Word Alignment Using GIZA++ on Windows

Liang Tian
Department of Computer and Information Science
University of Macau, Macau S.A.R, China
ma96572@umac.mo

Fai Wong
Department of Computer and Information Science
University of Macau, Macau S.A.R, China
derekfw@umac.mo

Sam Chao
Department of Computer and Information Science
University of Macau, Macau S.A.R, China
lidiasc@umac.mo

Abstract

In Linux system environment, it is very common and convenient to use the word alignment generated from GIZA++ for most statistical machine translation (SMT) systems. While it is not the story for many researchers used to conducting their research under the Windows platform. Although either Cygwin or Virtual Machine (VM) can be used to emulate Linux environment in Windows environment, it is not always a good method. In this paper, we mainly want to share our experiences on how to use GIZA++ on Windows.

1 Introduction

There are many applications for word alignment in natural language processing, and most of them depend on the quality of word alignment (Och and Ney, 2000; Yarowsky and Wicentowski, 2000). A frequently used application system for word alignment is the automatic extraction of bilingual lexicon and terminology from parallel corpus (Smadja et al.,1996; Melamed, 2000). Och and Ney (2003) compare various methods for computing word alignments using statistical and heuristic models and then develop a statistical word alignment toolkit, GIZA++, which is the mostly used package in statistical machine translation (SMT) nowadays.

GIZA++ is part of the statistical machine translation toolkit used to train IBM Model 1 to Model 5 (Brown et al., 1993) and the Hidden Markov Model (HMM) (Och et al., 2003). It is part of the SMT toolkit EGYPT which was developed by the SMT team during the summer workshop in 1999 at the Center for Language and Speech Processing at Johns-Hopkins University (CLSP/JHU) *.

With the help of Expectation-Maximization (EM) algorithm, final word alignment results can be obtained after GIZA++ trains the parallel corpus several iterations from two directions (source to target language and vice-versa). Various heuristics, such as *grow-diag-final* (Koehn et al., 2003) can be applied to obtain a better symmetrical alignment *a* from those two directions. Word alignment results can be used to build phrase table if *biphrases* (*e'*, *f'*) alone with their alignment *a'* satisfy the following two conditions (Galbrun, 2009):

a) *e'* and *f'* are consecutive word subsequences in the target sentences *e* and source sentence *f* respectively and neither of them is longer than *k* words.

b) *a'* is the alignment between the words of *e'* and *f'* induced by *a*, which contains at least one link from *a*.

Table 1 shows an example of extracting phrases (refer to consecutive sequence of words) according to the English-Chinese word alignment in Figure 1. Here the phrase length *k* is not more than 7 words.

Suppose the Chinese word "能够" in Figure 1 is only aligned to "able" and the other alignments are the same, as shown in Figure 2. After the phrase extraction in line with the two conditions given above, *biphrase* (我 能够 ||| I am able to) can still be aligned, while the *biphrase* (能够 ||| am able to) cannot. In other words, "我 能够" can be correctly translated into "I am able to", while "能够" cannot

*http://www-i6.informatik.rwth-aachen.de/Colleagues/och/software/GIZA++.html

give the right translation "am able to". That is to say better word alignment can obtain a better phrase table. Figure 3 illustrates various phrases translation process.

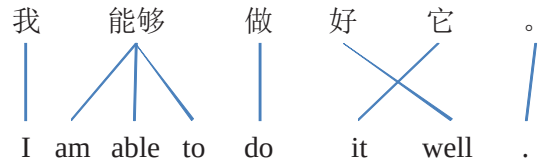

Figure 1. An example of word alignments

(我 ||| I);
(我 能够 ||| I am able to);
(我 能够 做 ||| I am able to do);
(我 能够 做 好 它 ||| I am able to do it well);
(我 能够 做 好 它 。||| I am able to do it well .);
(能够 ||| am able to);
(能够 做 ||| am able to do);
(能够 做 好 它 ||| am able to do it well);
(能够 做 好 它 。||| am able to do it well .);
(做 ||| do);
(做 好 它 ||| do it well);
(做 好 它 。 ||| do it well .);
(好 ||| well);
(好 它 ||| it well);
(好 它 。||| it well .);
(它 ||| it);
(它 。 ||| it .);
(。 ||| .)

