Machine Translation **13:** 59–80, 1998.

Bilingual Sentence Alignment: Balancing Robustness and Accuracy

MICHEL SIMARD and PIERRE PLAMONDON
Laboratoire de recherche appliquée en linguistique informatique (RALI), Département d'informatique et de recherche opérationnelle, Université de Montréal, CP 6128, Succ. Centre-Ville, Montréal, Québec, Canada H3C 3J7
{simardm,plamondo}@iro.umontreal.ca

Abstract. Sentence alignment is the problem of making explicit the relations that exist between the sentences of two texts that are known to be mutual translations. Automatic sentence-alignment methods typically face two kinds of difficulties. First, there is the question of robustness. In real life, discrepancies between a source text and its translation are quite common: differences in layout, omissions, inversions, etc. Sentence-alignment programs must be ready to deal with such phenomena. Then, there is the question of accuracy. Even when translations are "clean", alignment is still not a trivial matter: some decisions are hard to make, even for humans. We report here on the current state of our ongoing efforts to produce a sentence-alignment program that is both robust and accurate. The method that we propose relies on two new alignment engines: one that produces highly reliable and robust character-level alignments, and one that relies on statistical lexical knowledge to produce accurate mappings. Experimental results are presented which demonstrate the method's effectiveness, and highlight where problems remain to be solved.

Key words: bilingual sentence alignment, bitext, translation analysis, statistical translation model

1. Introduction

The "bitext correspondence problem" can be informally described as that of making explicit the relations that exist between two texts that are known to be mutual translations. The result of this operation can take many forms, but the output of most existing bitext-correspondence methods falls into one of two categories (Figure 1):

- An alignment is a parallel segmentation of the two texts, typically into sentences, such that the nth segment of the first text and the nth segment of the second are mutual translations.
- A bitext map is a set of pairs (x, y), where x and y refer to precise locations in the first and second texts respectively, with the intention of denoting portions of the texts that correspond to one another.

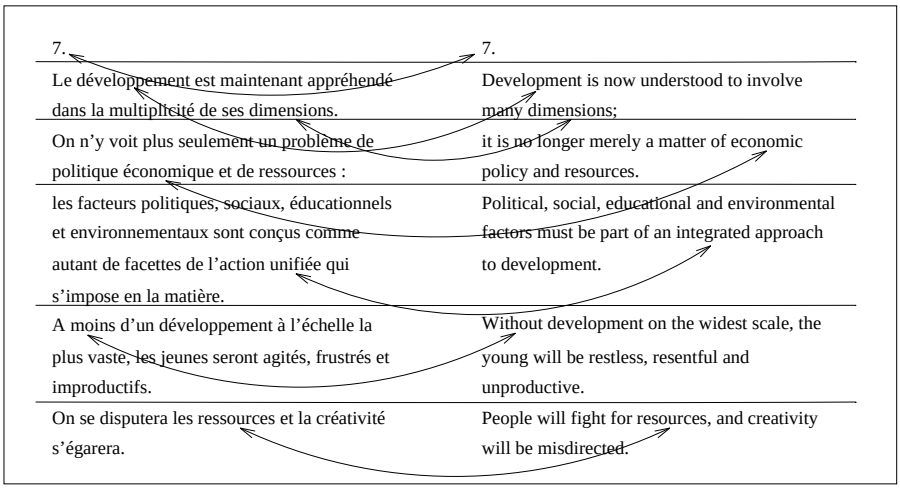

Figure 1. Alignment vs. Bitext Map. Horizontal lines denote the segmentations of a sentence alignment and arrows denote a word-level mapping.

Bitext correspondences are of vital interest to anyone who wishes to exploit existing translations as an active source of information; see, for example, Isabelle et al. (1993) and Klavans & Tzoukermann (1995). Which of an alignment and a bitext map is best usually depends on the intended application. By definition, an alignment covers the totality of the bitext. In this sense, it is both exhaustive and exact: for each segment of text, it says something like "The translation of this segment is exactly that segment." The same cannot be said of bitext maps. Some methods, such as those proposed by Church (1993) or Fung & McKeown (1994), produce approximate maps (i.e. not exact), that say something like "The translation of the text around point x is somewhere around point y." Other methods, such as those proposed by Dagan et al. (1993) or Melamed (1996) produce maps that are exact ("The translation of the object at position x is the object at position y") but not exhaustive. On the other hand, what they lack in exactness or exhaustiveness, bitext maps usually make up for in resolution: they give a closer view of the correspondence.

There are many situations where alignments are preferable, however. In particular, this appears to be true of applications where the bitext correspondence is directly intended for a human. An example of such an application is the bilingual concordance system described by Simard et al. (1993). This system allows a user to query a large corpus of bitext for specific expressions in one or both languages. Most often, the goal of the user is to find out how a given expression is translated. Using a sentence alignment for such a system has two advantages: first, given that exhaustive and accurate mappings at the level of expressions are not yet available, it ensures that the excerpts of bitext returned by the system contain both the queried expression and its translation; second, it allows the system to return the expression

and its translation within a coherent context, so that the users can evaluate the relevance of each returned item with regard to their own problem.

On the other hand, aligned pairs of sentences appear to constitute the most appropriate type of material for the training of self-organizing methods, such as the statistical translation models proposed by Brown et al. (1993). In fact, one of the first sentence-alignment programs was developed precisely for that purpose (Brown et al., 1991).

In recent years, our research group has been involved in the development of a new generation of translation support tools. In the process, we were lead to experiment with a number of bitext-correspondence techniques. One aspect that emerged from these experiments is how we can benefit from combining different approaches. In the following pages, we describe a sentence-alignment technique which, by combining two distinct approaches, inherits both the robustness of so-called "character alignment" methods, and a level of accuracy that can only be achieved with the use of bilingual lexical resources.

2. Background

Interest in the bitext-correspondence problem seems to have begun sometime during the 1980s, at a time when large text corpora were becoming more and more available to researchers. By the end of the decade, independent efforts on the subject were being pursued concurrently in many places, most notably at Xerox PARC (Kay & Röscheisen, 1993), IBM's Thomas J. Watson Research Center (Brown et al., 1991), AT&T Bell Laboratories in Murray Hill (Gale & Church, 1991) and in Geneva, at ISSCO (Catizone et al., 1989). Interestingly, all these early efforts focussed on sentence alignments rather than other types of bitext maps.

The best known of these early sentence-alignment methods were published in 1991. Coincidentally, there were two of them, they were presented at the same conference, and described alignment methods that were virtually identical. Both were based on a statistical modeling of translations that took into account only the length of the text segments (sentences, paragraphs), and relied on a dynamic programming scheme to find the most likely alignment. The main difference between the two approaches was how length was measured: while Brown et al. (1991) counted words, Gale & Church (1991) counted characters.

The early successes obtained using these methods almost gave the impression that the problem had been solved. Of course, this was not the case, and although it is true that sentence alignment is mostly an easy problem, anyone who has attempted to align by hand a sufficient amount of text knows that there are situations where even humans have a hard time making a decision (see Figure 2 for an example). The truth of the matter is that the bitext-correspondence problem is just one instance of the more general *translation analysis* problem (Isabelle et al., 1993), which is known to be AI-complete. In other words, solving the bitext correspondence problem is no easier than solving any other significant AI problem. Why is that so?

