Char_align: A Program for Aligning Parallel Texts at the Character Level

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

Abstract

There have been a number of recent papers on aligning parallel texts at the sentence level, e.g., Brown *et al* (1991), Gale and Church (to appear), Isabelle (1992), Kay and Rösenschein (to appear), Simard *et al* (1992), Warwick−Armstrong and Russell (1990). On clean inputs, such as the Canadian Hansards, these methods have been very successful (at least 96% correct by sentence). Unfortunately, if the input is noisy (due to OCR and/or unknown markup conventions), then these methods tend to break down because the noise can make it difficult to find paragraph boundaries, let alone sentences. This paper describes a new program, *char_align*, that aligns texts at the character level rather than at the sentence/paragraph level, based on the cognate approach proposed by Simard *et al*.

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. Translators find it extremely embarrassing when ‘‘store’’ (in the computer sense) is translated as ‘‘grocery,’’ or when ‘‘magnetic fields’’ is translated as ‘‘magnetic meadows.’’ Terminology errors of this kind are all too common because the translator is generally not as familiar with the subject domain as the author of the source text or the readers 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 expressions and see how they were translated in the past.

While pursuing this possibility with a commercial translation organization, AT&T Language Line Services, we discovered that we needed to completely redesign our alignment programs in order to deal more effectively with texts supplied by AT&T Language Line’s customers in whatever format they happen to be available in. All too often these texts are not available in electronic form. And even if they are available in electronic form, it may not be worth the effort to clean them up by hand.

2. Real Texts are Noisy

Most previous work depends on being able to identify paragraph and sentence boundaries with fairly high reliability. We have found it so difficult to find paragraph boundaries in texts that have been OCRed that we have decided to abandon the paragraph/sentence approach. Figure 1, for example, shows some parallel text (selected from the official record of the European Parliament) that has been processed with the Xerox ScanWorX OCR program. The OCR output is remarkably good, but nevertheless, the paragraphs are more elusive than it might appear at first.

The first problem we encountered was the missing blank line between the second and third paragraphs in the French (Figure 1b). Although this missing line might obscure the boundary between the two paragraphs, one could imagine methods that could overcome missing blank lines.

A more serious problem is illustrated by two phrases highlighted in italics in Figure 1, ‘‘Petitions Documents received...,’’ and its French equivalent, ‘‘Pétittons − Dépôt de documents....’’ When we first read the OCR output, we found these two expressions somewhat confusing, and didn’t understand why they ended up in such different places in the OCR output. After inspecting the original hardcopy, we realized that they were footnotes, and that their location in the OCR output depends on the location of the page breaks. Page breaks are extremely complicated. Most alignment programs don’t attempt to deal with issues such as footnotes, headers, footers, tables, figures and other types of floating displays.

One might believe that these layout problems could be avoided if only we could obtain the texts in electronic format. Perhaps so. But ironically, electronic formats are also problematic, though for different reasons.

Figure 1a: An Example of OCRed English

4. Agenda

PRESIDENT. − We now come to the agenda for this week.

SEAL (5). − Mr President, I should like to protest most strongly against the fact that there is no debate on topical and urgent subjects on the agenda for this part−session. I know that this decision was taken by the enlarged Bureau because this is an extraordinary meeting. None the less, how can we be taken seriously as a Parliament if we are going to consider only inter− nal matters while the world goes on outside? I would like to ask you to ask the enlarged Bureau to look at how we might have extra sittings in which urgencies would be included.

Having said that to the Chair and bearing in mind that there are no urgencies, I should like to ask the Com− mission to make statements on two items. First of all, what action is the Community taking to help the peo− ple of Nicaragua, who have suffered a most enormous natural disaster which has left one−third of the popula− tion homeless? Secondly, would Commissioner Suth− erland make a statement on the situation that has ari− sen in the United Kingdom, where the British Govern− ment has subsidized Aerospace to the tune of UKL 1 billion by selling them the Royal Ordnance factories at a knockdown price and allowing them to asset−strip in order to get this kind of cash?

