Aligning Parallel Texts: Do Methods Developed for English-French Generalize to Asian Languages?

Kenneth Church
Ido Dagan
William Gale
Pascale Fung
Jon Helfman
Bala Satish

AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill NJ, 07974-0636
kwc@research.att.com

1. Introduction

Parallel texts have recently received considerable attention in machine translation (e.g., Brown *et al*, 1990), bilingual lexicography (e.g., Klavans and Tzoukermann, 1990), and terminology research for human translators (e.g., Isabelle, 1992). We have been most interested in the terminology application. How would Microsoft, or some other software vendor, want the term "dialog box" to be translated in their manuals? Technical terms such as "dialog box" are difficult for translators because they are generally not as familiar with the subject domain as either the author of the source text or the reader of the target text. Parallel texts could be used to help translators overcome their lack of domain expertise by providing them with the ability to search previously translated documents for examples of potentially difficult terminology and see how they were translated in the past. "Existing translations contain more solutions to more translation problems than any other existing resource." (Isabelle, 1992)

There has recently been quite a bit of interest in programs that align parallel texts such as the Canadian Parliamentary Debates (Hansards), which are available in both English and French, e.g., Brown *et al.* (1991), Chen (1993), Church (1993), Gale and Church (1993), Dagan *et al.* (1993), Kay and Rösenschein (to appear), Kupiec (1993), Matsumoto *et al.* (1993), Simard *et al.* (1992), Warwick-Armstrong and Russell (1990). It remains an open question how well these methods might generalize to other language pairs, especially pairs such as English-Japanese and English-Chinese. Some of these methods depend on being able to count the number of words in a sentence, a measure that is fairly easy to compute for French and most European languages, but not for many Asian languages. Other methods make use of cognates, another measure that may be restricted to historically related language pairs that share a common character set.

Nevertheless, we have had some preliminary success in aligning certain texts in English and Japanese, especially texts such as the *AWK* manual (Aho, Kernighan, Weinberger (1980)), which have been translated from English into Japanese. This particular text contains a large number of examples and technical words that are the same in both the English source and target Japanese, a fact that greatly simplifies the alignment task. We will show that methods that were originally developed to identify cognates turn out to work surprisingly well on a text such as the *AWK* manual.

Moreover, it should also be possible to align sentences using character-based methods such as Gale and Church (1993) since it turns out that the number of characters in an English sentence is highly correlated with the number of bytes in the Japanese translation. One might speculate that similar results might apply to other Asian languages. That is, the number of bits required to express a concept in English is probably, on average, highly correlated with the number of bits required to express the same concept in practically any other language. In other words, it might well be the case that all of the worlds' writing systems are about equally compact encodings of meaning (modulo a few constants).

2. Align, Char_align and Word_align

We have developed three different alignment programs: (1) *align* (Gale and Church, 1993), (2) *word_align* (Dagan *et al.*, 1993) and (3) *char_align* (Church, 1993). *align* is intended to produce quick-and-dirty paragraph and sentence-level alignments, using the fact that longer paragraphs (and sentences) in one language tend to be translated into longer paragraphs (and sentences) in the other language, and that shorter paragraphs (and sentences) tend to be translated into shorter paragraphs (and sentences). A probabilistic score is assigned to each proposed correspondence of sentences based on the ratio of lengths of the two sentences (in bytes) and the variance of this ratio. This probabilistic score is used in a dynamic programming framework to find the maximum likelihood alignment of sentences. An evaluation showed that *align* correctly aligned 96% of the sentences.

Sentence-based methods have become popular in the literature because they are so easy to implement, and they produce reasonably accurate results with relatively little cost. These methods are based entirely on the lengths of the sentences in the two texts, and completely ignore the words. Of course, the words are very important. It would really help to know that the English word *government* is very likely to be translated into French as *gouvernement*. Word-based methods such as Kay and Rösenschein (to appear) are potentially more accurate than sentence-based methods since they make use of more information, although they are probably also more expensive. Hybrid methods have recently become popular; Shemtov (1993) and Brown *et al.* (to appear), for example, both use a sentence-based method to produce a rough initial starting point, and then word-based methods are used to refine the results. Our *word_align* program is also a hybrid method, though we tend to run it on the output of *char_align* rather than on the output of a sentence-based alignment program such as *align*.

char_align is intended as a robust alternative to *align*. Sentence-based methods have been very successful on clean inputs such as the Canadian Hansards. Unfortunately, if the input is noisy (due to OCR and/or unknown markup conventions), these methods tend to break down because the noise can make it difficult to find paragraph boundaries, let alone sentences and words. To deal more effectively with these robustness issues, we developed yet another alignment program, *char_align*, that produces quick-and-dirty rough alignments at the character level rather than at the sentence/paragraph level, based on the cognate approach proposed by Simard *et al* (1992). The new program, *char_align*, looks for 4-byte sequences in the source language that are identical to some 4-byte sequence in the target language. There are often quite a large number of these matching 4-byte combinations along the correct alignment path. Proper nouns are often the same in both the source and target text. Numbers, dates, times are also nearly the same in both texts. Likewise for punctuation. 4-grams are especially powerful in historically related language pairs such as English and French, where there are large numbers of cognates, e.g., *government* and *gouvernement*.