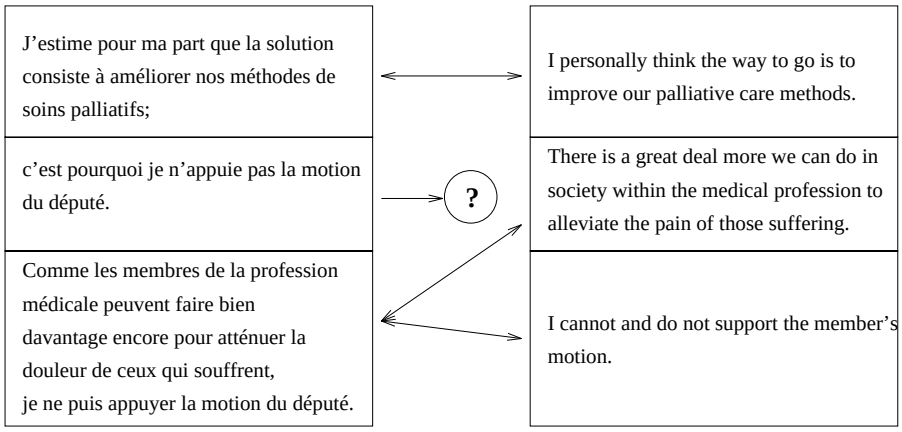

Figure 2. Example of a Non-trivial Sentence Alignment. Was the second French sentence omitted in the English?

The first issue is that of robustness. For a long time, almost everybody in the field was working with the same set of data, namely the Canadian Hansards (parliamentary proceedings). As other multilingual corpora became available, it quickly became obvious that the Hansards were exceptionally clean translations. As pointed out in Church (1993), "Real texts are noisy." Earlier methods are likely to wander off track when faced with deviations from the standard linear progression of translation, as for instance when parts of the source text do not make their way into the translation (omissions), or end up in a different order (inversions).

To deal with the robustness issue, Church took a very straightforward and intuitive approach, exploiting an alignment criterion that was first proposed by Simard et al. (1992): cognate words. Cognates are pairs of words of different languages that have close etymological ties. Often, this tie will be reflected both in the meanings and orthography of these words. As a result, they are likely mutual translations, and they are fairly easy to detect, even for someone who is familiar with neither of the languages involved. Church's program, called *char_align*, does not rely on a formal definition of cognates, but rather on a more general notion of *resemblance* between source text and translation. Interestingly, what *char_align* does could very well be compared to what a human would do to get a rough bitext mapping, i.e. take a certain distance from the texts, and look for similarities in layout, or for obvious clues such as numbers, proper names, etc.

Unlike its predecessors, *char_align* will usually not be fooled by omissions, inversions and other oddities. At the same time, again for the sake of robustness, the program does not rely on an *a priori* segmentation of the texts into paragraphs and sentences. Church realized that this was one of the major problems with earlier approaches: proper segmentation is not a trivial problem, and incorrect segmentations are likely to lead to incorrect alignments. He resolved the problem by building a program that completely ignores logical text divisions. (As a matter

of fact, *char_align* is not even interested in words—what it aligns are *bytes*.) As a result, the output of the program is also radically different from that produced by previous methods: it consists of an approximate mapping between regions in the two texts.

The second issue is that of accuracy: even when the input texts are clean, alignment programs are sometimes faced with hard decisions. The alignment process can be seen as figuring out the translator's interpretation of the source text, an exercise which, in certain situations, is not very different from that of figuring out the author's original intentions.

Clearly, this problem cannot be solved without some sort of bilingual lexical knowledge. A machine-readable bilingual dictionary, even of modest size, would be of invaluable help in this regard. Unfortunately, such resources are not yet generally available. We therefore have to turn to alternatives.

So far, the most promising of these alternatives are stochastic translation models, such as those that were proposed by Brown et al. (1993). With the goal of developing an entirely self-organizing MT system, Brown et al. elaborated a series of statistical models, whose purpose was to estimate the likelihood of two texts being mutual translations.

Statistical models of this type have been used to produce bitext correspondences. For example, to compute sentence alignments, Chen (1993) replaces the simple length-based models of earlier methods by a more elaborate model that takes into account the words of the text. Dagan et al. (1993) use a similar model to obtain word-level mappings. These methods are interesting with regard to bitext correspondences, because they focus on the lexical content of the texts. Furthermore, they have the ability to learn: every new pair of texts that is aligned can theoretically be used to refine the parameters that make up the model, hopefully permitting better alignments.

3. Robust and Accurate Sentence Alignments

We now describe our approach to the sentence-alignment problem. Our idea is to combine the robustness of character-based methods, such as *char_align*, and the accuracy of lexical-based methods. This idea is implemented as a two-step strategy: first compute a bitext map, working on robustness rather than accuracy; then use this map to constrain the search space for the computation of the sentence alignment, this time relying on a method that favors accuracy over robustness or efficiency.

3.1. FIRST STEP: INITIAL BITEXT MAPPING

The initial bitext map is computed using a program that we call *Jacal* (Just another cognate alignment program). This program is in the same line as Church's *char_align* and Melamed's (1996) SIMR program. Like these programs, it looks

for similar patterns of characters to produce a bitext map that is highly reliable and is independent of the texts' logical divisions (sections, paragraphs, sentences).

A bitext-mapping program that looks at the texts too closely is likely to miss out on even the most obvious deviation from the expected progression of translation (a case of not seeing the forest for the trees). On the other hand, one that is able to take a certain distance from the texts and capitalize on the most reliable evidence first is likely to attain a greater degree of robustness.

Suppose you are looking at a pair of texts A and B, and that you find some word in text A that looks like no other words in its vicinity, but that does look a lot like some other word in text B, which itself looks like no other words in its vicinity. Together, these words form what we call a pair of *isolated cognates*. Obviously, such pairs are very likely matches in a bitext map.

What *Jacal* does is to try to match these isolated cognates:

- *Jacal* considers two word forms of different languages to be *cognates* if their four first characters are identical, disregarding letter-case or diacritics (words four characters long and less must match in their entirety). In spite of its simplicity, this operational definition of cognates works well for related pairs of languages such as French and English, as demonstrated by Simard et al. (1992).
- An occurrence of a word form is said to be *isolated* if no occurrence of *resembling* word forms appears within a certain window around this occurrence. This isolation window is measured in characters, and is set to cover a given fraction of the text considered, say 30%.
- As for the notion of *resemblance* between word forms, *Jacal* uses the same definition as for *cognateness*, except this time applied to pairs of word forms of the same language. Once again, this is a very simple-minded solution, but it works well in practice, and it allows the program to do without language-specific morphological dictionaries.

Jacal does not systematically add all pairs of isolated cognates to the bitext map. To explain how the selection is done, it is useful to look at bitext maps from a graphical point of view, as if both texts to map were respectively laid out along the x and y axes in the plane (see Figure 3).

Jacal initially includes two points in the map: those that correspond to the beginnings and ends of the texts. Assuming that the alignment is going to lie somewhere along the line segment that connects these two points, it draws this line, and then a "corridor" around it, whose width is proportional to the distance between the two initial points. It then adds to the set only those points corresponding to pairs of isolated cognates that lie within this search corridor.

