POVERTY DRIVEN BILINGUAL ALIGNMENT

KIM GERDES

1. Introduction

The sheer unlimited usefulness of aligned bilingual corpora in all areas of translation sciences from theoretical corpus work to dictionary development or machine translation cannot be overstated. Unfortunately, however, many translation researchers end up aligning large parts of their parallel corpora manually, lacking tools with basic heuristics to simplify this task.

It is well known that cognate alignment paired with bilingual dictionaries can give astonishingly good results and constitute the state of the art of current bilingual alignment algorithms. However, most of these systems are out of reach for a common translation researcher, because the parameterization requires insight in the underlying statistics and, even more obstructive, the systems require adaptation of the dictionaries – dictionaries that are often expensive or inexistent for the (sub)language pair.

In order to explore the possibilities of aligning parallel corpora without any resources, we have to ask the following questions:

Is there any "visible" feature shared by any piece of written text and its translation? The common meaning of the two texts is not easily accessible, and the great variety of syntactic structures and writing systems makes any further affirmation very difficult. However, the discrete and linear nature of all languages gives us a basic access point to alignment.

Words exist. This means that language has chunks of indissociable segments while words have a unique written representation or a finite set of representations.[1] Moreover, these forms that correspond to one word (allomorphs) are often graphically similar.

Of course, every pair of a text with its translation (a bitext) has some untranslated words or words that are translated by complex constructions, distributed over different words. However, we assume that even in very distant pairs of languages most words are translated by words or contiguous sets of words. But among these words with a linearly constraint translation, even non-ambiguous words often constitute translation ambiguities (because the target language forces us to be more or less specific). We can postulate, however, that in every sufficiently long text we find words (or groups of graphically similar words) that have an "easy" translation in the sense that they correspond to a unique word (or a group of graphically similar words) in the translation (see Figure 12-1). The central hypothesis of the present study is that these words occur at similar linear positions in both texts.

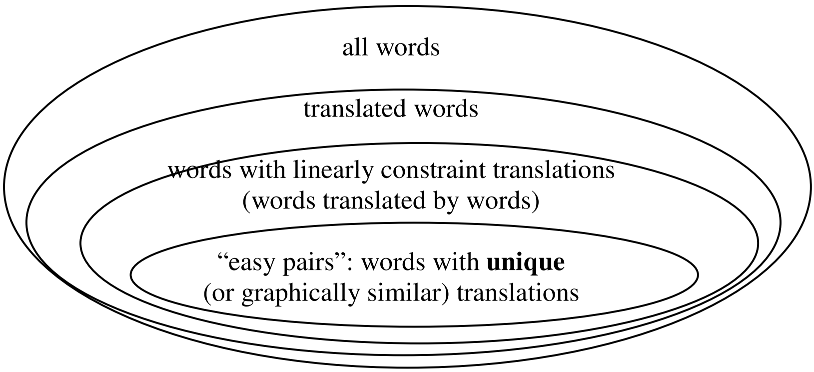

Figure 12-1. Underlying hypothesis: "easy pairs"

We can consider the positions of occurrences of forms in a text as a signal. And "easy pairs" of words will have similar signals.

We will thus attempt to detect these "easy" pairs by similarity measures on all (reasonable) candidates. This chapter presents how this can be done, how to improve some known algorithms of word distance computation in order to include grouped signals of allomorphs and finally how to use these couples as anchor points for the paragraph alignment. The system presented here allows aligning any bitext on the paragraph level without any linguistic parameterization and in particular without any linguistic resources. The final version of the system will be accessible online and thus needs no installation on the user's machine, which gives all users of bitexts easy access to alignment, even without any knowledge in computer programming.