(Protests from the right)

Petitions Documents received − Texts of treaties for− warded by the Council: see minutes. [italics added]

No 2−370/6 Debates of the European [...]

PRESIDENT. − I think you have just raised about four urgencies in one there. We cannot allow this. The enlarged Bureau made a decision. This decision came to this House and the House has confirmed it. This is a special part−session. We have an enormous amount of work to do and I suggest we get on with it.

Figure 1b: An Example of OCRed French

4. Ordre du jour

Le Président. − Nous passons maintenant à l'or− dre du jour de cette semaine.

Seal (s). − (EN> Monsieur le Président, je pro− teste énergiquement contre le fait que l'ordre du jour de cette session ne prévoit pas de débat d'actualité et d'urgence. Je sais que cette décision a été prise par le Bureau élargi parce qu'il s'agit d'une session extraordinaire. Néanmoins, comment pourrions−nous, en tant que Parlement, être pris au sérieux si nous ne nous occupons que de nos petits problèmes internes sans nous soucier de ce qui se passe dans le monde? Je vous serais recon− naissant de bien vouloir demander au Bureau élar− gi de voir comment nous pourrions avoir des séances supplémentaires pour aborder les questions urgentes.
Cela dit, et puisqu'il n'y a pas de problèmes urgents, je voudrais demander à la Commission de faire des déclarations sur deux points. Première− ment: quelles actions la Communauté envisage−t− elle pour venir en aide au peuple du Nicaragua,

Pétittons − Dépôt de documents Transmission par le Conseil de textes d'accords: CE. procès−verbai. [italics added]

qui vient de subir une immense catastrophe natu− relle laissant sans abri le tiers de la population? Deuxièmement: le commissaire Sutherland pour− rait−il faire une déclaration au sujet de la situation créée au Royaume−Uni par la décision du gouver− nement britannique d'accorder à la société Aero− space une subvention s'élevant à un milliard de livres sterling en lui vendant les Royal Ordinance Factories à un prix cadeau et en lui permettant de brader des éléments d'actif afin de réunir des liquidités de cet ordre?
(Protestations à droite>

Le Président. − Je pense que vous venez de parler de quatre urgences en une seule. Nous ne pouvons le permettre. Le Bureau élargi a pris une décision. Cette décision a été transmise à l'Assem− blée et l'Assemblée l'a entérinée. La présente pé− riode de session est une période de session spé− ciale. Nous avons beaucoup de pain sur la planche et je vous propose d'avancer.

There are a large number of different markup languages, conventions, implementations, platforms, etc., many of which are obscure and some of which are proprietary. In more than one instance, we have decided that the electronic format was more trouble than it was worth, and have resorted to OCR. Even when we did end up using the electronic format, much of the markup had to be treated as noise since we haven't been able to build interpreters to handle all of the world's markup languages, or even a large percentage of them.

3. Aligning at the Character Level

Because of the noise issues, we decided to look for an alternative to paragraph−based alignment methods. The resulting program, *char_align*, works at the character level using an approach inspired by the cognate method proposed in Simard *et al* (1992).

Figures 2 show 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 2 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 fairly closely, though there are small but important residuals, which may be easier to see in Figure 3.

Figure 3 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 (0.91). The residuals usually have fairly small magnitudes, rarely more than a few percent of the length of the file. In Figure 3, for example, residuals have magnitudes less than 2% of the length of the target file.

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 help translators catch potentially embarrassing errors of this kind.

Figure 4 illustrates this use of the residuals for the European Parliamentary text presented in Figure 1. Note that the residuals have relatively large magnitudes, e.g., 10% of the length of the file, compared with the 2% magnitudes in Figure 3. Moreover, the residuals in Figure 4 have two 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 were caused by the two troublesome footnotes discussed in section 2.