While most of the points found using this method are true correspondences, some may be wrong. We have found that most of these erroneous points are easy to detect, because they are usually not in line with their neighbors. To eliminate these

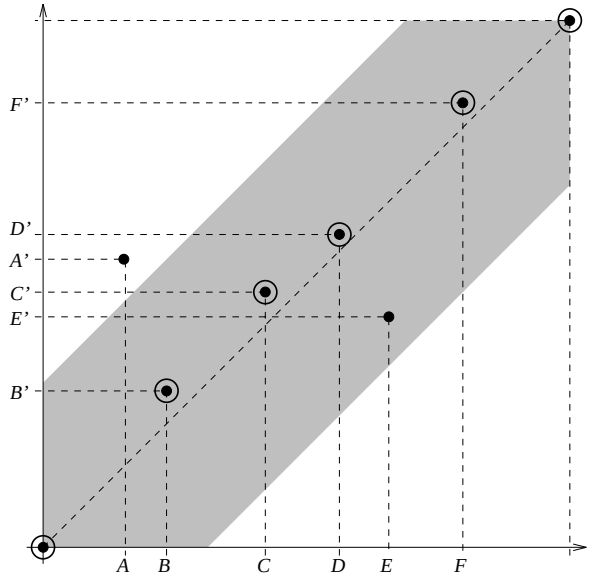

Figure 3. *Jacal*'s Basic Mechanism: First, identify a search corridor (shaded region); then, identify pairs of isolated cognates (points $(A, A'), (B, B'), \ldots, (F, F')$); finally, eliminate points that lie outside the corridor (such as point (A, A')) and points that are not in line with their neighbors (such as point (E, E')).

points, *Jacal* relies on a simple smoothing technique, based on linear regression (Figure 4). The general idea is to take short sequences of consecutive alignment points, draw a line that approximates this set of points, and then eliminate those points that lie too far away from the line. Of course, some particular point may be out of line with, say, neighboring points to its left, but be perfectly in line with points immediately to its right. Our smoothing method takes care of this situation by examining all sequences to which a particular point belongs, as follows:

1. Examine all sequences of n consecutive points (relative to the x-axis).
2. For each, find the least-squares line, and then draw a rectangle along this line, whose width is proportional to the geometric distance between the first and last points of the sequence.
3. Then verify which points of the sequence fall within the rectangle.
4. Once this is done for all sequences, count in how many rectangles each point of the alignment falls.
5. Smooth out all points that fall within less than some number $k < n$ of rectangles.

We found that optimal results were obtained with $k = 5$, $n = 6$, and rectangles whose width is 0.18 of the distance between the first and last point of each sequence. The values for these parameters, as for all others affecting the behavior of the *Jacal* program, were obtained through a stochastic optimization technique known as *simulated annealing* (see, for example, Aarts & Korst (1989)). This tech-

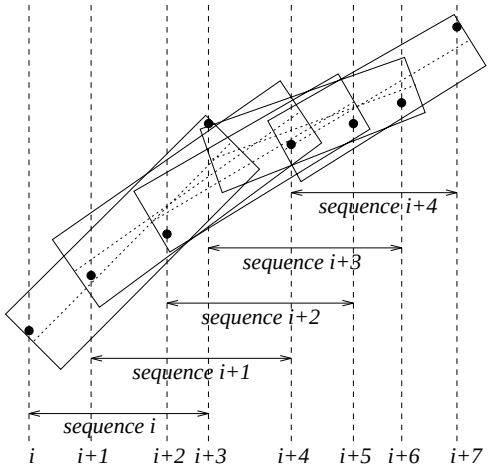

Figure 4. *Jacal* Smoothing Example: For each sequence of n consecutive points (here, $n = 4$), a rectangle is drawn along the linear regression line, whose width is proportional to the distance between the first and last points in the sequence. We then determine which points of the sequence fall within this rectangle. Each point can potentially belong to at most n rectangles. We smooth out those points that belong to less than a certain proportion of this maximum (e.g. 3 out of 4). In this example, all points belong to at least 3 rectangles, except point $i + 3$, which would be smoothed out. (Rectangles corresponding to sequences $< i$ and $> i + 4$ are not drawn in this example.)

nique calls for the alignment program to be run repeatedly on the same texts, each time with a different set of parameters. We start out with random values for all parameters, run the program, then measure the quality of the resulting alignment (more on evaluation in Section 4). We then introduce a small random change on the parameter set, and run the program again. If this improves the results, then we unconditionally accept this new set of parameters. If the results deteriorate, then acceptance is conditional on the output of some random function. As this process is repeated, the behavior of the random acceptance function changes, so as to become increasingly discriminant. The process normally ends when the system has reached some sort of equilibrium, hopefully corresponding to a (possibly local) optimum for the set of parameters.

Of course, *Jacal*'s word-matching criterion is quite strict, and very few word-pairs are actually selected, making the bitext map usually very sparse. But because it is also extremely reliable, we can now repeat the process: successively take as *anchors* each consecutive pair of points already in the map, disregard all surrounding text, and re-apply the method between anchors, i.e. find isolated cognates along the search corridor, and then smooth out rogue points. As the regions of text in question become increasingly small, the size of the isolation window also decreases, and thus new pairs of cognates become eligible. *Jacal* applies this process recursively until no more points can be found.

Once this is accomplished, we have found it useful to apply a final, two-pass smoothing. The first pass is identical to what is done during the recursive search, and eliminates aberrations that sometimes appear when the final result is pieced together. The second pass is based on the simple observation that isolated points in the map, say those that are more than 150 characters away from their closest neighbors, are often wrong, even if they are in line with those neighbors. These points are therefore suppressed.

A straightforward implementation of this algorithm would look like this:

1. Given two texts A and B, of size M and N respectively, Insert the start- and end-points (a_0, b_0) and (a_M, b_N) in the alignment.
2. Match isolated cognates: for each word-token a_i in text A, look for the previous and next resembling tokens, to verify that a_i is isolated. If this is the case, then find the corresponding point in text B (linear interpolation between the start- and end-points), and then look for an occurrence of a cognate b_j around that point. If a cognate is found, and is also isolated, then add point (a_i, b_j) to the alignment.
3. Eliminate alignment points that are out of line by smoothing.
4. If the alignment contains new points, then segment A and B at each point in the alignment, apply steps (1) to (4) on each pair of sub-segments, and merge the resulting alignments.
5. Apply the two-pass final smoothing.

In this straightforward implementation, since we operate within a search corridor whose width is proportional to the size of the texts, the computational cost of verifying if a word is isolated is $\mathcal{O}(N)$, as is that of finding a cognate for a given word (we assume M and N to be in the same order). Therefore, the complexity of step 2 is $\mathcal{O}(N^2)$. Our smoothing method, on the other hand, is a linear-time operation ($\mathcal{O}(N)$). In the worst of cases, the recursive process would cause steps 1 to 4 to be repeated N times on sub-segments of decreasing size, but the average is more like $\log N$. All things considered, the algorithm has a worst-case time complexity of $\mathcal{O}(N^3)$, while space requirement is $\mathcal{O}(N)$.

