains different classifiers for nodes with different number of children.

However, these approaches which are bases on POS tag sequences or syntax trees are mostly designed for German and are not especially adapted for Chinese translation. As Chinese has very different word orders, a reordering approach, which can further explore the syntactic structure of Chinese and utilize this information for reordering, is desirable.

Oracle reordering has also shown values for evaluating the potential of preordering. Birch et al. (2010) introduced the permutation distance metrics which can be used to measure reordering quality. And Birch (2011) described how we can construct permutations from the word alignment as oracle reordering.

1.4. Structure

In this thesis, we first explain some fundamental concepts and knowledge in chapter 2, which are relevant for the understanding of this thesis. Then we introduce our reordering methods in detail in chapter 3, including the problems of translating between English and Chinese and the motivation of our reordering approach. The results and evaluation of our method are presented in chapter 4. In chapter 5 we conclude this work with an overall discussion of our approach and results. And we also point out the possible directions for future research.

2. Foundations

This chapter provides an introduction to fundamental knowledge and concepts that are relevant to this thesis. We start with the overall SMT system and preordering system first, then followed by the information we used to create the reordering rules including alignment, POS tag, syntax tree. At the end we shows the different rule types, the oracle reordering, the way to build word lattices and the evaluation metrics.

Detailed Descriptions of word alignment, POS tagging, syntax tree, reordering rules, oracle reordering, word lattices and evluation metrics can be found in the corresponding sections. Koehn (2010) also provides a good introduction to statistical machine translation in general, including different kinds of theories and methods that are relevant to this work. The preordering approach we used for rule extraction and application is introduced in the chapter 3.

2.1. Statistical Machine Translation (SMT) System

Statistical Machine Translation (SMT) is the state-of-the-art paradigm for machine translation. It uses a typical log-linear model which is composed of a decoder and different statistical models including phrase table, reordering model and language model. All the models are weighted with parameters which are tuned from the development data. Besides development data, training data are used for training the alignment, phrase table and other models. And test data are used for evaluation purpose. In phrase-based SMT system sequences of words are used as basic blocks for translation. The phrases are found by using statistical methods from training corpus. The architecture of a SMT system could be illustrated as figure 2.1.

2.2. Rules Based Preodering

Our preordering method is based on reordering rules. Reordering rules show how sentences should be reordered in source language before translation. In our system, the rules are generated by using the word alignment, POS tags and syntax tree, all of which are calculated based on the training data. After reordering rules are applied to the source sentences, word lattices are generated. A word lattice contains all the reordering possibilities of a source sentence and is further passed to the decoder for translating. The preordering system could be illustrated as figure 2.2.

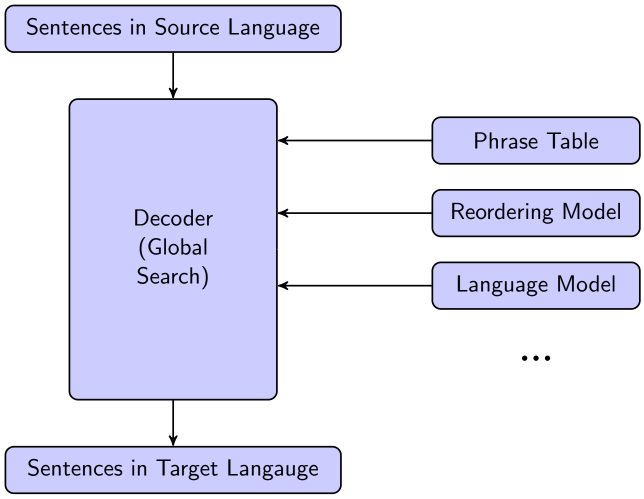


Figure 2.1.: Architecture of a SMT system

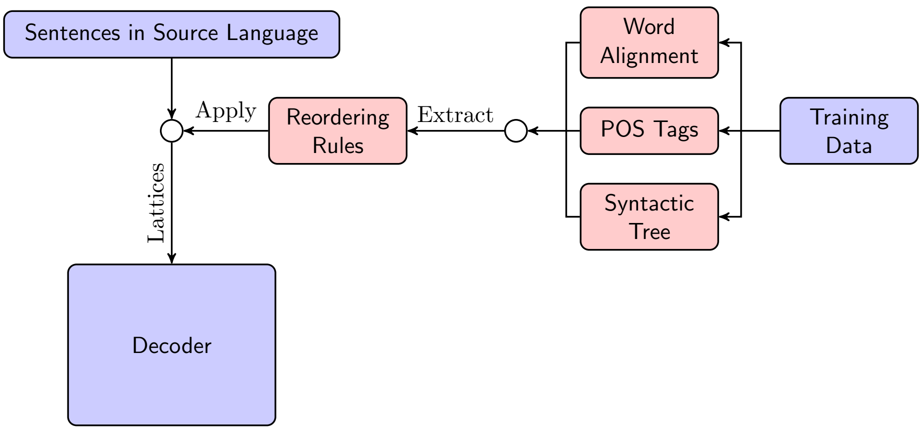


Figure 2.2.: Preordering system

2.3. Word Alignment

Word alignment indicates the possible alignment between words in the source sentences and words in the target sentences. For example, figure 2.3 shows a word alignment between an English sentence and a Chinese sentence, which is generated by using the *GIZA++* toolkit*.

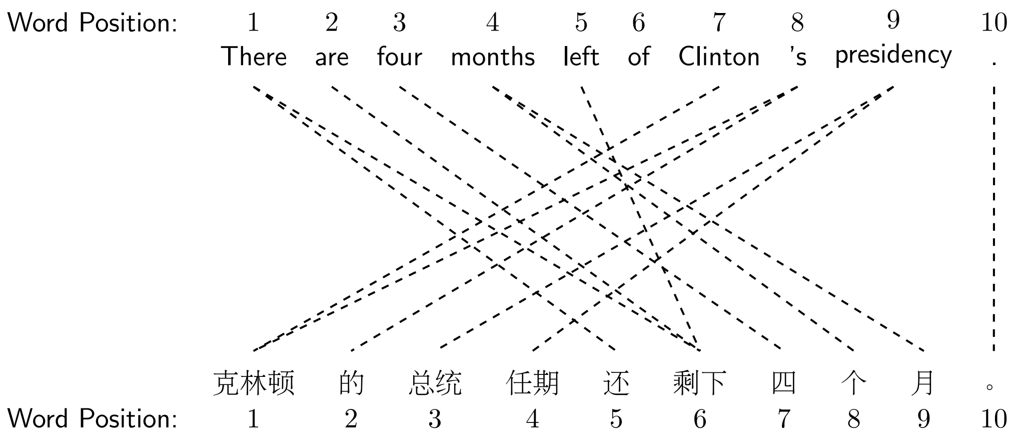


Figure 2.3.: Word alignment

Or it may be simply presented as index pairs:

```
1-5 1-6 2-6 3-7 4-8 4-9 5-6 7-1 8-1 8-2 9-3 9-4 10-10
```

In figure 2.3, the words that are aligned through lines in two languages have related meaning. The word reordering can be clearly seen from the figure. For example, the noun clause *Clinton's presidency* is moved forward to the front of the sentence as a whole.

Word alignment doesn't always present one-to-one word matching. In the example, the word *of* is not aligned at all and the word *months* is aligned to two words 个 and 月 in Chinese. We can define the **aligned range** as the range from the first word a certain word is aligned to to the last word it is aligned to. For example, the aligned range of word *there* is $5-6$. The aligned ranges can collide with each other, such as the 6th word 剩下 in Chinese is also aligned to *are* and *left* at the same time besides *there*, so the aligned ranges of the three words collide. The collision sometimes makes the detection of reordering patterns more difficult, because the word order can not be clearly decided in these cases.

The word alignment could be trained with the GIZA++ tool by using Expectation-Maximization (EM) algorithm. From the word alignment of parallel data, reordering patterns of how the words are reordered between the source language and target language can be detected. Therefore, reordering rules can be extracted from the corpus and applied to the text to be translated.

2.4. Part-of-Speech Tagging

Part-of-Speech (POS) tags are markups of words in the text, which indicates the syntactic role of part of the speech. The markups are based on words' definitions and their context.

*GIZA++ toolkit: `http://code.google.com/p/giza-pp`

1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	IN	Preposition/subordinating conjunction
5.	JJ	Adjective
6.	JJR	Adjective, comparative
7.	JJS	Adjective, superlative
8.	MD	Modal verb
9.	NN	Noun, singular or mass
10.	NNS	Noun, plural
11.	RB	Adverb
12.	VB	Verb, base form
13.	VBP	Verb, non-3rd person singular present
14.	VBZ	Verb, 3rd person singular present
15.	WRB	*wh*-adverb
16.	.	Sentence-final punctuation
17.	ADJP	Adjective phrase
18.	ADVP	Adverb phrase
19.	NP	Noun phrase
20.	PP	Prepositional phrase
21.	QP	Quantity phrase
22.	S	Simple declarative clause
23.	VP	Verb phrase

Table 2.1.: Penn Treebank tagset

Figure 2.4 shows a sentence with the POS tags. Table 2.1 lists part of the Penn Treebank tagset for quick reference. A complete list can be viewed in appendix B.

The	domestic	consumption	market	for	animal	products	is	very	great	.
DT	JJ	NN	NN	IN	NN	NNS	VBZ	RB	JJ	.

Figure 2.4.: POS tagging

2.5. Syntax Tree

The syntax tree shows the syntactic structure of a sentence and can be very useful for word reordering. A syntax tree contains two kinds of nodes: the leaves and the internal nodes. Each leaf presents a word in the sentence, and is annotated with a POS tag. And each internal node presents a constituents, which is also annotated to indicate its category or syntactic role. In the Penn treebank (Marcus et al., 1993; Santorini, 1990), for example, the annotation *NP* means noun phrase and the annotation *S* means simple declarative clause. Figure 2.5 is an example of a syntax tree. In the example, leaf nodes are presented by red rectangles and internal nodes are presented by blue rectangles with rounded corners.

