Fast-Champollion: A Fast and Robust Sentence Alignment Algorithm

Peng Li and **Maosong Sun**
Department of Computer Science and Technology
State Key Lab on Intelligent Technology and Systems
National Lab for Information Science and Technology
pengli09@gmail.com, sms@tsinghua.edu.cn

Ping Xue
The Boeing Company
ping.xue@boeing.com

Abstract

Sentence-level aligned parallel texts are important resources for a number of natural language processing (NLP) tasks and applications such as statistical machine translation and cross-language information retrieval. With the rapid growth of online parallel texts, efficient and robust sentence alignment algorithms become increasingly important. In this paper, we propose a fast and robust sentence alignment algorithm, i.e., Fast-Champollion, which employs a combination of both length-based and lexicon-based algorithm. By optimizing the process of splitting the input bilingual texts into small fragments for alignment, Fast-Champollion, as our extensive experiments show, is 4.0 to 5.1 times as fast as the current baseline methods such as Champollion (Ma, 2006) on short texts and achieves about 39.4 times as fast on long texts, and Fast-Champollion is as robust as Champollion.

1 Introduction

Sentence level aligned parallel corpora are very important resources for NLP tasks including machine translation, cross-language information retrieval and so on. These tasks typically require support by large aligned corpora. In general, the more aligned text we have, the better result we achieve. Although there is a huge amount of bilingual text on the Internet, most of them are either only aligned at article level or even not aligned at all. Sentence alignment is a process mapping sentences in the source text to their corresponding units in the translated text. Manual sentence alignment operation is both expensive and time-consuming, and thus automated sentence alignment techniques are necessary. A sentence alignment algorithm for practical use should be (1) fast enough to process large corpora, (2) robust enough to tackle noise commonly present in the real data, and (3) effective enough to make as few mistakes as possible.

Various sentence alignment algorithms have been proposed, which generally fall into three types: length-based, lexicon-based, and the hybrid of the above two types. Length-based algorithms align sentences according to their length (measured by character or word). The first length-based algorithm was proposed in (Brown et al., 1991). This algorithm is fast and has a good performance if there is minimal noise (e.g., sentence or paragraph omission) in the input bilingual texts. As this algorithm does not use any lexical information, it is not robust. Lexicon-based algorithms are usually more robust than the length-based algorithm, because they use the lexical information from source and translation lexicons instead of solely sentence length to determine the translation relationship between sentences in the source text and the target text. However, lexicon-based algorithms are slower than length-based sentence alignment algorithms, because they require much more expensive computation. Typical lexicon-based algorithms include (Ma, 2006; Chen, 1993; Utsuro et al., 1994; Melamed, 1996). Sentence length and lexical information are also combined to achieve more efficient algorithms in two ways. One way is to use both sentence length and lex-

Coling 2008: Poster Volume, pages 710–718,
Beijing, August 2010

ical information together to determine whether two sentences should be directly aligned or not (Simard et al., 1993; Wu, 1994). The other way is to produce a rough alignment based on sentence length (and possibly some lexical information at the same time), and then build more precise alignment by using more effective lexicon-based algorithms (Moore, 2002; Varga et al., 2005). But both of the two ways suffer from high computational cost and are not fast enough for processing large corpora.

Lexical information is necessary for improving robustness of a sentence alignment algorithm, but use of lexical information will introduce higher computational cost and cause a lower speed. A common fact is that the shorter the text is, the less combination possibilities it would introduce and the less computational cost it would need. So if we can first split the input bilingual texts into small aligned fragments reliably with a reasonable amount of computational cost, and then further align these fragments one by one, we can speed up these algorithms remarkably. This is the main idea of our algorithm Fast-Champollion.

The rest of this paper is organized as follows: Section 2 presents formal definitions of sentence alignment problem, and briefly reviews the length-based sentence alignment algorithm and Champollion algorithm; Section 3 proposes the Fast-Champollion algorithm. Section 4 shows the experiment results; and Section 5 is the conclusion.

