Report of Assignment #2 – Translation System

WUJUNCHAO
University of Macao
Mc25653@umac.mo

Abstract

MOSES is an implementation method of statistical machine translation (SMT). In this report, we construct a Chinese-English translation system based on MOSES, compare the scores of the system before and after tuning and explore the translation results. The experimental results show that high-quality parameter tuning can help improve the performance of statistical translation models, while the development data used for tuning contains a lot of long text will damage the performance of the model. In addition, the statistical translation model has the problems of not being smooth, semantically incomprehensible, has many grammatical errors, many translation errors and is difficult to deal with long texts, even though it has a strong ability for lexical translation selection and phrase translation selection.

1 Experimental Environment

Basic Packages Use the *sudo apt-get install [package name]* command to install the following packages: g++, git, subversion, automake, libtool, zlib1g-dev, libicu-dev, libboost-all-dev, libbz2-dev, liblzma-dev, python-dev, graphviz, imagemagick, make, cmake, libgoogle-perftools-dev (for tcmalloc), autoconf, doxygen.

Minimum Software Requirements are as follows:

a) GIZA++16, for word-aligning your parallel corpus
b) Moses
c) KenLM (included in Moses) for language model estimation.

See the appendix for the specific commands of GIZA++ and Moses installation.

Manually Installed Softwares see the appendix for the specific commans:

a) Boost
b) cmph
c) xmlrpc-c

Compile Moses with bjam:

```
./bjam --with-boost=~/boost
--with-cmph=~/cmph
--with-xmlrpc-c=~/xmlrpc-c -j4
```

2 Data Preprocessing

Data The data for this report comes from Professor Derek and is provided on UMMoodle, which comes from the International Workshop on Spoken Language Translation (IWSLT) 2014 evaluation campaign, including training, development and test data.

Tokenize Parallel Corpus I tokenize the acquired parallel corpus data by inserting spaces between words and punctuation. For Chinese data, I use the ANJS toolkit to call the ansjTokenizer.jar JAVA program of ANJS to tokenize. For English data, I use Moses' tokenize.perl script for tokenize. See the appendix for the specific commands.

Specific Parallel Corpus (Truecase) After corpus is tokenized, we convert the phrases and words of each sentence of the parallel corpus into a unified form without format, such as unified capitalization, etc., to reduce the sparsity of the data. The specific steps are:

1). Train the model based on the built-in train-truecaser.perl of moses, and extract the statistical information of the text;

2). Use the built-in truecase.perl of moses to perform truecase on the file after tokenising. See the appendix for the specific commands.

Corpus cleaning In order to control the size of the parallel corpus, the problems in the training process that may be caused by long sentences and empty sentences are reduced. We screen the Chinese and English texts after tokenising, retain only parallel corpora with text vector lengths less than 50, and delete sentences that are obviously misaligned. Its implementation is based on moses's built-in clean-corpus-n.perl. See th appendix for the specific commands.

	Train	Test	Dev
Source	182479	903	1092
Processed	**173918**	903	**815**

Table 1 Comparison of data volume of training, development and test before and after preprocessing

3 Experiments

3.1 Language Model Building

Language models are trained to ensure good output, so use the target language to build. We conduct language model training on English (target language). =The language model tool used is the built-in language model tool KenLM of Moses, and the monolingual corpus trained is the English pair from the Chinese-English translation training set of the International Workshop on Spoken Language Translation (IWSLT) 2014 evaluation campaign.

We build a 3-gram language model and binarize its language model result file to improve its loading speed (see the appendix for the specific code).

The different n-gram language models constructed are shown in Table x. As the number of N increases, the scale of its words increases, but the frequency of occurrence decreases.

LM	Number
1-gram	61240
2-gram	703202
3-gram	1881127

Table 2 N number of different n-gram LMs.

3.2 Translation Model Training

We build Chinese-English translation models for a Phrase-based translation system using the Moses based on the created 3-gram language models. (see appendix for the specific commands)

The training process of the translation model includes :

a) word alignment (with GIZA++),

b) phrase extraction, scoring,

c) lexical reordering table creation,

d) Moses configuration file (moses.ini).

We use the MOSES built-in train-model.perl script to train the model with the default options: alignment heuristic parameter for word alignment is “grow-diag-final”; the reordering model is “msd-bidirectional-fe”.

After training, a translation model (moses.ini) is obtained in the train folder. The translation model’ parameters are shown in table 3 (Un-tune TM).