We can see the syntactic structure of the sentence from the syntax tree very clear. In this example, The words *math and biology exams* make up a noun clause, which plays the roll of subject. The predicate has a nested structure of verb clauses, because it contains the modal verb *will* and the verb *be*. And *on the* 27*th* is a preposition clause nested in the verb clause, which is again composed of a preposition *in* and a noun clause *the* 27*th*.

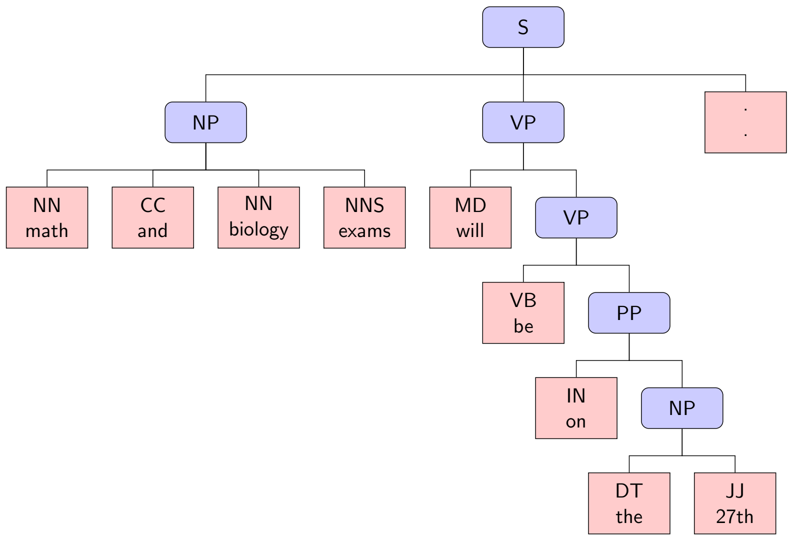


Figure 2.5.: An example of parse tree

2.6. Reordering Rules

Based on Rottmann and Vogel (2007), Niehues and Kolss (2009), Herrmann et al. (2013b) and Herrmann et al. (2013a), we introduce the different rule types, rule combination, how to decide if rules can be extracted as well as how to calculate the probability of the reordering rules.

2.6.1. Short Rules

Short rules are extracted based on the sequences of adjacent words or their POS tags in the sentence from training data. Sequences of adjacent words or tags are observed, rules are then extracted if the same reordering patterns appear frequently. Following are some examples:

```
after the accident -> the accident after (0.5)

WRB MD DT -> DT WRB MD (0.3)
```

The first rule in this example shows, if the word sequence *after the accident* appears in the text, it should be reordered to *the accident after* with a probability 0.5, so the word order will be more consistent with the translation. The second rule shows, if the word sequence of a *wh*-adverb (*when*, *where*, *why*, etc.), a modal verb (*MD*) and a determiner (*DT*) appears, the determiner should be moved before the *wh*-adverb with a probablity of 0.3.

In addition, Short rules have some different varieties (Rottmann and Vogel, 2007):

- **Tag sequence:** rules are extracted based on adjacent tag sequence
- **Word sequence:** rules are extracted based on adjacent word sequence
- **Context of one or two tags before and/or after the tag sequence**
- **Context of one or two words before and/or after the tag sequence**

2.6.2. Long Rules

Long rules are specially designed to improve the long distance word reordering for translation between English and German. The rules are based on POS tags of the text, and following is an exapmle:

```
NN X MD : VBP -> X MD NN (0.14)
```

The *X* in the example is a placeholder, which presents one or more words. *VBP* is the right context, which create a restriction for applying this rule and is sometimes helpful to define the reordering boundary. *NN* means noun, *MD* means modal verb and *VBP* is the word *have*. In this example, the tag sequence *NN X MD* with right context *VBP* should be reordered as sequence *X MD NN* with a likelihood of 0.14.

Rules are extracted by first finding the location of the reordering rule and then putting the placeholder. Depends on the location, where the placeholder is put, and how much the placeholder replace, the long rules also have some varieties:

- **Left/right rules:** depends on if the placeholder is put on the left part or right part
- **All/part replacement:** depends on if the placeholder replaces all the words in a part

2.6.3. Tree Rules

While short rules and long rules are based on the flat structure of sentences, tree rules reorder sentences by using information from sentences' syntactic structure. The syntax tree and word alignment of the training corpus are used to train the reordering rules. The tree rules reorder the words both on the word level and on the constituent level. Following is an example:

```
NP ( ADJP JJ NN ) -> JJ NN ADJP (0.16)
```

The parenthesis in the example represents the hierarchy in the syntax tree. The left side of the rule corresponds a tree with the root labeled with *NP* and three children, each labeled with *ADJP*, *JJ* and *NN*. When this structure appears as a subtree in the syntax tree of the sentence to translate, the order of its subtrees should be changed into *JJ*, *NN* and *ADJP* with probability 0.16. The change is illustrated in figure 2.6.

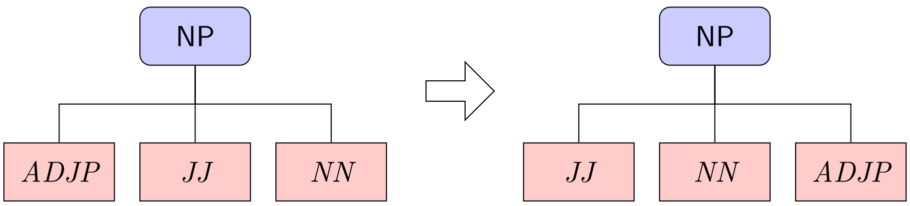


Figure 2.6.: Order change of children based on tree rules

Tree rules have some different varieties too:

- **Partial Rules:** the relatively flat syntactic structure of languages like German may make the rule extraction difficult, because the extraction requires that the whole subtree including all its children is matched. In order to extract more useful information for reordering, rules are also extracted from any partial child sequence in a constituent.
- **Recursive rule application:** the rules may be applied recursively to already reordered sentence. And all paths of the reorderings are added to the lattice.

2.6.4. Rule Extraction and Application

Rules are extracted by scanning all the training data and detecting the reordering patterns. A valid reordering pattern that can count for a reordering rule needs to fulfill the two requirements in general:

- **Order change exists:** otherwise, there's no need for reordering rules.

- **Aligned ranges don't collide:** the collision makes it hard to decide new word orders in the target language.

Reordering rules are not always extracted upon discovering of reordering pattern. In order to avoid too excessively concrete rules which can't be applied well in general, we extract reordering rules only when the same reordering pattern appears more than a certain threshold, so it won't lead to overfitting.

The associated probability of a reordering rules is the frequency how often the sequence in the rules are reordered in the same manner. For example, if the sequence *after the accident* appears certain times in the training data, by half of it's appearance, it's reordered as *the accident after*, then the probability of this reordering rule is 50%.

Rules are applied by scanning the text to be translated. When there's a sequence coincides the left side of the reordering rules, rules will be applied, and a path in the word lattice representing the reordered sentence will be added.

2.6.5. Rule Combination

In order to further explore the probability of improvement, different types of reordering rules can be combined to achieve better translation. This is done by training the different types of rules separately and applying them on the monotone path of the sentence independently. They generate different paths in the word lattice.

2.7. Oracle Reordering

In order to evaluate the potential of word reordering, we introduce the oracle reordering. Oracle reordering is considered to be an optimally reordered sentence as input to the SMT system and do not allow additional reordering during decoding. (Herrmann et al., 2013a) The oracle reordering is created by using the permutation of source sentences, which is extracted from the word alignment between the source text and reference.

In order to abstract the permutation from the word alignment, some cases need to be considered, since word alignment is generally not a one-to-one word mapping. There are the following four cases: (Birch, 2011)

- **Unaligned source words:** are assigned to the target word position immediately after the target word position of the previous source word, or to position 1 if they are at the beginning of source sentences
- **Unaligned target words:** are ignored
- **Many-to-one alignment:** the target ordering is assumed to be monotone
- **One-to-many alignment:** the source word is assumed to be aligned to the first target word

Because it is considered as an optimally reordering, we can use it as input of the SMT system and the scores are the optimal results that can be achieved by word reordering, from which we can evaluate the potential of reordering methods.

2.8. Word Lattice

A word lattice could be presented with a directed acyclic graph. The graph contains nodes and transitions, with each transition labeled with a word and a probability. The outgoing transitions from a node indicate different options, which words can come after this point. The annotation on the transition indicates the word that can be in that position, together with the probability of this option. Each reordering of a sentence corresponds a path from the beginning node to the end node in a word lattice.

Word lattice for reordered word sequences is build gradually while applying the reordering rules on the sentence. It starts with a monotone path presenting the sentence to be translated. Every time when a rule is applied and part of the sentence is reordered. We add a parallel path to the corresponding part of the initial monotone path. The parallel path is labeled with reordered words on its transitions. The probability of this new reordering is subtracted from the first transition after the splitting point on monotone path, and assigned to the first transition of the new path. All the other transitions on the new path that follow have a probability of 1.

Paths with very low probability are removed, in order to save space for storing the lattice and reduce decoding time later, without compromising too much translation quality.

An example of a word lattice is showed in figure 2.7. In the figure, if the probability of a transition is 1, the label of probability is omitted.

2.9. Evaluation Metrics

Bilingual Evaluation Understudy (BLEU) is an algorithm to evaluate the translation quality of machine-translated text. It shows a high correlation with human judgments of translation quality (Papineni et al., 2002), and remains one of the most popular metrics in statistical machine translation.

As described in Birch et al. (2010): BLEU is the de facto standard in machine translation. It captures the n-gram precision of the translation. Shorter n-gram precision captures the lexical coverage of the translation and word order is evaluated by the higher order n-grams. The final score is an interpolation of these precisions and it is adjusted by a brevity penalty. BLEU score is always a number between 0 and 1. This value measures how close the translation and reference are, with 1 indicting the two are identical.

Translation Edit Rate (TER) measures the amount of editing that has to be performed to change a system output so it matches the reference. TER is adequate for research purposes as it correlates reasonably well with human judgments (Snover et al., 2006).