Table 1. Content of phrase table

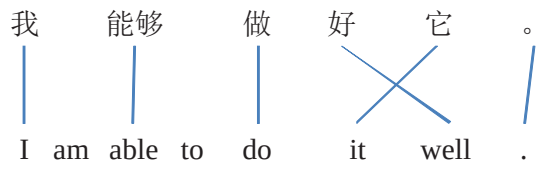

Figure 2. An alternative alignment of words, where "能够" is aligned to "able" instead of "am able to"

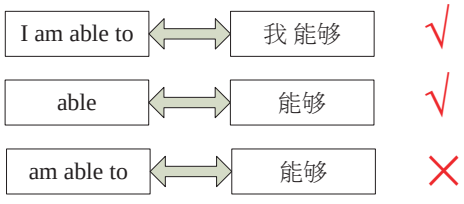

Figure 3. Translations driven by alignments in Figure 2

Sometimes some GIZA++ experiment results will be dealt in Windows system, however, GIZA++ is now compiled in Linux, Irix and Sun operating systems. Although Cygwin and Virtual Machine can help us to achieve the purpose, it is quite complicated to carry out experiments by switching the intermediate results between the Linux and Windows platforms. As we know, some attempts have been made to transfer GIZA++ from Linux to Windows with the help of Visual Studio 2003 and STLport4.6.2. However, some files cannot be successfully compiled in our trial. Hence, we have a motive to make GIZA++ run in the Windows environment. In this paper, we would like to share the experience in developing an online word alignment system (using the GIZA++) that run under the Windows environment.

In the following sections, main techniques will be described in the second part. Conclusion and future work will be presented in the third section.

2 Implementation Details

In this section, the developing steps will be described. To get the GIZA++ executable files, some preparation work should be done. Then the GIZA++ required parameters should be integrated in a batch processing file. Finally, the batch file will be called by the `ShellExecute()` function in Visual Studio 2008 (VS 2008).

Software	Function
`Cygwin`	Compile GIZA++ file;
`VS 2008`	Dialog-based MFC; Call executable files; Output the alignment result
`STLport 4.6.2`	Serve for VS 2003/2008 to compile GIZA++

Table 2. Software required for the development

2.1 Preparation Tasks

The main purpose of this step is to obtain some executable files that can be run in Windows. As different compilation environment will be used between Linux and Windows, files compiled on Linux system cannot directly execute in Windows. Fortunately, Windows recognizable file (.exe file) can be obtained from Cygwin. Besides this method, some executable files can also be prepared by using VS 2003 and STLport 4.6.2. Table 2 lists the software packages that we have applied to the implementation the alignment program for run in

```
mkcls.exe -c80 -n10 -pchinese -V pchinese .vcb.classes opt
mkcls.exe -c80 -n10 -penglish -Venglish.vcb.classes opt
plain2snt.out  chinese  english
GIZA++.exe -S chinese.vcb -T english.vcb -C chinese_english.snt
```

Table 3. Content of `giza.bat`

`WinExec()` 1) `UINT WinExec(LPCSTR lpCmdLine, UINT uCmdShow);` 2) Two parameters and provide only for compatibility with 16-bit Windows.
`ShellExecute()` 1) `ShellExecute(HWND hwnd, LPCTSTR lpOperation, LPCTSTR lpFile, LPCTSTR lpParameters, LPCTSTR lpDirectory, INT nShowCmd);` 2) Six parameters and easy to use for most cases.
`CreatProcess()` 1) Ten parameters 2) Creates a new process and its primary thread, most of the time we will not use so many parameters.

Table 4. Functions used to call executable file

Windows XP and 7 systems. According to our experiment, it is quite straight forward to compile GIZA++ in Cygwin, while the second method not only needs to make some changes of the GIZA++ codes, but also cannot compile all the files, such as the word classes toolkit `mkcls` and the word counting program `plain2snt.out`.

In order to fully use the parameters of GIZA++, all of those parameters will be put in a single batch file, `giza.bat`. Table 3 shows an example. Here "`chinese`" and "`english`" are the file names of the bilingual corpora.

2.2 Calling Executable Files

Generally speaking, there are three different ways for calling the executable files, as shown in Table 4. Considering the number of parameters and simplicity, `ShellExecute()` is chosen as our favor. The following shows an example about how the parameters of `ShellExecute()` are defined and used to execute the `GIZA++.exe`.

```
ShellExecute(NULL,"open","GIZA++.exe",FilePath,NULL,SW_HIDE);
```

The function will run the `GIZA++.exe` from the location defined in `FilePath` without showing the interface of the program (as disable with `SW_HIDE`). Unfortunately, there is a problem that it is unable to keep track of the execution status of `GIZA++.exe`, whether the program has finished the process or not. For example, when we use the statements provided in Table 5 to read the word alignment results from the "`chinese.txt`" generated by `GIZA++.exe`, there is no way to tell whether the alignment results has finished writing or not, before the reading process starts. However, this can be solved by using a single thread. Table 6 presents an alternative way to define the parameter values for `ShellExecute()` to monitor the execution status of the invoked program.