In our implementation, a preliminary pass is performed, in which we scan both texts in parallel, linking each word token with the previous and next resembling tokens as we go along, and linking tokens of text A with the closest cognate in text B (if such a token exists). This trick allows us to verify the "isolation criterion" from step 2 in constant time, and to find cognates in near constant time. The use of a hash table enables us to carry out the preliminary step in $\mathcal{O}(N)$ time and space. As a result, although the worst-case time complexity of our implementation remains $\mathcal{O}(N^3)$, the average is closer to $\mathcal{O}(N \log N)$. Space requirements remain unchanged at $\mathcal{O}(N)$.

In practice, we also found that it was desirable to fix an upper bound on the sizes of the isolation window and cognate corridor. This means that it would also be possible to process the texts in overlapping segments of texts of constant size,

and the program could run in constant space. So far, we have not found it necessary to venture into this complex optimization exercise.

Although bitext maps as such were not our primary goal, we did conduct a number of experiments to evaluate the robustness of *Jacal* as a stand-alone bitext-mapping program. Essentially, we ran the program on a set of pairs of texts, collectively known as the BAF corpus, for which valid sentence alignments exist (this corpus is described in more detail in Section 4.1). The resulting bitext maps were then evaluated with regard to the reference sentence alignment. Essentially, the following results emerged:

- Between 99.4% and 100% of all points produced by *Jacal* fell within pairs of sentences that were identified as mutual translations in the reference alignment. Since these points always connect pairs of words that satisfy our operational definition of cognate, their chance of being erroneous is statistically insignificant.
- The average distance between points varied between 16 and 43 characters, or from 3 to 9 words. It should be pointed out, however, that many matches did not connect true word forms, but rather items such as numerical expressions and punctuation symbols (mostly periods).
- Most bitext maps produced by *Jacal* contained a few large gaps, i.e. regions of texts of anywhere between 200 and 1000 characters where no matches appeared. These were usually found around "difficult" regions, i.e. areas where omissions or inversions occurred.
- The number of sentences that did not contain a single match varied between 1.3% and 28%, but in most documents, it was closer to 5%, with about 85% containing two or more.

3.2. INTERMEDIATE STEPS: SEGMENTATION AND SEARCH SPACE DETERMINATION

Once we have a partial, but reliable bitext map, the question becomes: How do we get from there to an accurate sentence alignment? Let us recall that our idea is to use the bitext map to constrain the search space for an elaborate sentence-alignment system. This means that we first have to segment the texts into sentences, and then determine which pairs of sentences are potentially part of the correct alignment.

For the time being, we rely on a rather simplistic segmentation method, based almost exclusively on language-independent data. Essentially, the data consists of a set of rules which encode general knowledge about the structure of electronic texts. Some language-specific lists of abbreviations and acronyms are also included, and these are used to determine whether a period following a word belongs to the word itself or serves to end a sentence (ignoring the possibility of a period simultaneously serving both purposes).

To determine the search space for the final alignment (which sentences can be paired, and which ones cannot), we consider a sentence alignment as a special case of a bitext map, i.e. one in which the mapped points must coincide with sentence boundaries. We can assume that the points of the correct sentence alignment will not be too far from the points produced by a program such as *Jacal*. In practice, we have also observed that when the points in *Jacal*'s output are dense, the correct sentence-alignment points are likely to be close by. Conversely, as the points become scarcer, we have to widen our search area.

In practice, we run *Jacal* on the pair of texts, then draw a corridor along each pair of adjacent points in the resulting bitext map. The width of this corridor is proportional to the distance between the two points it connects. We then include all pairs of sentence boundaries that fall within the corridor in the search space. This is illustrated in Figure 5.

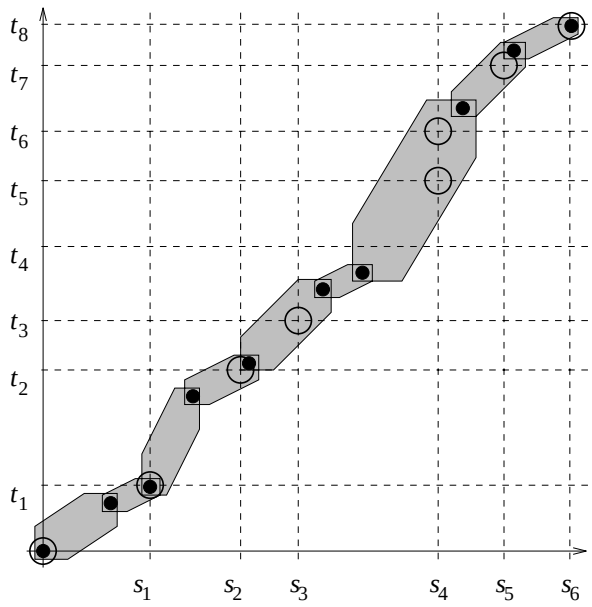

Figure 5. Computing the Search Space from a Bitext Map (Black Dots): Overlapping search regions (corridors) are drawn between each consecutive pair of points (shaded areas). The regions' width is proportional to the distance between the points; only those pairs of sentence boundaries that fall within these regions are included (circled intersections).

The resulting set of points constitutes the search space for the final sentence alignment: it determines exactly those points where the bitext may be segmented. Theoretically, for a pair of texts of sizes N and M, the number of points produced is $\mathcal{O}(NM)$. But since the bitext maps generated by *Jacal* are generally quite dense, for a large number of sentence boundaries of one text, there is only one possible match in the other. As a result, on average, both the computation time for this step and the size of the search space produced will be linear in the sizes of the texts.

3.3. SECOND STEP: FINAL SENTENCE ALIGNMENT

From this point on, any sentence-alignment program that is capable of working within such a restricted search space can be used to finish up the job. Following the ideas of Chen (1993) and Dagan et al. (1993), we have developed a method that is based on a statistical lexical translation model, namely Brown et al.'s (1993) Model 1. This model incorporates a set of parameters $\mathbf{tr}(x|y)$, that estimate $\mathrm{P}(x|y)$, the probability of observing some word x in a text of language L_T, given that some other word y appears in the corresponding text of language L_S.

The parameters of the model can be combined so as to estimate the probability of observing some sequence of words in one language, given a sequence in the other language. Suppose we have a pair of sequences $s_1 \ldots s_n$ and $t_1 \ldots t_m$, then

$$\mathrm{P}(t_1 \ldots t_m | s_1 \ldots s_n) = \mathrm{P}(m|n) \prod_{j=1}^{m} \sum_{i=1}^{n} \frac{\mathbf{tr}(t_j|s_i)}{n} \tag{1}$$

where $\mathrm{P}(m|n)$ is the probability of observing a string of size m as the translation of a string of size n.