Both the BLEU and TER of the SMT systems are measured in our experiments.

2.10. Summary

In this chapter, we've introduced some fundamental knowledge and concepts that are relevant to this thesis, which include the architecture of a SMT system, the preordering system, word alignment, POS tagging, syntax tree, different types of reordering rules, oracle reordering, word lattice and evaluation metrics. For the reordering rules, we've introduced three different types: short rules, long rules and tree rules. In the following chapters, we'll introduce our MLT reordering rules and compare these different reordering rules.

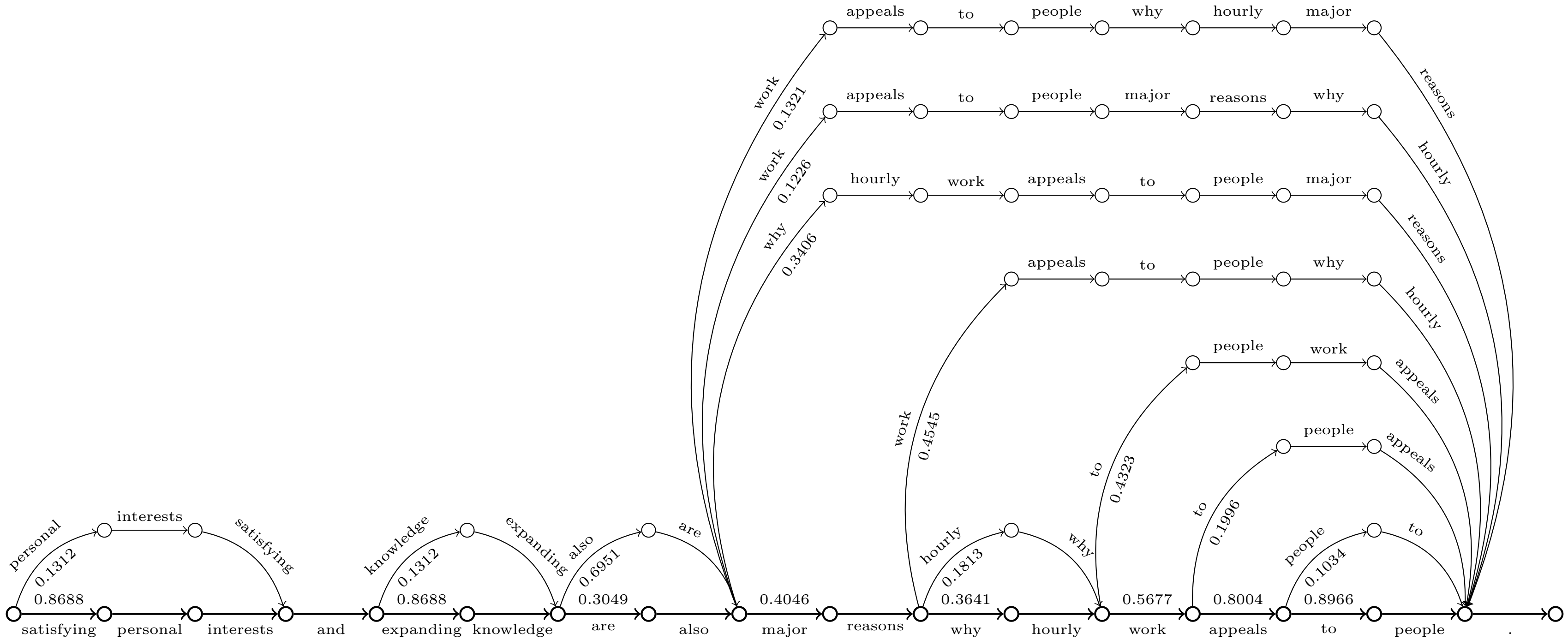


Figure 2.7.: An exapmle of word lattice

3. Reordering Approach

3.1. Reordering Problems in Chinese Translation

English and Chinese belong to different language families. Chinese belongs to the Sino-Tibetan language family, while English belongs to the Indo-European language family. Both of them have developed separately for long period of time. Because of their different origins and development, they are two very different languages.

Unlike the most languages in the Indo-European language family, which are more similar to English than Chinese, Chinese has some properties those languages don't have, such as Characters as basic linguistic element instead of letters, the tones, no word separation by writing, the usage of measure words, less inflection and conjugation. All of them raise further problems for machine translation.

The word order between English and Chinese differs very significantly. For one, the words in Chinese have generally different origins as those in English, which leads to very different vocabulary and word construction. Sometimes it is very hard to find corresponding words in the other language. For example, some prepositions in Chinese have very different usage than those prepositions in English. Also the continuous writing of Chinese without spaces makes this problem more severe, since word boundaries are not always so clear in Chinese. The text needs to be segmented first before translation. A word segmentation process is used to separate the words, but the results may not always be ideal.

For the other, both languages have sometimes very different sentence structures. Thus, a word-for-word translation between English and Chinese is often unnatural or difficult to understand. Each of them has some sentence patterns that don't exist or rarely used in the other. In Chinese, a modifier is often put before the part that it modifies. While in English, it is very common that the modifier is put after the part that it modifies. Besides, English sentences with a lot of long clauses may be more suitable to translate into several Chinese sentences, because in Chinese people don't tend to use long clauses in general.

Some literature (Wang et al., 2007) has discussed or analyzed the differences in word orders between English and Chinese. Through analyzing the data we have and the study of literature, we have found several major types of differences in word orders between English and Chinese, which are typical in the data and often lead to translation problems. They are listed as follows.

Relative clauses

Typically a relative clause modifies a noun or noun phrase. In English a relative clause is normally put after the noun or noun phrase that it modifies. While in Chinese, it's normally put before the noun or noun phrase. But sometimes, a relative clause may also be detached to form another sentence if it is too long. This makes the sentence look more balance in Chinese.

Figure 3.1 shows an example of how the position of a relative clause changes. In the example, the relative clause *that go around* is put after the object *conveyor belts* which is modified. While in the Chinese translation, the part 在 运转 的 which corresponds the relative clause is put before the part 传送 带 which corresponds the *conveyor belts*.

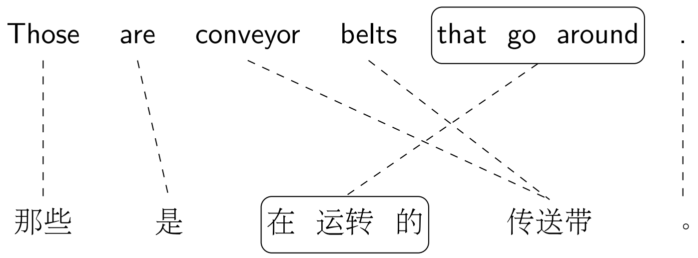


Figure 3.1.: Position change of a relative clause

Adverbials

An adverbial can be an adverb, an adverbial phrase or an adverbial clause that modifies the verb or the whole sentence. The position of adverbials is a complicated topic. In general, the location of adverbials in a sentence can be very flexible. They can be placed in different locations of a sentence both in English and Chinese, such as at the beginning, in the middle or at the end of a sentence, before or after the verb. The locations vary and it often depends on the situation. When comparing English and Chinese word orders, the location of adverbials in one language doesn't automatically implies the same location in the other. Typical examples are adverbials of time, location and frequency. It's often put after the verb in English, but before the verb in Chinese.

Figure 3.2 shows an example of how the position of an adverbial phrase changes. In the English sentence, the adverbial phrase *in five to ten years* is put after the verb *happen*. While in Chinese, the corresponding adverbial phrase 在 五 到 十 年 内 is put before the verb 发生, which means *happen*.

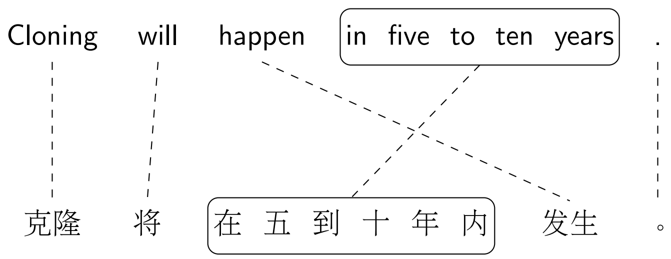


Figure 3.2.: Position change of an adverbial

Preposition Phrases

A preposition phrase functions sometimes as an adverbial, and it's commonly placed before the verb in Chinese. But a preposition phrase can also modify a noun or noun phrase sometimes. When a preposition phrase modifies a noun or noun phrase, it's typically located after the noun or noun phrase that it modifies in English. While in Chinese it's typically located before the noun or noun phrase that it modifies.

Figure 3.3 shows an example of how the position of a preposition phrase changes. In the English sentence, the preposition phrase *of perception* is put after the noun *repetition*. While in Chinese, the part 感知 的 which corresponds the preposition phrase is put before the noun 重复, which means *repetition*.

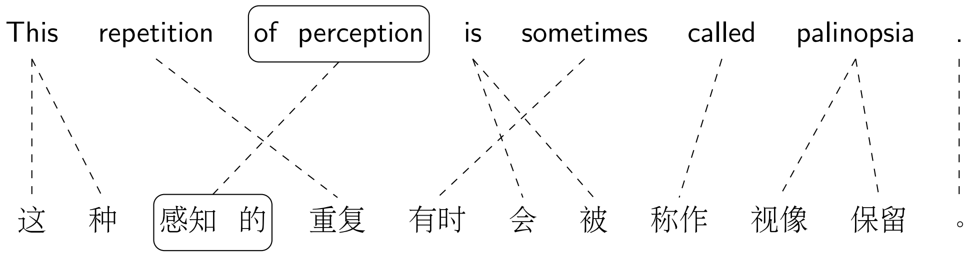


Figure 3.3.: Position change of a preposition phrase

Questions

Questions are often formed by moving the auxiliary verb before the subject or adding *do* (*does*, *did*) before the subject if there's no auxiliary verb in a English sentence. And interrogative word such as *where*, *what*, *how*, etc. is added if it's an interrogative question. However, building a question in Chinese doesn't affect the sentence structure so much. Generally, the questioned part is replaced with an interrogative word and a question denominator is put at the end of a sentence to indicate the question.

