Word Alignment of Parallel Texts

Joshua Berney
Department of Computer Science
Swarthmore College
berney@cs.swarthmore.edu

Jason Perini
Department of Computer Science
Swarthmore College
perini@cs.swarthmore.edu

Abstract

We induced a word-aligned dictionary of English and French using parallel texts. Our texts were the Hansards corpus and a small literary text corpus. We performed phrase alignment by the use of identical words in both texts as anchor points and improved the distribution of our anchor points with lexically similar words. We then performed statistical word-alignment using ϕ statistical correlation to locate translation word pairs in the parallel corpora. Our results show that ϕ correlation works reasonably well when a large number of small parallel phrases are available.

1 Introduction

The goal of this project is to induce a translation dictionary between two similar languages using parallel corpora. The two languages we chose were English and French, as they mostly share the same character set and have significant linguistic similarities. One ready source of large blocks of parallel texts in English and French are classic literary works that have been translated. These have the advantages of being in the public domain and are long documents with consistent word usage and translation style throughout. In addition, the translation of a literary text will leave a large number of words untouched and untranslated such as characters' names, locations, etc. We will need these and any French-English cognates in our phrase alignment algorithm. The disadvantages of using literary works is that the translators, in an attempt to reproduce the style of the original texts, are less likely to produce exact translations, will use less common words, and will repeat words less often.

The literary text we used was "Swann's Way", the first volume of Marcel Proust's *Remembrance of Things Past*, which is approximately 200,000 words in French and English. We took the text from the Project Gutenberg website[1]. We also ran our system against part of the Hansards corpus[2], which is the proceedings of the Canadian Parliament and is in both English and French. This corpus was appealing because it was already split into sentence alignments and was very large (approximately 1 million words).

We started with phrase and sentence alignments using anchor points, which were words that are identical in either text. We then increased the number of anchor points we used by finding likely matches using lexical similarity. Armed with a large number of aligned phrases, we then match likely pairs using the ϕ statistic correlation method.

2 Previous Work

Dmitriy's (2005) work has a number of similarities to ours in his intentions, his system induces dictionaries for languages with few machine translation resources from parallel texts in linguistically similar languages. He aligned his text on a character-to-character basis, not word tokens, and he then per-

[1] Project Gutenberg main site: http://www.gutenberg.org. The specific Proust text can be found at: http://www.gutenberg.org/etext/2650 and http://www.gutenberg.org/etext/7178

[2] http://www.isi.edu/natural-language/download/hansard/1

Appeared in: *Proceedings of the Class of 2005 Senior Conference*, pages 42–47
Computer Science Department, Swarthmore College

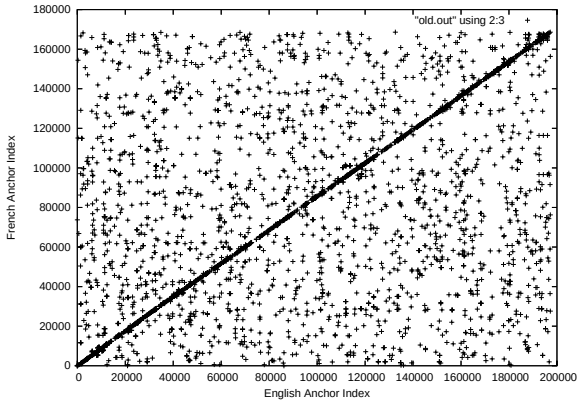

Figure 1: Output from the original anchor alignment procedure: English vs. French Anchor Position

forms a number of post-processing steps to improve the matches he receives from the GIZA++ software, which is a statistical alignment model updated by Franz Joseph Och. One of these steps is the use of lexically similar words, determined by edit distance, as 'seed' words for the alignment models.

Melamed (1999) discusses a much more complicated alignment process for bilingual parallel corpora. He uses cognates and lexically similar words with lexical similarity being determined by the Longest Common Subsequence Ratio method. The alignment is further refined using methods taken from signal noise filtering, as well as several-pass segment alignment and subsection deviations.

Gale and Church (1991) discuss using the ϕ statistic for determining word correspondences. However, their paper is preliminary and provides only vague numbers. There does not appear to be any followup work.

3 Phrase Alignment

The first step in our system is finding equivalent phrases in the source and target languages. This is done to reduce the total number of comparisons that must be made to find translations and to avoid false matches of words that are very far apart from each other in the text. Most post-segment-alignment matching algorithms increase much faster than O(n), where n is the number of words in a corpus, our phrase-alignment method can significantly reduce the time required later in the system.