2. Approaches to alignment

2.1. Other approaches

Most alignment systems are based on some kind of graphic similarity between the source and target texts. The most common approach to alignment is based on cognates (lexical or punctuational) (see Simard *et al.* 1992). The basic idea is the exploitation of graphic similarities between a word and its translation. Proper nouns but also many words of Greco-Latin origin have similar graphic forms in many European languages. As an example, the English word *chair* corresponds to the French word *chaise*, a pair which has sufficiently similar forms to be recognized as cognates. The English-Chinese translation pair *chair* – 椅, however, cannot be detected in this way. Cognate-based systems have achieved a quite high reliability rate for most studied (European) languages although most works are highly specific to an application and a language pair, because cognate distances differ among European language pairs and the best definition of the underlying metric remains a subject of debate (see for example Ribeiro *et al.* 2001).

It is clearly more difficult to extend this idea to cognates in language pairs with different writing systems. But even for languages like Russian and Japanese, this approach remains interesting as has been shown by Knight and Graehl (1998), because the transcription rules (for example of katakanas in Japanese) are quite simple, although specific metrics for the computation of word distances are needed.[2] Chinese language, on the contrary, does not have a simple phonetic transcription system. For each word of foreign origin the translator has the choice of a multitude of homophone characters that transcribe the foreign sounds in a satisfying manner. The choice is then often based on "beauty" or semantic appropriateness of the characters. In order to obtain a certain degree of coherence among different translations, the Chinese translator uses enormous specialized transcription dictionaries (for example Zhou 2003).

Thus, finding “similar” words in most language pairs requires considerable linguistic resources, while simple cognate alignment remains a privilege of the European languages with their closeness of vocabulary and their uniformity of the writing system.

First attempts to align bilingual corpora without recourse to linguistic resources have been made by Brown *et al.* (1991), Gale and Church (1991), and Kay and Röscheisen (1993). All three aim at an alignment on a sentence level and work on technical texts or particularly literal translations (Hansards). The first two are based on the closeness of the length of forms (words and sentences), while the latter, closer to our approach, describes a dynamic programming algorithm that makes hypotheses based on the overall frequency of words and enhances dynamically these hypotheses by taking into account the possible alignments of the sentences containing these words.

The hypothesis of word length similarity, confirmed for example by the English-French pair, is dubious already for pairs like German-French, because German compound nouns usually have a “noun *de* noun” translation in French.

Sentence length, too, depends on the syntactic structure of the languages; and for distant languages, we can expect to find greater difference in sentence length. Moreover, punctuation symbols vary across languages. For example the full stop indicating the end of a sentence is often represented by a small circle in Asian languages; and even if the symbols are graphically identical, they are often listed in the Unicode tables with the language they are used in, creating completely different objects from the computer’s point of view.

The segmentation of texts into paragraphs seems to be the only common point between practically all modern texts. The new line character is thus the only “universal” cognate.

In this chapter we will attempt an alignment only at the paragraph level, and our approach is thus less ambitious than most approaches to alignment. Note however, that practically all current approaches are “tweaked” for a specific language pair and they do not aspire to any universality. Moreover the set of language pairs is very limited (mainly English and a major European language, Chinese, or Japanese).

Paragraphs constitute the next step after the alignment of chapters or sections. It seems reasonable to assume that paragraph boundaries are more often respected in the translation process than sentence boundaries, because paragraphs correspond to semantic units, whereas sentences constitute syntactic units. However, the sentence alignment can be done in a subsequent step, the task being considerably easier once paragraphs are aligned. Indeed, some sentence alignment approaches are even based on a previous paragraph alignment, which is often achieved by hand or semi-automatically (e.g. Lebart and Salem 1994, Zimina 2000).

We put one further limitation on our goal: We just try to find the best alignment of paragraph boundaries, i.e. no paragraph will remain orphan. We can thus obtain any combination of paragraph numbers being aligned. Again, finding an untranslated paragraph or inversely, the translator's insertion can be done once the best paragraph alignment is achieved.

