Karlsruhe Institute of Technology

Analyzing the Potential of Source Sentence Reordering in Statistical Machine Translation for Chinese

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 Febrary 2014 – 25th August 2014

Abstract

todo

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.

25th August 2014

[illegible]

Contents

1. Introduction

1.1. Motivation

Word reordering 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 are isolated from each other. Depend on the languages, different word orders could have very distinguish features. For example, 45% of the languages in the world has a subject-object-verb (SOV) order. 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. An example from the paper Lerner and Petrov (2013) shows how people would order the words differently:

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 that is very different from English, because of their separate origins and development. Both languages have a SOV order, but they also have a lot of differences in word order. Especially sometimes a sentence translated in both languages could have totally different syntactic structures. The differences could involves 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 pre-reordering method, which uses rules based on syntax tree. The method is called Multi-Level-Tree (MLT) reordering, which orders the constituents on multiple levels of the syntax tree all together. This pre-reordering method rearrange the words in source language into a similar order as they are supposed to appear in the target language before translation. with the appropriate word order, better translation quality can be achieved.

In addition, we combine this new reordering method with other existing rule based reordering methods and evaluate the results on translation between English and Chinese.

1.2. Objective and Contribution

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

> We establish a new data driven rule based pre-reordering method for translation between English and Chinese. The method reorders the source text of a SMT system before translation by using the information from alignment, POS tags and syntax tree of the training data. Also we evaluate this method by checking the resulted translation quality and by comparing it with other existing rule based pre-reordering method.

The ground of this thesis are three papers about rule based pre-reordering: 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 to German, they are not necessarily suitable for Chinese translation, which belongs to a totally different language branch and has some distinguishable features. Or at least, there may be still space for improvement on translating for Chinese.

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 the syntax trees' subtrees 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 these existing methods. Through the evaluation and comparison, we'll see the improvement on translation between English and Chinese.

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 reorder the words. The words can be reorder 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 is including use 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.

It's worth mentioning the hierarchical phrase-based translation model (Chiang (2007)) is especially suitable for Chinese, and provide very good 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. But the drawback of this model is also apparent, it could be time consuming due to the amount of calculation during decoding.

Rottmann and Vogel (2007) introduced the rule based pre-reordering by using reordering information based on adjacent words.

special problem of chinese: segmentation

1.4. Structure

In this chapter we mainly describe the background and objective of this thesis, including the related research. In chapter 2 we explain some fundamental concept and knowledge, which is relevant for understanding this thesis. In chapter 3 we introduce our reordering methods in detail, including the problem of translating between English and Chinese and the initiation of our reordering method. The results and evaluation of our method are present in chapter 4. In the last chapter we conclude this work with an overall discussion of our methods. We also point out some possible directions for future research in this chapter.

2. Foundations

This chapter provides an introduction to fundamental knowledge that are relevant to this work. We start with the whole SMT system and pre-reordering 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 and how to build lattices for translation. Koehn (2010) also provide a good introduction to statistical machine translation, including different kinds of theories and methods that are relevant to this work.

2.1. SMT System

Statistical machine translation (SMT) is the state of art machine translation paradigm. 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. The architecture of a SMT system could be illustrated as figure 2.1.

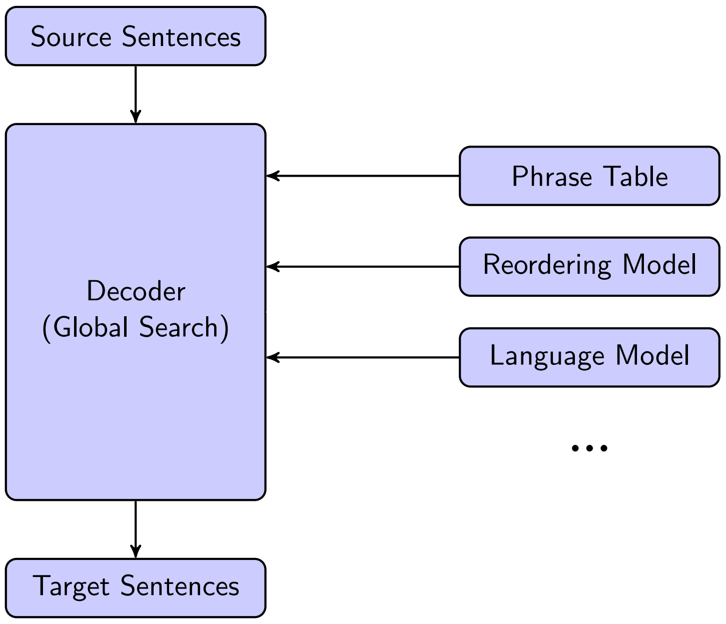


Figure 2.1.: Architecture of SMT system

2.2. Rules Based Pre-Reodering

Our pre-reordering method is based on reordering rules. Reordering rules are rules that tell us how we should reordering the sentences in source language before translating them. In our system, the rules are generated by using the word alignment, part-of-speech (POS) tag and syntax tree, all of which are calculated based on the training data. After we apply the reordering rules to the source sentences, word lattices are generated. The word lattices contains all the reordering possibilities of the source sentences and are further passed to the decoder for translating. The pre-reodering system could be illustrated as figure 2.2.