3. Do These Methods Generalize Beyond European Languages?

It might appear that all three of these methods might be limited to languages based on the Roman alphabet. In particular, *char_align* depends on finding 4-grams that are the same in both the source and target text. This assumption is unlikely to work very well when the source and target text are written in different alphabets.[1]

align makes a somewhat weaker assumption that the length of a source text (in characters) can be used to predict the length of the target text (in characters). Although this assumption has been very useful for the European languages that we have investigated thus far, it may run into trouble when we consider Japanese and Chinese, since these languages have a very different notion of what counts as a character.

word_align also might run into trouble because the European notion of a ''word'' may not translate very well into non-European languages like Japanese and Chinese. The first problem, of course, is that Japanese and Chinese do not use spaces to delimit words, and therefore, one cannot tokenize Japanese and Chinese into words by simply looking for spaces as one might for a European language. One could, of course,

1. Umemura (personal communication) hypothesized that although *char_align* might not work very well when one language is based on the Roman alphabet and the other is based on the Chinese alphabet, it might work just as well when both languages are based on the Chinese alphabet as it does when both languages are based on the Roman alphabet. This is a fascinating hypothesis that deserves further investigation.

overcome this first stumbling block with a more sophisticated tokenizer such as Juman (Matsumoto *et al.*, 1991) for Japanese or some equivalent for Chinese. Even so, one might feel uncomfortable equating the output of such a program with words in English. The Juman program is designed to output "bunsetsu," which correspond better to phrases in English than to words. The tokenizing problem is probably even more problematic in Chinese where they have only one character set rather than the four in Japanese. Moreover, it appears that native speakers have less clear intuitions about how a text should be segmented into word/phrase-like tokens than in Japanese or English.[2] European word-like units might not be a very useful concept when aligning non-European languages since the very concept of a European word might not be all that meaningful in non-European languages.

Nevertheless, despite numerous potentially serious concerns, we believe that many of these alignment methods are actually much more general than it might appear at first. We will review *align, char_align* and *word_align* and discuss how the methods might be applied to non-European texts.

4. Align

As previously mentioned, *align* makes use of the very strong correlation between the length of a piece of text (e.g., sentence or paragraph) and the length of its translation. This correlation is illustrated in Figure 1 below, which plots the length (in characters) of 10,000 English and French sentences, selected from the Canadian Hansards. The correlation is 0.97. We have found even higher correlations at the paragraph level than at the sentence level.

2. We tend to think that it is a trivial matter to tokenize English text into "words," but in fact, even in English, there are some interested challenges for tokenization. For example, are hyphenated words, one word or two? Spaces usually delimit words, but not always. For example, phrasal verbs such as *cut down* and complex nominals such as *White House* and *greenhouse effect* are often treated as a single "word," even though there are spaces in the middle of these "words." In fact, half of the entries in Roget's Thesaurus (Chapman, 1992) contain spaces. It is not easy to come up with a rigorous definition for a "word," even in a language like English where the writing system makes it fairly obvious, at least in most cases, and where native speakers usually have very strong (and consistent) intuitions.

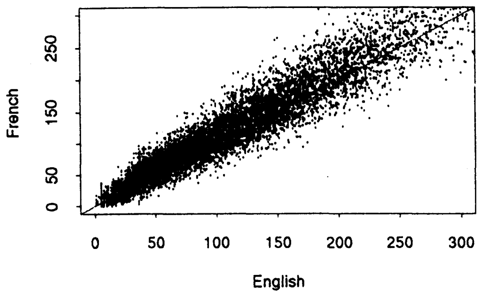

Figure 1: The length of an English sentence (in characters) is highly correlated (0.97) with the length of the corresponding French sentence. ($x = y$ line superimposed for comparison.)

The *align* program uses this correlation in a dynamic programming framework to decide which pieces of the source text correspond to which pieces of the target text. Figure 2 (below) shows the results of *align* on a particularly difficult paragraph where there are several multi-sentence alignments. A sentence in one language normally matches exactly one sentence in the other language (1-1), though other possibilities are not uncommon. The 1-1 case accounts for 90% of the alignments, with most of the remainder consisting of 2-1 and 1-2 alignments.

English	French
According to our survey, 1988 sales of mineral water and soft drinks were much higher than in 1987, reflecting the growing popularity of these products. Cola drink manufacturers in particular achieved above-average growth rates.	Quant aux eaux minérales et aux limonades, elles rencontrent toujours plus d'adeptes. En effet, notre sondage fait ressortir des ventes nettement supérieures à celles de 1987, pour les boissons à base de cola notamment.
The higher turnover was largely due to an increase in the sales volume.	La progression des chiffres d'affaires résulte en grande partie de l'accroissement du volume des ventes.
Employment and investment levels also climbed.	L'emploi et les investissements ont également augmenté.
Following a two-year transitional period, the new Foodstuffs Ordinance for Mineral Water came into effect on April 1, 1988. Specifically, it contains more stringent requirements regarding quality consistency and purity guarantees.	La nouvelle ordonnance fédérale sur les denrées alimentaires concernant entre autres les eaux minérales, entrée en vigueur le 1er avril 1988 après une période transitoire de deux ans, exige surtout une plus grande constance dans la qualité et une garantie de la pureté.

Figure 2: Sentence Alignment (from Union Bank of Switzerland)

What Next?
1.9 おつぎは何?

You have now seen the essentials of awk.
これで読者は awk の本質的な部分をひととおり見てきたことになる.

Each program in this chapter has been a sequence of pattern-action statements.
この章のそれぞれのプログラムは，パターン－アクション文の列であった.

Awk tests every input line against the patterns, and when a pattern matches, performs the corresponding action.
Awk はすべての入力行に対してそのパターンに適合しているかどうかを検査し，適合した場合は対応するアクションを実行する.

Patterns can involve numeric and string comparisons, and actions can include computation and formatted printing.
パターンは数値および文字列の比較を，またアクションは計算や書式つき印字をそれぞれ含んでもよい.