2.2. Paragraph alignment by length

Even though paragraphs constitute semantic units, a naïve algorithm that simply aligns the first paragraph of the source language with the first paragraph of the target language and so on will not work well as long as the paragraph correspondence is not one-to-one and it is natural to want to take into account the length of paragraphs.

The first approach to paragraph alignment of a text and its translation, which will be the basis of our method, consists in finding the best alignment of the paragraph marks based simply on the length of each paragraph. The underlying idea is that aligned paragraphs should have approximately the same length. This length can neither be taken to be the number of words as the segmentation of the text in words is not always readily available, nor the number of characters, as the length in characters varies markedly between languages (see Figure 12-2). We have to take into account the relative position of paragraph marks as a fraction of the whole text. In other words, we must normalize the text length. Each paragraph position is thus taken to be a percentage of the whole text. We will now show how to find the best pairings of these percentage points.

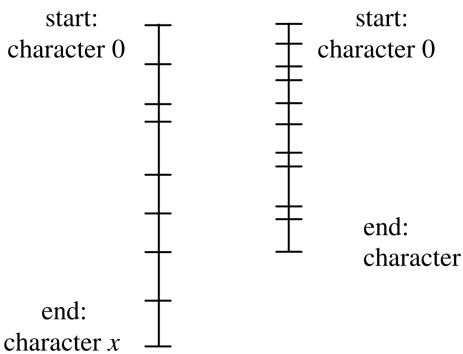

Figure 12-2. Paragraphs as marks relative to the number of characters of the text

In Figure 12-3, we indicate the proceeding graphically. An arrow goes from each paragraph mark in the source language to its closest correspondent in the target language and vice versa. We only take into account the bidirectional arrows, i.e. those arrows that correspond to a pairing of paragraph marks that are mutually their closest homologue. It is possible to obtain non-trivial pairings in this way as the multi-correspondence 2-3 indicated with curly brackets in the Figure 12-3.

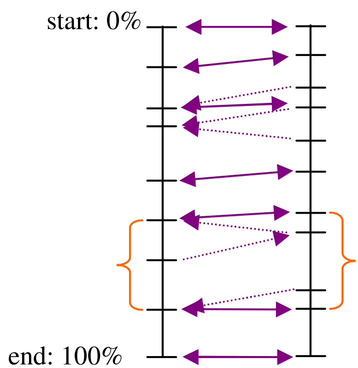

Figure 12-3. Positional alignment

From a computational point of view, this is a standard dynamic algorithm searching for the shortest path in a lattice diagram. We look for the closest path to the diagonal (i.e. the thin line in Figure 12-4) that passes through all paragraph marks of both sides if the corresponding point is a local maximum in the sense that we cannot find a horizontal or vertical neighbour point that is closer to the diagonal.

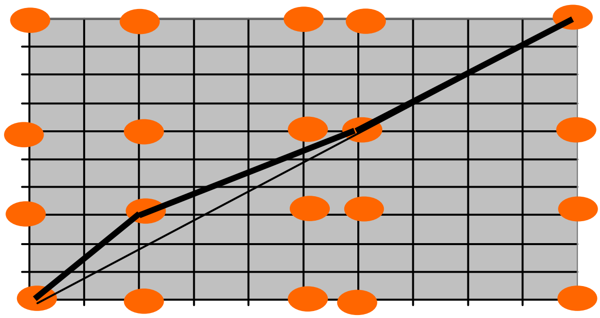

Figure 12-4. Trellis for the alignment of (0, 2, 5, 6, 10) and (0, 3, 6, 10)

This shortest path is shown as the thick line in Figure 12-4, aligning the marks (0, 2, 5, 6, 10) (horizontal) and (0, 3, 6, 10) (vertical).[3] The path starts with the alignment of the beginning of the two texts [0-0].[4] Then we obtain a two-to-one correspondence: [2, 5-3] and finally we obtain the [6-6] alignment (and the obligatory final alignment [10-10] that corresponds to no paragraph). In Figure 12-5, showing the alignment of (0, 1, 9, 10) and (0, 6, 10), no points but the start and end points are local maxima on our lattice and we obtain the grouping of three paragraphs with two paragraphs from Figure 12-3: [0, 1, 9-0, 6]