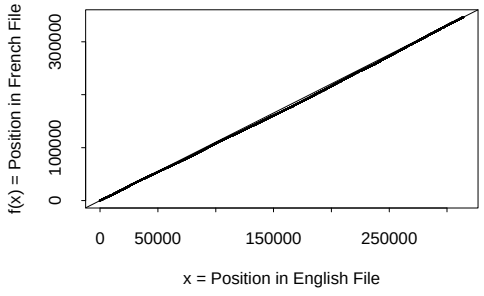

Figure 2: *char_align* output on the ‘‘Hard’’ Dataset

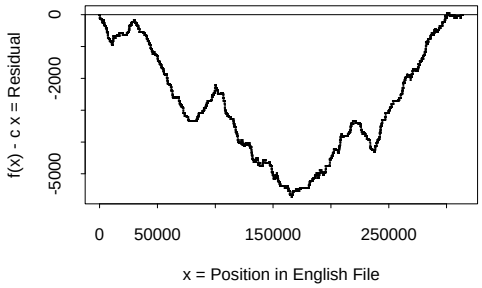

Figure 3: rotated version of Figure 2

Figure 4: Residuals for text in Figure 1
(large discontinuities correspond to footnotes)

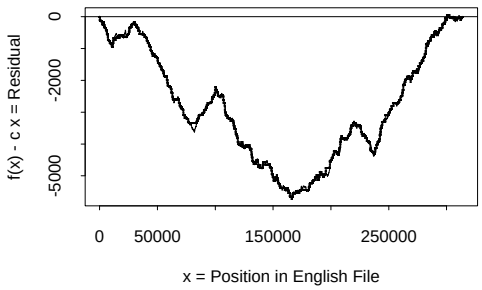

Figure 5: Figure 3 with judges' alignments

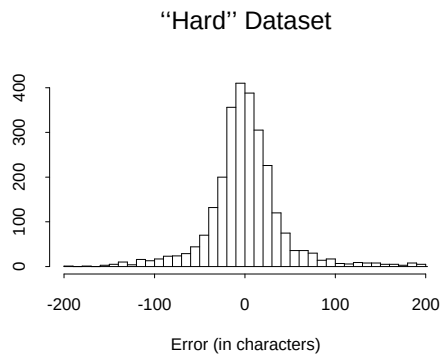

Figure 6: histogram of errors

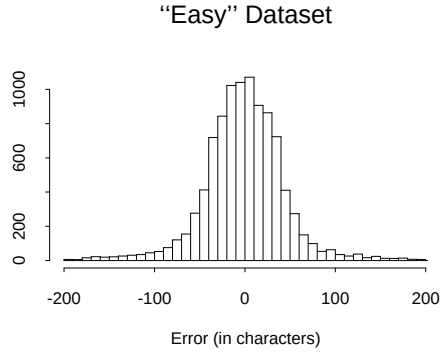

Figure 7: histogram of errors

Figure 5 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 6, 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 (2 ± 46 bytes) are much smaller than the length of a sentence (129 ± 84 bytes). Half of the errors are less than 18 characters.

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 7 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 6, errors with an absolute value greater than 200 have been omitted; less than 1% of the data fall into this category.)

4. Cognates

How does *char_align* work? The program assumes that there will often be quite a number of words near x that will be the same as, or nearly the same as some word near $f(x)$. This is especially true for historically related language pairs such as English and French, which share quite a number of cognates, e.g., *government* and *gouvernement*, though it also holds fairly well for almost any language pair that makes 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 8 and 9 below demonstrate the cognate property 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 implementation of dotplots are discussed in more detail in section 7.

The dotplots in Figures 8 and 9 look very similar, with diagonal lines superimposed over squares, though the features are somewhat sharper in Figure 8 because the input is much larger. Figure 8 shows a dotplot of 3 years of Canadian Hansards (37 million words) in English and French, tokenized by words. Figure 9 shows a dotplot of a short article (25 kbytes) that appeared in a Christian Science magazine in both English and German, tokenized into 4−grams of characters.