Besides reading through your input files automatically, awk splits each input line into fields.
読者の入力ファイルを自動的に読むだけでなく，awk はそれぞれの入力行を欄に分解する.

It also provides a number of built-in variables and functions, and lets you define your own as well.
Awk はまた，数々の組み込み変数や組み込み関数，それに読者独自の変数や関数を定義する機能も提供してくれる.

With this combination of features, quite a few useful computations can be expressed by short programs \(em many of the details that would be needed in another language are handled implicitly in an awk program.
これらの特徴を組み合わせることによって，実にたくさんの有用な計算が短いプログラムによって表現できる．それというのも他の言語で記述するならば必要になるであろう細かい部分の多くは，awkプログラムでは暗黙のうちに処理されるからである.

The rest of the book elaborates on these basic ideas.
この本の残りの部分では，これらの基本的なアイデアについて詳しく述べる.

Since some of the examples are quite a bit bigger than anything in this chapter, we encourage you strongly to begin writing programs as soon as possible.
いくつかの例はこの章にあるものよりもかなり大きいので，われわれは読者ができるだけ早くプログラムを書き始めるのを強く勧める.

This will give you famil\%iarity with the language and make it easier to understand the larger programs.
そうすることによって読者はこの言語に慣れ，その結果より大きなプログラムを理解するのが簡単になるであろう.

Furthermore, nothing answers questions so well as some simple experiments.
実際，簡単な経験以上のよき解答者はいないものである.

You should also browse through the whole book; each example conveys something about the language, either about how to use a particular feature, or how to create an interesting program.
読者はまたこの本の全体を通して，一つ一つの例をゆっくり眺めるべきである：これらの例は，この言語の特定の機能の使い方や興味深いプログラムの作製方法についての，なにがしかの情報を提供している.

Figure 3: An Example of Sentence-aligned English/Japanese (from AWK manual, p. 19)

How well do these results generalize to other languages? We have looked at several other European languages (e.g., German, Spanish) and found quite similar results. But, one might suspect that Japanese would be quite different. However, Figures 3 and 4 show that Japanese is actually not unlike French in this respect. Figure 3 shows a short excerpt from the AWK manual (Aho *et al.*, 1980, p. 19), and its Japanese translation, manually aligned sentence by sentence. Figure 4 shows the lengths of these sentences (in bytes), as well as the lengths of those in Figure 2. The J's denote the lengths of the Japanese sentences, and the F's denote the lengths of the French sentences. Note that both the J's and the F's seem to follow the $x = y$ line. Again, there is a very strong correlation, and the slope is fairly close to 1. The J's and F's together have a correlation of 0.95, and the J's alone have a correlation of 0.96. We conclude that the basic approach in *align* seems promising, even for language combinations such as English & Japanese and English & Chinese, because of the high correlation in sentence lengths, and because it is relatively easy to identify sentence boundaries in these languages.

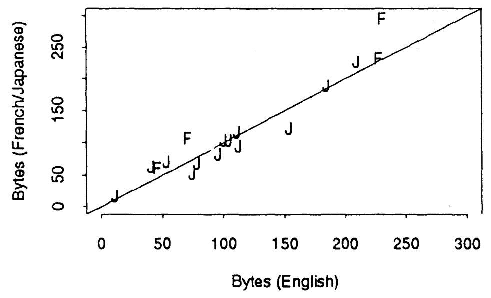

Figure 4: Lengths (in bytes) of sentences in Figures 2 and 3. ($x = y$ line superimposed for comparison.)

5. Char_align

Although Figure 4 suggests that we probably could have developed a sentence-based alignment program for Japanese (and possibly other non-Europen languages, as well) following similar methods that have been developed for the Canadian Hansards, we decided not to, because of a number of robustness issues mentioned above. Most real texts are not nearly as clean as the Hansards. *char_align* was designed as an alternative to *align*. It is intended to produce similar quick-and-dirty results, but it is more robust to the kinds of noise that we find in real texts (e.g., OCR noise, missing figures, etc.).

Figure 5 shows the results of *char_align* on a sample of Canadian Hansard data, kindly provided by Simard *et al*, along with alignments as determined by their panel of 8 judges. Simard *et al* (1992) refer to this dataset as the "hard" dataset and their other dataset as the "easy" dataset, so-named to reflect the fact that the former dataset was relatively more difficult than the latter for the class of alignment methods that they were evaluating. Figure 5 plots $f(x)$ as a function of x, where x is a byte position in the English text and $f(x)$ is the corresponding byte position in the French text, as determined by *char_align*. For comparison's sake, the plot also shows a straight line connecting the two endpoints of the file. Note that $f(x)$ follows the straight line quite closely, though there are small but important residuals, which may be easier to see in Figure 6.

Figure 6 plots the residuals from the straight line. The residuals can be computed as $f(x) - cx$, where c is the ratio of the lengths of the two files (1.1). The residuals usually have fairly small magnitudes, rarely more than a few percent of the length of the target file. In Figure 6, for example, the residuals have magnitudes less than 2% of the length of the target file.