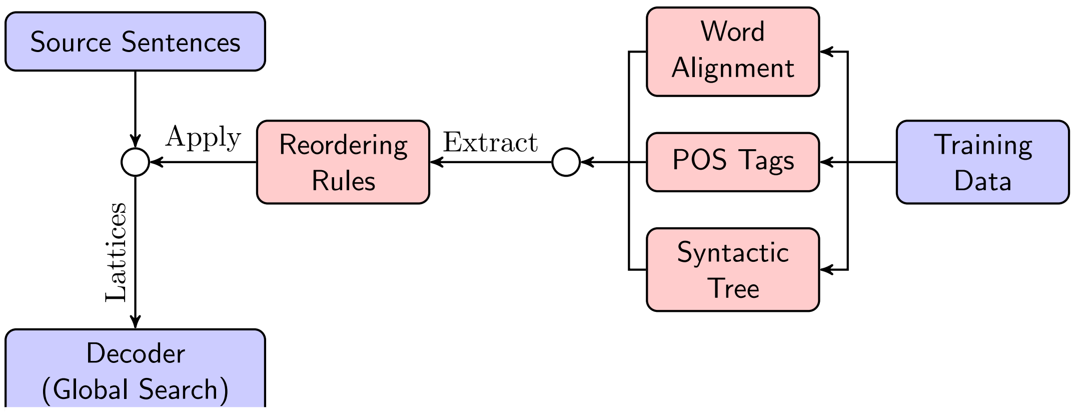


Figure 2.2.: Pre-reordering system

A more detailed description of word alignment, POS tag, syntax tree, reordering rules and word lattices is also clarified in the following sections.

The reordering approach we used for extracting and applying the rules is introduced in the next chapter.

2.3. Word Alignment

Word alignment indicate the possible alignment between words in the source sentence and words in the target sentence. For example, figure 2.3 shows an alignment between an English sentence and a Chinese sentence.

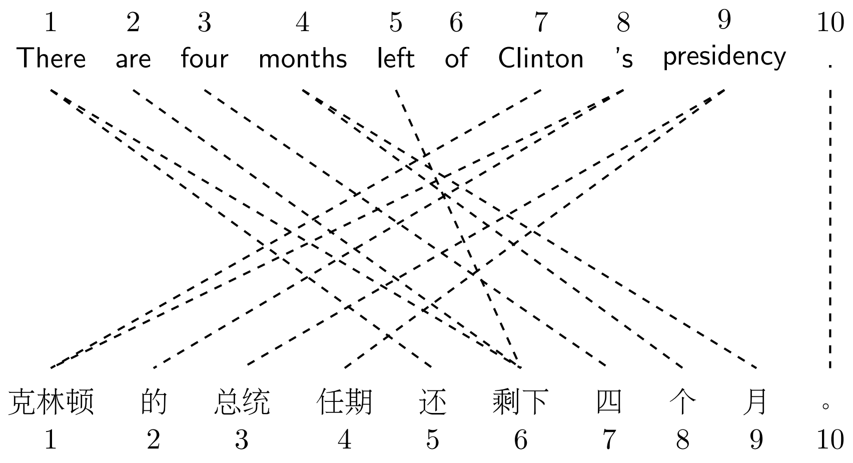


Figure 2.3.: Example of word alignment

Or it may be simply presented as index pairs.

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

In the figure, the words that are aligned through lines in two languages have related meaning. We can see how the words are reordered from the figure. For exmaple, the noun clause "Clinton's presidency" is completely moved forward to the front of the sentence.

It is noteworthy that the alignment is not always a one-to-one match. In the example, the word "of" is not aligned at all and the word "months" is aligned to two word "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 range** can be coincided, such as the 6th word "剩下" in Chinese is also aligned to "are" and "left" at the same time besides "there". The coincidence sometimes makes the detection of rules more difficult, because the word order is not clear any more.

The word alignment could be trained with the GIZA++ tool by using Expectation Maximization (EM) algorithm. From the word alignment of training data we can see the patterns how the words are reordered before and after translation. Therefore, we could extract these reordering rules and apply them on the text, which is to be translated.

2.4. Part-of-Speech Tag

Part-of-speech (POS) tags are markups of words in the text, which corresponds their linguistic role in the text. Depends on the definition of the roles, the set of POS tags could be different. Besides, different languages may have different POS tag set, since they may have different linguistic features, which are relevant to translation.

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

Figure 2.4.: Example of POS tags

Figure 2.4 shows an tagged English sentence.

Tagset? English & Chinese how to tag?

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 the 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), 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.