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 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.

Figure 10 shows the upper−right quadrant of Figure 9, enhanced by standard signal processing techniques (e.g., low−pass filtering and thresholding). The diagonal line in Figure 10 is almost straight, but not quite. The minor deviations in this line are crucial for determining the alignment of the two texts. Figures 11 and 12 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.

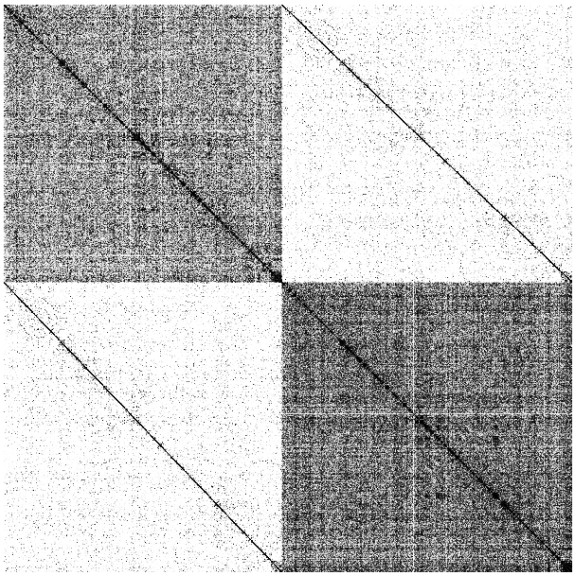

Figure 8: A dotplot demonstrating the cognate property
(37 million words of Canadian Hansards)

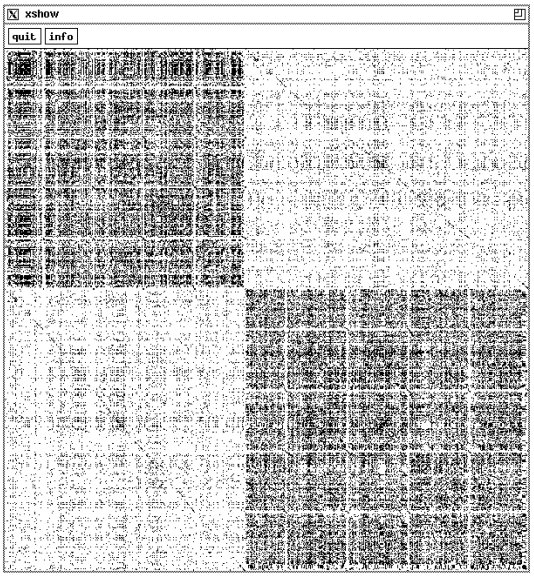

Figure 9: A dotplot demonstrating the cognate property
(25 kbytes selected of Christian Science material)

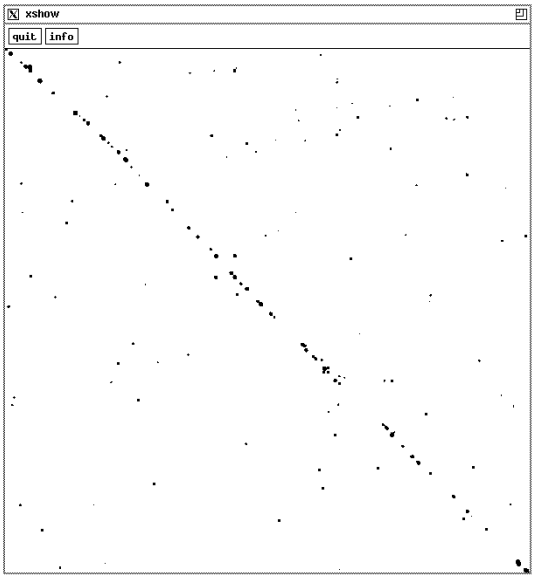

Figure 10: Upper−right quadrant of Figure 9
(enhanced by signal processing)