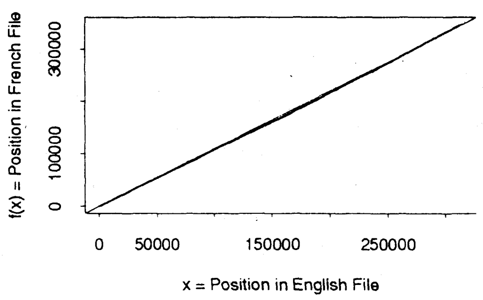

Figure 5: *char_align* output on the "Hard" Dataset

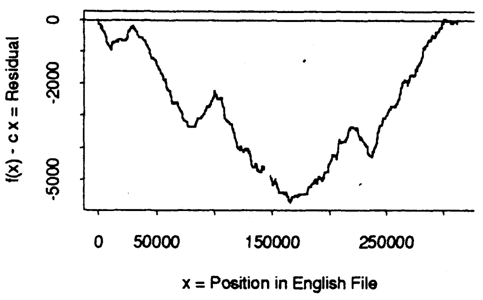

Figure 6: rotated version of Figure 5

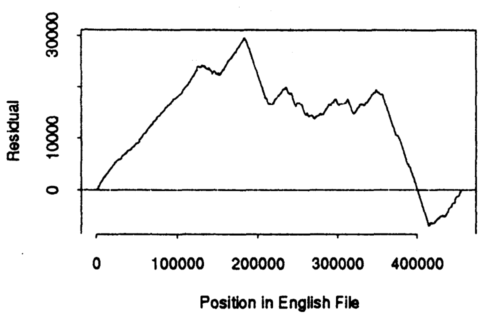

Figure 7: *char_align* output on English/Japanese (AWK manual)

If the residuals are large, or if they show a sharp discontinuity, then it is very likely that the two texts don't match up in some way (e.g., a page/figure is missing or misplaced). We have used the residuals in this way to catch errors of this kind. Figure 7 illustrates this use of the residuals for the AWK manual. Note that the residuals have relatively large magnitudes, e.g., 10% of the length of the file, compared with the 2% magnitudes in Figure 6. Moreover, the residuals in Figure 7 have several very sharp discontinuities. The location of these sharp discontinuities is an important diagnostic clue for identifying the location of the problem. In this case, the discontinuities are caused by several large figures (17,000 bytes) which were left out of the translation. One of the strengths of *char_align* over sentence-based methods is that it can cope with these sorts of realities. When we tried to transfer our alignment technology to AT&T Language Line, a commercial translation service within AT&T, we were rather surprised to discover that *align* didn't work because most texts are not nearly as clean as the Canadian Hansards. Most texts are full of large sections that don't align for one reason or another.

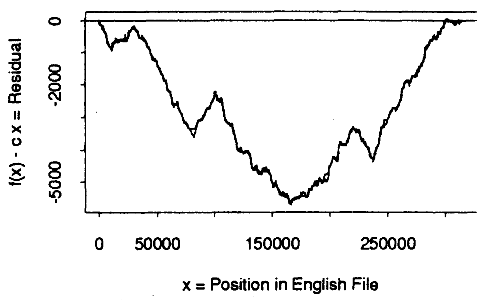

Figure 8: Figure 6 with judges' alignments

Figure 8 shows the correct alignments, as determined by Simard *et al*'s panel of 8 judges (sampled at sentence boundaries), superimposed over *char_align*'s output. *Char_align*'s results are so close to the judge's alignments that it is hard to see the differences between the two. *Char_align*'s errors may be easier to see in Figure 9, which shows a histogram of *char_align*'s errors. (Errors with an absolute value greater than 200 have been omitted; less than 1% of the data fall into this category.) The errors (mean of 2 characters ± a standard deviation of 46 characters) are much smaller than the length of a sentence (129±84 bytes). Half of the errors are less than 18 characters.

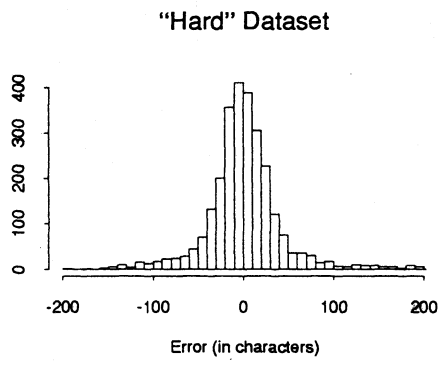

Figure 9: histogram of errors (from Canadian Hansards)

In general, performance is slightly better on shorter files than on longer files because *char_align* doesn't use paragraph boundaries to break up long files into short chunks. Figure 10 shows the errors for the "easy" dataset (−1±57 bytes), which ironically, happens to be somewhat harder for *char_align* because the "easy" set is 2.75 times longer than the "hard" dataset. (As in Figure 9, errors with an absolute value greater than 200 have been omitted; less than 1% of the data fall into this category.)

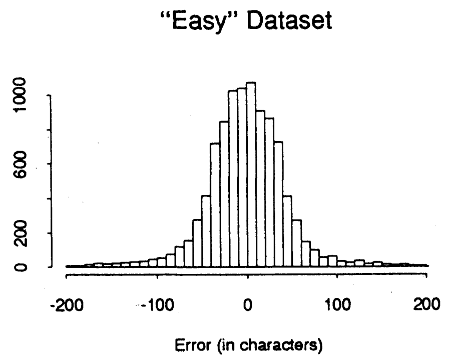

Figure 10: histogram of errors (from Canadian Hansards)

6. 4-grams

As mentioned above, *char_align* looks for 4-grams that are the same in both the source and target text. This method works fairly well for language pairs that make use of the Roman alphabet since there will usually be a fair number of proper nouns (e.g., surnames, company names, place names) and numbers (e.g., dates, times) that will be nearly the same in the two texts. We have found that it can even work on some texts in English and Japanese such as the AWK manual, because many of the technical terms (e.g., *awk, BEGIN, END, getline, print, printf*) are the same in both texts. We have also found that it can work on electronic texts in the same markup language, but different alphabets (e.g., English and Russian versions of 5ESS® telephone switch manuals, formatted in troff).