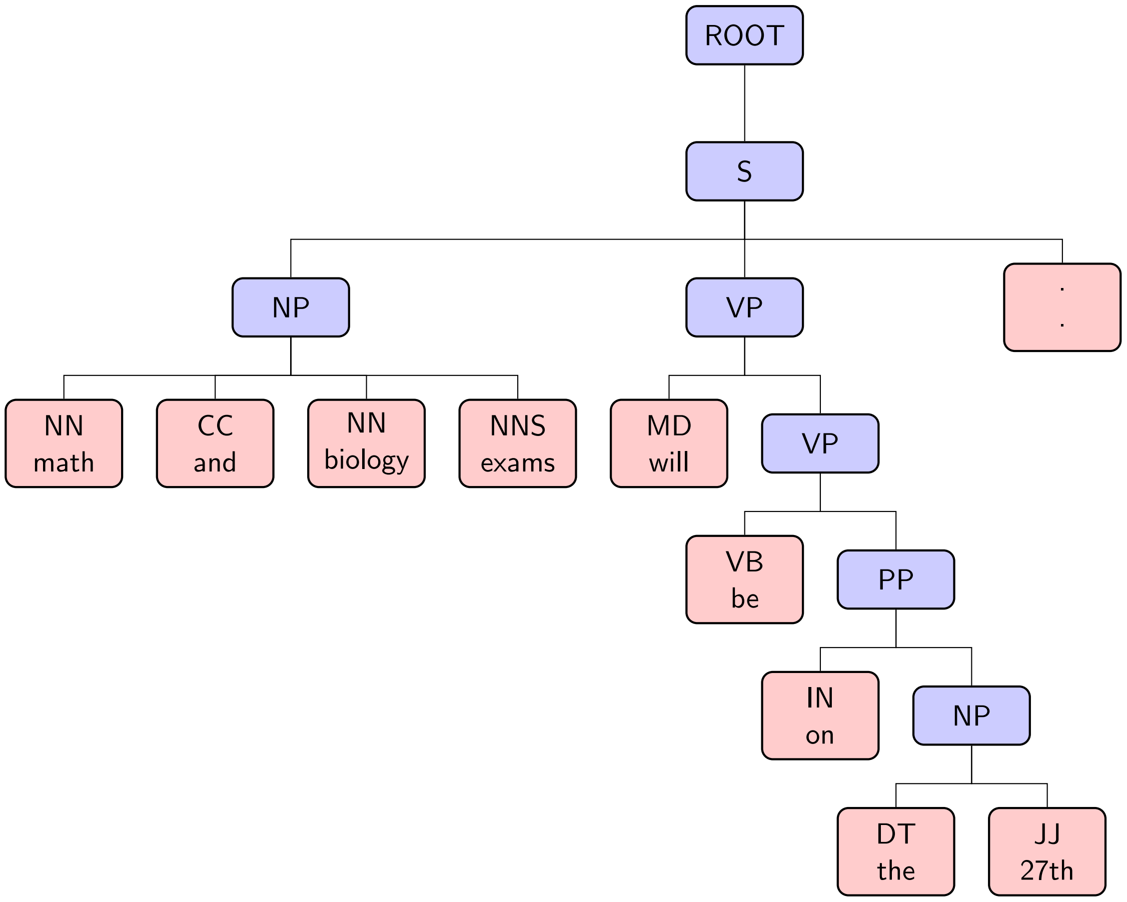


Figure 2.5.: Example of a parse tree

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 clause, because it’s compound with the modal verb “will” and the word “be”. And “on the 27th” is a preposition clause in the verb clause, which is again composed of a preposition “in” and a noun clause “the 27th”.

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 to apply the rules.

2.6.1. Short Rules

Short rules are extracted based on the sequences of adjacent words or their POS tags in the sentence to translate. 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 DT (0.3)
```

The first rule in this example shows, if a the word sequence "after the accident" ever appears in the text, it should be reordered to "the accident after" with a probability 0.5, so will the word order 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 also 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 help the long distance word reordering for translation between English and German. The rules are based on POS tags of the text, and an example is as following:

```
NN X MD VHP -> X MD NN VHP (0.14)
```

The "X" in the example is an placeholder, which can replace one or more words. "VHP" is the right context, which is sometimes helpful to define the reordering boundary. "NN" means noun, "MD" means modal verb and "VHP" is the word "have". In this example, the tag sequence "NN X MD VHP" will be reordered as "X MD NN VHP" with a likelihood of 0.1429.

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 all the words in a part is replaced by the placeholder

2.6.3. Tree Rules

While short rules and long rules are based on the flat structure of the sentence, tree rules reorders the sentence by using information from the sentence's syntactic structure. The syntax tree and alignment of the training corpus are used to train the 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 (0.06)
```

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

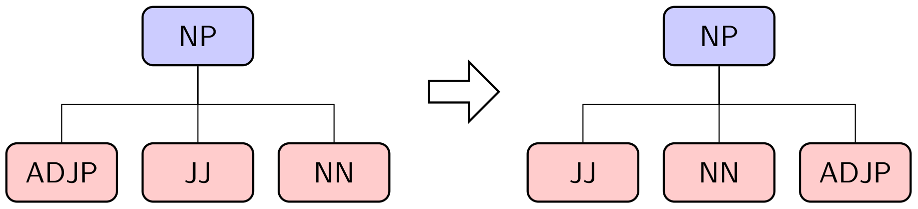


Figure 2.6.: Change subtree order based on tree rules

Tree rules also have some different varieties:

- **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 word order change. A valid word order change that can count for reordering rule needs to fulfill the two criteria in general:

- **Order change exists:** otherwise, there's no need for reordering rules.
- **No aligned ranges coincidence:** the coincidence makes it hard to decide the new word orders in target language.