3.3 Translation Model Tuning

Tuning of a model refers to using data to find the best model or parameters for a given task. In this report, we use the development data to tune the translation model. In the pre-training part, our development data has been full word segmentation and specific to truecase. We use the built-in mert-moses.pl script of Moses for tuning (see the appendix for the specific code), and end up with a translation model (moses.ini) in the *mert-work* folder.

In order to better test the model effect and influencing factors, we set up three groups of experiments:

a) Un-tuned TM: Translation Model without tuning
b) Tune-true TM: Translation Model tuning with truecase but un-cleaned development data
c) Tune-clean TM: Translation Model tuning with cleaned development data

The model size and parameters trained by the three sets of experiments are shown in Table 3.

	Un-tuned TM	Tune-true TM	Tune-clean TM
Size	**535.2M**	**535.2M +78.7M**	**535.2M +75.3M**
Lexical Reordering	0.300	0.060	0.028
	0.300	0.049	0.018
	0.300	0.127	0.116
	0.300	0.074	0.093
	0.300	-0.021	0.004
	0.300	0.051	0.073
Distortion	0.300	0.029	0.015
LM	0.500	0.136	0.089
Word Penalty	-1.000	-0.223	**0.334**
Phrase Penalty	0.200	0.006	0.047
Translation Model	0.200	0.066	0.017
	0.200	0.067	0.043
	0.200	0.037	0.038
	0.200	0.053	0.085
UnknowWord Penalty	1.000	1.000	1.000

Table 3 The size and specific parameters of the model trained by the three sets of experiments. The parameters of Un-tune TM are initialized, while the parameters of Tune-true TM and Tune-clean TM are fine-tuned on the basis of Un-tune TM.

3.4 Translation Model Test

Extra Process for Test Data Before the test of our models, we also tokenize and truecase the test data. Then, we filter the test data (see the appendix for the specific commands) to keep only the items that need to be transformed into the test data, which further improves the translation speed of our models.

Binarize the Phrase Table and Lexical Reordering Model We translate the test data into English using the trained MOSES model to test the performance of our translation model. In the pre-training part, the test data we will use for testing has been fully tokenized. It takes a lot of time and resources to directly start the translation model obtained by training. In order to make it start quickly, before that, we binarize the phrase table and the lexical reordering model, based on the *processPhraseTableMin* provided by Moses and cmph (See the appendix for the specific code). After processing, we get a new Translation Model(moses.ini) that greatly speeds up loading and running.

It should be noted that in the new Moses.ini, we need to modify two paths("~" refers to the user path):

a) PhraseDictionaryCompact:
`~/mosesdecoder/corpus/working/binarised-model/phrase-table.minphr`

b) LexicalReordering:
`~/mosesdecoder/corpus/working/binarised-model/reordering-table`

Test Performance of Models We use the test set to measure the performance of our models and use the new translation models (Moses.ini) processed above to translate the test data (see the appendix for the specific commands).

The results of the translation model need certain indicators to evaluate, and we choose the BLEU standard to evaluate the results. BLEU is a type of machine translation result evaluation. Its essence is to calculate the frequency of co-occurrence words in two sentences and get a comprehensive score, which is mainly used to measure the degree of consistency between the two sentences. The higher the score, the better the machine translation effect.

Run the BLEU script *multi-bleu.perl* to test the decoder and we get the BLEU scores of three models that are shown in table 4.

Model	BLEU Score
Un-tuned TM	11.98
Tune-true TM	**11.80 (- 0.18)**
Tune-clean TM	**12.24 (+0.26)**

Table 4 BLEU scores of 3 models. For the Translation Model tuning with cleaned development data, its performance is significantly improved compared to the un-tuned model. While the performance of the model tuned on the uncleaned development data is impaired.

4 Analysis

Table 5 shows some cases of the final translation results.