Figures 11 and 12 below demonstrate the usefulness of 4-grams using a scatter plot technique which we call *dotplots* (Church and Helfman, to appear). The source text (N_x bytes) is concatenated to the target text (N_y bytes) to form a single input sequence of N_x+N_y bytes. A dot is placed in position i,j whenever the input token at position i is the same as the input token at position j. (The origin is placed in the upper left corner for reasons that need not concern us here.) Various signal processing techniques are used to compress dotplots for large N_x+N_y.

The dotplots in Figures 11 and 12 look very similar, with diagonal lines superimposed over squares, though the features are somewhat sharper in Figure 12 because the input is much larger. Figure 11 shows a short article from a Christian Science magazine, 11 kbytes in English and 13 kbytes in German, whereas Figure 12 shows the entire AWK manual, English (455 kbytes) and Japanese (330 kbytes). The features become even sharper still in dotplots of much larger inputs. See Church and Helfman (to appear) for a dotplot of 3 years of the Canadian Hansards.

The diagonals and squares are commonly found in dotplots of parallel text. The squares have a very simple explanation. The upper-left quadrant and the lower-right quadrant are darker than the other two quadrants because the source text and the target text are more like themselves than either is like the other. This fact, of course, is not very surprising, and is not particularly useful for our purposes here. However, the diagonal line running through the upper-right quadrant is very important. This line indicates how the two texts should be aligned.[3]

Figure 13 shows the upper-right quadrant of Figure 11, enhanced by standard signal processing techniques (e.g., low-pass filtering and thresholding). The diagonal line in Figure 13 is almost straight, but not quite. The minor deviations in this line are crucial for determining the alignment of the two texts. Figures 14 and 15 make it easier to see these deviations by first rotating the image and increasing the vertical resolution by an order of magnitude. The alignment program makes use of both of these transformation in order to track the alignment path with as much precision as possible.

3. The discontinuities in the diagonal in Figure 12 are caused by several large figures that appear in the English source text but not in the Japanese target. These discontinuities can also be seen in Figure 7.

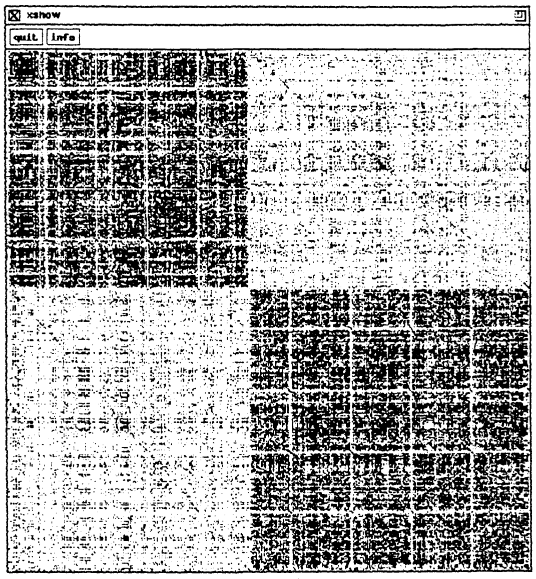

Figure 11: A dotplot demonstrating the usefulness of 4-grams (Christian Science material in English & German)

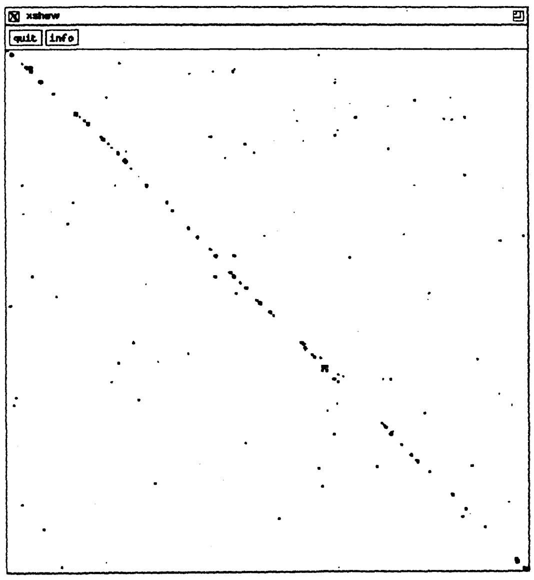

Figure 13: Upper-right quadrant of Figure 11 (enhanced by signal processing)

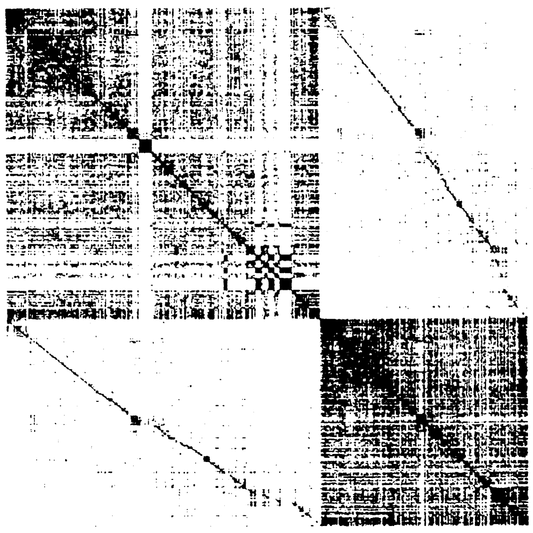

Figure 12: A dotplot demonstrating the usefulness of 4-grams (AWK manual in English & Japanese)