Our sentence-alignment program, called *Salign*, formulates the problem of finding the best alignment for some pair of texts S and T as that of finding the most probable one. This can be expressed as in (2).

$$\begin{aligned} A' &= \mathrm{argmax}_A \, \mathrm{P}(A|S, T) \\ &= \mathrm{argmax}_A \, \mathrm{P}(T, S|A)\mathrm{P}(A) \\ &\approx \mathrm{argmax}_A \left[\textstyle\prod_{a \in A} \mathrm{P}(t_a, s_a)\right] \mathrm{P}(A) \end{aligned} \tag{2}$$

where t_a and s_a are the segments of T and S put into correspondence by the component a of alignment A, under the assumption that all such correspondences are independent. In turn, $\mathrm{P}(t_a, s_a)$ can be calculated as the product of $\mathrm{P}(t_a|s_a)$ and $\mathrm{P}(s_a)$. In practice, we found that we could assume the *a priori* probability of the source sentences $\mathrm{P}(s_a)$ and of the alignments $\mathrm{P}(A)$ to be constant, and still obtain good results (3).

$$A' = \mathrm{argmax}_A \prod_{a \in A} \mathrm{P}(t_a|s_a) \tag{3}$$

By using the $\mathbf{tr}(x|y)$ parameters in (1) to estimate $\mathrm{P}(t_a|s_a)$ in (3), we obtain a general method for finding an optimal alignment, based on Brown et al.'s Model 1.

Theoretically, the number of alignments that we need to examine in order to find the most probable one is exponential in the sizes of the texts. However, dynamic programming allows us to carry out this task in polynomial time. This is achieved by calculating optimal partial alignments, covering prefixes of the texts of increasing size. Individual alignment segments, consisting of pairs of word sequences

$s_i \ldots s_{I-1}$ and $t_j \ldots t_{J-1}$, are scored using a function W, whose value corresponds to minus the logarithm of $\mathrm{P}(t_j \ldots t_{J-1} | s_i \ldots s_{I-1})$ (4).

$$\mathrm{W}(i, I, j, J) = \delta(I - i, J - j) + \sum_{k=j}^{J-1} - \log \frac{\sum_{l=i}^{I-1} \mathbf{tr}(t_k | s_l)}{I - 1} \tag{4}$$

where $\delta(n, m)$ is an approximation of $-\log \mathrm{P}(m|n)$ (ignoring, for the moment, how this is calculated).

We define $\mathrm{W}_A(I, J)$ to be the best alignment over the pair of text prefixes $s_0 \ldots s_I$ and $t_0 \ldots t_J$. As a result of the mathematical properties of scoring function W, this can be calculated as in (5).

$$\mathrm{W}_A(I, J) = \min_{i<I, j<J} [\mathrm{W}_A(i, j) + \mathrm{W}(i, I, j, J)] \tag{5}$$

Suppose we are given a pair of texts S and T, of sizes N and M respectively. By computing $\mathrm{W}_A(I, J)$ for all values of I and J, we eventually arrive at the value of $\mathrm{W}_A(N, M)$, which corresponds to the probability of the best overall alignment for S and T. Of course, it is not the probability of the best alignment that interests us, but rather the alignment itself: this can easily be obtained by storing those values of i and j that produce the minimum $\mathrm{W}_A(I, J)$ at each step in the computation.

In theory, the computational complexity of this algorithm is quite steep, at $\mathcal{O}(N^3M^3)$, while space complexity is $\mathcal{O}(NM)$. It is possible, however, to reduce the time complexity to $\mathcal{O}(N^2M)$, without increasing the space complexity of the algorithm, if we assume $\mathrm{P}(m|n)$ in (1) to be the same for all values of m and n. In practice, it would appear that this can be done without incurring too great a loss of accuracy. Interestingly, this is the opposite of what Gale & Church (1991) and others have done in disregarding lexical content and taking only length ratios into consideration for alignment. The mathematical details of this optimization are presented in the Appendix.

It is interesting to note that for *Salign*, a prior division of the texts into sentences is not necessary. In fact, unless explicitly instructed to restrict itself to segmenting the texts at sentence boundaries, the program will usually produce an alignment that establishes correspondences between much smaller segments, typically sequences of one to three words.

Of course, when aligning sentences, execution times and space requirements will go down significantly, because the program can restrict itself to computing $\mathrm{W}_A(I, J)$ only for values of I and J that correspond to valid sentence boundaries (although the theoretical complexity remains the same). Furthermore, when dealing with a restricted search space such as that produced from the output of the *Jacal* program, the number of values of $\mathrm{W}_A(I, J)$ that must be calculated is reduced to exactly the size of the search space. This means that in most cases, the complexity of the sentence alignment will go down to $\mathcal{O}(NM)$.

One of the underlying assumptions in this complex optimization exercise is that the savings that result will make up for the fact that we first intend to run *Jacal* on the texts. After all, the worst-case computational time complexity of *Jacal* is comparable to that of *Salign*. However, computational time and space are not our only concern here, and as we will see in the next section, gains can also be realized with regard to the quality of the alignments obtained.

Most alignment methods that make use of lexical information assume this information not to be available *a priori*: Kay & Röscheisen (1993), Fung & McKeown (1994), Chen (1993), Dagan et al. (1993) all go to great lengths to infer the parameters of their models directly from the pair of texts to align. This is a very interesting approach, especially when dealing with many language pairs. But for language pairs such as English and French, for which large quantities of aligned bitext already exist, it seems a little bit like re-inventing the wheel every time. Furthermore, with such methods, aligning short texts can become a problem.

Salign normally assumes the existence of a trained translation model. In fact, its implementation allows it to deal with large models, covering vocabularies of tens of thousands of word forms. For example, the model we used was trained on approximately three years of Hansard proceedings, sentence-aligned with the Gale & Church (1991) alignment program. However, nothing in the method precludes a bootstrapping approach, which could, for instance, be implemented using the initial bitext map.

4. Evaluation

We have produced an implementation of the sentence-alignment method described in the previous section. In order to assess its performance, we needed two things: first, a corpus of text for which a *reference sentence alignment* exists, i.e. an alignment reputed to be "correct"; second, some way of measuring how the output of our program differed from the reference. We feel that this last aspect has been somewhat neglected in previous work, which makes it very hard to compare methods, or simply to know what to expect from a given program.

4.1. TEST CORPUS

The corpus we used is the BAF corpus (Simard, 1997): this is a collection of French–English bitexts, hand-aligned to the sentence level. The corpus consists of a dozen pairs of text files, totaling a little over 400, 000 words in each language. Most of the texts are of an institutional nature (Hansards, UN reports, etc.), but the corpus also contains scientific, technical and literary material.[1]

[1] One of the BAF bitexts, a technical manual, had to be discarded from the corpus for the tests, because it contained in an appendix a relatively large glossary of terms, sorted alphabetically. Since the order of the entries was entirely different in French and English, no attempt was made to hand-align this glossary. Therefore, the reference alignment contained one very large pair of segments. The

All documents of the corpus were split into two more or less equal parts. The first halves (the training corpus) were used to optimize the various parameters of the program, while the second halves (the test corpus) were kept for evaluation purposes.

