A fast and accurate method for detecting English-Japanese parallel texts

Ken'ichi Fukushima, Kenjiro Taura and Takashi Chikayama
University of Tokyo
`ken@tkl.iis.u-tokyo.ac.jp`
`{tau,chikayama}@logos.ic.i.u-tokyo.ac.jp`

Abstract

Parallel corpus is a valuable resource used in various fields of multilingual natural language processing. One of the most significant problems in using parallel corpora is the lack of their availability. Researchers have investigated approaches to collecting parallel texts from the Web. A basic component of these approaches is an algorithm that judges whether a pair of texts is parallel or not. In this paper, we propose an algorithm that accelerates this task without losing accuracy by preprocessing a bilingual dictionary as well as the collection of texts. This method achieved 250,000 pairs/sec throughput on a single CPU, with the best F_1 score of 0.960 for the task of detecting 200 Japanese-English translation pairs out of $40,000$. The method is applicable to texts of any format, and not specific to HTML documents labeled with URLs. We report details of these preprocessing methods and the fast comparison algorithm. To the best of our knowledge, this is the first reported experiment of extracting Japanese–English parallel texts from a large corpora based solely on linguistic content.

1 Introduction

"Parallel text" is a pair of texts which is written in different languages and is a translation of each other. A compilation of parallel texts offered in a serviceable form is called a "parallel corpus". Parallel corpora are very valuable resources in various fields of multilingual natural language processing such as statistical machine translation (Brown et al., 1990), cross-lingual IR (Chen and Nie, 2000), and construction of dictionary (Nagao, 1996).

However, it is generally difficult to obtain parallel corpora of enough quantity and quality. There have only been a few varieties of parallel corpora. In addition, their languages have been biased toward English–French and their contents toward official documents of governmental institutions or software manuals. Therefore, it is often difficult to find a parallel corpus that meets the needs of specific researches.

To solve this problem, approaches to collect parallel texts from the Web have been proposed. In the Web space, all sorts of languages are used though English is dominating, and the content of the texts seems to be as diverse as all activities of the human-beings. Therefore, this approach has a potential to break the limitation in the use of parallel corpora.

Previous works successfully built parallel corpora of interesting sizes. Most of them utilized URL strings or HTML tags as a clue to efficiently find parallel documents (Yang and Li, 2002; Nadeau and Foster, 2004). Depending on such information specific to webpages limits the applicability of the methods. Even for webpages, many parallel texts not conforming to the presupposed styles will be left undetected. In this work, we have therefore decided to focus on a generally applicable method, which is solely based on the textual content of the documents. The main challenge then is how to make judgements fast.

Our proposed method utilizes a bilingual dictionary which, for each word in tne language, gives the list of translations in the other. The method preprocesses both the bilingual dictionary and the collection of texts to make a comparison of text pairs in a subsequent stage faster. A comparison of a text pair is carried out simply by comparing two streams of integers without any dictionary or table lookup, in time linear in the sum of the two text sizes. With this method, we achieved 250,000 pairs/sec throughput on a single Xeon CPU (2.4GHz). The best F_1 score is 0.960, for a dataset which includes 200 true pairs out of 40,000 candidate pairs. Further comments on these numbers are given in Section 4.

In addition, to the best of our knowledge, this is the first reported experiment of extracitng Japanese–English parallel texts using a method solely based on their linguistic contents.

2 Related Work

There have been several attempts to collect parallel texts from the Web. We will mention two contrasting approaches among them.

2.1 BITS

Ma and Liberman collected English–German parallel webpages (Ma and Liberman, 1999). They began with a list of websites that belong to a domain accosiated with German–speaking areas and searched for parallel webpages in these sites. For each site, they downloaded a subset of the site to investigate what language it is written in, and then, downloaded all pages if it was proved to be English–German bilingual. For each pair of English and German document, they judged whether it is a mutual translation. They made a decision in the following manner. First, they searched a bilingual dictionary for all English–German word pairs in the text pair. If a word pair is found in the dictionary, it is recognized as an evidence of translation. Finally, they divided the number of recognized pairs by the sum of the length of the two texts and regard this value as a score of translationality. When this score is greater than a given threshold, the pair is judged as a mutual translation. They succeeded in creating about 63MB parallel corpus with 10 machines through 20 days.

