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 – 18th 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.

18th August 2014

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 [LP13] 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.

Table 1.1.: Word orders of three different languages

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 [RV], [NK] and [HWNW], we further propose a new data driven, rule based pre-reordering method, which uses rules based on syntactic tree. The method is called Multi-Layer-Tree (MLT) reordering, which orders the constituents on multiple layers of the syntactic 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. Related Work

todo

special problem of chinese: segmentation

1.3. Structure

In this chapter we mainly describe the background and objective of this thesis, including the related research in the next section of this chapter. In the chapter 2 we shows the fundamental knowledge, which is related and relevant to our research. In chapter 3 we introduce our reordering methods in detail. The experiment setup and results are present in chapter 4, together with the evaluation of the methods we use. 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.

2. Foundations

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.

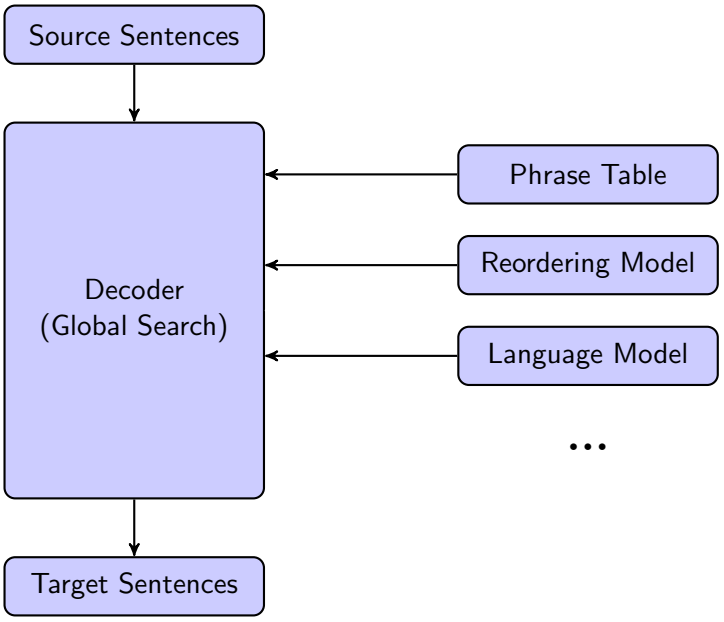


Figure 2.1.: Architecture of SMT system

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 syntactic 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.

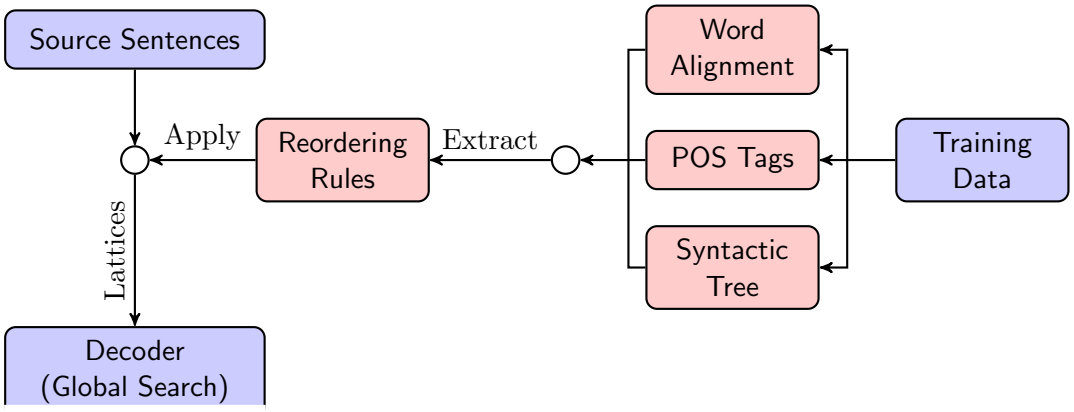


Figure 2.2.: Pre-reordering system

A more detailed description of word alignment, POS tag, syntactic 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.

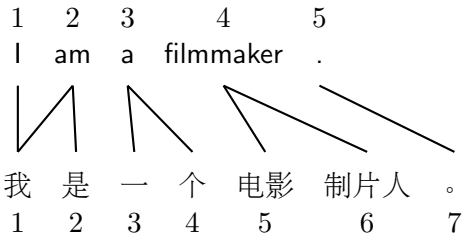


Figure 2.3.: Example of word alignment

Or it may be simply presented as index pair in the file system.