2 Definitions and Related Work

2.1 Definitions and Key Points

A **segment** is one or more consecutive sentence(s). A **fragment** consists of one segment of the source text (denoted by S) and one segment of the target text (denoted by T), and a fragment can be further divided into one or more beads. A **bead** represents a group of one or more sentences in the source text and the corresponding sentence(s) in the target text, denoted by $A_i = (S_{Ai}; T_{Ai}) = (S_{a_{i-1}+1}, S_{a_{i-1}+2}, \cdots, S_{a_i}; T_{b_{i-1}+1}, T_{b_{i-1}+2}, \cdots, T_{b_i})$, where S_i and T_j are the i$^{\text{th}}$ and j$^{\text{th}}$ sentence of S and T respectively.

In practice, we rarely encounter crossing alignment, e.g., sentences S_i and S_j of the source language are aligned to the sentences T_j and T_i of the target language respectively. But much more effort has to be taken for an algorithm to process crossing alignment well. So we do not consider crossing alignment here.

In addition, only a few type of beads are frequently observed in the real world. As it can save significantly in terms of computational cost and it would not do significant harm to algorithm without considering rare bead types, a common practice for designing sentence alignment algorithms is to only consider the frequently observed types of beads. Following this practice, we only consider beads of 1-to-0, 0-to-1, 1-to-1, 1-to-2, 2-to-1, 1-to-3, 3-to-1, 1-to-4, 4-to-1 and 2-to-2 types in our algorithm, where n-to-m means the bead consists of n sentence(s) of the source language and m sentence(s) of the target language.

2.2 Length-based Sentence Alignment Algorithm

Length-based sentence alignment algorithm was first proposed in (Brown et al., 1991). This algorithm captures the idea that long or short sentences tend to be translated into long or short sentences. A probability is produced for each bead based on the sentence length, and a dynamic programming algorithm is used to search for the alignment with the highest probability, which is treated as the best alignment.

This algorithm is fast and can produce good alignment when the input bilingual texts do not contain too much noise, but it is not robust, because it only uses the sentence length information. When there is too much noise in the input bilingual texts, sentence length information will be no longer reliable.

2.3 Champollion Aligner

Champollion aligner was proposed in (Ma, 2006). It borrows the idea of *tf-idf* value, which is widely used in information retrieval, to weight term[1] pair similarity. Greater weight is assigned to the less frequent translation term pairs, because these term

[1]Here terms are not limited to linguistic words, but also can be tokens like "QX6800"

pairs have much stronger evidence for two segments to be aligned. For any two segments, a similarity is assigned based on the term pair weight, sentence number and sentence length. And the dynamic programming algorithm is used to search for the alignment with the greatest total similarity. This alignment is treated as the best alignment.

Champollion aligner can produce good alignment even on noisy input as reported in (Ma, 2006). Its simplicity and robustness make it a good candidate for practical use. But this aligner is slow. Because its time complexity is $O(n^2)$ and it has to look up the dictionary multiple times in each step of the dynamic programming algorithm, which needs higher computational cost.

3 Fast-Champollion Algorithm

In this section we propose a new sentence alignment algorithm: Fast-Champollion. Its basis is splitting the input bilingual texts into small aligned fragments and then further aligning them one by one to reduce running time while maintaining Champollion-equivalent (or better) alignment quality; it takes the advantages of both length-based and lexicon-based algorithms to the maximum extent. The outline of the algorithm is that first the *length-based splitting module* is used to split the input bilingual texts into aligned fragments, and then the components of each of these fragments will be identified and aligned by a Champollion-based algorithm. The details are described in the following sections.

3.1 Length-based Splitting Module

Although length-based sentence alignment algorithm is not robust enough, it can produce rough alignment very fast with a certain number of reliably translated beads. Length-based splitting module is designed to select these reliably translated beads to be used for delimiting and splitting the input bilingual texts into fragments. These beads will be referred to as *anchor beads* in the remaining sections.