Rules are not always extracted upon discovering of word change. In order to avoid too excessively concrete rules which don't apply well in general, we extract reordering rules only when the same reordering pattern appears more than a certain threshold. Thus it won't lead to overfitting.

The associated probabilities of 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 many times in the training data, and half of the time, it's reordered as "the accident after", then the probability of the reordering rule is calculated as 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 word will be added.

2.6.5. Rule Combination

In order to further explore the probability of improvement, we combine different rules for reordering in our experiment. This is done by training the different types of rules separately and applying them on the monotone path of the sentence independently. They result different paths in the lattice.

2.7. Oracle Reordering

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 could come after this point. The annotation on the transition indicate the word that could come, together with the probability of this option. The word lattice groups different word reorderings of the same English sentence together, with each reordering corresponding a path from the beginning node to the end node.

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 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 to much translation quality.

An example of a word lattice is showed in figure 2.6. If the probability of a transition is 1, it's left out in the example to keep the graph clear.

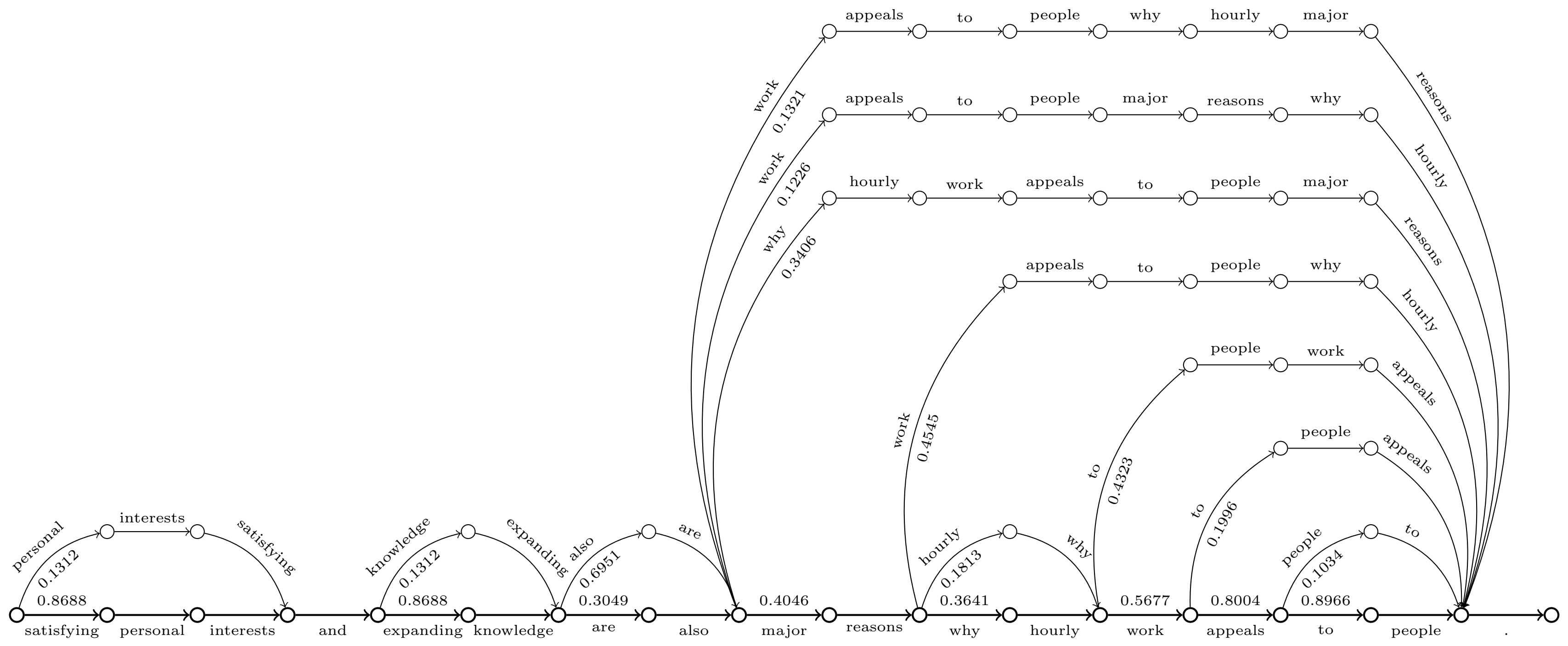


Figure 2.7.: Example of a word lattice

2.9. BLEU Score

BLEU is the de facto standard in machine translationBirch et al. (2010). We use BLEU score to evaluate the SMT system throughout this paper.

2.10. Summary

3. Reordering Approach

3.1. Reordering Problem in Chinese Translation

English and Chinese belong to different language groups. As Chinese belong to the Sino-Tibet language group, while English belong to the Indo-Germanic language group. And they have also developed separately for a long time. Because of their different origins and development, they've becoming two very different languages.

Unlike the most languages in the Indo-Germanic language group, which are more close to English than Chinese, Chinese has some properties those languages don't have, such as Characters as fundamental element instead of letters, the tones, no word separation, the usage of measure words,much more simple inflections and conjugations, which raises further problem for machine translation.