2.3 Word Aligner

Bilingual word alignment, as a first step, is also a necessary step in various NLP applications. This is

```
ShellExecute(NULL,"open","GIZA++.exe",
"chinese.txt",NULL,SW_HIDE);
Read each line in "chinese.txt"
{
     display the alignment result;
}
```

Table 5. Pseudo code of reading alignment results

```
SHELLEXECUTEINFO sei;
memset(&sei,0,sizeof(SHELLEXECUTEIN
FO));
sei.cbSize=sizeof(SHELLEXECUTEINFO;
sei.fMask = SEE_MASK_NOCLOSEPROCESS;
sei.lpVerb = "open"; //open the files
sei.lpFile = "GIZA++.exe";//call GIZA++
sei.nShow = SW_HIDE;
ShellExecuteEx(&sei); //call executable file
WaitForSingleObject(sei.hProcess,IN
FINITE); //wait until the execute file stops
CloseHandle(sei.hProcess); //close thread
```

Table 6. Alternative way to monitor program's status

especially true for the creation of parallel corpus. In order to incorporate word alignment module into our annotation system, the aligner is proposed to deal with two types of input source: pair of sentences and parallel texts. To get the alignment for parallel texts, this can be easily done with GIZA++ as it already supports. For case of taking a pair of sentences as input, the aligner is set to make use of a basic data as an extra resource in addition to the sentences pair to be aligned. The idea behind this trick is to provide GIZA++ a training (parallel) text with sufficient content to produce good alignments of words for bilingual sentences. Considering the alignment precision and training speed, 5,000 parallel sentences are used as the basic data for our experiment. During alignment, input bilingual sentences are appended to the basic data. After training with GIZA++, we can obtain the alignment results for the sentences. The basic data is then restored by removing the sentences. Figure 4 shows the program interface and an example of alignment results for a single pair of sentences.

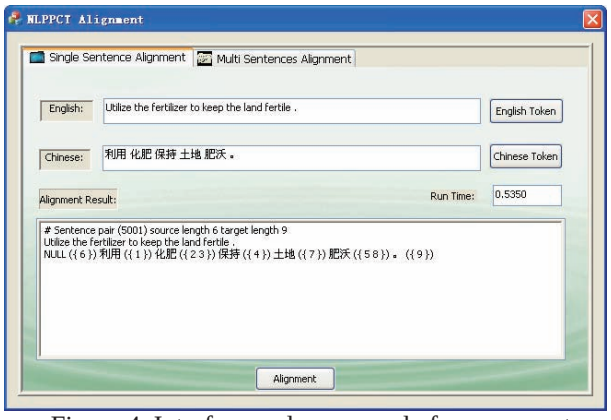

Figure 4. Interface and an example from our system

3 Conclusion and Future Work

The use of word alignment in natural language processing has increased dramatically in recent years, especially in the development of statistical machine translation. In this paper, we adapted the word alignment model GIZA++ to the development of tree annotation system for Windows environment. The first contribution of this work is to find a way in making GIZA++ run under the Windows. While the second one the realization of word alignment for a single pair of sentences instead of large batch of parallel. In the future, we will try to redevelop the aligner and give up using the GIZA++ program due to the fact that the alignment results are not satisfactory to us and, secondly, the current design using GIZA++ is not optimal.

Acknowledgments

This work is partially supported by the Research Committee of University of Macau under grant UL019/09-Y2/EEE/LYP01/FST and supported by Science and Technology Development Fund of Macau under grant 057/2009/A2.

References

Brown Peter, Della Pietra S., Della Pietra V., Mercer R. 1993. *The Mathematics of Statistical Machine Translation: Parameter Estimation*. Computational Linguistics, 19(2), pp. 263–311.

Esther, Galbrun. 2009. *Phrase table pruning for Statistical Machine Translation.* Series of Publications C. Report C-2009-22.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. *Statistical phrase-based translation*. In Proceedings of the North American Chapter of the Association for Computational Linguistics.

Liu, Yuan, Qiang Tan, and Kun Xu Shen. 1994. The *Word Segmentation Rules and Automatic Word Segmentation Methods for Chinese Information Processing* (in Chinese). Qing Hua University Press and Guang Xi Science and Technology Press.

Melamed, I. Dan. 2000. *Models of translational equivalence among words.* Computational Linguistics, 26 (2): 221–249.

Och, Franz J. 2000. Giza++: *Training of statistical translation models.*

Och, Franz J., Hermann Ney. 2000. *A Comparison of Alignment Models for Statistical Machine Translation*. In: Proc. of the 18th Int. Conf. on Computational Linguistics. Saarbrucken, Germany, pp. 1086–1090.

Och, Franz J., Hermann Ney. 2003. *A Systematic Comparison of Various Statistical Alignment Models*, Computational Linguistics, vol.,29(1)1, pp. 19-51.

Smadja, Frank, Kathleen R.McKeown, and Vasileios Hatzivassiloglou. 1996. *Translating collocations for bilingual lexicons: A statistical approach*. Computational Linguistics, 22(1):1–38.

Yarowsky, David and Richard Wicentowski. 2000. *Minimally supervised morphological analysis by multimodal alignment.* In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL), pages 207–216, Hong Kong.