In the example in figure 3.4, the verb 会 (*will*) stays after the subject, the interrogative word 怎么 (*how*) is put at the location where an adverbial of manner is generally put, and an question denominator 呢 is put at the end before the question mark.

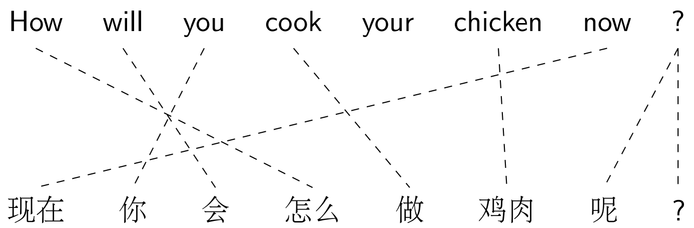


Figure 3.4.: Word reordering of a question

Special Sentence Constructions

Due to the lack of certain sentence patterns, the reordering could be untypical or it varies from case to case. In Chinese, there is generally no sentence construction corresponding to the inverted negative sentences or *there-be* sentences in English. Meanwhile, the Chinese *bǎ*-construction (把字句) doesn't really exist in English either.

Figure 3.5 shows word order change when a *there-be* sentence is translated into Chinese. In the example, the noun *diabetes* is moved to the front of the sentence. However, not all the *there-be* sentences are reordered in the same pattern when being translated.

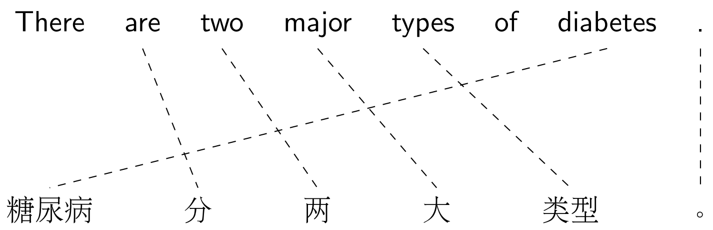


Figure 3.5.: Word reordering of a *there-be* sentence

3.2. Motivation of Multi-Level-Tree (MLT) Reordering

Because English and Chinese have different word orders and there are also some special cases of sentence patterns, the word reordering can be complicated and unsystematically. From the differences in word orders that we've discussed in the last section, we can see there are two typical issues in word reordering between English and Chinese.

3.2.1. Long-Distance Word Reordering

Because sentences are often restructured dramatically when being translated between English and Chinese, the reordering often involves long-distance word position changes. Not only the position change of a part of sentence may be a long-distance shift, but also, the part that is moved may be very long. For example, an adverbial clause of time may be located at the end of a English sentence, but in order to be preordered for translation into Chinese, it may need to be moved across the whole sentence to the front, and the adverbial clause itself may be very long too.

Example 1:
I find this very much disturbing *when we are talking about what is going on right and wrong with democracy these days*.
现在，每当我跟别人讨论我们的民主什么是对的，什么是错的我都为此觉得很无力。

Example 2:
You feel intense elation *when things are going well*; mood swings into horrible despair *when things are going poorly*.
当事情进展顺利的时候，你会觉得兴高采烈；当事情不顺利的时候，你又会陷入极度的失望和恐慌。

The adverbial clauses are underlined in the examples. We can see from both examples that the adverbial clauses are moved forward. These are common cases in translation between English and Chinese. In order to be able to handle the reordering between English and Chinese correctly, the reordering approach should allow part of the sentence to be shifted across long distance.

3.2.2. Word Reordering on Multiple Syntactic Levels

Sometimes the reordering involves word position changes on multiple syntactic levels. We can see this issue from the examples in figure 3.6 and figure 3.7. In the examples, the syntactic structures and alignment of the parallel text are presented in a intuitive way. In the figures, the nodes in the levels that are used to extract the reordering patterns are marked red and use italic font.

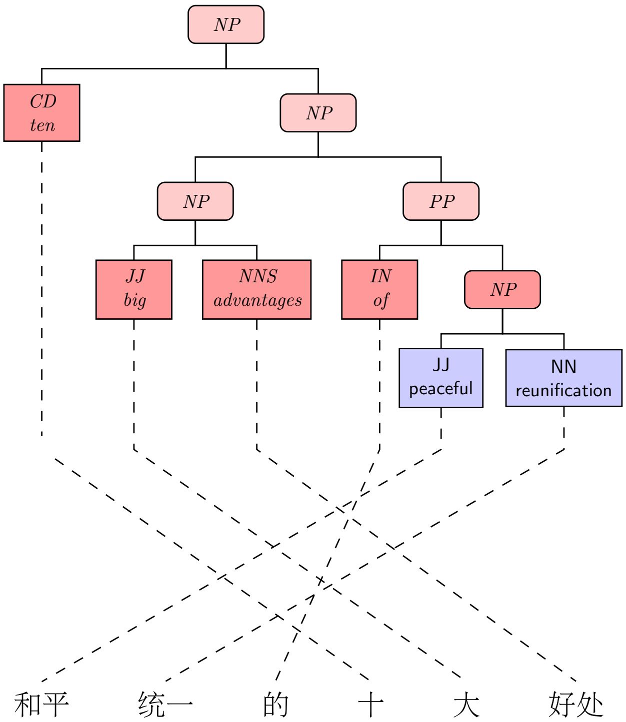


Figure 3.6.: Examples of reordering on multiple syntactic levels (a)

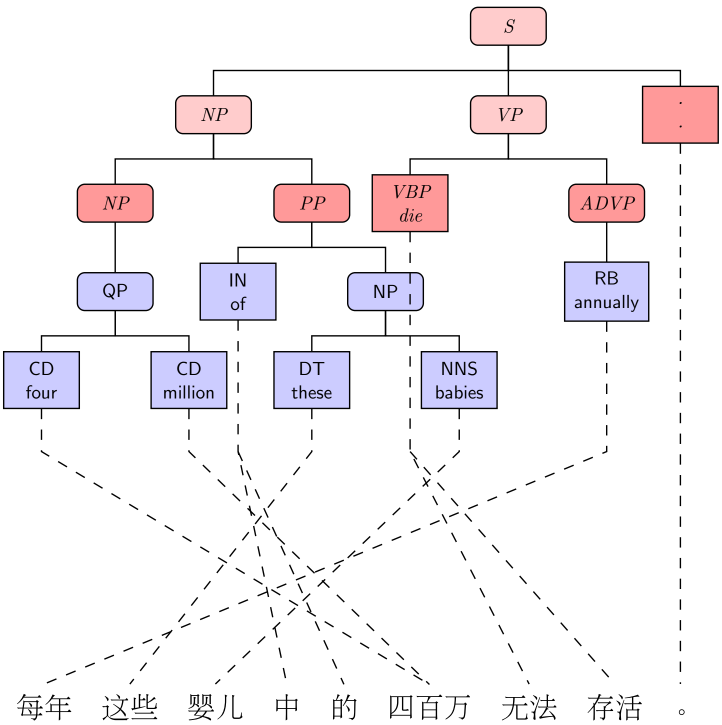


Figure 3.7.: Examples of reordering on multiple syntactic levels (b)

From the examples we can clearly see how reordering on multiple syntactic levels works. From the root of the syntax tree on the left, if we inspect 3 syntactic levels downwards, we can find the following pattern of word position changes, in comparison with the parallel Chinese text:

```
NP ( CD NP ( NP ( JJ NNS ) PP ( IN NP ) ) ) -> NP IN CD JJ NNS
```

The tags with underscore indicate the leaves, which are ordered. as we can see, we can't get this pattern if we only observing children on the same level of syntax tree. The node labeled with *CD ten* is inserted into the subtree of its sibling which is labeled with *NP*, between its sibling's two children. This position change can not be simply done by swapping children of the same node.

We can also observe the same phenomenon from the second example. If we inspect two levels downwards from the root node, we can find the following pattern:

```
S ( NP ( NP PP ) VP ( VBP ADVP) . ) -> ADVP NP PP VBP .
```

In this example, the adverbial phrase *annually* is moved forward to the front of the sentence, leaving the subtree *NP (NP PP)*, which is on a higher syntactic level, being shifted between the subtree with root *VBP* and the subtree with root *ADVP*.

Besides, there are several reasons why we need word reordering on multiple syntactic levels. First, the syntactic parser may make mistakes. As we found in the training corpus, it's not rare that sentences are misparsed, either the words are not correctly tagged, or the syntactic structure is simply wrong. Second, the syntax tree of the English sentence may not be suitable for translation into Chinese. In both case, MLT reordering can be used as a remedy for incorrect or improper parsed sentences. Besides, due to the very different word orders between English and Chinese, simply reordering the words by change children orders on the same syntax tree level may not do the job, and MLT reordering will be useful in this case.

In conclusion of the existing reordering rules we've introduced and the problems we've seen by translating between English and Chinese, a good approach for the reordering should both take long-distance reordering and reordering on multiple syntactic levels into account. A short or long rule based reordering may not utilize the syntactic information for reordering, so the structure of Chinese sentence may not be reconstructed. On the other side, a tree rule based reordering may not be enough helpful by too complicated structure changes, as we've found out there are cases that a reordering can not simply be done by swapping children in a syntax tree.

Inspired by the method of tree rules based reordering method, we've created the MLT source sentence preordering algorithm. The algorithm solely uses information of the syntax trees and the word alignments. It further explores the syntactic structure of text and detects reordering patterns from multiple levels of the syntax tree altogether.

3.3. MLT Reordering Algorithm

As we've already seen how the basic idea of finding reordering patterns on multiple syntactic levels generally works in the last section. We'll systematically explain the rule extraction and application in all details in this chapter.

3.3.1. Rule Extraction

In order to find as much information for reordering as possible. The algorithm of rule extraction detects the reordering patterns from all nodes in the syntax tree and it goes downwards for any number of hierarchies, until it reaches the lowest hierarchy in the subtrees.