The number of webpages is considered to have increased far more rapidly than the performance of computers in the past seven years. Therefore, we think it is important to reduce the cost of calculation of a system.

2.2 STRAND

If we simply make a dicision for all pairs in a collection of texts, the calculation takes $\Omega(n^2)$ comparisons of text pairs where n is the number of documents in the collection. In fact, most researches utilize properties peculiar to certain parallel webpages to reduce the number of candidate pairs in advance. Resnik and Smith focused on the fact that a page pair tends to be a mutual translation when their URL strings meet a certain condition, and examined only page pairs which satisfy it (Resnik and Smith, 2003). A URL string sometimes contains a substring which indicates the language in which the page is written. For example, a webpage written in Japanese sometimes have a substring such as `j`, `jp`, `jpn`, `n`, `euc` or `sjis` in its URL. They regard a pair of pages as a candidate when their URLs match completely after removing such language-specific substrings and, only for these candidates, did they make a detailed comparison with bilingual dictionary. They were successful in collecting 2190 parallel pairs from 8294 candidates. However, this URL condition seems so strict for the purpose that they found 8294 candidate pairs from as much as 20 Tera bytes of webpages.

3 Proposed Method

3.1 Problem settings

There are several evaluation criteria for parallel text mining algorithms. They include accuracy, execution speed, and generality. We say an algorithm is general when it can be applied to texts of any format, not only to webpages with associated information specific to webpages (e.g., URLs and tags). In this paper, we focus on developing a fast and general algorithm for determining if a pair of texts is parallel.

In general, there are two complementary ways to improve the speed of parallel text mining. One is to reduce the number of "candidate pairs" to be compared. The other is to make a single comparison of two texts faster. An example of the former is Resnik and Smith's URL matching method, which is able to mine parallel texts from a very large corpora of Tera bytes. However, this approach is very specific to the Web and, even if we restrict our interest to webpages, there may be a significant number of parallel pages whose URLs do not match the prescribed pattern and therefore are filtered out. Our method is in the latter category, and is generally applicable to texts of any format. The approach depends only on the linguistic content of texts. Reducing the number of

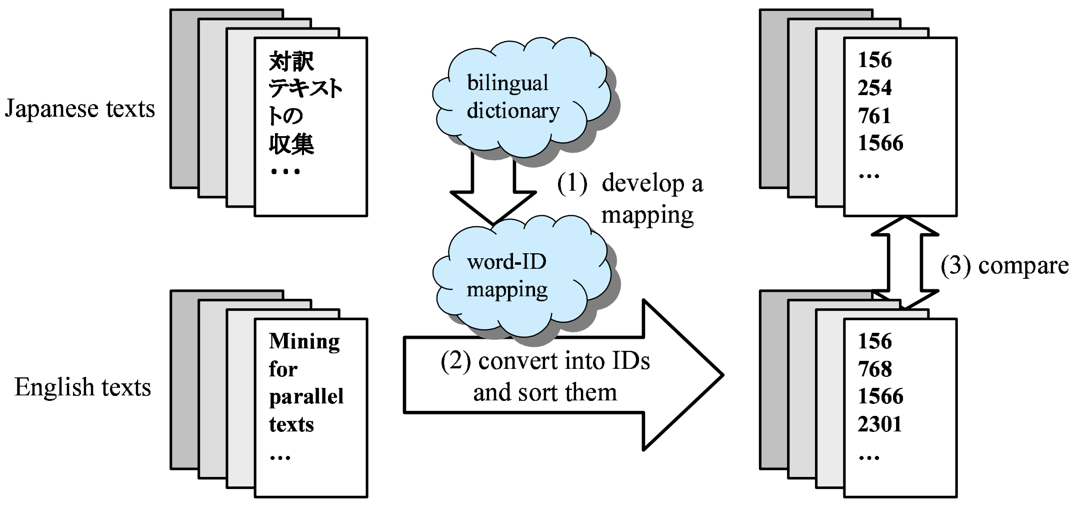


Figure 1: Outline of the method

comparisons while maintaining the generality will be one of our future works.