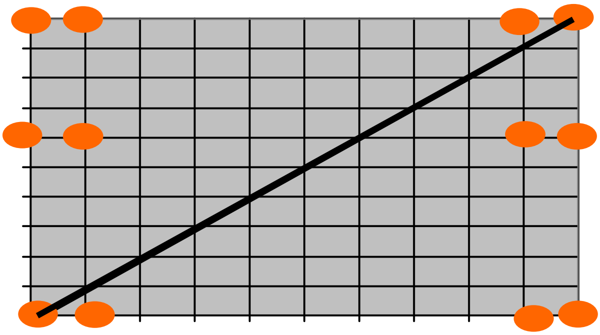

Figure 12-5. Trellis for the alignment of (0, 1, 9, 10) and (0, 6, 10)

The results obtained with this algorithm are better than those of the naïve algorithm counting paragraphs, but this approach is very sensitive to noise and will work well only on texts that are translated very precisely, homogeneously, and without omissions or insertions. If for example, the translation of a journalistic article contains an introductory paragraph that the original did not contain, all paragraph alignments will be shifted down one step too far and the alignment will thus be mostly wrong. It is clear that it is necessary to add other hints in the bitext that will make the alignment more robust.

3. Time warp

Dynamic time warping algorithms are also based on the distribution of a word in the whole text, but contrarily to the paragraph marks, we first have to establish the pairings. The intuition behind the time warping approach is that a word signal resembles the signal of its translation, even if the latter is "deformed" by the translation. The signal may be reduced, occur earlier or later or even miss certain points, but it still remains "recognizable" as being the translation of the original signal.

3.1. Illustrating the intuition behind dynamic time warping

To illustrate this intuition, let us consider a French text with its Chinese translation. We use the first volume *Aube* of the epic *Jean-Christophe* by Romain Rolland (1904-1912), amounting to 226,981 characters, and its Chinese translation by Fu Lei (1957) totalling 68,062 characters.[5]

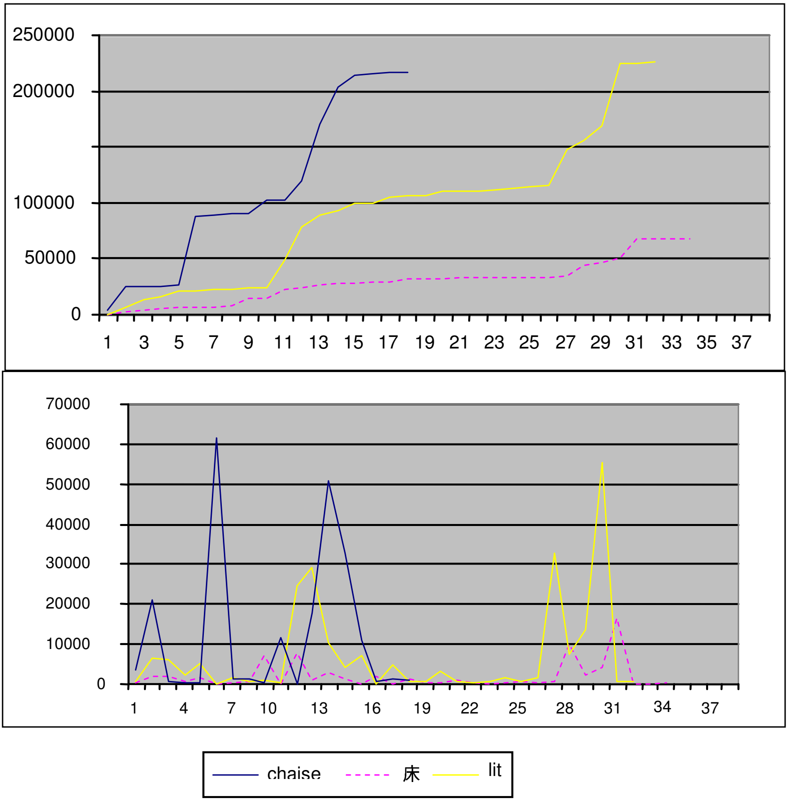

Figure 12-6. Occurrence vectors (top) and recency vectors (bottom) for three words