In the implementation, the program conducts a Depth-First Search (DFS) to traverse every node in a syntax tree. Every time when a node is reached, the program conducts another Iterative Deepening Depth-First Search (IDDFS) in its subtree with depth-limit from 1 to the subtree's depth. And the program detects if there are any patterns of word position changes at the same time, by using the alignment for comparison.

The detected word position changes are checked for their validity for reordering rules. As describe in section 2.6.4, a valid patterns for reordering should both involves actual reordered words and have clearly distinguishable new order from the target side, i.e. no collision of aligned ranges on the target side.

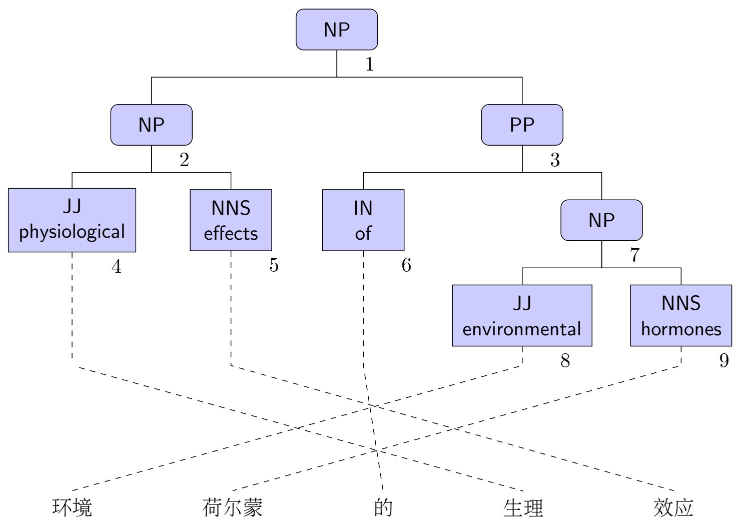


Figure 3.8.: Illustration of rule extraction

Figure 3.8 shows a phrase to be translated, together with its syntax tree and word alignment of parallel text. The leaf nodes are underlined and indexed according to the order they appear in the left side of the patterns. The index sequences on the right side of the patterns indicate the reordering. In this example, we can find the following reordering patterns:

From node 1:

`NP ( NP PP ) -> 1 0` [1 level]

`NP ( NP ( JJ NNS ) PP ( IN NP ) ) -> 3 2 0 1` ∗[2 levels]

`NP ( NP ( JJ NNS ) PP ( IN NP ( JJ NNS ) ) ) -> 3 4 2 0 1` ∗[3 levels]

From node 3:

`PP ( IN NP ) -> 1 0` [1 level]

`PP ( IN NP ( JJ NNS ) ) -> 1 2 0` ∗[2 levels]

As the example shows, this approach can detect more reordering patterns than the tree rule based approach. For example, the reordering patterns marked with ∗ above can not be detected with the tree rules based reordering approach (Herrmann et al., 2013b) directly.

The probability of the reordering patterns can be calculated as described in section 2.6.4. There are the left part and the right part of the reordering patterns separated by the arrow. The left part indicates the syntactic tags that should be reordered and the right part indicates how the new order should be like. The probability of the pattern is calculated by how often the left part is reordered into the right part among all its appearances in the training corpus. In addition, reordering patterns that appear less than a threshold are ignored to be used as reordering rules, in order to prevent too concrete rules without generalization capability and overfitting.

3.3.2. Rule Application

The syntax tree is traversed by DFS as the same in rule extraction. But from the root of each subtree, it's scanned with depth limit from its maximal levels, i.e. its depth, to 1. As it turns out, any rule can be applied for a subtree at some level, a new path for this reordering will be added to the word lattice for decoding, as introduced in section 2.8. As long as rules can be applied on a subtree for a certain depth, the rules are applied and the search for rule application on this subtree stops, and the search on the next subtree continues.

The reason for this is to prevent duplicate reorderings due to application of nested rules, which has overlapped effect with each other. These rules are normally patterns that are generated on the same subtree, but with different number of levels, which has different generalization effect on the same range of words in the text. For example, the following patterns can be detected from the syntax tree in figure 3.8:

```
PP ( IN NP ) -> 1 0
PP ( IN NP ( JJ NNS ) ) -> 1 2 0
```

Both patterns are detected from the same node, but the second pattern is detected by retrieving the nodes one level deeper and it's more concrete. So the first pattern can be seen as a generalization of the second pattern. Whenever a rule of the second pattern can be applied, a rule of the first pattern can be applied too. Because subtrees are checked from the highest number of levels in rule application, the more concrete rule is applied first. Because the more concrete rule fits the detected pattern better and contains more details of reordering, so it may be more suitable for rule application. In this example, the second rule is applied rather than the first rule.

To illustrate how the rule application works in a more intuitive way, we present an example of reordering an English source sentence. Normally the algorithm generates a lot of reordering rules from the training data, from different nodes with different search depths. But in order to make the example simple, we only concentrate on application of the following 2 reordering rules and ignore the others. The sentence for reordering is also presented as follows:

Rules:
[1] `VP ( VBZ NP ( NP PP ) ) -> 2 0 1 (0.18)`
[2] `NP ( NP ( NN NN ) PP ( IN NP ) ) -> 3 2 0 1 (0.17)`

Sentence:
`world bank plans debt relief for poorest countries`

The syntax tree and monotone path as initial word lattice are presented in figure 3.9. In all the examples, leaf nodes have rectangle shape, and the red nodes with italic font indicate the part of the syntax tree where a rule is applicable.

By using DFS to traverse the syntax tree, the program first finds out the pattern started from node *VP* with 2 levels corresponds the left part of the first rule listed above. This indicates the rule is applicable at this location. According to the reordering rule, the order of the three constituents labeled with *VBZ*, *NP* and *PP* should be changed to *PP VBZ NP* with probability 0.18. Thus part of the sentence is reordered into *for poorest countries plans debt relief*, and the new path with this probability is added to the word lattice.

As the program keeps running, the second rule is found applicable at another node labeled with *NP* with 2 levels. Again the rule is applied with probability 0.17 and the new path is added.

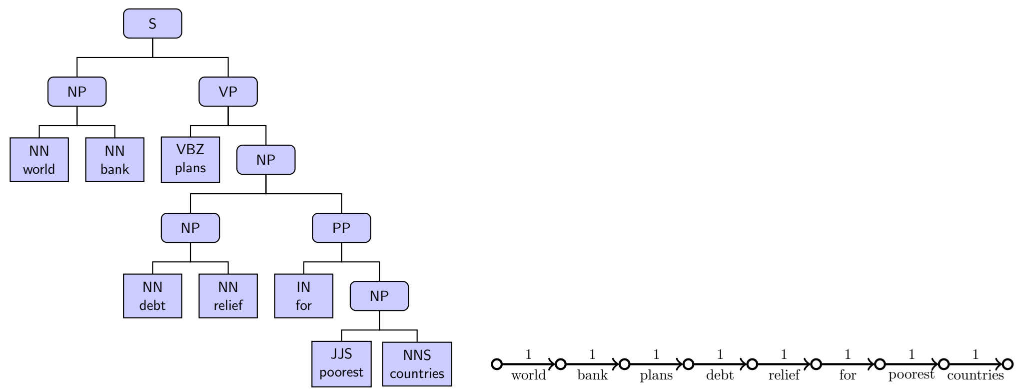


Figure 3.9.: Illustration of rule application (a)

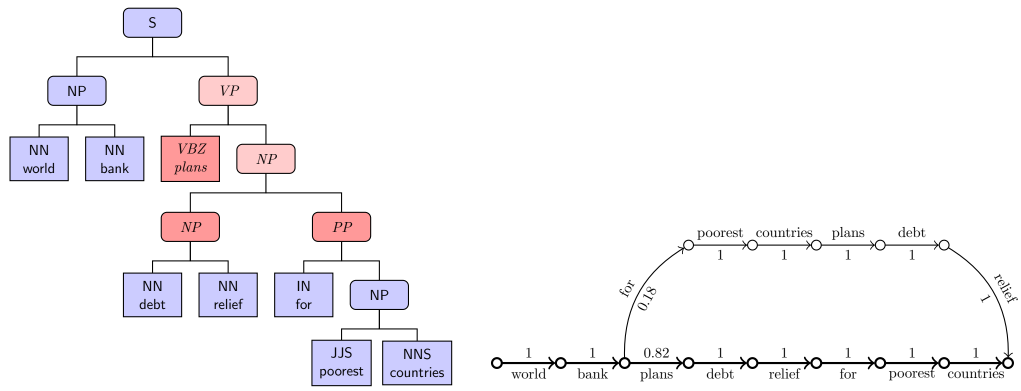


Figure 3.10.: Illustration of rule application (b)

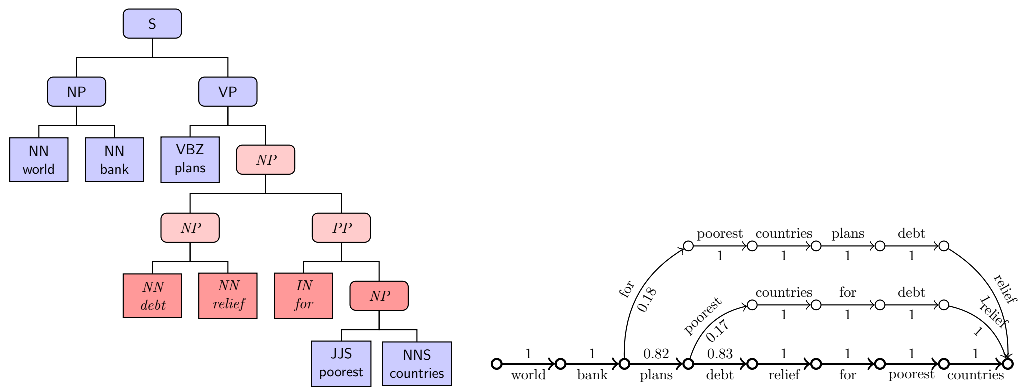


Figure 3.11.: Illustration of rule application (c)

In the end we get the word lattice for the reordered source sentence, where each path from the beginning node to the end node in the lattice presents a possible reordering.