The outline of the method is as follows. First we preprocess a bilingual dictionary and build a mapping from words to integers, which we call "semantic ID." Texts are then preprocessed, converting each word to its corresponding semantic ID plus its position of the occurrence. Then we compare all pairs of texts, using their converted representations (Figure 1). Comparing a pair of texts is fast because it is performed in time linear in the length of the texts and does not need any table lookup or string manipulation.

3.2 Preprocessing a bilingual dictionary

We take only nouns into account in our algorithm. For the language pair of English and Japanese, a correspondence of parts of speech of a word and its translation is not so clear and may make the problem more difficult. A result was actually worse when every open-class word was considered than when only nouns were.

The first stage of the method is to assign an integer called semantic ID to every word (in both languages) that appears in a bilingual dictionary. The goal is to assign the same ID to a pair of words that are translations of each other. In an ideal situation where each word of one language corresponds one-to-one with a word of the other language, all you need to do is to assign differnt IDs to every translational relationship between two words. The main purpose of this conversion is to make a comparison of two texts in a subsequent stage faster.

However, it's not exactly that simple. A word very often has more than one words as its translation so the naive method described above is not directly applicable. We devised an approximate solution to address this complexity. We build a bigraph whose nodes are words in the dictionary and edges translational relationships between them. This graph consists of many small connected components, each representing a group of words that are expected to have similar meanings. We then make a mapping from a word to its semantic ID. Two words are considered translations of each other when they have the same semantic ID.

This method causes a side-effect of connecting two words not directly related in the dictionary. It has both good and bad effects. A good effect is that it may connect two words that do not explicitly appear as translations in the dictionary, but are used as translations in practice (see section 4.3). In other words, new translational word pairs are detected. A bad effect, on the other hand, is that it potentially connects many words that do not share meanings at all. Figure 2 shows an actual example of such an undesirable component observed in our experiment. You can go from *fruit* to *army* through several hops and these words are treated as identical entity in subsequent steps of our technique. Futhermore, in the most extreme case, a very large connected component can be created. Table 1 shows the statistics of the component sizes for the English-Japanese dictionary we have used in our experiment (EDR Electronic Dictionary).

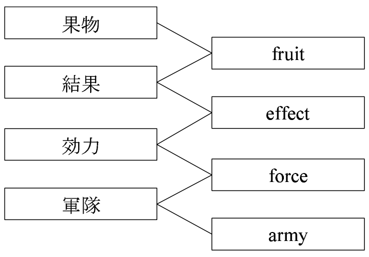


Figure 2: Example of a undesirable graph

Most components are fairly small (< 10 words). The largest connected component, however, consisted of 3563 nodes out of the total 28001 nodes in the entire graph and 3943 edges out of 19413. As we will see in the next section, this had a devastating effect on the quality of judgement so we clearly need a method that circumvents the situation. One possibility is to simply drop very large components. Another is to divide the graph into small components. We have tried both approaches.

Table 1: Statistics of the component sizes

# of nodes	# of components
2	6629
3	1498
4	463
5	212
6	125
7	69
8	44
9	32
10∼	106

For partitioning graphs, we used a very simple greedy method. Even though a more complex method may be possible that takes advantages of linguistic insights, this work uses a very simple partitioning method that only looks at the graph structure in this work. A graph is partitioned into two parts having an equal number of nodes and a partition is recursively performed until each part becomes smaller than a given threshold. The threshold is chosen so that it yields the best result for a training set and then applied to a test data. For each bisection, we begin with a random partition and improves it by a local greedy search. Given the current partition, it seeks a pair of nodes which, if swapped, maximumly reduces the number of edges crossing the two parts. Ties are broken arbitrarily when there are many such pairs. If no single swap reduces the number of edges across parts, we simply stop (i.e., local search). A semantic ID is then given to each part.