Even the word order between English and Chinese is more distinct. For one, the words in Chinese have generally different origins as those in English, which leads to different vocabulary and sometimes it's very hard to found corresponding words in the other language. For example, Chinese has a lot of different prepositions and adverbs, which have very distinct usage as those in English. Also the continuous writing of Chinese without space makes this problem more severe, since word boundaries are not always so clear in Chinese. Text needs to be segmented before translation, and a word segmentation program decide how to separate the words and the result is not always ideal.

For the other,

And sometimes, a long English sentence with clauses is more suitable to be translated into two of more Chinese sentences. Through analyzing the data we have, we've found several major word order differences between English and Chinese, which leads to low translation quality and should be changed.

Wang et al. (2007) also systematic analyzing about Chinese syntactic reordering. Through our analyzing / research, examples below, worth noticing / improving

"Quote":

Mandarin Chinese sentence structure is quite different than English or other European languages. Since the word order doesn't match, sentences which are translated word-for-word to Mandarin will be difficult to understand. You must learn to think in Mandarin Chinese when speaking the language.

There're plenty of literatures discussing the sentence strucutes of chinese cite.. please.

Preposition phrases

relation clauses

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 are very flexible. They can be located in different position of a sentence both in English and Chinese, such as beginning, middle or end of a sentence, before or after the verb. The location varies and it often depends on the situation. When comparing English and Chinese word order, 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 3.1.: Position change of an adverbial of time

questions

There be ...

3.2. Motivation of Reordering on Multiple Syntactic Levels

3.2.1. Long Distance Word Reordering

efficient than hier

use tree and alignment only

3.2.2. Unstructured Word Reordering

Because English and Chinese have different word orders and there're also some special cases of sentence patterns, the word reordering can be complicated and unsystematically. Sometimes it involves word position changes on multiple syntactic levels. We can see this problem from the examples in figure 3.1. In the figure, the syntactic structure and alignment of the parallel text are presented in a intuitive way.

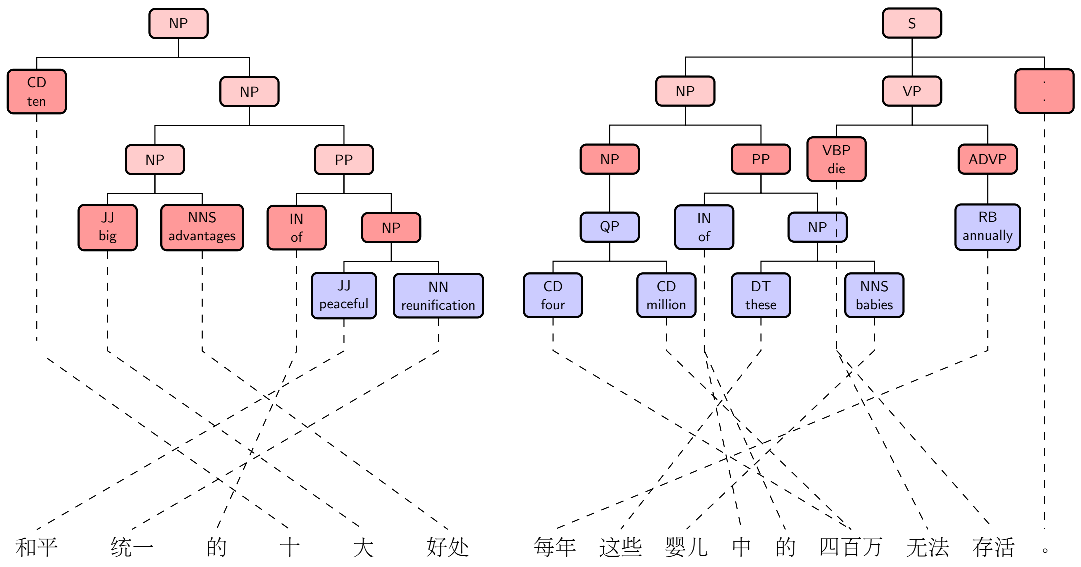


Figure 3.2.: Examples of reordering on multiple syntactic levels

From the examples we can also clearly see how to 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 change, comparing the parallel Chinese text:

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

As we can see, we can't get this pattern if we only observing subtrees on the same level of syntax tree. The node labeled with "CD ten" is inserted into the subtree of its sibling labeled with "NP", between its sibling's two children. This position change can not simply be 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 level, between it and its sibling.

So as a whole, 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 wrong. Second, the syntax tree of the English sentence may not be suitable for translation into Chinese. In these case, MLT reordering could be used as a remedy for incorrect parsing patterns in the syntax tree. Besides due to the very different word orders between English and Chinese, simply reordering the words by change nodes on the same syntax tree level may not do the job, and MLT reordering will be useful in this case.

3.3. The SMT Reordering Algorithm

Inspired the method of tree rules based of reordering and hierarchical SMT model, we created this reordering algorithm. The algorithm solely uses information of the syntax tree and alignment of the source side. It further explores the syntactic structure of text to be translated, and detecting reordering patterns based on multiple hierarchies of the syntax tree.

As we've ready 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 as information for reordering as possible. The algorithm of rule extraction detects the reordering pattern from all nodes in the syntax tree, goes downwards for any number of hierarchies, until it reaches the last hierarchy in the subtrees.