We consider three words: *lit* ‘bed’, its Chinese translation 床 and the word *chaise* ‘chair’. If we represent the points where these words occur simply as the number of characters from the start of the text, we obtain the graph at the top of Figure 12-6. The simple fact that *lit* and 床 occur a similar number of times (respectively 32 and 34 times) causes their curves (the light and the dotted curve) to be more similar but this similarity seems difficult to discern. It is preferable to use a recency vector. That is, instead of representing distances of occurrences of the word from the beginning of the text, we take into account the distance (in number of characters) between each apparition of the word. The representation of this vector makes the similarity of the lines of *lit* and 床 stand out much more clearly (the graph at the bottom of Figure 12-6). However, the fact that French uses many more characters than Chinese still appears in the graph as higher amplitude of the French curves.

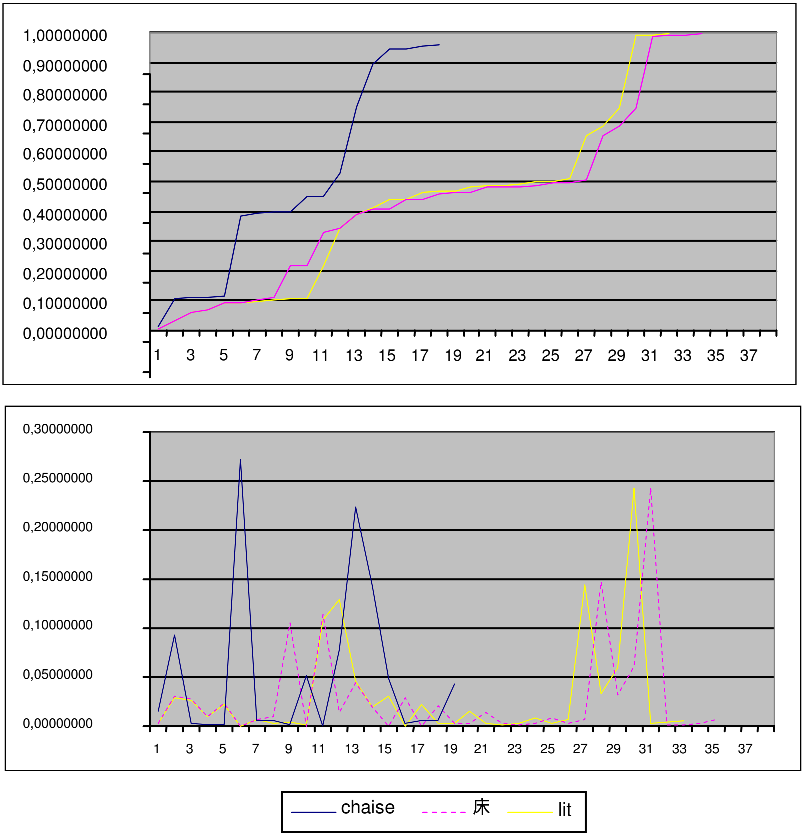

Figure 12-7. Occurrence and recency of three words in a normalized bitext

Using fractions of the whole texts instead of absolute values allows for a normalization of the curves. Now the link between the two words *lit* and 床 stands out clearly in comparison with *chaise* just as well in the fractional diagram (the top graph of Figure 12-7) as with the normalized recency vectors (using the distances between each occurrence of the words expressed in fractions of the text (the graph at the bottom of Figure 12-7). In this latter diagram one recognizes easily the slight movement to the right of the 床 curve, which is caused by two supplementary apparitions of 床 around its

9th and 10th apparitions. The time warping algorithm will allow us to establish a distance measure between two words that counts only once this right movement of the *lit* curve in relation to the 床 curve. Intuitively, the time warping distance will only count the "stretching" needed around positions 9 and 10 to superpose the two curves rather than the constant offset of the two curves. Clearly, normalization can bring the similarity of curves of 床 and *lit* to the fore.