This process would lose connections between words that are originally translations in the dictionary but are separated by the partitioning. We will describe a method to partially recover this loss in the end of the next section, after describing how texts are preprocessed.

3.3 Preprocessing texts

Each text (document) is preprocessed as follows. Texts are segmented into words and tagged with a part-of-speech. Inflection problems are addressed with lemmatization. Each word is converted into the pair (*nid*, *pos*), where *nid* is the semantic ID of the partition containing the word and *pos* its position of occurrence. The position is normalized and represented as a floating point number between 0.0 and 1.0. Any word which does not appear in the dictionary is simply ignored. The position is used to judge if words having an equal ID occur in similar positions in both texts, so they suggest a translation.

After converting each word, all (*nid*, *pos*) pairs are sorted first by their semantic IDs breaking ties with positions. This sorting takes $O(n \log n)$ time for a document of n words. This preprocessing needs to be performed only once for each document.

We recover the connections between word pairs separated by the partitioning in the following manner. Suppose words J and E are translations of each other in the dictionary, J is in a partition whose semantic ID is x and E in another partition whose semantic ID is y. In this case, we translate J into two elements x and y. This result is as if two separate words, one in component x and another in y, appeared in the original text, so it may potentially have an undesirable side-effect on the quality of judgement. It is therefore important to keep the number of such pairs reasonably small. We experimented with both cases, one in which we recover separate connections and the other in which we don't.

3.4 Comparing document pairs

We judge if a text pair is likely to be a translation by comparing two sequences obtained by the preprocessing. We count the number of word pairs that have an equal semantic ID and whose positions are within a distance threshold. The best threshold is chosen to yield the best result for a training set and then applied to test set. This process takes time linear in the length of texts since the sequences are sorted. First, we set cursors at the first element of each of the two sequences. When the semantic IDs of the elements under the cursors are equal and the difference between their positions is within a threshold, we count them as an evidence of translationality and move both cursors forward. Otherwise, the cursor on the element which is less according to the sorting criteria is moved forward. In this step, we do not perform any further search to determine if original words of the elements were related directly in the bilingual dictionary giving preference to speed over accuracy. We repeat this operation until any of the cursors reaches the end of the sequence. Finally, we divide the number of matching elements by the sum of the lengths of the two documents. We define this value as "tscore," which stands for translational score. At least one cursor moves after each comparison, so this algorithm finishes in time linear in the length of the texts.

4 Experiments

4.1 Preparation

To evaluate our method, we used The EDR Electronic Dictionary[1] for a bilingual dictionary and Fry's Japanese-English parallel web corpus (Fry, 2005) for sample data. In this experiment, we considered only nouns (see section 3.2) and got a graph which consists of 28001 nodes, 19413 edges and 9178 connected components of which the largest has 3563 nodes and 3943 edges. Large components including it need to be partitioned.

We conducted partitioning with differnt thresholds and developed various word–ID mappings. For each mapping, we made several variations in two respect. One is whether cut connections are recovered or not. The other is whether and how many numerals, which can be easily utilized to boost the vocaburary of the dictionary, are added to a bilingual dictionary.

The parallel corpus we used had been collected by Fry from four news sites. Most texts in the corpus are news report on computer technology and the rest is on various fields of science. A single document is typically 1,000–6,000 bytes. He detected parallel texts based only on HTML tags and link structures, which depend on websites, without looking at textual content, so there are many false pairs in his corpus. Therefore, to evaluate our method precisely, we used only 400 true parallel pairs that are randomly selected and checked by human inspection. We divided them evenly and randomly into two parts and use one half for a training set and the other for a test set. In experiments described in section 4.4 and 4.5, we used other portion of the corpus to scale experiments.

For tokenization and pos-tagging, we used MeCab[2] to Japanese texts and SS Tagger[3] to English texts. Because SS Tagger doesn't act as lemmatizer, we used `morphstr()` function in WordNet library[4].