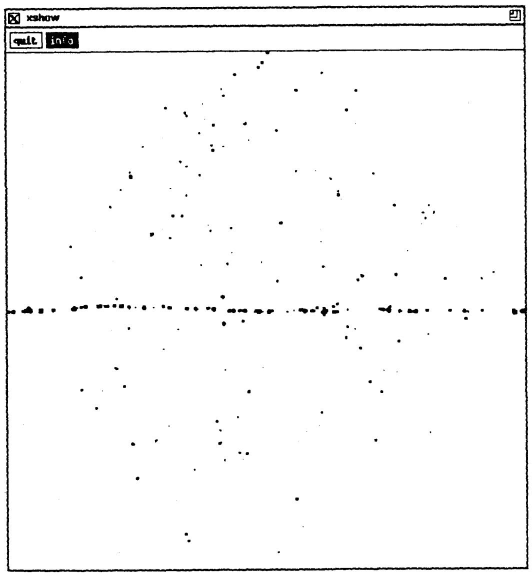

Figure 14: Rotated version of Figure 13

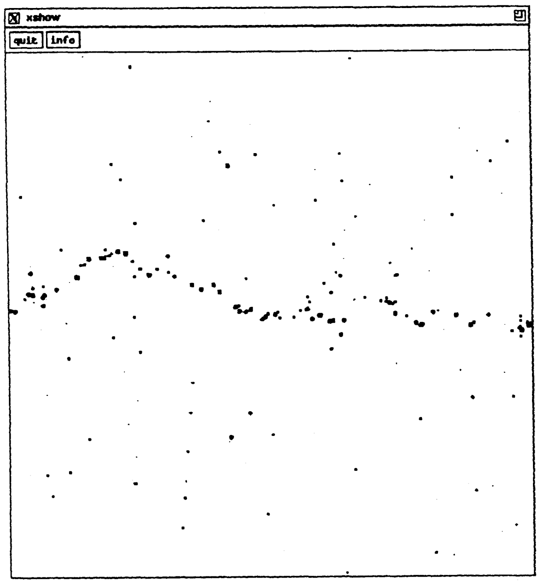

Figure 15: Figure 14 with 10x gain on vertical axis

It is difficult to know in advance how much dynamic range to set aside for the vertical axis. Setting the range too high wastes memory, and setting it too low causes the signal to be clipped. We use an iterative solution to find the optimal range. On the first iteration, we set the bounds on the search space very wide and see where the signal goes. On subsequent iterations, the bounds are reduced as the algorithm obtains tighter estimates on the dynamic range of the signal. The memory that was saved by shrinking the bounds in this way can now be used to enhance the horizontal resolution. We keep iterating in this fashion as long as it is possible to improve the resolution by tightening the bounds on the signal. Figure 16 shows the four iterations that were required for the Christian Science text and Figure 17 shows the three iterations that were required for the AWK manual.

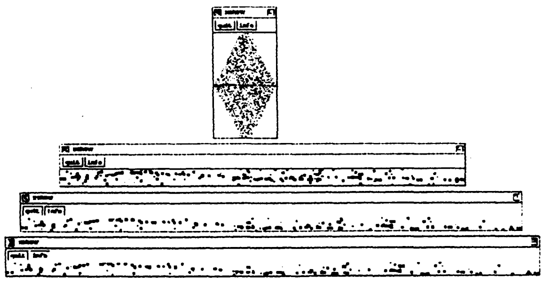

Figure 16: Four iterations (for Christian Science text)
(last three iterations were enhanced by signal processing)

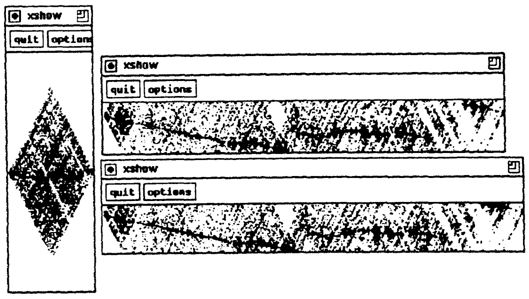

Figure 17: Three iterations (for AWK manual)

7. Word_align

As mentioned above, *word_align* is a hybrid method. It is intended to take the output of a quick-and-dirty alignment method such as *align* or *char_align* and improve the results using word-level constraints. The program estimates two sets of parameters: (1) translation probabilities, t, as illustrated in Figure 18, and (2) offset probabilities, o, as illustrated in Figure 19. The translation probabilities indicate the correspondences between words in one language and words in the other language, and the offset probabilities indicate how far *word_align* should search from the initial quick-and-dirty input alignment, I. Both sets of parameters are estimated using the EM algorithm, following Brown *et al.* (to appear).

French	English	$t(e\|f)$
zone	box	0.58
zone	area	0.28
zone	want	0.04
fermer	close	0.44
fermer	when	0.08
fermer	Close	0.07
insertion	insertion	0.61
insertion	point	0.23
insertion	Edit	0.06

Figure 18: Translation Probabilities
(from Microsoft Windows manual in English & French)

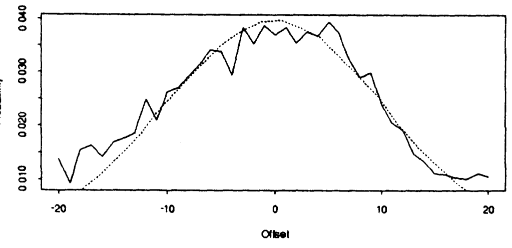

Figure 19: Offset Probabilities (from Canadian Hansards in English & French). The smooth line shows a normal distribution with a standard deviation of 10, suggesting that the input quick-and-dirty alignment has an error of ± 10 words.