After a short summary of works using dynamic time warping approaches, we will determine the metrics to measure the distance between curves of this type. Then we will expose the algorithm used to find word couples based on the similarity of their signals.

3.2. The use of time warping

Dynamic time warping (DTW) algorithms attempt to find optimal monotone (non-crossing) alignments of two sequences of variable length. The optimal alignment minimizes the distortion between signals. DTW is used in a wide range of domains for the recognition of forms that can be extended or contracted while preserving the information for easier recognition. Its "classic" use is in speech recognition, today usually combined with Hidden Markov Models (Jelinek 1997); moreover it is used in image or form recognition (where the deformation can be multidimensional) as for example in signature or face recognition or even in data mining (Ratanamahatana 2004).

Concerning the use of DTW for bilingual corpus alignments, the first attempts in this direction have been made by Fung and McKeown (1994), who work on English-Chinese alignment (see also Somers (1998) for a comparison of similar approaches). Their work shows that the DTW algorithm can find pairs of words that are mutual translations.

Note that Fung and McKeown's (1994) algorithm starts with a Chinese text that is already segmented into words. They do not indicate, however, how this segmentation has been obtained, nor do they state the linguistic premises for this segmentation. This is important for two reasons. First, the use of pre-segmentation makes their algorithm dependent of linguistic resources because all segmentation systems of Chinese rely heavily on usually large-scale dictionaries to accomplish this task.[6] Second, the notion of "word" has an important influence on the results, because their algorithm uses the words directly as aligned units. If we wanted to obtain a Chinese-German alignment, for example, a "German-style" segmentation of the Chinese texts (i.e. a system where compound nouns constitute single words) would certainly give much better results than an "English-style" segmentation, another Germanic language, where compound nouns are written with spaces between the nouns.[7]

Thus, without any explication of this preliminary step of segmentation, Fung and McKeown's (1994) results are neither verifiable nor reproducible. We have to add here that the task of alignment, as indicated in the title of their paper, is never accomplished. They find good pairs of words that could be used as anchor points for the alignment but two points remain obscure: 1) the type of alignment (on paragraph, sentence, phrase, or word level) they want to achieve with the pairs; and 2) the actual alignment method that uses the pairs.[8]

3.3. The good distance between signals

The computation of the global distance between two sequences is based on the sum of local distances (between two elements of the two sequences). It is primordial to find a good metric of local distances, because errors will multiply up in the computation of the global distance and we have to watch out that long sequences will not have a greater distortion just because of their length (as it is the case in Fung and McKeown's (1994) metric that uses word numbers).

Let us develop this point in greater detail. In Figure 12-8, graph A shows two texts of identical length (language 1 on the left, and language 2 on the right) with a word pair that has an identical distribution (three occurrences in both languages at identical positions). These words are of course very good candidates for being mutual translations and we want to attribute 0 as the distance between these signals. Graph B shows the same pair of words in a bitext where the second text is shorter. It is clear that in this case, the word pair is a less good candidate for being a translation than in the A case. Graph C shows another bitext where a word pairing looks just as good as pairing A, because the second text is shorter. In order to obtain a distance measure that corresponds to this intuition where distance in B is greater than distance in A which equals that in C, we again have to normalize and use fractional instead of absolute positions.