ID	Translation Pair
1	我的一个同学的第一个病人是一个纵火犯。 one of my classmates first patient is a 纵火犯.
2	如果你和二十几岁的人工作，如果你爱一个二十几岁的人， 如果你因为二十几岁的人而失眠，我就想看到— 好的。棒极了，二十几岁的人非常重要。 if you and 二十几岁 people , if you love 一个二十几岁 , if you because 二十几岁 people lost , I want to see -- okay . awesome , 二十几岁 people very important .
3	我在微软应用科学实验室， 和我的导师卡迪•布兰葛 重新设计了这台计算机 把键盘上方的小空间 挪进数字工作区内。 I was at Microsoft lab , and all of my mentors 卡迪•布兰葛 to design this computer , on the keyboard to move into the space , in numbers .
4	因为，我这里有个增强版在线试衣间。 because I , here 's an enhanced version 试衣间 in line .
5	我们知道一份职业中的前 10 年 对于你将会挣多少钱 有非常大的影响。 we know a career in for 10 years ago , how much money you 're going to make a very big impact .
6	因此当我们说到儿童发展， 我们都知道前 5 年是大脑发展 语言和爱慕的关键时期。 so when we say children , we all know about five years ago , the brain is the key admiration and language , and development period .

Table 5 some cases of the final translation results

Comparative analysis of the three models

a) It is obvious that high-quality tuning can help improve the performance of statistical translation models, however, when the development data of the tuning contains a lot of long text, it will actually damage the performance of the model.

b) At the same time, we can infer that long text is a big challenge for statistical machine translation. Once long-distance sequencing is involved, it is very difficult to generate translations, and the effect is often not good enough.

c) Word Penalty can be a parameter that has a large impact on the model.

Analysis of specific translation examples

a) As shown in the cases of translation results, the results of statistical translation often present a situation that is blunt, fluent on the surface but semantically incomprehensible, and it is difficult to even be called “human language”.

b) Statistical translation models are prone to many translation errors because they cannot handle complex semantic phrases. For example, in both cases 5 and 6, "before x years" is translated into "x years ago", which should actually be translated as "in the x years of".

c) Statistical machine translation cannot translate unregistered words that do not appear in the training set or words that appear very rarely. As in the translation ccase 2, the OOV “纵火犯” and the proper noun “卡迪•布兰葛”. “二十几岁” is also an untranslatable word that occurs only once in the training set and cannot be statistically aligned effectively. Therefore, there is reasonable that word segmentation has a great influence on statistical machine translation.

d) We can also infer that material that is not similar in content to the training data or that is cross-domain presents significant challenges for statistical machine translation. Phrases will not be well aligned because there will be a lot of proper nouns, slang, idioms, or text with a completely different overall style.

e) Statistical machine translation has a strong ability to select lexical translations and phrase translations, but once it involves long-distance sequencing, it is very difficult to generate translations, and the effect is often not good enough. This leads to the results of statistical machine translation. If you observe carefully, you feel that most words and phrases are translated well. When you read the whole translation, you often feel that the sentences are not smooth, and at the same time, the problem of missing translation will be introduced. As in case 6.

5 Conclusion

In conclusion, this experiment implements a statistical machine translation system based on Moses and explores its model parameters and model performance before and after tuning. In addition, the performance of statistical translation is also analyzed according to the translation results.

We found that tuning the initialized statistical model with high-quality development data is effective, but hurts the performance of the model when there is a lot of long text in the development data.

We also found that the results of statistical translation often present a blunt, unsmooth, and semantically incomprehensible situation, and at the same time have a large number of grammatical errors, and have great defects, even if it has a great ability to select lexical translations and phrase translations. strong. In addition, the results of statistical translation are affected by the content correlation between the training corpus and the test corpus, the text including a large number of proper nouns, slang words, idioms or texts with completely different overall styles, and phrases will not be well translated. Lastly, word segmentation is also a key factor affecting the quality of translation.

References

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, and others. 2007. Moses: Open source toolkit for statistical machine translation. In Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, pages 177–180.

Appendix

Commands All the hyperparameters are kept as the same as the default setting. On the command line, ~ refers to my user path: /home/wujunchao

Environment Installation For installation of GIZA+, my command is:

```
git clone https://github.com/moses-smt/
giza-pp. git
cd giza-pp
make
```

For installation of GIZA+, my command is:

```
git clone https://github.com/moses-smt/
mosesdecoder.git
cd mosesdecoder
mkdir tools
cp ~/giza-pp/GIZA++-v2/GIZA++ \
~/giza-pp/GIZA++-v2/snt2cooc.out \
~/giza-pp/mkcls-v2/mkcls tools
```

For installation of Boost, my command is:

```
tar zxvf boost_1_64_0.tar.gz
cd boost_1_64_0/
```

```
./bootstrap.sh
./b2
```

For installation of cmph, my command is:

```
tar zxvf cmph-2.0.2.tar.gz
cd cmph-2.0.2/
./configure --prefix= ~/cmph
make
make install
```