In the implementation, the program conducts a depth-first search (DFS) to traverse every node in the 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're 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 be involved with actual reordered words and have clearly distinguished new order from the target side, i.e. no coincidence of aligned range on the target side.

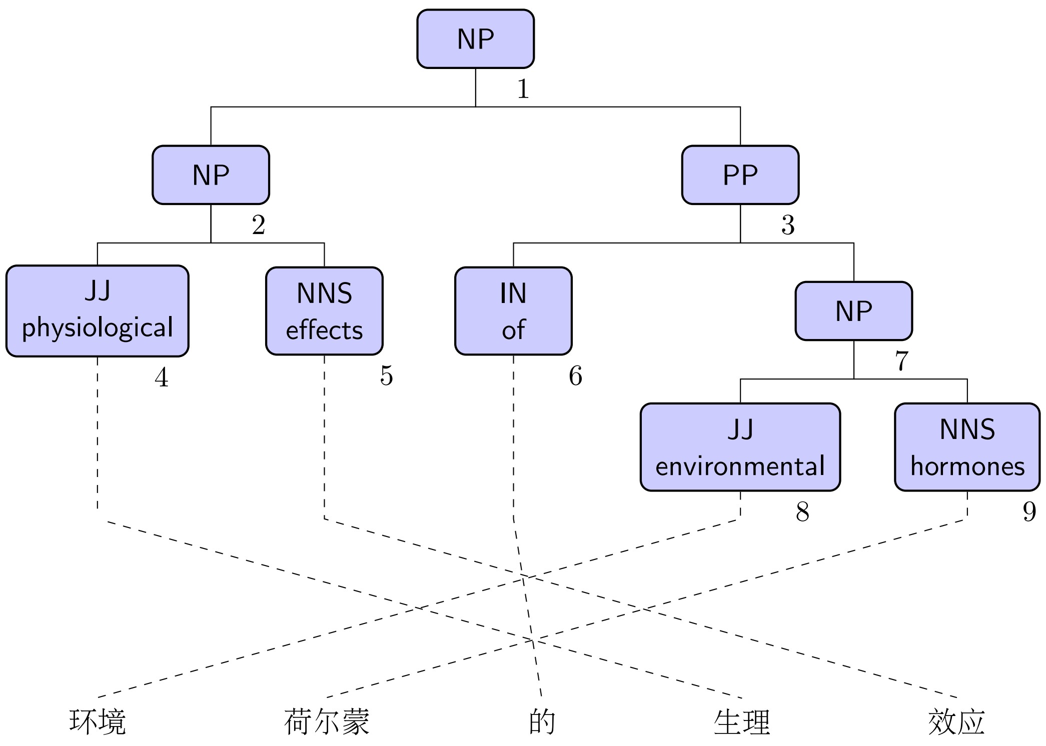


Figure 3.3.: Illustration of rule extraction

Figure 3.2 shows a phrase to be translated together with is syntax tree and alignment to its parallel text in Chinese. The nodes are labels with numbers. In this example, we can find the following reordering patterns:

From node 1:

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

From node 3:

```
PP ( IN NP ) -> NP IN                                        [1 level]
PP ( IN NP ( JJ NNS ) ) -> JJ NNS IN                        *[2 levels]
```

It's noteworthy that our method can detect more reordering patterns than the tree rule based method. For example, the patterns marked with $*$ above can not be extracted with the tree rules based method 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 sequence 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 appearance in the training corpus. In addition, reordering patterns that appear less than a threshold are ignored to be used as reordering rule, in order to prevent too concrete rules without generalization capability and overfitting.

3.3.2. Rule Application

The syntax tree is traverse 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. If 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 2.8.1. 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 rules, which has overlapped effect with each other. These rules are normally patterns that are generated on the same tree, but with different number of levels, which has different generalization effect on the same range of words in the text. For exmaple, the following patterns can be detected from the syntax tree in figure 3.2 in last section:

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

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 by 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 work in a more intuitive way, we present an example. Assume we have several reordering rules and a pre-processed sentence for reordering as follows:

Rules:

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

Sentence:

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

The syntax tree and monotone path as initial word lattice looks as follows:

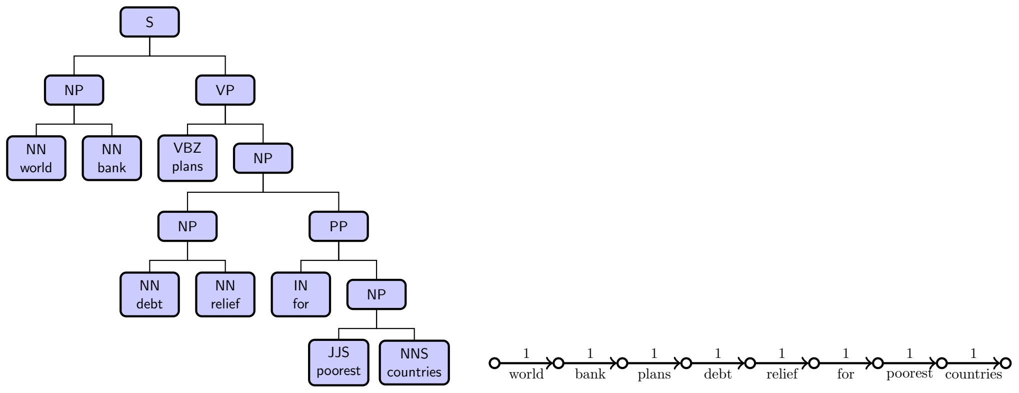


Figure 3.4.: Illustration of rule application (1)

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 position. 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 the words are reordered into "for poorest countries plans debt relief", and the new path with this probability is added to the word lattice.