There are four steps in this module as described below in detail.

Step 1: decide whether to skip step 2-4 or not

When there is too much noise in the input bilingual texts, the percentage of reliably translated beads in the alignment produced by the length-based algorithm will be very low. In this case, we will skip step 2 through 4.

An evidence for such a situation is that the difference between the sentence numbers of the source and target language is too big. Suppose N_S and N_T are the number of sentences of the source and target language respectively. We specify $r = |N_S - N_T| / \min\{N_S, N_T\}$ as a measure of the difference, where min means minimum. If r is bigger than a threshold, we say the difference is too big. In our experiments, the threshold is set as 0.4 empirically.

Step 2: align the input texts using length-based algorithm

In this step, length-based sentence alignment algorithm is used to align the input bilingual texts. Brown, et al. (1991) models the process of sentence alignment as two steps. First, a bead is generated according to a fixed probability distribution over bead types, and then sentence length in the bead is generated according to this model: for the 0-to-1 and 1-to-0 type of beads, it is assumed that the sentence lengths are distributed according to a probability distribution estimated from the data. For other type of beads, the lengths of sentences of the source language are generated independently from the probability distribution for the 0-to-1 and 1-to-0 type of beads, and the total length of sentences of the target language is generated according to a probability distribution conditioned on the total length of sentences of the source language. For a bead $A_i = (S_{Ai}; T_{Ai})$, $l_{S_{Ai}}$ and $l_{T_{Ai}}$ are the total lengths of sentences in S_{Ai} and T_{Ai} respectively, which are measured by word[2]. Brown, et al. (1991) assumed this conditioned probability distribution is

$$Prob(l_{T_{Ai}}|l_{S_{Ai}}) = \alpha \exp\left(-\frac{(\lambda_i - \mu)^2}{2\sigma^2}\right),$$

where $\lambda_i = \log(l_{T_Ai}/l_{S_Ai})$ and α is a normalization factor. Moore (2002) assumed the condi-

[2]For Chinese, word segmentation should be done first to identify words.

tioned probability distribution is

$$Prob(l_{T_{Ai}}|l_{S_{Ai}}) = \frac{\exp(-l_{S_{Ai}}r)(l_{S_{Ai}}r)^{l_{T_{Ai}}}}{l_{T_{Ai}}!},$$

where r is the ratio of the mean length of sentences of the target language to the mean length of sentences of the source language. We tested the two models on our development corpus and the result shows that the first model performs better, so we choose the first one.

Step 3: determine the anchor beads

In this step, the reliably translated beads in the alignment produced by the length-based algorithm in Step 2 will be selected as anchor beads.

The length-based algorithm can generate a probability for each bead it produces. So a trivial way is to choose the beads with a probability above certain threshold as anchor beads. But as pointed out before, when there is too much noise, the alignment produced by the length-based algorithm is no longer reliable, and so is it with the probability. A fact is that if we select a non-translated bead as an anchor bead, we will split the input bilingual texts into wrong fragments and may cause many errors. So we have to make decision conservatively in this step and we decide to use lexical information instead of the probability to determine the anchor beads.

For a bead $A_i = (S_{Ai}; T_{Ai})$, the proportion of translation term-pairs is a good measure for determine whether this bead is reliably translated or not. In addition, use of *local information* will also be greatly helpful. To explain the use of "local information", let's define the fingerprint of a sentence first. Suppose we have a sequence of sentences $S_1, S_2, \cdots, S_m$, and $W(S_i)$is the set of distinct words in S_i, then the fingerprint of S_i is

$$f(S_i) = W(S_i) - W(S_{i-1}) - W(S_{i+1}),$$

and specially

$$f(S_1) = W(S_1) - W(S_2),$$

$$f(S_m) = W(S_m) - W(S_{m-1}).$$

The fingerprints of S_{Ai} and T_{Ai}, denoted by $f(S_{Ai})$ and $f(T_{Ai})$, are the unions of all the fingerprints of sentences in S_{Ai} and T_{Ai} respectively. As you can see, the fingerprint of a sentence is the set of words in the sentence that do not appear in the adjacent sentence(s), and thus can distinguish this sentence from its neighbors. So fingerprint is also a good measure. By combining these two measures together, we can select out more reliably translated beads.