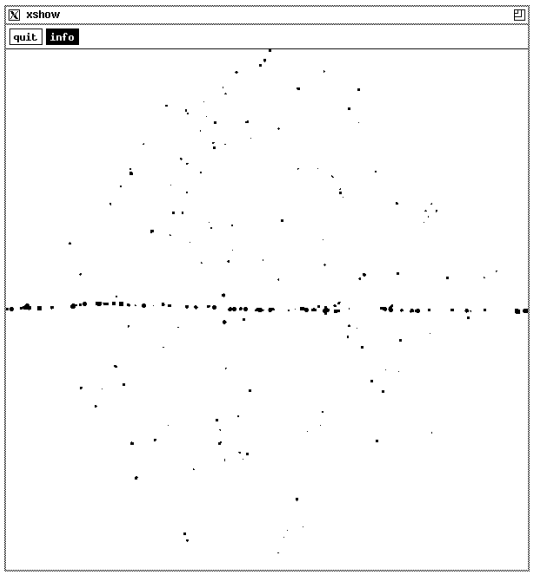

Figure 11: Rotated version of Figure 10

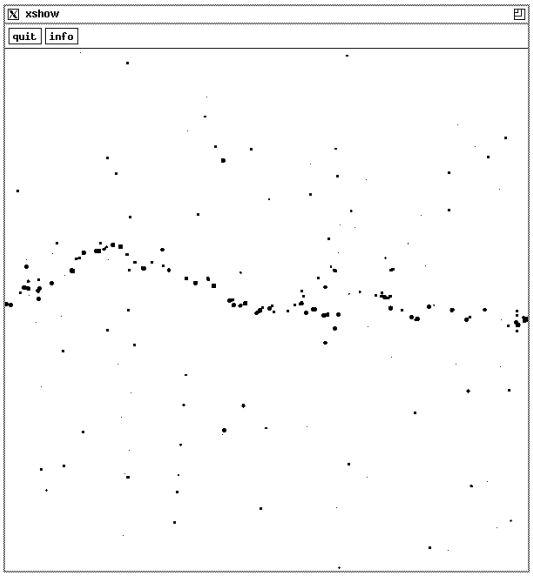

Figure 12: Figure 11 with 10x gain on vertical axis

5. Bounds Estimation

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, $B_{\min}$ and $B_{\max}$, very wide and see where the signal goes. The search will consider matching any byte x in the source file with some byte in the target file between $f(x) - B_{\min}$ and $f(x) + B_{\max}$, where $f(x)$ is the current best estimate of the position in the target file that corresponds to position x in the source file. 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.

```
while(making_progress) {
   Estimate_Bounds: Bmin, Bmax
   Estimate_Resolution_Factor: r
   Compute_Dotplot
   Compute_Alignment_Path
   }
```

Figure 13 shows the four iterations that were required for the Christian Science text. For expository convenience, the last three iterations were enhanced with a low−pass filter to make it easier to see the signal.

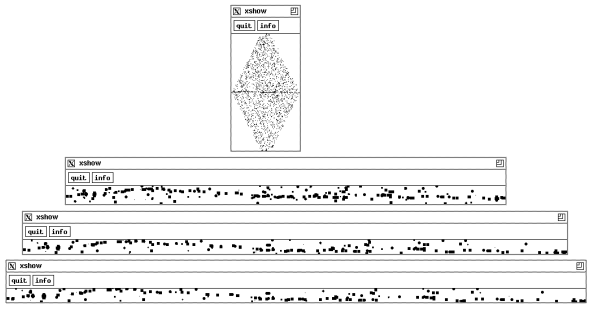

Figure 13: Four iterations

6. Resolution Factor Estimation

We need to allocate an array to hold the dots. Ideally, we would like to have enough memory so that no two points in the search space corresponded to the same cell in the array. That is, we would like to allocate the dotplot array with a width of $w = N_x + N_y$ and a height of $h = B_{\max} + B_{\min}$. (The array is stored in rotated coordinates.) Unfortunately, this is generally not possible. Therefore, we compute a ‘‘resolution’’ factor, *r*, which indicates how much we have to compromise from this ideal. The resolution factor, *r*, which depends on the available amount of memory *M*, indicates the resolution of the dotplot array in units of bytes per cell.