The offset probabilities can be thought of as an estimate of the accuracy of the quick-and-dirty input alignment. Figure 20 shows a histogram of *char_align*'s errors, when compared against Simard *et al.*'s panel of eight judges. Note that Figure 20 is remarkably similar to the offset probabilities in Figure 19, which were computed using the EM algorithm, an unsupervised method, and did not make use of human judges.

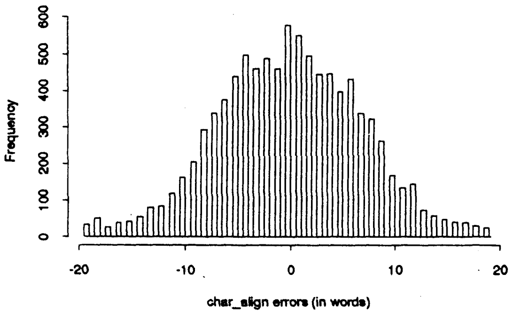

Figure 20: Histogram of *char_align*'s errors, computed on same material as Figure 19. Note the similarity between Figures 19 and 20.

Figure 21 shows that the combination of *word_align* and *char_align* significantly improves over the performance of *char_align* alone. Note that the variance is reduced by a factor of 5, and that more than half of the words are correctly aligned.

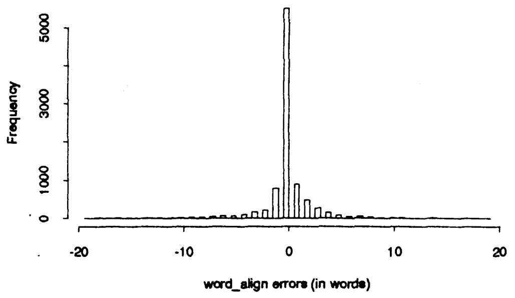

Figure 21: Word_align Errors (from Canadian Hansards). Note the improvement over Figure 20.

How are the translation and offset probabilities estimated? (The following is a brief summary of the algorithm; see Dagan *et al.* (1993) for more details.) Let i and j be positions in the two languages, say English and French, respectively, and let $Pr(con_{j,i})$ be the probability that the French word in position j is connected to the English word in position i. Let $CON_{f,e}$ be the set of connections between the French word f and the English word e. That is, $CON_{f,e}$ denotes a set of pairs (j,i) such that e is in position i and f is in position j and $Pr(con_{j,i}) > 0$.

Then the transition counts, $tcount(f,e)$, can be computed as:

$$tcounts(e,f) = \sum_{con_{j,i} \text{ in } CON_{f,e}} Pr(con_{j,i})$$

These counts can be turned into probabilities by normalizing in the obvious way:

$$t(e|f) = \frac{tcounts(e,f)}{\sum_{e'} tcounts(e',f)}$$

The offset probabilities, $o(k)$, can also be computed from the connections. A connection, $con_{j,i}$ has an offset $k = i - I(j)$. We use CON_k to denote the set of connections with offset k. Then the offset counts, *ocounts*, can be computed as:

$$ocounts(k) = \sum_{con_{j,i} \text{ in } CON_k} Pr(con_{j,i})$$

These counts are turned into probabilities by normalizing in the obvious way:

$$o(k) = \frac{ocounts(k)}{\sum\limits_{k'} ocounts(k')}$$

Thus, both t and o are defined in terms of connection probabilities, $Pr(con_{j,i})$. How do we estimate $Pr(con_{j,i})$? Let $conweights(j,i)$ be:

$$conweights(j,i) = t(f_j \mid e_i)\ o(i - I(j))$$

where e_i is the English word at position i, and f_j is the French word at position j. These *conweights* are turned into probabilities by normalizing over all English words that are within a window, w, of the initial rough alignment, I. That is,

$$Pr(con_{j,i}) = \frac{conweights(j,i)}{\sum\limits_{I(j)-w<i'\le I(j)+w} conweights(j,i')}$$

The equations for t, o and $Pr(con_{j,i})$ are used in the EM algorithm in an iterative fashion. We start with an initial guess for t and o. Then, we compute $Pr(con_{j,i})$, which is then used to recompute t and o. We typically iterate this process 10 times.

To find the best alignment, the program uses a refinement where the offsets are defined in terms of where the previous word was aligned, rather than in terms of the initial alignment. Dynamic programming is used to find the maximum likelihood alignment.

word_align has been run on the AWK manual in English and Japanese. The very same code was used on English & Japanese as we used on English & French, except that the Juman program was used to tokenize the Japanese text into "words" (and numerous uninteresting "minor modifications" for manipulating Japanese characters).

The results are shown in Figures 21 and 22, which correspond to Figures 18 and 19. Figures 23 and 24 use *word_align*'s results to produce a bilingual concordance. The translators at AT&T Language Line have found these kinds of bilingual concordances to be useful when compiling glossaries for European languages. Perhaps they will also be useful for other language combinations, as well.

```
0.03 0.21 0.36 0.52 0.65 0.74 0.81 0.86 0.89 program プログラム
0.02 0.06 0.06 0.05 0.04 0.04 0.03 0.03 0.03 program と
0.02 0.04 0.04 0.04 0.03 0.03 0.02 0.02 0.02 program で
0.01 0.02 0.02 0.02 0.01 0.01 0.01 0.01 0.01 program ある
0.01 0.02 0.02 0.02 0.02 0.01 0.01 0.01 0.01 program さ
0.01 0.02 0.02 0.02 0.02 0.02 0.01 0.01 0.01 program である
0.01 0.03 0.03 0.03 0.02 0.02 0.02 0.01 0.01 program この
0.01 0.03 0.03 0.03 0.03 0.02 0.02 0.01 0.01 program いる

0.02 0.18 0.37 0.57 0.70 0.80 0.85 0.89 0.91 input 入力
0.01 0.03 0.03 0.03 0.03 0.02 0.02 0.02 0.02 input :
0.02 0.06 0.07 0.06 0.05 0.04 0.03 0.03 0.02 input 行
0.01 0.02 0.02 0.02 0.01 0.01 0.01 0.01 0.01 input から
0.01 0.02 0.02 0.02 0.01 0.01 0.01 0.01 0.01 input さ
0.01 0.03 0.02 0.02 0.01 0.01 0.01 0.01 0.01 input P1
0.02 0.03 0.02 0.02 0.01 0.01 0.01 0.01 0.01 input と
0.02 0.04 0.03 0.03 0.02 0.02 0.02 0.01 0.01 input プログラム
```