For a word w, we use $d_D(w)$ to denote the set of all its translations in a bilingual dictionary D, and use $t_D(w)$ to denote the union of $\{w\}$ and $d_D(w)$, i.e., $t_D(w) = \{w\} \cup d_D(w)$. Given two sets of words A and B. We say a word w of A is translated by B if either one of its translations in the dictionary D or the word itself appears in B, i.e., $t_D(w) \cap B \neq \emptyset$. The set of all the words of A that are translated by B is:

$$h_D(A, B) = \{w \in A \; and \; t_D(w) \cap B \neq \emptyset\}.$$

Then the proportion of terms in A that are translated by B is

$$r_D(A, B) = \frac{|h_D(A, B)|}{|A|}.$$

We specify the proportion of translation term pairs in a bead, denoted as $ar_D(A_i)$, to be $\min\{r_D(W(S_{Ai}), W(T_{Ai})), r_D(W(T_{Ai}), W(S_{Ai}))\}$, where $W(S_{Ai})$ and $W(T_{Ai})$ are the sets of distinct words in S_{Ai} and T_{Ai} respectively. Also we specify the proportion of translation term-pairs in the fingerprint, denoted as $fr_D(A_i)$, to be $\min\{r_D(f(S_{Ai}), f(T_{Ai})), r_D(f(T_{Ai}), f(S_{Ai}))\}$. Given thresholds TH_{ar} and TH_{fr}, a bead is selected as an anchor bead when $ar_D(A_i)$ and $fr_D(A_i)$ are not smaller than TH_{ar} and TH_{fr} respectively. We will show that Fast-Champollion algorithm is not sensitive to TH_{ar} and TH_{fr} to some extent in Section 4.2.

Step 4: split the input bilingual texts

The anchor beads determined in Step 3 are used to split the input texts into fragments. The ending location of each anchor bead is regarded as a splitting point, resulting in two fragments.

3.2 Aligning Fragments with Champollion Aligner

The similarity function used by Champollion aligner is defined as follows. Given two (source

and target language) groups of sentences in a fragment, denoted by G_S=S_1, S_2,$\cdots$,S_m and G_T=T_1, T_2,$\cdots$,T_n, suppose there are k pairs of translated terms in G_S and G_T denoted by (ws_1, wt_1),(ws_2, wt_2),$\cdots$,(ws_k, wt_k), where ws_i is in G_S and wt_i is in G_T. For each pair of the translated terms (ws_i, wt_i), define $idtf(ws_i)$ to be

$$\frac{Total\ \#\ of\ terms\ in\ the\ whole\ document}{\#\ occurrences\ of\ ws_i\ in\ G_S},$$

and define

$$stf(ws_i, wt_i) = \min\{stf(ws_i), stf(wt_i)\},$$

where $stf(ws_i)$ and $stf(wt_i)$ are the frequency of ws_i and wt_i in G_S and G_T respectively. The similarity between G_S and G_T is defined as

$$\sum_{i=1}^{k} \log\left(idtf(ws_i) \times stf(ws_i, wt_i)\right) \times alignment\_penalty \times length\_penalty,$$

where $alignment\_penalty$ is 1 for 1-to-1 alignment type of beads and a number between 0 and 1 for other type of beads, $length\_penalty$ is a function of the total sentence lengths of G_S and G_T.