$$1-1 \quad 2-1 \quad 2-2 \quad 3-3 \quad 3-4 \quad 4-5 \quad 4-6 \quad 5-7$$

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. Syntactic Tree

TODO

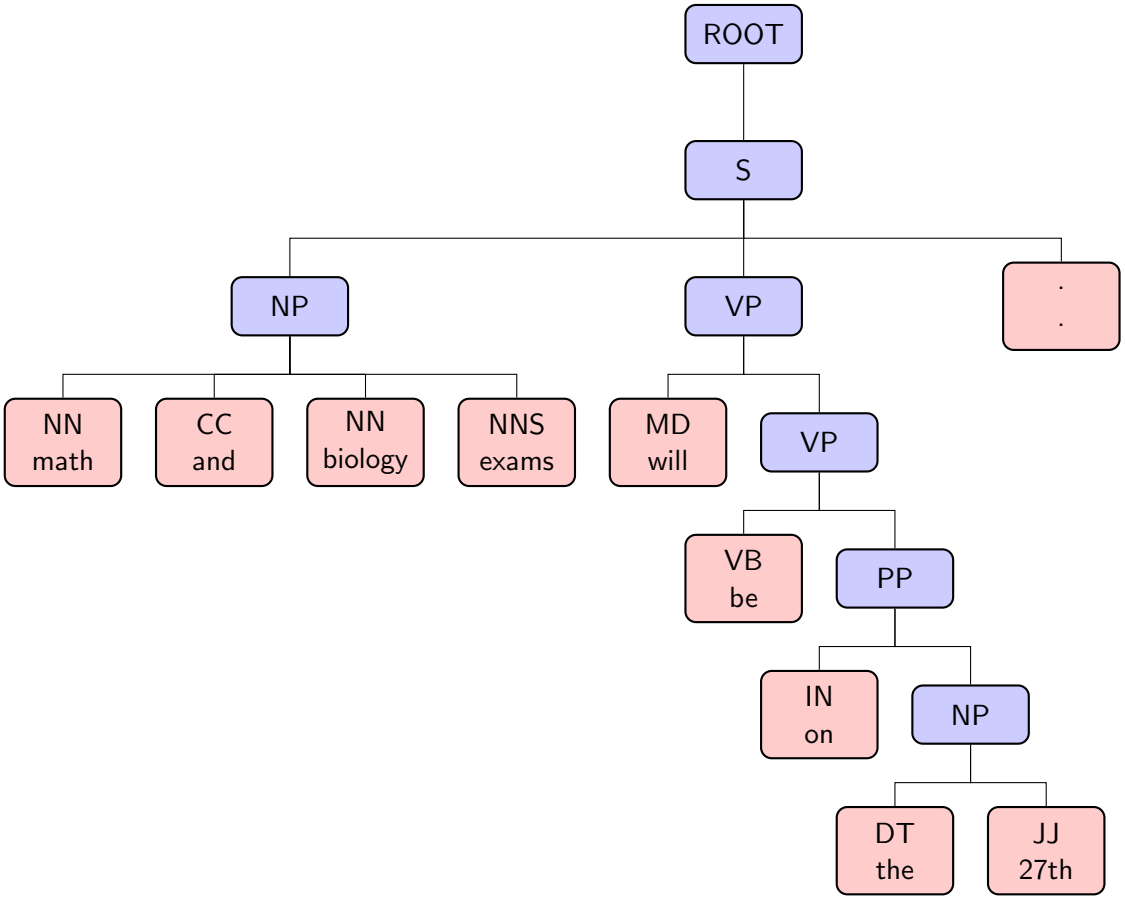


Figure 2.5.: Example of a parse tree

2.6. Reordering Rule Types

2.7. Oracle Reordering

2.8. Lattices

A word lattice could be presented with a directed acyclic graph. The graph contains nodes and transitions, with each transition labeled with a word. One example is showed in the next page. The word lattice in the example groups different word reorderings of the same English sentence together, with each reordering corresponding a path from the beginning node to the end node. The word lattice provides different p pass to decoder, probability on transitions, probability of reordering.