3.4. Summary

In this chapter, we've deeply analyzed the differences in word orders between English and Chinese, followed by an introduction of the MLT reordering algorithm. Because of the different origins and separate development of English and Chinese, they have very distinct sentence structures. To adjust the word order in one language to the other as preordering for translation between the two languages. It often involves long-distance position change and reordering on multiple level of syntactic structures. Inspired by the method of tree rule based reordering method, we created the MLT reordering algorithm which rearranges the words as a pre-process for translation. The algorithm uses the information of syntactic trees and word alignments. It extracts and applies the reordering rules by detecting patterns from the subtrees with different search levels in syntax trees.

4. Evaluation

We've conducted two sets of experiments to test our reordering methods. The first set of experiments is designed for testing the English-to-Chinese translation, which is described in section 4.1. The second set of experiments is designed for the Chinese-to-English translation direction, which is described in section 4.2. Through the experiments in both translation directions, we could get a better overview of the MLT reordering's effect.

Both sections are composed of two parts: experimental setup, results. The first part describes details of the system configurations and experimental data. The second part shows the results of different systems for comparison. And we evaluates the translation systems in section 4.3 in overall.

4.1. English to Chinese System

4.1.1. Experimental Setup

We performed experiments with or without different reordering methods covering the English-to-Chinese translation direction. The reordering methods included our MLT reordering approach and the existing reordering approaches with short rules, long rules and tree rules. The system was trained on news text from the LDC corpus and subtitles from TED talks. The development data and test data were both news text from the LDC corpus. The system was a phrase-based SMT system, which used a 6-gram language model with Knersey-Ney smoothing. Besides the preordering, no lexical reordering or other reordering method in decoding phase was used. The text was translated through a monotone decoder. The Chinese text were first segmented into words with the *Stanford Word Segmenter*[⋆] before use.

The reordering rules were extracted by using the word alignments, POS tags and syntax trees from the training data. One reference of the test data was used for evaluating the results. The threshold for rule extraction is set as 5 times and reordering paths with probability less than 0.1 are not added to word lattices. The decoder was a monotone decoder. Table 4.1 shows the size of data used in the system.

[⋆]Stanford Word Segmenter: `http://nlp.stanford.edu/software/segmenter.shtml`

Data Set		Sentence Count	Word Count		Size (Byte)	
			English	Chinese	English	Chinese
Training Data	LDC	303K	10.96M	8.56M	60.88M	47.27M
	TED	151K	2.58M	2.86M	14.24M	15.63K
Development Data		919	30K	25K	164K	142K
Test Data		1663	47K	38K	263K	220K

Table 4.1.: Data details in English-to-Chinese system

4.1.2. Results

	BLEU Score (%)	Improvement	TER (%)
Baseline	12.07		72.15
+Short Rules	12.50	0.43	71.41
+Long Rules	12.99	0.92	70.71
+Tree Rules	13.38	1.31	68.27
+MLT Rules	**13.81**	**1.74**	**68.20**
Oracle Reordering	18.58	6.51	62.13
Long Rules	12.31	0.24	71.81
Tree Rules	13.30	1.23	70.42
MLT Rules	**13.68**	**1.61**	**70.25**

Table 4.2.: Result overview of English-to-Chinese system

Table 4.2 shows the BLEU scores, absolute improvements of BLEU scores and TERs for configurations with different reordering methods. The table consist of 2 sections. the first row of the top section shows results of the baseline, which used no preordering at all. In the following rows of the top section, different types of reordering rules are combined gradually, with each type per row, and the results are showed. For example, the row with +*MLT* Rules presents the configuration with all the rule types including MLT rules and all the other rules in the rows above. All improvements are absolute improvements of BLEU scores in comparison to the baseline. Each row with a certain reordering type presents all the different variations of this type and the best score under these configurations are shown. For example, long rules include the left rules and right rules, and the tree rules include the partial rules and recursive application. The baseline used a monotone decoder and no preordering. The row with *oracle reordering* shows the results from the configuration that used the oracle reordering as input. The results of oracle reordering can be used for analyzing the potential of source sentence reordering. In the lower section of the table, different rule types are not combined and the effect of each rule type is shown.

4.2. Chinese to English System

4.2.1. Experimental Setup

The experiments for Chinese-to-English systems had a similar setup as described in the last section. The parallel data used in the English-to-Chinese system was also used in this experiment by switching the source language and the target language. We only used the LDC data set for training, and no TED data were used in this system. The test data had three English references for evaluating the results instead of one as in the previous system. The data used are summarized in table 4.3.

Data Set	Sentence Count	Word Count		Size (Byte)	
		Chinese	English	Chinese	English
Training Data	303K	8.56M	10.96M	47.27M	60.88M
Development Data	919	25K	30K	142K	164K
Test Data	1663	38K	47K	220K	263K

Table 4.3.: Data details in Chinese-to-English system

4.2.2. Results

	BLEU Score (%)	Improvement	TER (%)
Baseline	21.80		62.09
+Short Rules	22.90	1.10	61.64
+Long Rules	23.13	1.33	61.43
+Tree Rules	23.84	2.04	60.95
+MLT Rules	**24.14**	**2.34**	**60.79**
Oracle Reordering	26.80	5.00	56.97
Long Rules	22.10	0.30	62.21
Tree Rules	23.35	1.55	61.52
MLT Rules	**23.96**	**2.16**	**60.83**

Table 4.4.: Result overview of Chinese to English systems

Table 4.4 shows the results for configurations with different reordering methods for the Chinese-to-English translation. The table can be interpreted in the same manner as table 4.2 in the previous section.

4.3. Evaluation

The results shows increasing scores as we used reordering methods from short rules, long rules, tree rules to MLT rules. And better BLEU scores were achieved as we combined the different reordering rules. The MLT rules achieved better BLEU scores and TERs in both translation directions, not only when it was used alone, but also it was added to the other reordering rules. When our MLT reordering rules were combined with the other existing reordering rules, a further improvement of 0.43 in BLEU score (from 13.38 to 13.81) was shown in the English-to-Chinese translation direction, as well as a further improvement of 0.3 in BLEU score (from 23.84 to 24.14) in the Chinese-to-English translation direction.

We've also found improvements in the sentence structure. Table 4.5 shows some translation examples in both translation directions. Sections are separated by double lines in the table. Each section of this table shows one translation example with the source sentence (*source*), translation without using MLT reordering (*no MLT*), translation with MLT reordering (*MLT*) and the reference (*reference*). The translation without using MLT reordering comes from the configuration with highest BLEU score that didn't use MLT reordering. And the translation with MLT reordering comes from the configuration with highest BLEU score that used MLT reordering. From the examples, we can clearly see the improvements in sentence structure.

However, by taking a close look at the gap between the scores of oracle reordering and the best scores achieved by MLT reordering, we can also see, there's still potential for improvements of translation between English and Chinese through better reordering methods.

Source	hu jintao also extended deep condolences on the death of the chinese victims and expressed sincere sympathy to the bereaved families .
No MLT	胡锦涛 还 表示 深切 哀悼 的 受害者 家属 的 死亡 , 向 迂难者 家属 表示 诚挚 的 慰问 。
MLT	胡锦涛 还 对 中国 迂难者 表示 哀悼 , 向 迂难者 家属 表示 诚挚 的 慰问 。
Reference	胡锦涛 还 对 中方 不幸 遇难 人员 表示 深切 的 哀悼 , 并 向 遇难 着 的 亲属 致以 诚挚 的 慰问 。
Source	satisfying personal interests and expanding knowledge are also major reasons why hourly work appeals to people .
No MLT	满足 个人 利益 和 扩大 知识 也 是 主要 原因 小时 工作 吸引 人 。
MLT	满足 个人 利益 和 扩大 知识 也 是 为什么 学生 工作 吸引 人 的 主要 原因 。
Reference	满足 个人 兴趣 , 扩大 自己 的 知识面 也 是 兼职 小时 工 受 青睐 的 一 个 重要 原因 。
Source	the dalai lama will go to visit washington this month .
No MLT	达赖 喇嘛 将 访问 华盛顿 的 这 一 个 月 。
MLT	达赖 喇嘛 将 本 月 访问 华盛顿 。
Reference	达赖 喇嘛 将 在 本 月 前往 华盛顿 访问 。
Source	陈至立 说 , 古巴 是 拉美 和 加勒比 地区 有 重要 影响 的 国家 。
No MLT	chen zhili said : cuba is the latin america and the caribbean region has an important influence on the state .
MLT	chen zhili said : cuba is a country of important influence latin america and the caribbean region .
Reference	chen zhili said that cuba is a country of great influence in the latin american and caribbean region .
Source	近年 来 , 两 国 教育 交流 日益 密切 , 人员 来往 频繁 。
no MLT	in recent years , the two countries education have been increasingly close exchanges and personnel contacts have been frequent .
MLT	in recent years , the educational exchanges between the two countries have become increasingly frequent , and have had frequent contacts .
Reference	in recent years , the educational exchange between the two countries has become increasingly close with frequent personnel visits .

Table 4.5.: Examples of translations

From this experiments we can draw the conclusion that our reordering method obviously improves the sentence structure and translation quality in both English-to-Chinese and Chinese-to-English translation directions, no matter when we apply it alone or when we combine it with the reordering methods based on short rules, long rules and tree rules.

4.4. Summary

In this chapter, we've presented and evaluated the results of the English-to-Chinese and Chinese-to-English SMT system. In both system, our MLT reordering method shows obvious improvements on translation quality.

When our MLT reordering rules were combined with the other existing reordering rules, a further improvement of 0.43 in BLEU score was shown in the English-to-Chinese translation direction, as well as a further improvement of 0.3 in BLEU score in the Chinese-to-English translation direction.