The reason for choosing Champollion aligner instead of other algorithms will be given in Section 4.2. And another question is how $idtf$ values should be calculated. $idtf$ is used to estimate how widely a term is used. An intuition is that $idtf$ will work better if the texts are longer, because if the texts are short, most words will have a low frequency and will seem to only appear locally. In Fast-Champollion, we calculate $idtf$ according to the whole document instead of each fragment. In this way, a better performance is achieved.

3.3 Parameter Estimation

A development corpus is used to estimate the parameters needed by Fast-Champollion.

For the length-based algorithm, there are five parameters that need to be estimated. The first one is the probability distribution over bead types. The ratio of different types of beads in the development corpus is used as the basis for the estimation. The second and third parameters are the probability distributions over the sentence length of the source language and the target language. These distributions are estimated as the distributions observed from the input bilingual texts. That is to say, these two distributions will not be the same for different bilingual input texts. The forth and fifth are μ and σ. They are estimated as the mean and variance of λ_i over the development corpus.

For Champollion aligner, $alignment\_penalty$ and $length\_penalty$ need to be determined. Because the Perl version of Champollion aligner[3] is well developed, we borrow the two definitions from it directly.

4 Experiments

4.1 Datasets and Evaluation Metrics

We have two English-Chinese parallel corpora, one for the development purpose and one for the testing purpose. Both of the two corpora are collected from the Internet and are manually aligned.

The development corpus has 2,004 beads. Given the space constraint, detailed information about the development corpus is omitted here.

The testing corpus contains 26 English-Chinese bilingual articles collected from the Internet, including news reports, novels, science articles, television documentary subtitles and the record of government meetings. There are 9,130 English sentences and 9,052 Chinese sentences in these articles[4]. The number of different type of beads in the golden standard answer is shown in Table 1.

Type	Number	Percentage(%)
1:1	7275	83.19
1:2 2:1	846	9.67
1:3 3:1	77	0.88
1:4 4:1	16	0.18
2:2	32	0.37
1:0 0:1	482	5.51
others	17	0.19
total	8745	100.00

Table 1: Types of beads in the golden standard

Both the Fast-Champollion algorithm and the Champollion aligner need a bilingual dictionary and we supply the same bidirectional dictionary to

[3] http://champollion.sourceforge.net

[4] The definition of "sentence" is slightly different from the common sense here. We also treat semicolon and colon as the end of a sentence.

them in the following evaluations. This dictionary contains 45,439 pair of English-Chinese translation terms.

We use four commonly used measures for evaluating the performance of a sentence alignment algorithm, which are the *running time*,

$$\textit{Precision} = \frac{|GB \cap PB|}{|PB|},$$

$$\textit{Recall} = \frac{|GB \cap PB|}{|GB|},$$

and

$$\textit{F1-measure} = \frac{2 \times Precision \times Recall}{Precision + Recall},$$

where GB is the set of beads in the golden standard, and PB is the set of beads produced by the algorithm.

All the following experiments are taken on a PC with an Intel QX6800 CPU and 8GB memory.

4.2 Algorithm Design Issues

Why Choose Champollion?

We compared Champollion aligner with two other sentence alignment algorithms which also make use of lexical information. And the result is shown in Table 2. "Moore-1-to-1" and "Moore-all" are corresponding to the algorithm proposed in (Moore, 2002). The difference between them is how Recall is calculated. Moore's algorithm can only output 1-to-1 type of beads. For "Moore-1-to-1", we only consider beads of 1-to-1 type in the golden standard when calculating Recall, but all types of beads are considered for "Moore-all". The result suggests that ignoring the beads that are not of 1-to-1 type does have much negative effect on the overall performance of Moore's algorithm. Our goal is to design a general purpose sentence alignment algorithm that can process frequently observed types of beads. So Moore's algorithm is not a good choice. Hunalign refers to the hunalign algorithm proposed in (Varga et al., 2005). The resources provided to Champollion aligner and hunalign algorithm are the same in the test, but hunalign algorithm's performance is much lower. So hunalign algorithm is not a good choice either. Champollion algorithm is simple and has a high overall performance. So it is a better choice for us.