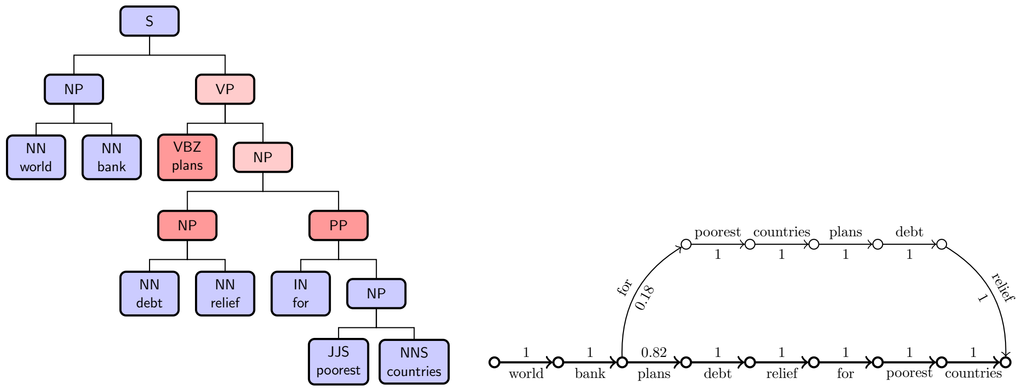


Figure 3.5.: Illustration of rule application (2)

As the program keeps going, it turns out the second rule is 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. It should be noted that there're numbers added to the two "NN" tags in the rule, in order to distinguish them for the reordering. The program uses index for the internal presentation of words, which doesn't cause any confusion.

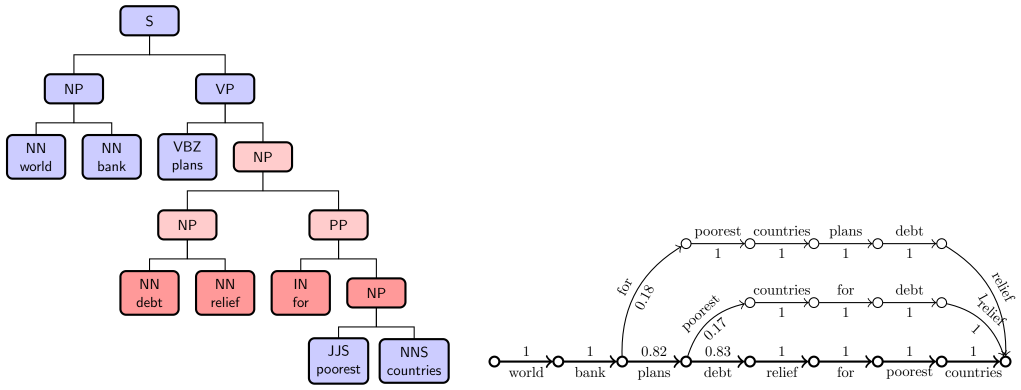


Figure 3.6.: Illustration of rule application (3)

3.4. Summary

advantages

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 the first section of this chapter. The second set of experiments is designed for the other translating direction, which is described in the second section. Through the experiments on both translating direction, we could get a better overview of our methods' effect.

Both sections are composed of three parts: experimental setup, result and evaluation. The first part describes details of the system configurations and experimental data. The second part shows the BLEU scores of different systems for comparison. And the last part evaluates the improvement with translation examples from the experiments.

4.1. English to Chinese Systems

4.1.1. Experimental Setup

We performed experiments with or without different reordering methods covering the English to Chines translation direction. The reordering methods included the reordering with short rules, long rules, tree rules and our MLT 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 pre-reordering, no lexical reordering or other reordering method was used. The text was translated through a monotone decoder. The Chinese text were first segmented into words before use. The reordering rules were extracted based on the alignment, POS tags and syntax trees from the training data. One reference of the test data was used for evaluating the BLEU score. The threshold for rule extraction is set as 5 times and reordering paths with probability less than 0.1 are not added to the lattice. Table 4.1 shows the size of data used in the system.

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.: Details of data in English-to-Chinese system

4.1.2. Results

	BLEU Score	Improvement
Baseline	12.07	
+Short Rules	12.50	3.56 %
+Long Rules	12.99	7.62 %
+Tree Rules	13.38	10.85 %
+MLT Rules	13.81	14.42 %
Oracle Reordering	18.58	53.94 %
Long Rules	12.31	1.99 %
Tree Rules	13.30	10.19 %
MLT Rules	13.68	13.34 %

Table 4.2.: BLEU score overview of English-to-Chinese system

Table 4.2 shows the BLEU scores 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 involves no reordering. In the following rows of the top section, different types of reordering rules are combined gradually, each type per row, and the improvements 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 the improvements are calculated comparing to the baseline in percentage. Each row with a certain reordering type presents all the different variations of the type and the best score under these configurations are shown. For example, long rules also presents the left rules and right rules type. In the lower section of the table, rules types are not combined and the effect of each rule type is shown.