4.2 Effect of large components and a partitioning

Figure 3 shows the results of experiments on several conditions. There are three groups of bars; (A) treat every connected component equally regardless of its size, (B) simply drop the largest component and (C) divide large components into smaller parts. In each group, the upper bar corresponds to the case the algorithm works without a distance threshold and the lower with it (0.2). The figures attached to each bar are the $\max F_1$ score, which is a popular measure to evaluate a classification algorithm, and indicate how accurately a method is able to detect 200 true text pairs from the test set of 40,000 pairs. We didn't recover word connections broken in the partitioning step and didn't add any numerals to the vocabrary of the bilingual dictionary this time.

The significant difference between (A) and (B) clearly shows the devastating effect of large components. The difference between (B) and (C) shows that the accurary can be further improved if large components are partitioned into small ones in order to utilize as much information as possible. In addtion, the accuracy consistently improves by using the distance threshold.

Next, we determined the best word–ID mapping

[1]EDR Electronic Dictionary.
`http://www2.nict.go.jp/kk/e416/EDR/`

[2]MeCab: Yet Another Part-of-Speech and Morphological Analyzer.
`http://mecab.sourceforge.jp/`

[3]SS Tagger - a part-of-speech tagger for English.
`http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/postagger/`

[4]WordNet - a lexical database for the English language.
`http://wordnet.princeton.edu/`

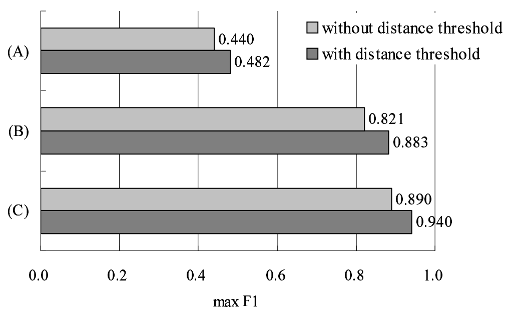


Figure 3: Effect of the graph partitioning

and distance threshold and tested its performance through a 2–fold cross validation. The best mapping among those was the one which

- divides a component recursively until the number of nodes of each language becomes no more than 30,
- does not recover connections that are cut in the partitioning, and
- adds numerals from 0 to 999.

The best distance threshold was 0.2, and tscore threshold 0.102. We tested this rule and thresholds on the test set. The result was $F_1 = 0.960$.

4.3 Effect of false translation pairs

Our method of matching words differs from Ma and Liberman's one. While they only count word pairs that directly appear in a bilingual dictionary, we identify all words having the same semantic ID. Potential merits and drawbacks to accuracy have been described in the section 3.2. We compared the accuracy of the two algorithms to investigate the effect of our approximate matching. To this end, we implemented Ma and Liberman's method with all other conditions and input data being equal to the one in the last section. We got $\max F_1 = 0.933$ as a result, which is slightly worse than the figure reported in their paper. Though it is difficult to conclude where the difference stems from, there are several factors worth pointing out. First, our experiment is done for English-Japanese, while Ma and Liberman's experiment for English-German, which are more similar than English and Japanese are. Second, their data set contains much more true pairs (240 out of 300) than our data set does (200 out of 40,000).