Aligner	Precision	Recall	F1-measure
Champollion	0.9456	0.9546	0.9501
Moore-1-to-1	0.9529	0.9436	0.9482
Moore-all	0.9529	0.7680	0.8505
Hunalign	0.8813	0.9037	0.8923

Table 2: The performance of different aligners on the development corpus

The Effect of TH_{ar} and TH_{fr}

TH_{ar} and TH_{fr} are two thresholds for selecting anchor beads in Step 3 of length-based splitting module. In order to investigate the effect of these two thresholds on the performance of Fast-Champollion, we run Fast-Champollion on the development corpus with different TH_{ar} and TH_{fr}. Both TH_{ar} and TH_{fr} vary from 0 to 1 with step 0.05. And the running time and F1-measure are shown in Figure 1 and Figure 2 respectively.

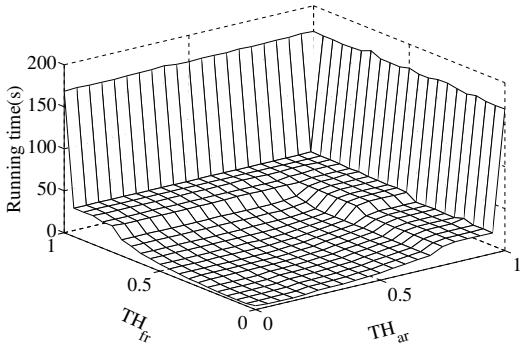

Figure 1: The running time corresponding to different TH_{ar} and TH_{fr}

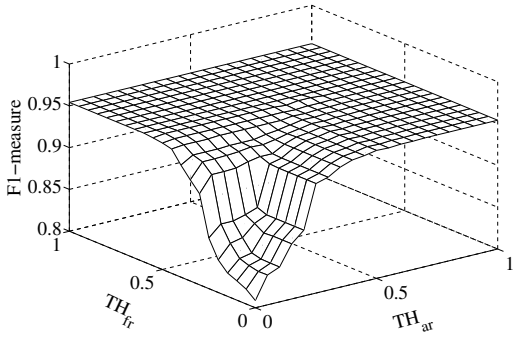

Figure 2: The F1-measure corresponding to different TH_{ar} and TH_{fr}

From Figure 1 and Figure 2, we see that for a large range of the possible values of TH_{ar} and TH_{fr}, the running time of Fast-Champollion increases slowly while F1-measure are nearly the same. In other words, Fast-Champollion are not sensitive to TH_{ar} and TH_{fr} to some extent. So making choice for the exact values of TH_{ar} and TH_{fr} becomes simple. And we use 0.5 for both of them in the following experiments.

4.3 Performance of Fast-Champollion

We use three baselines in the following evaluations. One is an implementation of the length-based algorithm in Java, one is a re-implemented Champollion aligner in Java according to the Perl version, and the last one is Fast-Champollion-Recal. Fast-Champollion-Recal is the same as Fast-Champollion except that it calculates $idtf$ values according to the fragments themselves independently instead of the whole document, and the Java versions of the length-based algorithm and Champollion aligner are used for evaluation.

Performance on Texts from the Internet

Table 3 shows the performance of Fast-Champollion and the baselines on the testing corpus. The result shows that Fast-Champollion achieves slightly better performance than Fast-Champollion-Recal. The running time of Champollion is about 2.6 times longer than Fast-Champollion with lower Precision, Recall and F1-measure. It should be pointed out that Fast-Champollion achieves better Precision, Recall and F1-measure than Champollion does because the splitting process may split the regions hard to align into different fragments and reduces the chance for making mistakes. Because of the noise in the corpus, the F1-measure of the length-based algorithm is low. This result suggests that Fast-Champollion is fast, robust and effective enough for aligning texts from the Internet.