4.2. EVALUATION METRIC

Performance was measured using a method based on a metric proposed by Isabelle (personal communication): Consider two texts, S and T, viewed as unordered sets of sentences: $S = \{s_1, s_2, \ldots, s_n\}$ and $T = \{t_1, t_2, \ldots, t_m\}$. An alignment A may be represented as a subset of $S \times T$ (6)

$$A = \{(s_1, t_1), (s_2, t_2), (s_2, t_3), \ldots, (s_n, t_m)\} \tag{6}$$

with the interpretation that $(s_i, t_j) \in A$ iff s_i and t_j share a common clause (in (6), the fact that s_2 appears in two couples simply means that s_2 is translated partly by t_2 and partly by t_3). What we need is a way of measuring the difference between some alignment A and a correct reference alignment A_R. Borrowing from the terminology of Information Retrieval (IR), we define alignment recall $\mathrm{R}(A, A_R)$ (7) and alignment precision $\mathrm{P}(A, A_R)$ (8).

$$\mathrm{R}(A, A_R) = \frac{|A_R \cap A|}{|A_R|} \tag{7}$$

$$\mathrm{P}(A, A_R) = \frac{|A_R \cap A|}{|A|} \tag{8}$$

As in IR, recall measures how much of the correct alignment A_R is found in alignment A, and precision measures how much of alignment A is correct, i.e. what proportion also appears in the reference A_R.

In IR applications, the relevance of a given document with regard to a query depends on the informative content of the document, something that is sometimes difficult to quantify. In the case of alignments, what interests us is usually not the informative content of the aligned texts, but rather their linguistic content. Since large segments of text, on average, are richer in lexical content than small segments, we argue that the relative importance of a certain correspondence within an alignment is in direct relation to the sizes of the segments of texts that it connects. For this reason, we use a variant of the evaluation metrics proposed in (7)–(8), which considers correspondences at the level of characters rather than whole sentences. Texts are then viewed as sets of character tokens rather than sets of sentences, and alignments are sets of pairs of character tokens.

---

presence of this region distorted the performance measures to the point where they were no longer significant.

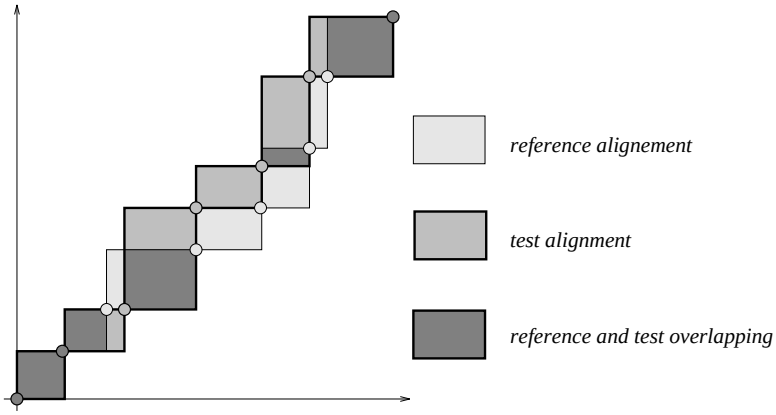

Figure 6. Graphical Interpretation of Alignment Recall and Precision: *Recall* is measured as the area of the overlapping region over the area of the reference alignment; *precision* is the area of the overlapping region over the area of the test alignment.

When alignment recall and precision are formulated in these terms, there is also a graphical interpretation to these notions: if the characters of the texts are laid along the x and y axes in the plane, and if we draw dots in the plane for each pair of characters in an alignment, then pairs of aligned sentences will appear as rectangular regions in the plane. Alignments can then be compared by examining how these regions overlap. Recall and precision denote the relative importance of the areas of overlapping and non-overlapping regions (Figure 6).

4.3. EXPERIMENTS AND RESULTS

In order to understand better the effect of each component of our program, we designed a number of experiments, the results of which are summarized below.

First, as points of comparison, the test corpus was submitted to the Gale–Church (GC) and Simard et al. (SFI) sentence-alignment programs. The same texts were then submitted to a combination of *Jacal* and Gale–Church (J+GC), in which GC operates on the reduced search space produced by *Jacal*, instead of *Salign*, to produce the final sentence alignment. The aim of this first experiment was to evaluate how using a *Jacal* bitext map to reduce the search space for a length-based sentence-alignment program could improve its robustness. The comparison with SFI is also interesting, because in a sense, SFI and J+GC represent opposite strategies: while the former method tries to improve length-based alignments using cognates, the latter combination proceeds the other way around.

As can be seen in Table I, using the output of *Jacal* to guide a length-based alignment technique generally improves alignment recall. In some cases the improvement is minor, but there are situations where this makes the difference between an alignment that is literally beyond repair, and one that is acceptable. For example, in the first scientific article, both the GC and SFI programs com-

Table I. Alignment Results for Various Programs on Individual Texts of the BAF Corpus. Recall and precision values are given as percentages; average values are weighted by text size.

Document	Text Size (words)			GC	SFI	J+GC	*Salign*	*Jacal + Salign*
	English	French						
Hansard	27654	29511	recall:	97.36	97.39	97.42	99.06	99.06
			precision:	98.77	99.23	97.54	98.30	98.30
UN Sec.-Gen.	24536	27850	recall:	96.72	96.79	97.03	97.23	97.23
Annual Report			precision	99.00	99.08	98.47	98.35	98.35
UN ILO Report	74414	75523	recall:	90.12	90.14	93.06	92.70	92.70
			precision	96.45	96.54	97.64	98.46	98.46
Court transcripts	15741	16508	recall:	92.15	92.40	90.26	92.94	92.94
			precision	97.46	97.30	94.88	95.81	95.81
Scientific articles	6344	6299	recall:	5.42	10.40	88.07	89.47	89.39
			precision	6.91	15.59	81.55	83.47	83.97
	9527	10397	recall:	83.05	83.62	84.45	85.25	85.25
			precision	97.86	98.55	96.81	96.82	96.82
	3293	3858	recall:	95.95	95.65	95.50	96.42	96.42
			precision	89.61	89.60	87.08	84.44	84.44
	3556	3551	recall:	86.52	91.94	94.44	93.19	93.19
			precision	90.62	97.01	97.01	96.97	96.97
	1667	1687	recall:	95.44	95.15	96.29	98.29	98.29
			precision	98.81	98.79	92.28	94.04	94.04
Literary	19597	26725	recall:	36.95	39.97	72.19	87.70	93.99
			precision	34.42	39.06	57.66	46.09	54.49
Total / Average	186329	201909	recall:	82.84	83.52	90.99	93.20	93.92
			precision	86.89	87.98	92.09	91.28	92.26

pletely lose track of the alignment, right at the paragraph level (both programs first align paragraphs, then sentences within paragraphs). In this case, *Jacal* obviously manages to keep track of the alignment, which in turn makes the task much easier for GC in the second step. A similar thing seems to happen with the literary text.

The situation with alignment precision is not as clear, however: although J+GC is more precise than SFI on average, the opposite appears to be true when the harder texts are discarded. One possible explanation lies in the fact that, because GC was operating on a very narrow search space, we allowed it to perform a deeper search, i.e. to consider more matching possibilities. Gale & Church originally restricted their program to matches connecting sentences 1:1, 1:0, 0:1, 1:2, 2:1 or 2:2. We allowed the program to consider all combinations, up to 6:6. This may have encouraged the program to play safe in some situations, by producing larger pairs of segments, therefore decreasing alignment precision.

