NTCIR-7 Experiments in Patent Translation based on Open Source Statistical Machine Translation Tools

Ze-Jing Chuang
WebGenie Information Ltd.
No.207-1, Sec. 3, Beisin Rd.,
Sindian, Taipei County, Taiwan
(R.O.C.)
terry@webgenie.com.tw

Yuen-Hsien Tseng
National Taiwan Normal University
No.162, Sec. 1, Heping East Road.,
Taiwan, Taiwan (R.O.C.)
samtseng@ntnu.edu.tw

Abstract

This paper describes our experiment methods and results in the NTCIR-7 Patent Translation Task [1]. *As the first step of our research in machine translation, we integrated a series of open source software to build a statistical translation model. The experiment results demonstrated that we still need to improve the performance and efficiency in both model training and testing.*

Keywords: *Patent Translation, GIZA++.*

1. Introduction

In this experiment, we create a statistical machine translation system by integrating several open sources tools. The tools used in our experiment include JUMAN [2], CMU-Cam Language Model Toolkit [3], GIZA++ [4], mkcls [5], and ISI Rewrite Decoder [6].

These tools were used as follows. Firstly, we applied a tokenization tools called JUMAN for word segmentation in Japanese sentences. The preprocessed corpus was then used for language model training directly by use of CMU-Cam Language Model Toolkit v2. For translation model training, an assistant tool called mkcls was applied to automatically define the classes of words in the preprocessed corpus before training. And the preprocessed bilingual corpus and pre-defined classes were fed to GIZA++ to build the translation model. Finally, the ISI Rewrite Decoder was used to translate Japanese language into English language based on the trained translation model. The details of each step are described in the following sections.

2. Corpus Preprocessing

To utilize the model training and translation tools, we need to modify the corpus format given by NTCIR. Our first step is language separating. We separated all Japanese-to-English mapped sentences into the Japanese part and the English part. The mapping between two languages is recorded individually to assure the correctness of model training. The second step is tokenization. This step in English is simply to divide all the words and punctuations by space in all sentences. But in Japanese language, to identify each "token" (or called word segmentation) is more complicated. To do this job, we install a Japanese word segmentation tool: JUMAN v.5.1 [2]. JUMAN is able to read a Japanese sentence and report the segmentation result. But it is not flexible in terms of input/output options: 1) It can only read files in Shift-JIS encoding; 2) It outputs segmented result in report format, not in word-separated sentence format. Therefore, we have to develop a "wrap program" to do Japanese word segmentation. This program wrapped the following process:

1. Converting the Japanese sentence from EUC-JIS encoding to Shift-JIS encoding.
2. Executing JUMAN to segment all Japanese sentences.
3. Converting the result from Shift-JIS encoding to UTF-8 encoding.
4. Transfer the result format to word-segmented sentence format.

Since the tokenization step in English is very simple, it takes about 13 minutes to tokenize 1.8 million English sentences. But the word segmentation process is more complex for Japanese. It takes 5 hours and 15 minutes in word segmentation, including all processing in the wrap program.

3. Language Model Training

We use CMU-Cam Language Model Toolkit v2 [3] for language model training. The parameter setting is listed as follow:

1. Using binary file as input and output.
2. N-gram size is set to 3.
3. Out-of-vocabulary (OOV) handle: type 1 (i.e., OOV words are all mapped to the same symbol, and treats this symbol the same way as any other word in the vocabulary)
4. Discounting strategy: Good Tuning
5. Range of Good Tuning: 1, 7, and 7 for unigram, bigram, and trigram respectively.
6. Unigram probability of unseen words: 0.5.

Other parameters not in the above list are set to their default values in the CMU-Cam Language Model Toolkit.

The language model training with this tool is fast. It takes less than 1 minute when the corpus size is two hundred thousand sentences, and about 7 minutes when the corpus size is 1.8 millions.

4. Word Class Construction

To automatically define the class of each word, we invoke mkcls [5]. The parameters we input to mkcls are selected according to the usage manual:

1. Number of classes: 80.
2. Number of optimization runs: 10.

It spends lots of time word class generation. The corpus size and the corresponding time spending are shown in Table I.

Table I. Time spending in word class generation.

Corpus size	Language	
	English	Japanese
200,000	57 mines	2 hrs, 3 mines
500,000	2 hrs, 38 mines	4 hrs, 19 mines
1,800,000	11 hrs, 51 mimes	24hrs, 23 mines

5. Translation Model Training

Using the preprocessed corpus and pre-created word class, we apply GIZA++ [4] for Japanese-to-English translation model training. The only one parameter we set here for training is p0 of IBM model 5, the value is 0.98.

Since it is time-consuming for translation model training, we only finish the training for the corpus size of 200,000 sentences. The corpus size and the corresponding time spending are shown in Table II.

Table II. Time spending of translation model training.

Corpus size	Time spending
10,000	2 hrs, 12 mines
100,000	7 days, 2 hrs
200,000	4 weeks
More than 200,000	N/A

According to Table II, the time complexity of the translation model training is about $O(n^2)$. That means to train the 1.8 millions sentences will take about 324 weeks! As a result, we only use the translation model resulting from the training the 200,000 sentences.

6. Language Translation

After generating the language model and the translation model, we use ISI Rewrite Decoder [6] to translate Japanese to English. Except for the input of language model and translation model, there is no additional parameter to set for this tool. To fit the translation input/output format required by NTCIR (XML format for input file and report format for output file), we develop another wrap program to do the format transformation.

7. Conclusion

Because of the enormous time consumed in the translation model training, we can not finish our training for the full bilingual corpus in time. Therefore we only send the translation result based on the model trained using only 200,000 sentences. This leads to a result which almost gets the lowest performance in this year's NTCIR patent translation experiment, which is shown in Table III.

Table III. Evaluation results of MT task formal run.

Group-ID	K
RUN	1
BLUE	1.41
ADEQUACY	108.33
FLUENCY	104.33

We believe that there should be other ways to increase the performance of the system we build based on these open source tools. So far we still try different parameter settings and difference tools. We hope that we can build a statistical machine translation system better than the current one in the future.

References

[1] Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, and Takehito Utsuro. Overview of the Patent Translation Task at the NTCIR-7 Workshop. Proceedings of the 7th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-lingual Information Access, 2008.

[2] JUMAN v5.1 online manual: *http://www-nagao.kuee.kyoto-u.ac.jp/nl-resource/juman-eng.html*

[3] CMU-Cam Language Model Toolkit online manual: *http://svr-www.eng.cam.ac.uk/~prc14/toolkit_documentation.html*

[4] GIZA++ online manual: *http://www.fjoch.com/GIZA++.html*

[5] mkcls online manual: *http://www.fjoch.com/mkcls.html*

[6] ISI Rewrite Decoder online manual: *http://www.isi.edu/natural-language/software/decoder/manual.html*