Through analyzing the syntactic structure of the sentences closely, we've also found that the MLT reordering method improved the translation by changing the order of words, not only on the same syntax tree level but also between different levels, which could not be easily achieved by other reordering methods we've introduced so far. So it further justifies our claim, that the MLT reordering method improved the sentence structure obviously and led to better translation between English and Chinese.

5. Discussion

In this chapter, we first summarize what we've done in this work in section 5.1. Then we conclude this thesis in section 5.2. At the end, we point out possible directions of future work in section 5.3.

5.1. Summary

In this work, we've present a new reordering approach for pre-processing before translation between English and Chinese.

English and Chinese are two very different languages. Because of the different origins and separate development of the two languages, their sentence structures differ significantly, which makes the word reordering a especially difficult problem. Unlike the most other European languages, Chinese has some distinct languages features such as the use of characters instead of letters, the use of measure words, lack of space to separate the words, more pre-modifier than post-modifier, etc.

Through the analyze of differences in word orders between the two languages, we've found out the reordering is often involved with long distance word position change, such as the shift of the whole adverbial clause in the sentence, and reordering on multiple syntactic levels, such as reordering for sentences in some special pattern, which doesn't exist or rarely used in the other language.

In order to improve the sentence structure and translation quality, we've proposed the Multi-Level-Tree (MLT) algorithm for pre-processing the text before translation. Based on the differences in English and Chinese word orders, the algorithm detects and applies reordering rules from the syntax trees and word alignments. Reordering patterns are detected by checking if the nested tag sequences in subtrees with any number of search levels have clearly new orders in the aligned text in the target language.

At last, we've established two different SMT systems with different data set and configurations, to conduct experiments on this reordering approach for both translation directions. The approach is tested on different configurations, with or without combining other reordering approaches. And the results of different configurations are compared and evaluated.

5.2. Conclusion

We've conducted experiments in both translation directions with different SMT configurations. From the results we can see the BLEU scores was improved no matter when we applied our SMT reordering method to the baseline directly or when we combined it with the other reordering methods we introduced before, i.e. short rules, long rules and tree rules based reordering methods.

When our approach was applied alone, it achieved the best BLEU score under all these reordering methods in both translation directions. The BLEU score of the baseline was improved by 1.61 in the English-to-Chinese translation direction, which maked up 13.34% in comparison with the baseline's BLEU score 12.07. And the improvement in the Chinese-to-English translation direction is 2.16, which maked up 9.91% in comparison of the baseline's BLEU score 21.80.

When our approach was combined with the other reordering methods, further improvements were achieved for both translation direction. Our approach improved the BLEU score further by 0.43 on the English-to-Chinese translation direction, which maked up 3.57% in comparison with the baseline's BLEU score 12.07. The BLEU score in the Chinese-to-English direction was further improved with our approach by 0.30, which showed a 1.37% improvement in comparison with the baseline's BLEU score 21.80.

Our reordering approach also has some other advantages. As the translation examples we've presented, there were obvious improvements in sentence structures with our reordering approach. Besides, the approach is very efficient because it reorders the words in a pre-process, rather than during decoding phase as the hierarchical phrase-based SMT model.

As the BLEU score was used as a measurement for the translation quality, we conclude that the Multiple-Levels-Tree reordering approach achieved obvious improvement in the word reordering and led to better translation quality between English and Chinese.

5.3. Outlook

Although the translation quality was obviously improved by our reordering approach, there's still much space for further improvements. As in the results, the BLEU scores that was achieved by the oracle reordering was still much higher than the BLEU scores achieved by our approach. This was partially because Chinese is a very different language from English and it's also not researched so much as English. However, there are still possibilities for further research.

One direction is to design better algorithm for word reordering. Design other reordering rule types which suit translation between English and Chinese better may be possible. On the other side, it's also possible to have reordering methods other than rule-based, such as training classificators for reordering for different circumstances (Lerner and Petrov, 2013).

The other direction is to design good reordering method use less information such as syntax tree. Because syntactic parser may not be available for some unpopular languages, due to lack of research and training data, this approach enables easier adaptation to other languages.

Besides, vector representation is currently a popular topic for various tasks too (Blunsom et al., 2014; Mikolov et al., 2013). One possible way is to use the vector representation as the feature instead of the POS tags, but details also need to be discussed, in order to make this approach perform well in practice. First, the vectors are continue values rather than discrete values as POS tags, so some metric may need to be defined in order to extract reordering rules from similar patterns. The detection of similar patterns may also be time-consuming or even impossible, if the metric is complicated or not suitable for grouping similar pattern. Second if syntax tree is used for reordering, consideration may need to be taken for what is good vector representation of internal nodes or constituents as well as how to calculate it. If syntax tree is not used, information of syntactic structure may not be fully utilized, long distance reordering or syntactic structure change may not be detected. One way out is probably to use the dependency tree (De Marneffe et al., 2006), because each internal node is labeled with the head word of its subtree, which can be used for the vector representation.

If a algorithm gets too complicated, it's also questionable if it will perform well in practice, since it may not be intuitive and will pose a problem for implementation sometimes. So another way to make use of vector representation for word reordering is probably design some algorithms which can utilize the vector representation in a more direct way, rather than using the rule based reordering.

Appendix

A. Documentation of Preordering System

Here we explain the details of our preordering system and how to integrate it into the SMT system of our faculty at Karlsruhe Institute of Technology (KIT), as well as other issues. A summary about how to integrate and use the code can be view in the section A.5 of the documentation.

A.1. System Integration

The source code `Configuration.py` and `ReorderingRules.py` of the SMT system are modified. The both modified versions are located at:
`/home/gwu/src/trunk/systemBuilder/src/Configuration.py`
`/home/gwu/src/trunk/systemBuilder/src/Components/ReorderingRules.py`

In the file `Configuration.py`, the condition statements at line 628 and 634 are modified.

In the file `ReorderingRules.py`, code at multiple locations are modified, which enable us to integrate the MLT reordering into the system.

Other source code of the SMT system is untouched.

In order to use the system, both `Configuration.py` or `ReorderingRules.py` files in one's SMT system should be changed or replaced accordingly.

A.2. Source Code of Reordering

Two lines in the `ReorderingRules.py` file are the entry points of the reordering algorithm, the two lines start separately with:

```
command = "/home/gwu/src/MLTRules.mode/extract " + ...
command = "/home/gwu/src/MLTRules.mode/apply " + ...
```

The last part of the lines is left out due to their length. The two lines point to the source code for extracting and applying the MLT reordering rules, which together with other related source code is located at:
`/home/gwu/src/MLTRules.mode/`

It's possible to move the whole source code directory to other locations and change the corresponding paths in `ReorderingRules.py`.

There is an `makefile` in the directory, which is used for compiling the source code.

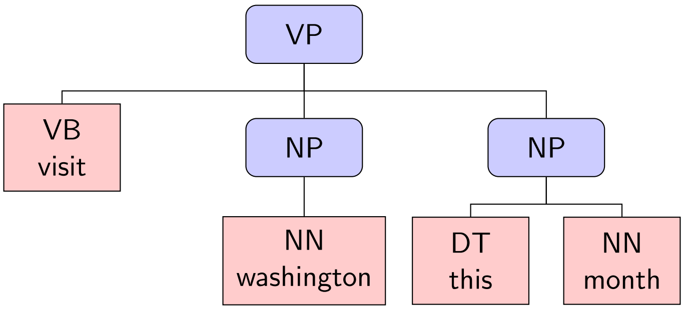
Command for Rule Extraction and Application

The executable file `extract` has the following usage format:

```
extract <Alignment> <SourceText> <SourceTrees> <Mode> <minOcc>
```

The parameter `<Alignment>` should be the path of alignment file of the parallel data, which are used for training the rules. The word indexes in the alignment file should start with 1. The parameter <SourceText> is the path of source text, <SourceTrees> is the parse tree file and <minOCC> is the minimum occurrences that a rule should have, in order to be extracted. Rules that don't occur so often are ignored. The parameter <Mode> is an integer between 0 and 3, which indicates four rule variations.

When mode is 0, the rule includes with POS tags of the internal nodes. In the other modes, the POS tags of internal nodes are ignored. In mode 2, the hierarchies of the syntax trees are compressed, so the rules contains less parenthesis. Parenthesis are removed, when a node is a single child of its parent or when the removal doesn't affect the word grouping. Furthermore, all the parenthesis are completely removed in mode 3.




```
Mode 0: VP ( VB ) ( NP ( NN ) ) ( NP ( DT ) ( NN ) ) ---- ...
Mode 1: ( VB ) ( ( NN ) ) ( ( DT ) ( NN ) ) ---- ...
Mode 2: VB NN ( DT NN ) ---- ...
Mode 3: VB NN DT NN ---- ...
```

Figure A.1.: Variations of MLT rules

Figure A.1 shows how the rules are presented in different modes.

The standard output of the program will list all the extracted rules including the reordering pattern, probability and occurrences. For example,

```
VB NN ( DT NN ) ---- 2 3 0 1 ---- 0.5 ---- 5 / 10
```

means the sequence VB NN (DT NN) in mode 2 will be reordered as DT NN VB NN with a probability 0.5, because on 5 out of all the sequence's 10 occurrences in the training data, it is ordered in this manner.

The executable file apply has the following usage format:
apply <RuleFile> <TargetText> <TargetTag> <TargetTrees> <TargetDir> <Mode> [<LatticesDir>]

The parameter <RuleFile> is the path of the rule file created by the the extract commnad. The parameter <TargetText> is the path of text to be reordered and translated, with one sentence per line. <TargetTag> and <TargetTree> are separately the POS tags and syntax trees of the text. <TargetDir> is the directory, where the resulted lattices should be put. <Mode> is the same parameter as described in the extract command, it should be consistent with mode used for extracting the rules. The last parameter <LatticesDir> is optional, it's intended to enable the rule combination. If this parameter is given, the program will apply the rules on top of the existing lattices under the specified directory <LatticeDir>. The existing lattices should be also extracted for the same target text.