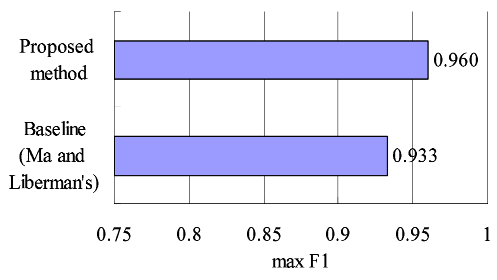


Figure 4: The two word-matching policy

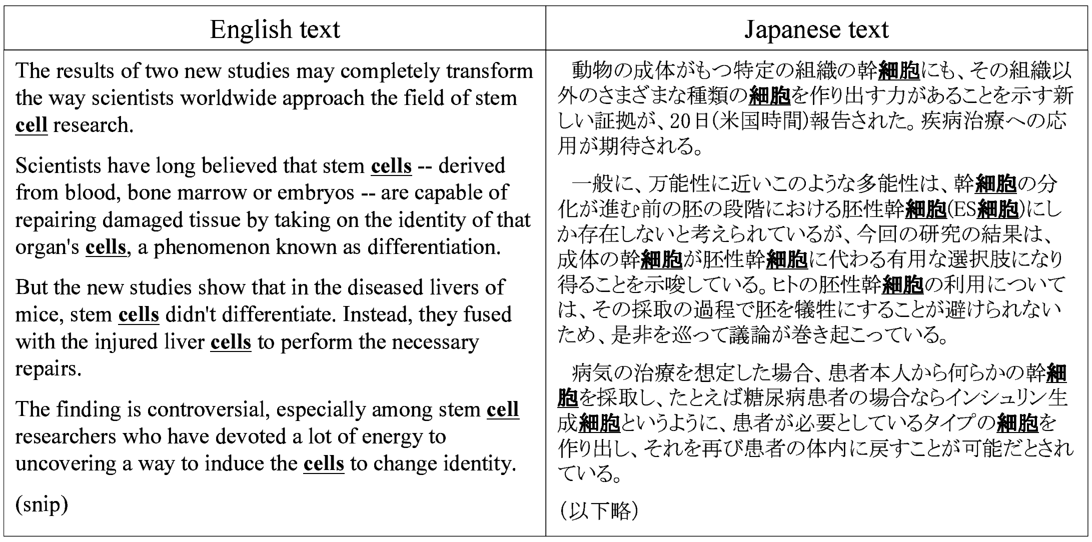
This number is also worse than that of our experiment (Figure 4). This shows that, at least in the experiment, our approach of identifying more pairs than the original dictionary causes more good effects than bad in total. We looked at word pairs which are not matched in Ma and Liberman's method but in ours. While most of the pairs can be hardly considered as a strict translation, some of them are pairs practically used as translations. Examples of such pairs are shown in Figure 5.

English word	Japanese word
issue	課題
competition	コンテスト
dicision	決意
sum	総額
benefit	利点
phone	電話器
device	発明
client	買手

Figure 5: Word pairs not in the dictionary

4.4 Execution Speed

We have argued that the execution speed is a major advantage of our method. We achieved 250,000 pairs/sec throughput on single Xeon (2.4GHz) processor. It's difficult to make a fair comparison of the execution speed because Ma and Liberman's paper does not describe enough details about their experimants other than processing 3145 websites with 10 sparc stations for 10 days. Just for a rough estimate, we introduce some bold assumptions. Say, there were a thousand pages for each language in a website or, in other words, a million page pairs, and the performance of processors has grown by 32 times in the past seven years, our method works more than 40 times faster than Ma and Liberman's one. This difference seems



English text	Japanese text
The results of two new studies may completely transform the way scientists worldwide approach the field of stem **cell** research. Scientists have long believed that stem **cells** -- derived from blood, bone marrow or embryos -- are capable of repairing damaged tissue by taking on the identity of that organ's **cells**, a phenomenon known as differentiation. But the new studies show that in the diseased livers of mice, stem **cells** didn't differentiate. Instead, they fused with the injured liver **cells** to perform the necessary repairs. The finding is controversial, especially among stem **cell** researchers who have devoted a lot of energy to uncovering a way to induce the **cells** to change identity. (snip)	動物の成体がもつ特定の組織の幹**細胞**にも、その組織以外のさまざまな種類の**細胞**を作り出す力があることを示す新しい証拠が、20日(米国時間)報告された。疾病治療への応用が期待される。 一般に、万能性に近いこのような多能性は、幹**細胞**の分化が進む前の胚の段階における胚性幹**細胞**(ES**細胞**)にしか存在しないと考えられているが、今回の研究の結果は、成体の幹**細胞**が胚性幹**細胞**に代わる有用な選択肢になり得ることを示唆している。ヒトの胚性幹**細胞**の利用については、その採取の過程で胚を犠牲にすることが避けられないため、是非を巡って議論が巻き起こっている。 病気の治療を想定した場合、患者本人から何らかの幹**細胞**を採取し、たとえば糖尿病患者の場合ならインシュリン生成**細胞**というように、患者が必要としているタイプの**細胞**を作り出し、それを再び患者の体内に戻すことが可能だとされている。 (以下略)