Robustness of Fast-Champollion

In order to make a more precise investigation on the robustness of Fast-Champollion against noise, we made the following evaluation. First we manually removed all the 1-to-0 and 0-to-1 type of beads from the testing corpus to produce a clean corpus. This corpus contains 8,263 beads. Then we added 8263×n% 1-to-0 or 0-to-1 type of beads to this corpus at arbitrary positions to produce a series of noisy corpora, with n having the values of 5,10,...,100. Finally we ran Fast-Champollion algorithm and the baselines on these corpora respectively and the results are shown in Figure 3 and Figure 4, which indicate that for Fast-Champollion, when n increases 1, Precision drops 0.0021, Recall drops 0.0038 and F1-measure drops 0.0030 on average, which are very similar to those of Champollion, but Fast-Champollion is 4.0 to 5.1 times as fast as Champollion. This evaluation proves that Fast-Champollion is robust against noise and is a more reasonable choice for practical use.

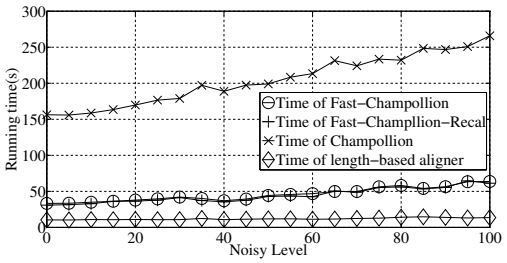

Figure 3: Running Time of Fast-Champollion, Fast-Champollion-Recal, Champollion and the length-based algorithm

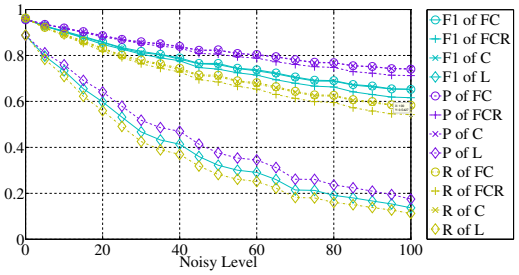

Figure 4: Precision (P), Recall (R) and F1-measure (F1) of Fast-Champollion (FC), Fast-Champollion-Recall (FCR), Champollion (C) and the length-base algorithm (L)

Performance on Long Texts

In order to test the scalability of Fast-Champollion algorithm, we evaluated it on long texts. We merged all the articles in the testing cor-

Aligner	Precision	Recall	F1-measure	Running time(s)
Fast-Champollion	0.9458	0.9408	0.9433	48.0
Fast-Champollion-Recal	0.9470	0.9373	0.9421	45.4
Champollion	0.9450	0.9385	0.9417	173.5
Length-based	0.8154	0.7878	0.8013	11.3

Table 3: Performance on texts from the Internet

Aligner	Precision	Recall	F1-measure	Running time(s)
Fast-Champollion	0.9457	0.9418	0.9437	51.5
Fast-Champollion-Recall	0.9456	0.9362	0.9409	50.7
Champollion	0.9464	0.9412	0.9438	2029.0
Length-based	0.8031	0.7729	0.7877	23.8

Table 4: Performance on long text

pus into a single long "article". Its length is comparable to that of the novel of Wuthering Heights. Table 4 shows the evaluation results on this long article. Fast-Champollion is about 39.4 times as fast as Champollion with slightly lower Precision, Recall and F1-measure, and is just about 1.2 times slower than the length-based algorithm, which has much lower Precision, Recall and F1-measure. So Fast-Champollion is also applicable for long text, and has a significantly higher speed.

4.4 Evaluation of the Length-based Splitting Module

The reason for Fast-Champollion can achieve relatively high speed is that the length-based splitting module can split the bilingual input texts into many small fragments reliably. We investigate the fragments produced by the length-based splitting module when aligning the long article used in Section 4.3. The length-based splitting module splits the long article at 1,993 places, and 1,972 segments are correct. The numbers of Chinese and English segments with no more than 30 Chinese and English sentences are shown in Figure 5. As there are only 27 and 29 segments with more than 30 sentences for Chinese and English respectively, we omit them in the figure. We can conclude that although the length-based splitting module is simple, it is efficient and reliable.