The output of of the command are the lattice files saved under TargetDir, one file per sentence. Each lattice file saves the specific information of nodes and edges of a lattice graph presenting a reordered sentence. The whole directory can be used as input for the decoder.

The reordering system uses 1 for the parameter `<mode>` and 5 for the parameter `<minOcc>` as default.

A.3. Description File

Two locations of the description file of the SMT system needs to be changed to use the reordering method. Here we show how to write the description with examples.

Reordering Rule Section

Following code is an example, which gives an idea about what should be added to this section of reordering rules.

```
<reorderingrules>
    <name> MLTRules.mode </name>
    <input>
        <alignment> gizaprepronc </alignment>
        <tags> SourceTreeTaggerenprepronc </tags>
        <tree> parseTreeenprepronc </tree>
    </input>
    <input>
        <alignment> gizapreproepps </alignment>
        <tags> SourceTreeTaggerenpreproepps </tags>
        <tree> parseTreeenpreproepps </tree>
    </input>
    <layers> 0,1,2,3 </layers>
    <thres> 5 </thres>
    <feature> pos </feature>
    <type> MLTRules </type>
    <maxMem> 30000 </maxMem>
</reorderingrules>
```

Content inside the `input` tag has the same format as in the tree rule reordering. The content in `alignment`, `tags` and `tree` tag should be changed accordingly. The `layers` tag indicates a list modes that you want to use. The `thres` tag indicates the threshold for extracting rules, which corresponds the `<minOCC>` parameter in the `extract` command. The `type` tag is used to distinguish this reordering from other reordering methods, the content of which should be set as `MLTRules`.

Configuration Section

The following examples are configurations that can be added to the description file.

```
<configuration>
    <name> MLTRules.mode.1.pos.5 </name>
    <configuration> Baseline </configuration>
    <latticecreator> MLTRules.mode </latticecreator>
    <rules> 1.pos.5 </rules>
</configuration>
```

```
<configuration>
    <name> MLTRules.mode.1.pos.5.pyLong </name>
    <configuration> Baseline </configuration>
    <latticecreator> MLTRules.mode </latticecreator>
    <rules> 1.pos.5.pyLong </rules>
</configuration>
```

The `latticecreator` should be the same as the `name` field in the reordering rule.

The content inside `rules` tag should have one of the following form:

- `mode.pos.threshold`
- `mode.pos.threshold.lattice_directory`

The `threshold` should be consistent as the one in the reordering rule section. The `mode` should be also included in the mode list in the reordering rule section. And the `pos` is simply the string *pos*.

The part `lattice_directory` is optional. When it's given, the lattices will be build on existing lattices, otherwise the lattices are built directly on the text. It should be the same as the `rules` field in the configuration with tree rules.

Note: the name of the tree rule configuration is supposed to be `TreeRules`. Otherwise, some directory paths should be change in `ReorderingRules.py`. In this case, `TreeRules` in the lines, where the variable `latdir` appears, should be changed to the correct name.

A complete example can be found under:
`/project/mt_rocks/user/gwu/EN/ZH/ende/description.xml`

A.4. Other Scripts

Here are some other scripts that I wrote throughout the time that I spent on this thesis. These scripts could be very helpful.

```
/home/gwu/ma/scripts/results.py
Usage: results.py <SystemPath> <Option>
```

This script is used to show the general outcome of all configurations. The parameter `<SystemPath>` should be the path of the system root direcotry. `<Option>` could be different strings.

The parameter `<Option>` should be one of the following strings:

- `test`: program lists the test scores of all configurations.
- `dev`: program lists the dev scores of the last training cycle.
- `devmax`: program lists the maximal dev scores among all training cycles.
- `translate`: program checks if the translations of all configurations are finished.
- `optimize`: program checks if the optimizations of all configurations are finished.
- `error`: program shows all the error and warning messages.

```
/home/gwu/ma/scripts/tree/getTreeInfo.py
Usage: getTreeInfo.py <TreeFile>
```

This script shows information of the depth and branch factor of syntax trees in a tree file specified by the parameter `<TreeFile>`.

```
/home/gwu/ma/scripts/generator/gentree
Usage: gentree
```

This program is used to get the tikz code for drawing a syntax tree. Executing this program will lead one into a command line mode. After the syntax tree is given, the program outputs the tikz code for the tree. Following example shows how a tree could be present with this code:
`/home/gwu/ma/scripts/generator/example_tree.tex`

The configurations of the tree may be altered according to demand, including the distance between levels, distance between children, how the nodes and edges look, etc.

```
/home/gwu/ma/scripts/generator/gengraph
Usage: gengraph <LatticeFile>
```

This program is used to get the tikz code for drawing a lattice graph. The `<LatticeFile>` is the lattice file to present in tikz code. An example using this code could be found at: `/home/gwu/ma/scripts/generator/example_graph.tex`

A.5. Summary

Here is a step for step summary about how to integrate and use our code for reordering.

System code directory: `/home/gwu/src/trunk/systemBuilder/src/`
The reordering code: `/home/gwu/src/MLTRules.mode/`

1. Modify or replacing source code `Configuration.py` and `ReorderingRules.py` in your SMT system accordingly.
2. Copy the directory of executable file `apply` and `exact` to your own directory and change the paths point to the executable files, or don't copy the directory and leave the paths as they are.
3. Modify the description file in the system, add new settings to the `reorderingrules` and `configuration` section, with the desired mode, threshold for rule extraction and optional existing lattice directory.
4. Check if the configuration for tree rules is called `TreeRules`. If it's not, some paths in `ReorderingRules.py` must be changed. See the note.
5. Build the system with the modified description file.

B. Penn Treebank Tagset

The Penn Treebank tagset is listed below for reference (Marcus et al., 1993; Santorini, 1990).

B.1. Penn Treebank POS tagset

1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition/subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal verb
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol (mathematical or scientific)
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund/present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	*wh*-determiner
34.	WP	*wh*-pronoun
35.	WP$	Possessive *wh*-pronoun
36.	WRB	*wh*-adverb
37.	#	Pound sign
38.	$	Dollar sign
39.	.	Sentence-final punctuation
40.	,	Comma
41.	:	Colon, semi-colon
42.	(	Left Parenthesis character
43.	)	Right Parenthesis character
44.	‘	Left open single quote
45.	’	Right close single quote
46.	“	Left open double quote
47.	”	Right close double quote

B.2. Penn Treebank Syntactic Tagset

1.	ADJP	Adjective phrase
2.	ADVP	Adverb phrase
3.	NP	Noun phrase
4.	PP	Prepositional phrase
5.	QP	Quantity phrase
6.	S	Simple declarative clause
7.	SBAR	Clause introduced by subordinating conjunction or *that*
8.	VP	Verb phrase
9.	X	Unknown, uncertain, or unbracketable

Acronyms

BLEU Bilingual Evaluation Understudy

DFS Depth-First Search

EM Expectation-Maximization (Algorithm)

IDDFS Iterative Deepening Depth-First Search

LDC Linguistic Data Consortium

MLT Multi-Level-Tree (Reordering)

POS Part-of-Speech (Tagging)

SMT Statistical Machine Translation

SOV Subject-Object-Verb (Language)

TED Technology, Entertainment, Design (Conference)

TER Translation Edit Rate

List of Tables

List of Figures

Bibliography

Birch, A. (2011). Reordering Metrics for Statistical Machine Translation.

Birch, A., Osborne, M., and Blunsom, P. (2010). Metrics for MT Evaluation: Evaluating Reordering. *Machine Translation*, 24(1).

Blunsom, P., Grefenstette, E., Kalchbrenner, N., et al. (2014). A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

Chiang, D. (2007). Hierarchical Phrase-Based Translation. *computational linguistics*, 33(2):201–228.

Collins, M., Koehn, P., and Kučerová, I. (2005). Clause Restructuring for Statistical Machine Translation. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 531–540. Association for Computational Linguistics.

Crego, J. M. and Habash, N. (2008). Using Shallow Syntax Information to Improve Word Alignment and Reordering for SMT. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 53–61. Association for Computational Linguistics.

De Marneffe, M.-C., MacCartney, B., Manning, C. D., et al. (2006). Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Habash, N. (2007). Syntactic Preprocessing for Statistical Machine Translation. *MT Summit XI*, pages 215–222.

Herrmann, T., Niehues, J., and Waibel, A. (2013a). Combining Word Reordering Methods on Different Linguistic Abstraction Levels for Statistical Machine Translation. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 39–47, Atlanta, Georgia. Association for Computational Linguistics.

Herrmann, T., Weiner, J., Niehues, J., and Waibel, A. (2013b). Analyzing the Potential of Source Sentence Reordering in Statistical Machine Translation.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Koehn, P., Axelrod, A., Birch, A., Callison-Burch, C., Osborne, M., Talbot, D., and White, M. (2005). Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *IWSLT*, pages 68–75.

Lerner, U. and Petrov, S. (2013). Source-Side Classifier Preordering for Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Niehues, J. and Kolss, M. (2009). A POS-Based Model for Long-Range Reorderings in SMT. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 206–214, Athens, Greece. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Popovic, M. and Ney, H. (2006). POS-Based Word Reorderings for Statistical Machine Translation. In *International Conference on Language Resources and Evaluation*, pages 1278–1283.

Rottmann, K. and Vogel, S. (2007). Word Reordering in Statistical Machine Translation with a POS-Based Distortion Model.

Santorini, B. (1990). Part-of-Speech Tagging Guidelines for the Penn Treebank Project (3rd revision).

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Tillmann, C. (2004). A Unigram Orientation Model for Statistical Machine Translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 101–104. Association for Computational Linguistics.

Wang, C., Collins, M., and Koehn, P. (2007). Chinese Syntactic Reordering for Statistical Machine Translation. In *EMNLP-CoNLL*, pages 737–745. Citeseer.

Zhang, Y., Zens, R., and Ney, H. (2007). Chunk-Level Reordering of Source Language Sentences with Automatically Learned Rules for Statistical Machine Translation. In *Proceedings of the NAACL-HLT 2007/AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 1–8. Association for Computational Linguistics.