Figure 21: $t(j|e)$ for nine iterations of EM-algorithm (from AWK manual in English & Japanese)

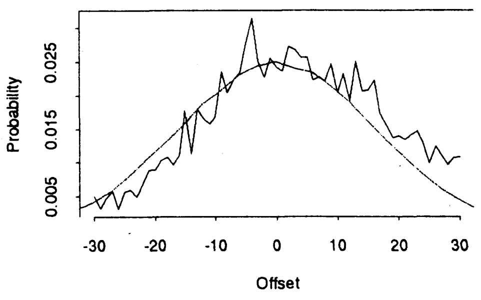

Figure 22: Offset Probabilities (from AWK manual in English & Japanese)

8. Conclusion

We have discussed how *align*, *char_align* and *word_align* might be generalized to handle non-European languages. *align* assumes that the length of a source text will be highly correlated with the length of its translation. Figure 4 showed that this assumption seems to hold for parallel text in Engish and Japanese.

char_align looks for 4-byte subsequences that are the same in both the source and target. This assumption is probably inappropriate when the source and target are written in different alphabets. Nevertheless, it seems to work reasonably well for the AWK manual because many of the technical terms (e.g., *awk, BEGIN, END, getline, print, printf*) are the same in both texts. Moreover, the assumption can probably be weakened in a fairly straightforward way to require merely that there are 4-byte sequences in the source text that are often found near some other 4-byte sequence in the target.

word_align estimates the translation probabilities and offset probabilities using the EM algorithm, following Brown *et al.* (to appear). The program appears to port relatively straightforwardly from English & French to English & Japanese.

Although we have shown that all three methods show promise for parallel texts such as the AWK manual in English & Japanese, much work remains to be done before we can reliably align documents when one language is based on the Roman alphabet and the other is based on the Chinese alphabet.

References

Aho, Kernighan, Weinberger (1980) "The AWK Programming Language," Addison-Wesley, Reading, Massachusetts, USA.

Brown, P., J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roossin, (1990) "A Statistical Approach to Machine Translation," *Computational Linguistics*, vol. 16, pp. 79-85.

Brown, P., Lai, J., and Mercer, R. (1991) "Aligning Sentences in Parallel Corpora," *ACL-91*.

Brown, P., Della Pietra, S., Della Pietra, V., and Mercer, R. (to appear), "The mathematics of machine translation: parameter estimation," *Computational Linguistics*.

Chapman, R. (1992) "Roget's International Thesaurus," fifth edition, HarperCollins, New York, USA.

Chen, S. (1993) "Aligning Sentences in Bilingual Corpora Using Lexical Information," *ACL-93*, pp. 9-16.

Church, K. (1993) "Char_align: A Program for Aligning Parallel Texts at the Character Level," *ACL-93*, pp. 1-8.

Church, K. and Helfman, J. (to appear) "Dotplot: A Program for Exploring Self-Similarity in Millions of Lines of Text and Code," *The Journal of Computational and Graphical Statistics*, also presented at *Interface-92*.

Dagan, I., Gale, W. and Church, K. (1993), "Robust Bilingual Word Alignment for Machine Aided Translation," in Proceedings of the Workshop on Very Large Corpora: Academic and Industrial Perspectives, available from the ACL.

Gale, W., and Church, K. (1993) "A Program for Aligning Sentences in Bilingual Corpora," *Computational Linguistics*, also presented at *ACL-91*.

Isabelle, P. (1992) "Bi-Textual Aids for Translators," in *Proceedings of the Eigth Annual Conference of the UW Centre for the New OED and Text Research*, available from the UW Centre for the New OED and Text Research, University of Waterloo, Waterloo, Ontario, Canada.

Kay, M. and Rösenschein, M. (to appear) "Text-Translation Alignment," *Computational Linguistics*.

Klavans, J., and Tzoukermann, E., (1990), "The BICORD System," *COLING-90*, pp 174-179.

Kupiec, J. (1993) "An Algorithm for Finding Noun Phrase Correspondences in Bilingual Corpora," *ACL-93*, pp. 17-22.

Matsumoto, Y., *et al.* (1991) Juman User's Manual (in Japanese), unpublished ms., Kyoto University, Japan.

Matsumoto, Y., Ishimoto, H., Utsuro, T. and Nagao, M. (1993) "Structural Matching of Parallel Texts," *ACL-93*, pp. 23-30.

Shemtov, H. (1993) "Text Alignment in a Tool for Translating Revised Documents," *EACL*, pp. 449-453.

Simard, M., Foster, G., and Isabelle, P. (1992) "Using Cognates to Align Sentences in Bilingual Corpora," *Fourth International Conference on Theoretical and Methodological Issues in Machine Translation* (*TMI-92*), Montreal, Canada.

Warwick-Armstrong, S. and G. Russell (1990) "Bilingual Concordancing and Bilingual Lexicography," *Euralex*.