4.1.3. Evaluation

The results shows increasing scores when we used reordering methods from short rules, long rules, tree rule to MLT rules. And better BLEU scores were achieved when we combined the different reordering rules. The MLT rules improved the BLEU score, not only when we used it alone, but also when we added to the other existing reordering rules. But taking a close look at the gap between BLEU score of oracle reordering and the best BLEU score we've achieved, we can also tell, there's still much potential for improvement.

We also found improvement in the translated text by analyzing it manually. Some examples are listed in table 4.3. Each section of table 4.3 shows translation of a sentence in its source language and target language. The translation in the row with "No MLT" comes from the configuration without using MLT reordering, and the translation in the row with "MLT" comes from the configuration with using MLT reordering. From the examples, we can see that the MLT reordering improved the sentence structure significantly.

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 迁难者 家属 表示 诚挚 的 慰问 。
MLT	胡锦涛 还 对 中国 迁难者 表示 哀悼 , 向 迁难者 家属 表示 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 。
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 。

Table 4.3.: Examples of improvements in translated text

From this experiments we can draw the conclusion that our reordering method indeed improves the English-to-Chinese translation quality obviously, no matter when we apply it alone or when we combine it with other reordering methods mentioned in this paper.

4.2. Chinese to English Systems

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 were also used in this experiment by changing the role of the source language and target language. We only used the LDC data set for training, and no TED data were used in this system. And the test data had three English references for evaluating the results instead of one as in the previous system. The data used are again summarized in table 4.4.

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 4.4.: Details of data in Chinese-to-English system

4.2.2. Results

	BLEU Score	Improvement
Baseline	21.80	
+Short Rules	22.90	5.05 %
+Long Rules	23.13	6.10 %
+Tree Rules	23.84	9.36 %
+MLT Rules	24.14	10.73 %
Oracle Reordering	26.80	22.94 %
Long Rules	22.10	6.10 %
Tree Rules	23.35	9.36 %
MLT Rules	23.96	10.73 %

Table 4.5.: BLEU score overview of Chinese to English systems

Table 4.5 shows the BLEU scores for configurations with different reordering methods for the Chinese-to-English translation. The table could be interpreted in the same manner as table 4.2 in the previous section.

4.2.3. Evaluation

We can tell from table 4.5 that MLT rules achieved the best BLEU score under all the rule types, when it was used separately. When the MLT rules was combined with other existing rule types, it also showed the effect to improve the translation. Like the results in the previous section, there's also still a large gap between the score we've achieved and the score of oracle reordering, which leaves further possibilities for improvement.

From this experiments we can draw the conclusion that our reordering method also improves the translation quality for Chinese-to-English direction.

4.3. 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 improvement on the translation quality. The MLT reordering helped to further improve the BLEU score by 3.57%, when it was combined with other reordering methods for translating from English to Chinese. And the improvement on the other translating direction was 1.37%.

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 our MLT reordering method changes the word order more significantly to improve the translation between Chinese and English.

5. Conclusion

And taking a close look at the BLEU score generated with oracle reordering, we can tell there's still potential for improvement.

5.1. Discussion

5.2. Conclusion

still space for improvement (look at oracle score) chinese not reseraches so much as english

5.3. Outlook

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

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

Besides, vector representation is currently also a popular topic for various tasks (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 not-so-good approach 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 more directly.

better algorithm for reordering between chinese english

distributive representation

Appendix

A. Documentation of Pre-Reordering System

Here we explain the details of our pre-reordering system and how to integrate it into the SMT system of our faculty at KIT, as well as other issues. A summary about how to integrate and use the code can be view in the last section 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. Reordering Source Code

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.

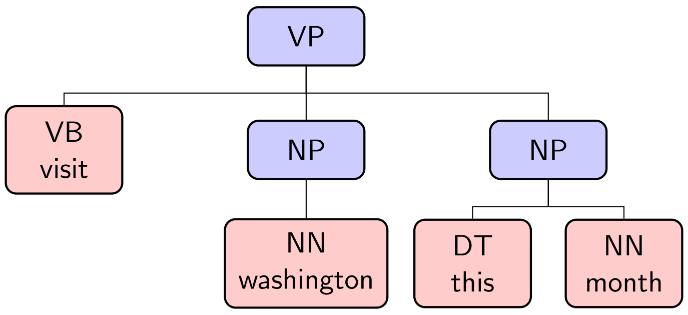
Extract and Apply Command

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.: Illustration of different MLT rule variations

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, probability and occurrences. For example,

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

means the sequence `VB NN (DT NN )` 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 rules 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. Configuration 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 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 tree rule configuration is called `TreeRules`. If it's not, some paths in `ReorderingRules.py` must be changed. See note.
5. Build the system with the modified description file.

List of Tables

List of Figures

Bibliography

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.

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.

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.

Rottmann, K. and Vogel, S. (2007). Word reordering in statistical machine translation with a pos-based distortion model.

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.