Our algorithm relies on the fact that some words are exactly lexically similar in the source and target languages, typically nouns. Common examples of such words are places, names, and recently developed concepts. Using these words, we can divide the source and target texts into equivalent phrases. Before beginning the main algorithm, we standardize or eliminate most punctuation. Next, we locate the indices of words that are exactly the same in the target and source language and record their indices. We limit the minimum length of words to exclude which are exactly lexically similar, but are actually different words, such as the English 'a' and French 'a' (the English 'a' is an indefinite article whereas 'a' in French can mean the singular third-person conjugation of 'avoir', 'to have'). We also limit the number of occurrences of words in hopes of limiting the number of times one word appears very close to itself and hence creates possible confusion over the actual anchor pair matching.

Examining a plot of English vs. French position generated from the above algorithm (Figure 1), we see a relatively clear line through the origin (number of English words, number of French words) and many scattered points throughout the plot. Considering the solid line in the figure and the structure of language, we make the assumption that a linear relation exists between the location of a given English word and the French equivalent. Similarly, the location of an English word should be approximately linearly related to the location of the equivalent French word. However, we must also consider there will be places where more English words per French word occur than normal or vice versa. We are also concerned with some target language sentences being out of order with respect to source sentences.

Combining these concepts, we say a given pair anchor points determined from the above algorithm must satisfy:

$$\$source\_word\_index = c \times \$target\_word\_index + \gamma + a\beta \quad (1)$$

where $c = \frac{\$number\_source\_words}{\$number\_target\_words}$, β is some constant, a varies from -1 to 1,

$$\gamma_{new} = \alpha \times c \times \$target\_word\_index + (1-\alpha) \times \gamma_{old} \quad (2)$$

for each valid anchor pair examined in order of occurrence and α is some constant.

Appeared in: *Proceedings of the Class of 2005 Senior Conference*, pages 42–47
Computer Science Department, Swarthmore College

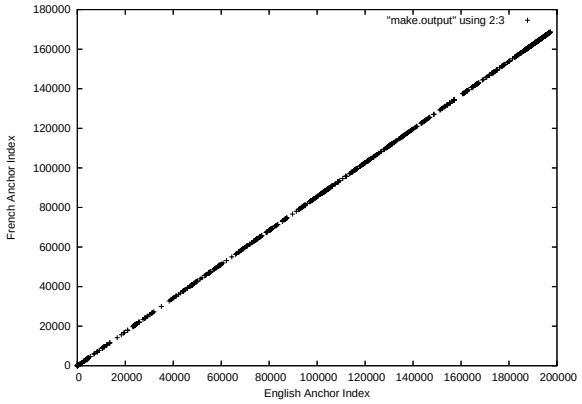

Figure 2: Output from the refined anchor alignment procedure: English vs. French Anchor Position

$source_word_index is our approximation of where the source word should appear. γ represents the current drift – that is the amount of deviation from a linear translation. β is a small constant that allows for some error in our approximation method. In our implement a does not exist, we instead test that $source_word_index without the β term fits within the interval of β. Equation 2 reflects that γ must be updated as anchor pairs as examined. We start with an initial value of γ at 0. Then we iterate through potential anchor word pairs in order of their source index. When a valid pair is found, the second equation is executed and γ_{new} is used until it is updated by finding a new valid pair. Empirically, we have found $\alpha = 0.15$ and $\beta = 40$ work well for the Proust corpus. Due to the nature of the Hansards, it has been difficult to determine optimal values.

Occasionally, a word that appears infrequently in the text will occur very close to itself. Consider the case of a character in a novel only encountered once. This can lead to the algorithm picking up several matches for the same word in a very small region of the text. When this occurs, we take only the first alignment for the word in the region. The output we receive is shown in Figure 2. We use this output for our later steps.

4 Lexical Word Alignment

For texts in similar languages, such as English and French, using lexical similarities can improve the alignment accuracy of other methods by finding words that are likely to be matches. Very good matches are added to the anchor point list along with the lexically identical words and then the anchor point list is passed onto the ϕ statistic correlation method. One simple method of finding lexically similar words is to measure their Levenshtein distances.

The Levenshtein distance between two words is the number of character alterations needed to change one word into another. Each substitution, insertion, or deletion of a character adds to the edit distance of the words. The specific implementation of the Levenshtein algorithm we used was written by Eli Bendersky (2003). It employs a (M+1) x (N+1) matrix where M and N are the lengths of the two strings. The algorithm starts with the word in the source language and calculates the cost for any move, following the least costly path until the minimum transformation cost from one of the strings to the other is found.