In the next experiment, we ran the *Salign* program on the test corpus, with the intention of seeing how this program would do on its own (in this setup, *Salign* operated on a fixed-width window along the diagonal that joins the beginnings and

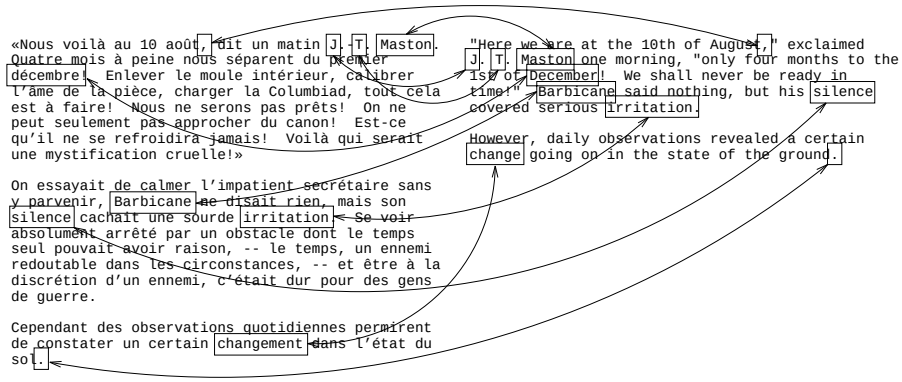

Figure 7. A Portion of Jules Verne's *De la terre à la lune*, along with Mappings Produced by the *Jacal* Program.

ends of the two texts). Here again, as far as alignment recall is concerned, the result is a general improvement over all other programs. As for alignment precision, it is approximately the same as for J+GC.

Finally, we ran *Jacal* and *Salign* together on the test corpus. Interestingly, the results are exactly the same as those obtained with *Salign* alone, except for the pair of literary texts, where using *Jacal* to guide the search significantly improved both recall and precision. This pair of texts (Jules Verne's *De la terre à la lune*) is particularly interesting, because it shows how a translation can sometimes diverge radically from the original. In this case, the translation is about 25% shorter than the source text, and in fact, it is not even clear whether the English version is indeed a translation of the French, or if it was based on an abridged version. Figure 7 shows a pair of matching segments that is quite typical of the kind of translations that can be found in this pair of texts; connections found by *Jacal* on these segments are also shown.

5. Conclusions

We have described our attempt to develop a method for aligning sentences that is both robust and accurate. Both the *Jacal* and *Salign* methods have been implemented. We recently used the J+GC combination to align eight years' worth of Hansard proceedings (approximately 70 million words in all) for our translation memory application. The whole process took about four hours on a Sun Sparc Ultra I. In theory, using *Salign* instead of the Gale–Church program would not have been much more costly in time. What prevented us from doing so was the lack of a clean and robust integration of the two programs, something that we obviously want to work on in the near future.

Our programs were tested on a hand-aligned corpus, and their performance was measured in terms of alignment recall and precision. The sentence-alignment methods of Gale & Church (1991) and Simard et al. (1992) were also submitted to

these tests, as points of comparison. It would certainly be interesting to extend the test to other, more recent methods as well.

As far as robustness is concerned, the test results are quite encouraging: our method was able to align all texts of the BAF corpus satisfactorily, even the more difficult ones. Accuracy, however, would appear to remain a problem. In fact, one thing that may come as a surprise is how poor the overall results are, regardless of which program is used. Performance levels below 95% are not exactly what the literature on the subject had us used to. It could be the case that this is just a consequence of our choice of performance metric. On the other hand, the figures obtained seem to confirm one of our earlier claims: that the Hansards are exceptionally easy to align when compared to other text genres. In fact, of all the texts in the BAF corpus, the Hansards are the only ones for which we managed to obtain recall and precision levels above 98%.

The low precision levels obtained with our methods can in part be linked with one of the current shortcomings of *Salign*: the program is unable to account for omissions or additions in a translation. As a result, text segments that do not have an equivalent in the other text are absorbed by neighboring segments in the alignment, thus reducing precision. This is due to the fact that the translation model on which the program is based was originally intended to be used for automatic translation, and originates from a "noisy channel" model, which is inherently directional. As Brown et al. (1993) conceive of the problem, the task of translating from French to English entails recuperating a source (English) signal, given a garbled (French) input. While this noisy-channel model has certainly proven its worth in voice recognition, it is not clear whether it is the most appropriate formulation for the bitext-correspondence problem. There have been some attempts to modify the model, for example, by making it symmetrical (Chen, 1993). We are currently investigating various solutions to this problem.

Another point that came to our attention while examining the alignment errors is that many of them are actually the result of an incorrect segmentation of the texts into sentences. In some cases, this can have a dramatic effect on recall and precision measures: typically, over-segmentation reduces alignment recall, while under-segmentation reduces alignment precision. It should be noted, however, that such errors are not necessarily catastrophic. From our experience, alignment errors resulting from over-segmentation usually separate unrelated portions of sentences, while under-segmentation simply results in the information becoming diluted rather than in genuine misalignments.

It is interesting to note that the problem of segmentation is a very general one, with implications not only for bitext alignment, but for many natural-language applications. Curiously, the NLP community has only recently begun to address it seriously. In our group, we are currently exploring the use of segmentation techniques like those employed by Palmer & Hearst (1994) for disambiguating periods. It should be noted, however, that ambiguous periods are only one aspect of the problem. In fact, it turns out that many of our segmentation errors come from

sentence-like units that do not end with a period, or any punctuation mark for that matter: titles, section headings, list and table items, etc. In the end, it seems that proper segmentation is no less difficult than natural-language understanding itself. In the near future, however, the increasing availability of electronically marked-up text (HTML, TEI, etc.) is likely to change the nature of the problem in an interesting way.

Acknowledgements

We would like to thank George Foster, Marie-Louise Hannan, Pierre Isabelle, Elliott Macklovitch, Dan Melamed, as well as three anonymous reviewers, for inspirational discussions and constructive comments. All trademarks are hereby acknowledged.

Appendix

Optimizations in the *Salign* Algorithm

To lower the computational complexity of the *Salign* search algorithm, it is necessary to reformulate the computation of the $\mathrm{W}_A(I, J)$, as presented in equation (5) in Section 3.3. This reformulation requires that we assume $\mathrm{P}(m|n)$, the probability of observing a string of size m as the translation of a string of size n, to be the same for all values of m and n. Then, the value of $\delta(I - i, J - j)$ becomes constant in the computation of the scoring function $\mathrm{W}(i, I, j, J)$, and the relation (9) holds.

$$W(i, I, j, J) = W(i, I, j, k) + W(i, I, k, J), \qquad j < k < J \tag{9}$$

We define $\mathrm{W}_p(I, J, i)$ to be the score of the best alignment over the pair of text prefixes $s_0 \ldots s_I$ and $t_0 \ldots t_J$, whose last correspondence covers segments $s_i \ldots s_{I-1}$ (10) (see Figure 8).

$$W_p(I, J, i) = \min_{0 \le j < J} [W_A(i, j) + W(i, I, j, J)] \tag{10}$$

It is then possible to express the value of $\mathrm{W}_A(I, J)$ in terms of $\mathrm{W}_p(I, J, i)$ (11).

$$W_A(I, J) = \min_{0 \le i < I} W_p(I, J, i) \tag{11}$$

What makes this intermediate notation interesting is that the computation of each value of $\mathrm{W}_p(I, J, i)$ can be greatly simplified (12).