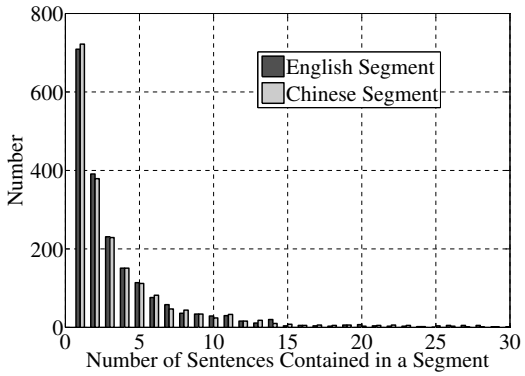

Figure 5: Numbers of Chinese/English segments with no more than 30 Chinese/English sentences

5 Conclusion and Future Work

In this paper we propose a new sentence alignment algorithm Fast-Champollion. It reduces the running time by first splitting the bilingual input texts into small aligned fragments and then further aligning them one by one. The evaluations show that Fast-Champollion is fast, robust and effective enough for practical use, especially for aligning large amount of bilingual texts or long bilingual texts.

Fast-Champollion needs a dictionary for aligning sentences, and shares the same problem of Champollion aligner as indicated in (Ma, 2006), that is the precision and recall will drop as the size of the dictionary decreases. So how to build bilingual dictionaries automatically is an important task for improving the performance of Fast-Champollion in practice, and is a critical problem for applying Fast-Champollion on language pairs without a ready to use dictionary.

Acknowledgement

This research is supported by the Boeing-Tsinghua Joint Research Project "Robust Chinese Word Segmentation and High Performance English-Chinese Bilingual Text Alignment".

References

Brown, Peter F., Jennifer C. Lai, and Robert L. Mercer. 1991. Aligning sentences in parallel corpora. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 169–176, Berkeley, California, USA, June. Association for Computational Linguistics.

Chen, Stanley F. 1993. Aligning sentences in bilingual corpora using lexical information. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 9–16, Columbus, Ohio, USA, June. Association for Computational Linguistics.

Ma, Xiaoyi. 2006. Champollion: A robust parallel text sentence aligner. In *Proceedings of LREC-2006: Fifth International Conference on Language Resources and Evaluation*, pages 489–492.

Melamed, I. Dan. 1996. A geometric approach to mapping bitext correspondence. In *Proceedings of the First Conference on Empirical Methods in Natural Language Processing*, pages 1–12.

Moore, Robert C. 2002. Fast and accurate sentence alignment of bilingual corpora. In *Proceedings of the 5th Conference of the Association for Machine Translation in the Americas on Machine Translation: From Research to Real Users*, pages 135–144, London, UK. Springer-Verlag.

Simard, Michel, George F. Foster, and Pierre Isabelle. 1993. Using cognates to align sentences in bilingual corpora. In *Proceedings of the 1993 Conference of the Centre for Advanced Studies on Collaborative Research*, pages 1071–1082. IBM Press.

Utsuro, Takehito, Hiroshi Ikeda, Masaya Yamane, Yuji Matsumoto, and Makoto Nagao. 1994. Bilingual text matching using bilingual dictionary and statistics. In *Proceedings of the 15th Conference on Computational Linguistics*, pages 1076–1082, Morristown, NJ, USA. Association for Computational Linguistics.

Varga, D., L. Nmeth, P. Halcsy, A. Kornai, V. Trn, and Nagy V. 2005. Parallel corpora for medium density languages. In *Proceedings of the RANLP 2005*, pages 590–596.

Wu, Dekai. 1994. Aligning a parallel english-chinese corpus statistically with lexical criteria. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 80–87, Las Cruces, New Mexico, USA, June. Association for Computational Linguistics.