$$r = \sqrt{\frac{(N_x + N_y)\,(B_{\max} + B_{\min})}{M}}$$

The dotplot array is then allocated to have a width of $w = \frac{N_x + N_y}{r}$ and a height of $h = \frac{B_{\max} + B_{\min}}{r}$. The dots are then computed, followed by the path, which is used to compute tighter bounds, if possible. As can be seen in Figure 13, this iteration has a tendency to start with a fairly square dotplot and generate ever wider and wider dotplots, until the signal extends to both the top and bottom of the dotplot.

In practice, the resolution places a lower bound on the error rate. For example, the alignments of the ''easy'' and ''hard'' datasets mentioned above had resolutions of 45 and 84 bytes per cell on the final iterations. It should not be surprising that the error rates are roughly comparable, ±46 and ±57 bytes, respectively. Increasing the resolution would probably reduce the error rate. This could be accomplished by adding memory (*M*) or by splitting the input into smaller chunks (e.g., parsing into paragraphs).

7. Dotplot Calculation

In principle, the dotplot could be computed by simply iterating through all pairs of positions in the two input files, *x* and *y*, and testing whether the 4−gram of characters in text *x* starting at position *i* are the same as the 4−gram of characters in text *y* starting at position *j*.

```
float dotplot[Nx][Ny];
for(i=0; i<Nx; i++)
  for(j=0; j<Ny; j++)
    if(chars4(x, i) == chars4(y, j))
      dotplot[i][j] = 1;
    else dotplot[i][j] = 0;
```

In fact, the dotplot calculation is actually somewhat more complicated. First, as suggested above, the dotplot is actually stored in rotated coordinates, with a limited resolution, *r*, and band limited between $B_{\min}$ and $B_{\max}$. These heuristics are necessary for space considerations.

In addition, another set of heuristics are used to save time. The dots are weighted to adjust for the fact that some matches are much more interesting than others. Matches are weighted inversely by the frequency of the token. Thus, low frequency tokens (e.g., content words) contribute more to the dotplot than high frequency tokens (e.g., function words). This weighting improves the quality of the results, but more importantly, it makes it possible to save time by ignoring the less important dots (e.g., those corresponding to tokens with a frequency greater than 100). This heuristic is extremely important, especially for large input files. See Church and Helfman (to appear) for more details and fragments of c code.

8. Alignment Path Calculation

The final step is to find the best path of dots. A sub−optimal heuristic search (with forward pruning) is used to find the path with the largest average weight. That is, each candidate path is scored by the sum of the weights along the path, divided by the length of the path, and the candidate path with the best score is returned. Admittedly, this criterion may seem a bit *ad hoc*, but it seems to work well in practice. It has the desirable property that it favors paths with more matches over paths with fewer matches. It also favors shorter paths over longer paths. It might be possible to justify the optimization criterion using a model where the weights are interpreted as variances.

9. Conclusion

The performance of *char_align* is encouraging. The error rates are often very small, usually well within the length of a sentence or the length of a concordance line. The program is currently being used by translators to produce bilingual concordances for terminology research. For this application, it is necessary that the alignment program accept noisy (realistic) input, e.g., raw OCR output, with little or no manual cleanup. It is also highly desirable that the program produce constructive diagnostics when confronted with texts that don't align very well because of various snafus such as missing and/or misplaced pages. *Char_align* has succeeded in meeting many of these goals because it works at the character level and does not depend on finding sentence and/or paragraph boundaries which are surprisingly elusive in realistic applications.

References

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*.

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*.

Gale, W., and Church, K. (to appear) ‘‘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.

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*.