The way we used the Levenshtein distance measure to find potential matches followed a partial bag-of-words approach. We looked at the two phrases surrounding an anchor word as unordered list of words, calculating the Levenshtein distance of each word against every other word. We took several steps to speed this process up and avoid calculating distances uselessly. We decided that finding words with a greater Levenshtein distance than 3 changes would result in too many false matches, and so we limited the length difference between two measured words. Since our phrases could be fairly long, oftentimes over 200 words, we found that shortening the window around the anchor word we looked at to between 60 and 80 words in either direction reduced the running time while keeping the algorithm from possibly finding matches where they would be unlikely to occur. We decided that translations would rarely move a word over 120 words from the word it was translated from. However, while most of the words in most phrases will be analyzed by the algorithm looking at the anchor points at the ends of the phrase, this will result in the middle portions of some large phrases being ignored. To capture these 'lost' words we ran the entire lexical matching system through several iterations, using the words we decided were very good matches as new anchors, thus reducing the size of the phrases.

The way we decided whether a match was 'very

Appeared in: *Proceedings of the Class of 2005 Senior Conference*, pages 42–47
Computer Science Department, Swarthmore College

good' was if the words in the match met three requirements:

1. They had a Levenshtein distance of three or less.

2. The ratio between the frequency of the words was not too large in the corpus. For example, if one word appears only two times, the word it is matched with should not appear a hundred times. For words with lexical distance of 1, the ratio is 2:5; for 2, 3:5; for 3, 4:5.

3. The words, if they have a close numerical ratio, should usually be matched with each other. In other words, the matched words should not appear apart from each other too frequently. For words with Levenshtein distance of 1, neither word in the match can appear more than 30 times the number of times the match appears; for 2, 20; for 3, 10.

These requirements were applied with differing strictness depending on their Levenshtein distance. Words that were very similar to each other were allowed to vary in their unmatched appearances and numerical ratio more than words that were less lexically similar.

5 ϕ^2 Word Alignment

The ϕ statistic is used to determine correlation between two binary variables. After separating the corpus into phrases, it is a generally good approximation that a given word will appear only once per phrase. By relating the occurrence (one or zero) of a word in a phrase we can hope to find the equivalent translated word in the target language.

The general form of the phi statistic is

$$\phi = \frac{ad - bc}{\sqrt{efgh}} \tag{3}$$

where

	X^-	X^+	Total
Y^-	a	b	e
Y^+	c	d	f
Total	g	h	n

It can be seen ϕ is close to 1 if x and y frequently do and do not occur in conjunction, near 0 if there is no correlation, and if one rarely occurs when the other occurs ϕ is close to -1. In practice, however, computing the square root is relatively computationally intensive. Furthermore, we make the assumption that words will not be negatively related, that is, the existence of one word in a source phrase should not imply that some other word does occur in the target phrase. Making these assumptions, computation time can be decreased by computing

$$\phi^2 = \frac{(ad - bc)^2}{efgh} \tag{4}$$

.

An issue with using the ϕ statistic is computation time. We must compute the ϕ value for every source, target word pair. At initialization, we determine the binary occurrence, either a word does or does not exist per phrase, for each word in the source and target corpora. We iterate through each source phrase for each source word counting the binary occurrence of each target word in the equivalent target phrases. From this, we learn d and using the pre-computed binary occurrences for the entire corpus we can determine the values of all variables. For each source word and target word that occurs in some parallel phrase to the source word, we compute a phi score. We take the highest phi score and treat this as a translation for the source word. A final refinement is to only consider words source words which occur greater than two times. If we consider source words that only occur once, we will frequently receive a large list of false good matches.

This algorithm works well for fairly limited size corpus (<300,000 words), but as the size increases the number of phrases a word occurs in increases approximately linearly and thus the number of phi ranks that must be computed increases very rapidly. This has limited the size of corpus that may be used for training. We believe in future work this problem can be eliminated.

6 Data and Results

We primarily used two corpora for testing: sections from the 2001 Hansards and Swann's Way by Marcel Proust. Each of these documents are available

Appeared in: *Proceedings of the Class of 2005 Senior Conference*, pages 42–47
Computer Science Department, Swarthmore College

online in French and English. We also used a sentence by sentence alignment of the 2001 Hansards.

6.1 Phrase Alignment

Phrase alignment has been found to be reasonably precise. Due to the nature of phrase alignment we have no standard data to which we can compare our performance, but examination of parallel phrases reflect that it is generally good at picking out appropriate anchor points. One problem is that not enough anchor points are selected. For the Proust corpus of approximately 200k words, 1000 anchor points are found which translates into phrases of around 200 words. Increasing the parameters to allow the algorithm to locate more anchor points greatly decreases the quality of phrases.