$$\begin{aligned}
\mathrm{W}_p(I, J, i) &= \min_{0 \le j < J} (\mathrm{W}_A(i, j) + \mathrm{W}(i, I, j, J)) \\
&= \min(\mathrm{W}_A(i, J-1) + \mathrm{W}(i, I, J-1, J), \\
&\qquad \min_{0 \le j < J-1} (\mathrm{W}_A(i, j) + \mathrm{W}(i, I, j, J))) \\
&= \min(\mathrm{W}_A(i, J-1) + \mathrm{W}(i, I, J-1, J), \\
&\qquad \min_{0 \le j < J-1} (\mathrm{W}_A(i, j) + \mathrm{W}(i, I, j, J-1) \\
&\qquad\qquad + \mathrm{W}(i, I, J-1, J)))
\end{aligned}$$

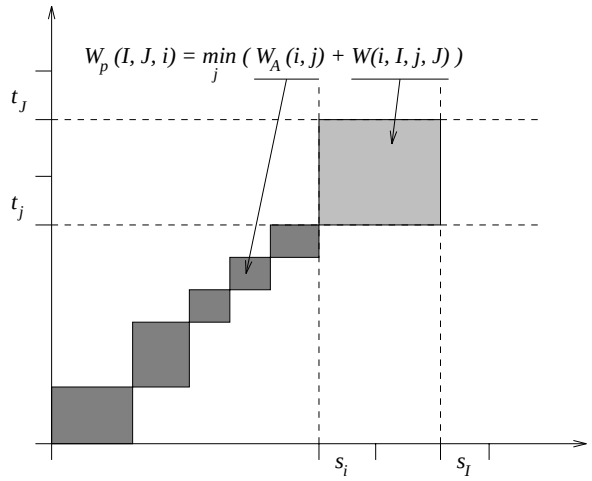

Figure 8. Computation of $\mathrm{W}_p(I, J, i)$: Find the value of j that produces the best alignment over $s_0 \ldots s_{I-1}$ and $t_0 \ldots t_{J-1}$, containing a single correspondence that covers text segment $s_i \ldots s_{I-1}$.

$$
\begin{aligned}
&= \mathrm{W}(i, I, J-1, J) + \\
&\quad \min(\mathrm{W}_A(i, J-1), \\
&\qquad \min_{0 \le j < J-1} (\mathrm{W}_A(i, j) \\
&\qquad\qquad + \mathrm{W}(i, I, j, J-1))) \\
&= \mathrm{W}(i, I, J-1, J) \\
&\quad + \min(\mathrm{W}_A(i, J-1), \mathrm{W}_p(I, J-1, i))
\end{aligned} \tag{12}
$$

Salign alternately computes rows of values of $\mathrm{W}_A(I, J)$ and $\mathrm{W}_p(I, J, i)$, and stores them in a vector. Computing each value of $\mathrm{W}_A(I, J)$ then requires $\mathcal{O}(N)$ time. Computing individual values of $\mathrm{W}_p(I, J, i)$ would normally require $\mathcal{O}(N^2)$ time, but this can be cut down to $\mathcal{O}(N)$, if values of $\sum_{l=i}^{I-1} \mathbf{tr}(t_k|s_l)$ are pre-computed and stored, for each possible pair of values of i and I. This pre-computation step requires $\mathcal{O}(N^2)$ time and space.

All things considered, the program must compute $\mathcal{O}(NM)$ values of $\mathrm{W}_A(I, J)$ and $\mathrm{W}_p(I, J, i)$, so the overall computational complexity of the algorithm is $\mathcal{O}(N^2M)$, while space complexity is $\mathcal{O}(N^2)$.

References

Aarts, E. and J. Korst: 1989, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley and Sons, Chichester, England.

Brown, P.F., J.C. Lai, and R.L. Mercer: 1991, 'Aligning Sentences in Parallel Corpora', in *29th Annual Meeting of the Association for Computational Linguistics*, Berkeley, Calif., pp. 89–94.

Brown, P.F., S.A. Della Pietra, V.J. Della Pietra, and R.L. Mercer: 1993, 'The Mathematics of Machine Translation: Parameter Estimation', *Computational Linguistics* **19**, 263–311.

Catizone, R., G. Russell, and S. Warwick: 1989, 'Deriving Translation Data from Bilingual Texts', in *Proceedings of the 1st International Lexical Acquisition Workshop*, Detroit, MI, pp. 15–21.

Chen, S.F.: 1993, 'Aligning Sentences in Bilingual Corpora Using Lexical Information', in *31st Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio, pp. 9–16.

Church, K.W.: 1993, '*Char_align*: A Program for Aligning Parallel Texts at the Character Level', in *31st Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio, pp. 1–8.

Dagan, I., K.W. Church, and W.A. Gale: 1993, 'Robust Bilingual Word Alignment for Machine Aided Translation', in *Proceedings of the Workshop on Very Large Corpora: Academic and Industrial Perspectives*, Columbus, Ohio, pp. 164–171.

Fung, P. and K.R. McKeown: 1994, 'Aligning Noisy Parallel Corpora Across Language Groups: Word Pair Feature by Dynamic Time Warping', in *Technology Partnerships for Crossing the Language Barrier: Proceedings of the First Conference of the Association for Machine Translation in the Americas*, Columbia, Maryland, pp. 81–89.

Gale, W.A. and K.W. Church: 1991, 'A Program for Aligning Sentences in Bilingual Corpora', in *29th Annual Meeting of the Association for Computational Linguistics*, Berkeley, Calif., pp. 177–183.

Isabelle, P., M. Dymetman, G. Foster, J-M. Jutras, E. Macklovitch, F. Perrault, X. Ren. and M. Simard: 1993, 'Translation Analysis and Translation Automation', in *Proceedings of Fifth International Conference on Theoretical and Methodological Issues in Machine Translation TMI '93: MT in the Next Generation*, Kyoto, Japan, pp. 15–22.

Kay, M. and M. Röscheisen: 1993, 'Text-Translation Alignment', *Computational Linguistics* **19**, 121–142.

Klavans, J. and E. Tzoukermann: 1995, 'Combining Corpus and Machine-Readable Dictionary Data for Building Bilingual Lexicons', *Machine Translation* **10**, 59–75.

Melamed, I.D.: 1996, 'A Geometric Approach to Mapping Bitext Correspondence', in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Philadelphia, PA, pp. 93–101.

Palmer, D.D. and M.A. Hearst: 1994, *Adaptive Sentence Boundary Disambiguation*, Report No. UCB/CSD 94/797, Computer Science Division (EECS), University of California, Berkeley, CA.

Simard, M.: 1997, *BAF : un corpus de bi-texte anglais-français annoté à la main*, `http://www-rali.iro.umontreal.ca/arc-a2/BAF`.

Simard, M., G. Foster, and P. Isabelle: 1992, 'Using Cognates to Align Sentences in Bilingual Corpora', in *Quatrième colloque international sur les aspects théoriques et méthodologiques de la traduction automatique, Fourth International Conference on Theoretical and Methodological Issues in Machine Translation: Méthodes empiristes versus méthodes rationalistes en TA, Empiricist vs. Rationalist Methods in MT, TMI-92*, Montréal, Canada, pp. 12–20.

Simard, M., G. Foster, and F. Perrault: 1993, *TransSearch: un concordancier bilingue*, CITI Technical Report, Montréal, Canada.