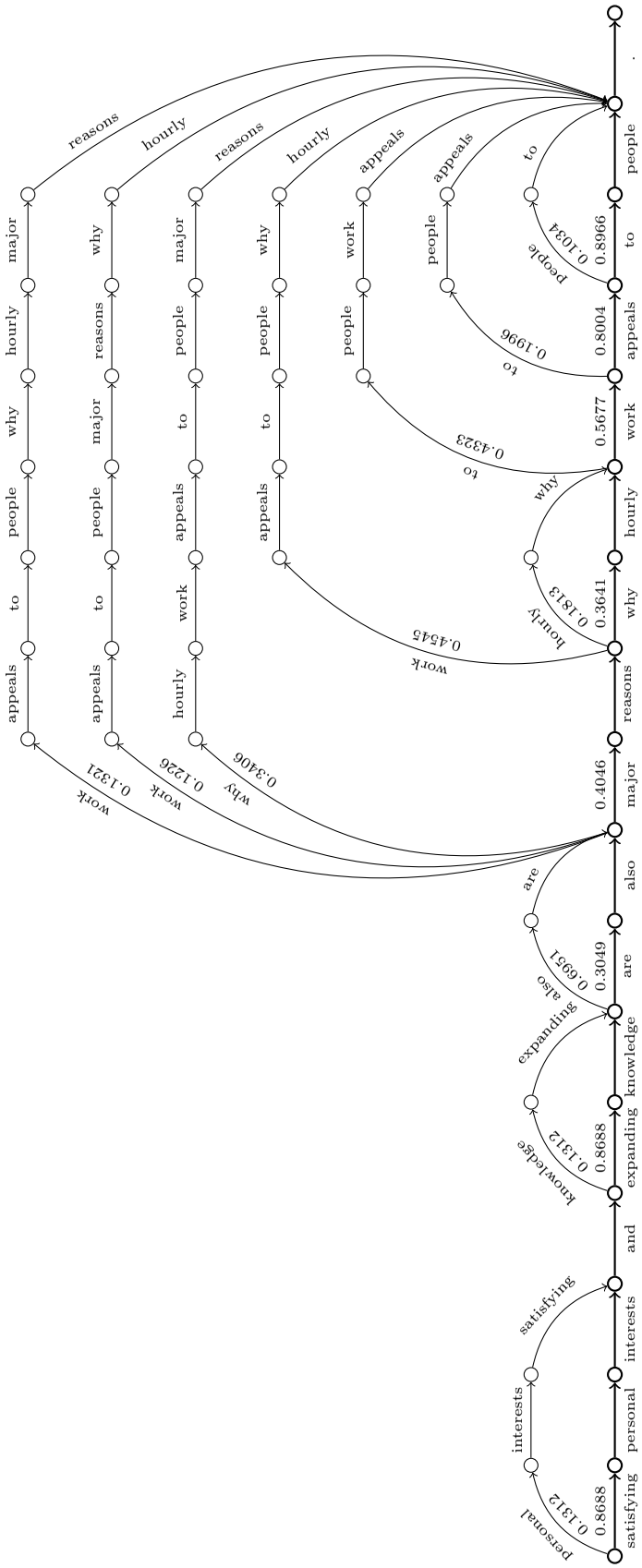


Figure 2.6.: Example of a word lattice

2.9. BLEU Score

BLEU is the de facto standard in machine translation[BOB10]. We use BLEU score to evaluate the SMT system throughout this paper.

3. Reordering Approach

3.1. Reordering Problem in Chinese-English Translation

3.2. Motivation of our Pre-reordering system

3.3. The reordering algorithm

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 syntactic trees from the training data. One reference of the test data was used for evaluating the BLEU score. 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.

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	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

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. In the last example, the part of the sentence "visit Washington this month" was parsed as the structur "(visit (Washington) ((this) (month)))" in the syntactic tree, but the correct Chinese translation has word order corresponds to "this month visit Washington". This reordering inserts the word "visit" between the words "this month" and "Washington", which are on a lower level of the syntactic tree. This behavior could not be done by the tree-rule-based reordering, which only changes order of constituents on the same tree level.

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. And we could further justify our claim that the MLT reordering method changes the word order more significantly to improve the translation quality.

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. Conclusion

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 syntactic 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

5.3. Outlook

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

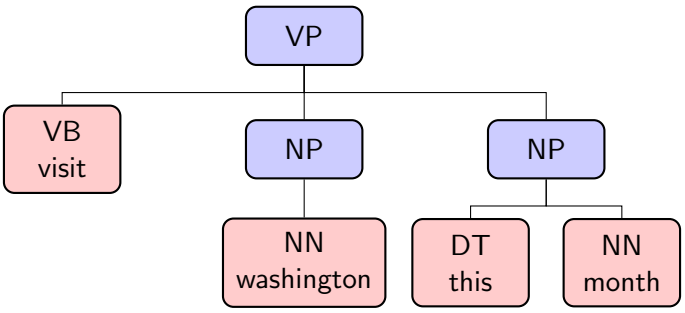
The executable file `extract` has the following usage format:

```
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 syntactic 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 syntactic 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 a format of

`mode.pos.threshold`

or

`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.

If the `<Option>` is the string "test", the test scores of all the configurations will be listed, with one record per line. If the `<Option>` is the string "dev", the dev scores of the last training cycle will be listed for all configurations. In a similar manner, best dev scores under all training cycles will be listed with the string "devmax". If "optimize" or "translate" is given, the program shows if the optimization or translation is finished by checking if the related log file is complete.

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

This script shows information of the depth and branch factor of syntactic 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 syntactic tree. Executing this program will lead one into a command line mode. After the syntactic 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

[BOB10] A. Birch, M. Osborne, and P. Blunsom, "Metrics for mt evaluation: Evaluating reordering," *Machine Translation*, vol. 24, no. 1, Mar. 2010. [Online]. Available: http://dx.doi.org/10.1007/s10590-009-9066-5

[Chi07] D. Chiang, "Hierarchical phrase-based translation," *computational linguistics*, vol. 33, no. 2, pp. 201–228, 2007.

[HWNW] T. Herrmann, J. Weiner, J. Niehues, and A. Waibel, "Analyzing the potential of source sentence reordering in statistical machine translation."

[LP13] U. Lerner and S. Petrov, "Source-side classifier preordering for machine translation," in *Proc. of EMNLP '13*, 2013.

[NK] J. Niehues and M. Kolss, "A pos-based model for long-range reorderings in smt."

[RV] K. Rottmann and S. Vogel, "Word reordering in statistical machine translation with a pos-based distortion model."