Figure 6: A example of false–positive text pairs

to be caused by a difference of the complexity between the two algorithms. To the extent written in their paper, Ma and Liberman calculated a score of translationality by enumerating all combinations of two words within a distance threshold and search a bilingual dictionary for each combination of words. This algorithm takes $\Omega(n^2)$ time where n is the length of a text, while our method takes $O(n)$ time. In addition, our method doesn't need any string manipulation in the comparison step.

4.5 Analysis of miss detections

We analyzed text pairs for which judgements differ between Fry's and ours.

Among pairs Fry determined as a translation, we examined the 10 pairs ranked highest in our algorithm. Two of them are in fact translations, which were not detected by Fry's method without any linguistic information. The rest eight pairs are not translations. Three of the eight pairs are about bioscience, and a word "cell" occurred many time (Figure 6). When words with an identical semantic ID appear repeatedly in two texts being compared, their distances are likely to be within a distance threshold and the pair gets unreasonably high tscore. Therefore, if we take the number of each semantic ID in a text into account, we might be able to improve the accuracy.

We performed the same examination on the 10 pairs ranked lowest among those Fry determined not to be a translation. But no interesting feature could be found at the moment.

5 Summary and Future Work

In this paper, we proposed a fast and accurate method for detecting parallel texts from a collection. This method consists of major three parts; preprocess a bilingual dictionary into word–ID conversion rule, convert texts into ID sequences, compare sequences. With this method, we achieved 250,000 pairs/sec on a single CPU and best F_1 score of 0.960. In addition, this method utilizes only linguistic information of a textual content so that it is generally applicable. This means it can detect parallel documents in any format. Furthermore, our method is independent on languages in essence. It can be applied to any pair of languages if a bilingual dictionary between the languages are available (a general language dictionary suffices.)

Our future study will include improving both accuracy and speed while retaining the generaility. For accuracy, as we described in Section 4.5, tscore tends to increase when an identical semantic ID appears many times in a text. We might be able to deal with this problem by taking into account the probability that the distance between words is within a threshold. Large connected components were partitioned by a very simple method at the present work. More involved partitioning methods may improve the accuracy of the judgement. For speed, reducing the number of comparisons is the most important issue that needs be addressed.

References

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Comput. Linguist.*, 16(2):79–85.

Jiang Chen and Jian-Yun Nie. 2000. Automatic construction of parallel english-chinese corpus for cross-language information retrieval. In *Proceedings of the sixth conference on Applied natural language processing*, pages 21–28, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

John Fry. 2005. Assembling a parallel corpus from RSS news feeds. In *Workshop on Example-Based Machine Translation, MT Summit X, Phuket, Thailand*, September.

Xiaoyi Ma and Mark Liberman. 1999. BITS: A method for bilingual text search over the web. In *Machine Translation Summit VII*, September.

David Nadeau and George Foster. 2004. Real-time identification of parallel texts from bilingual newsfeed. In *Computational Linguistic in the North-East (CLiNE 2004)*, pages 21–28.

Makoto Nagao, editor. 1996. *Natural Language Processing*. Number 15 in Iwanami Software Science. Iwanami Shoten. In Japanese.

Philip Resnik and Noah A. Smith. 2003. The web as a parallel corpus. *Comput. Linguist.*, 29(3):349–380.

C. C. Yang and K. W. Li. 2002. Mining English/ Chinese parallel documents from the World Wide Web. In *Proceedings of the International World Wide Web Conference*, Honolulu, Hawaii, May.