For installation of xmlrpc-c, my command is:

```
tar zxvf xmlrpc-c_1.33.14.orig.tar.gz
cd xmlrpc-c-1.33.14/
./configure --prefix= ~/xmlrpc-c
make
make install
```

Data Preprocessing Chinese tokenizer based on ANJS toolkit, my command is:

```
java -jar ansjTokenizer.jar train.zh \
tok_train.zh
```

English tokenizer based on tokenizer.perl built into Moses, my command is:

```
~/mosesdecoder/scripts/tokenizer/tokeniz
er.perl -l en \
<~/mosesdecoder/corpus/train.en> \
~/mosesdecoder/corpus/tok_train.en
```

Train the model based on train-truecaser.perl, my command is:

```
~/mosesdecoder/scripts/recaser/train-
  truecaser.perl \
--model   ~/mosesdecoder/corpus/truecase-
  model.train.zh \
--corpus
~/mosesdecoder/corpus/tok_train.tags.zh-
en.zh
```

Truecase based on truecase.perl, my command is:

```
/home/wujunchao/mosesdecoder/scripts/rec
  aser/truecase.perl \
--model
  /home/wujunchao/mosesdecoder/corpus/tr
  uecase-model.train.zh \
</home/wujunchao/mosesdecoder/corpus/tok
  _train.tags.zh-en.zh> \
/home/wujunchao/mosesdecoder/corpus/true
_train.tags.zh-en.zh
```

LM Training Train a 3-gram language model based on lmplz, my command is:

```
mkdir ~/mosesdecoder/corpus/lm
cd ~/mosesdecoder/corpus/lm
~/mosesdecoder/bin/lmplz -o 3 \
< ~/mosesdecoder/corpus/true_train.zh> \
arpa_train.zh
```

Use build_binary for the language model file binary:

```
~/mosesdecoder/bin/build_binary \
arpa_train.zh \
blm_train.zh
```

Translation Model Training Train the SMT model based on train-model.perl, my command is:

```
Nohup nice
~/mosesdecoder/corpus/mosesdecoder/scrip
ts/training/train-model.perl -root-dir
train \
-corpus
  ~/mosesdecoder/corpus/clean_dev.en \
-f en -e zh -alignment grow-diag-final-
  and -reordering msd-bidirectional-fe \
-lm 0:3:~/corpus/lm/blm_dev.en:8 \
-external-bin-dir
~/mosesdecoder/tools >& training.out &
```

Model Tuning Tune the SMT model based on mert-moses.pl, my command is:

```
nohup nice
/home/wujunchao/mosesdecoder/scripts/tra
ining/mert-moses.pl \
~/mosesdecoder/corpus/clean_dev.zh mo-
  sesdecoder/corpus/ true_de.zh \
~/mosesdecoder/bin/moses \
train/model/moses.ini --mertdir
  ~/mosesdecoder/bin/ \
&> mert.out &
```

Mode Testing Binarized model, my command is:

```
mkdir ~/mosesdecoder/working/binarised-
  model
~/mosesdecoder/bin/processPhraseTableMin
\
-in train/model/phrase-table.gz -nscores
  4 \
-out binarised-model/phrase-table
~/mosesdecoder/bin/processLexicalTableMi
  n \
-in train/model/reordering-table.wbe-
  msd-bidirectional-fe.gz \
-out binarised-model/reordering-table
```

Test set filtering, my command is:

```
~/mosesdecoder/scripts/training/filter-
model-given-input.pl \
filtered-corpus mert-work/moses.ini \
~/mosesdecoder /corpus/true_tst.zh \
-Binarizer
~/mosesdecoder/bin/processPhraseTableMin
```

Test set translation, my command is:

```
nohup nice ~/mosesdecoder/bin/moses \
-f ~/mosesdecoder/working/filtered-
corpus/moses.ini -i \
< ~/mosesdecoder/corpus/true_tst.en> \
~/mosesdecoder/corpus/working/trainslate
d_tst.zh \
2>
~/mosesdecoder/corpus/working/tst1.out
```

Evaluation of translation results(BLEU) , my command is:

```
~/mosesdecoder/scripts/generic/multi-
bleu.perl \
-lc ~/mosesdecoder/corpus/true_tst.zh \
<
```

```
~/mosesdecoder/corpus/working/trainslate
d_tst.zh
```