6.2 Lexical Word Alignment

The lexical word alignment was only somewhat successful. It did not end up adding many new anchor points to our phrase alignments, as we needed to constrain the matches greatly in order to reach a high accuracy rate (approximately 70-80% correct). We only allowed matches of up to a Levenshtein distance of 3 and small variations in their occurrence ratios. This generally resulted in the introduction of 400-600 new anchor points to a system with an average of 4000 anchor points produced from using identical words and our anchor phrase alignment algorithm. All these results are on the Proust corpus.

6.3 ϕ^2 Word Alignment

ϕ^2 word alignment was tested using our phrase alignment system for the Proust and Hansards corpora and using the sentence-aligned Hansards corpus. Determining the total number of possible words pairs would be by definition hence we do not include recall numbers. Regardless of the phrase alignment method, if we only considered alignment words with a ϕ^2 value greater than 0.5 almost 80% of the words pairs were correct. However, using our anchor point based alignment system, we will receive less than 100 of $>$4000 unique words which occur twice, resulting in 90% precision, but a very low recall. Yet, when using the sentence-alignment Hansards corpus of 60,000 sentence pairs we find 4000 translation pairs with a precision of 83% as found from a randomly selected sample of 50 word pairs.

7 Conclusion

Two of the defining features of our dictionary induction system were the two texts we used and the ϕ correlation. As discussed in the results section, we have found that ϕ correlation works well with a large number of small parallel phrases. Our system would work best on literary texts with many proper nouns, which would give us better anchor point coverage. Using the sentence-aligned Hansards text showed us how critical having a well-aligned work is and pointed towards one of the problems we had with the non-sentence aligned literary work.

The results of our system are very promising. While we found that lexical alignment did not improve our results greatly, we found that a well aligned corpora can be used to produce a very good translational dictionary using a statistical method. Future work using ϕ word alignment for sentence aligned parallel corpora could provide a highly accurate translation dictionary using no knowledge of the text other than they are linguistically related.

8 Future Work

8.1 Phrase Alignment

Phrase alignment based on sentence boundaries should be examined in depth. While using lexically identical words yields accurate parallel phrases, it fails to yield enough of them. This is especially crucial for ϕ^2 word alignment where increasing the number of phrases and decreasing their size improves the accuracy and running time of the algorithm. Cursory examination of the number of periods in the French vs. English version of Proust's corpus shows many more English than French sentences. However, we can use the fact that on average a given number of English words occur per French word and compare sentence lengths to determine sentence by sentence alignment. As is shown in the results section for ϕ^2 word alignment, if we could improve parallel phrase alignment, we would receive much better results.

8.2 Lexical Word Alignment

The lexical word alignment could undoubtedly be improved in its accuracy and its production of accurately matched words by more tweaking of the

Appeared in: *Proceedings of the Class of 2005 Senior Conference*, pages 42–47
Computer Science Department, Swarthmore College

various thresholds and restraints placed on the results. As for larger plans, using language-specific rules and morphological knowledge would be good, but would mean that we could not easily port the system to other linguistically similar languages. A more general approach that would greatly increase the accuracy of our matches would be to use a corpus with part-of-speech tagging, either pre-tagged or done with a readily available part-of-speech tagger. Lastly, testing the lexical word alignment algorithm on other texts and more importantly, different types of texts would reveal further improvements that could be applied to our system.

8.3 ϕ^2 Word Alignment

Several improvements can be made to the ϕ^2 word alignment algorithm. Clearly, this method will work better with large amounts of data. However, we are limited in the amount of data it can currently handle due to the algorithm computing ϕ^2 values for each target word in a target phrase parallel to a source phrase in which the source word exists. Examining only the first approximately 10 phrases in which a source word appears, we can determine the target words that might be translations of the given source word. From this, we can examine the rest of the phrases for a source word and only compute ϕ^2 values for the target words we have picked out. This would significantly reduce time required to run this algorithm and all us to examine very large corpora.

The ϕ correlation statistic is intended for use with binary variables. Many sentences will contain multiple occurrences of a given word. This additional information should be taken into account either by using a different correlation statistic or somehow incorporating this information to the existing ϕ rank statistic.

References

Genzel, D. 2005. Inducing a bilingual dictionary from a parallel corpus in related languages. Submitted to *ACL-05*.

Melamed, D. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics* 25(1).

Gale, W. and Church, K. 1991. Identifying Word Correspondences in Parallel Texts *Proceedings of the 4th Speech and Natural Language Workshop*.

Bendersky, E. *Levenshtein Distance Algorithm: Perl Implementation*, `http://www.merriampark.com/ldperl.htm`

Appeared in: *Proceedings of the Class of 2005 Senior Conference*, pages 42–47
Computer Science Department, Swarthmore College