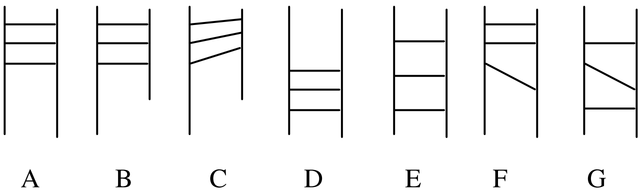

Figure 12-8. Word pairing vs. text length

A second point to take into account for the design of the distance measure is the recency vector. From the position vector of a given word (m_1, m_2, m_3, … m_n), Fung and McKeown (1994) compute the recency vector (m_1, m_2- m_1, m_3- m_2, … m_n- m_{n-1}). This recency vector is not symmetrical in the sense that it counts the distance between the beginning of the text and the first occurrence of the word (the value m_1) but it ignores the distance between the last word and the end of the text. This asymmetric recency vector does not always give bad results. For example, the couples shown in graphs D and E in Figure 12-8, which are clearly as good candidates as A or C, will all get 0 as a distance although the words occur at different positions in the text. However, our metric has to give the same value (>0) to the pairs F and G. Without taking into account the distance of the last occurrence of the word to the end of the text, the distortion of F will only be counted once (as the distance between the second and the last couple). In G, Fung and McKeown's (1994) metric will count it twice: once between the first and the second couple and another time between the second and the last couple of words. The F pairing will have a smaller distance than G, contrary to our intuition about the structure of occurrences of translations in bilingual texts.

In order to compute a correct recency vector, we use a position vector expressing fractions of the text (p_1, p_2, p_3, … p_n). The recency vector includes the distance to the end point (p_1, p_2- p_1, p_3- p_2, … p_n- p_{n-1}, 1- p_n).

Fung and McKeown's (1994) metric is not normalized length and is skewed by leaving out the final recency distance. But even if our metric seems more intuitive, we cannot compare our results directly. They start with a text segmented into words by a non-specified algorithm, as noted earlier, and moreover they only show some examples of anchor points they discover on the basis of heuristics that are not justified in the text (restriction to word frequencies of 10 to 300 words for an English-Chinese text of 700kb).

3.4. The algorithm for the computation of the time warped distance

The distance computation we use is a simple dynamic algorithm. Figure 12-9 gives the algorithm in pseudo-code: Given the position vectors of two words to be compared. After computing the recency vectors for each position vector as stated above, we construct a table crossing the two recency vectors and an additional line and colon filled with 1s (maximal distance) with the exception of the slot (0,0) containing 0 (see Figure 12-10).

```
timewarp(list1,list2):
    # takes two lists of numbers between 0 and 1
    # and computes a time warp distance
    rec1, rec2 = recency(list1), recency(list2)
    warp[(0,0)] = 0              # table initiation: corner
    for i=0 to length(rec1) do:
            warp[(i+1,0)] =  1 # table initiation: first line
    for j=0 to length(rec2) do:
            warp[(0,j+1)] =  1 # table initiation: first colon
    for i=0 to length(rec1) do:
           for j=0 to length(rec2) do:
             warp[(i+1,j+1)] = abs(rec1_i-rec2_j) + min(warp[(i,j+1)],
            warp[(i+1,j)], warp[(i,j)])
    return warp[(i+1,j+1)]
```

Figure 12-9. Pseudo-code for time warping

Then the rest of the table as filled line by line as illustrated in Figure 12-10. In each slot S we enter the distance between the corresponding recency vector values, to which we add the minimal value of the following 3 slots: left of S, above S, or diagonally left above S. These three possibilities correspond to a table traversal linking slot S to one of its neighbours on its left, above, or diagonally above. The restriction to these three directions reflects the monotonicity of time warping: we can distort the signal but not tear it apart. When the table is filled, the distance between the words appears in the lowest most right slot (length of recency 1, length of recency 2), symbolizing the less costly alignment between the two words, in the same way as shown in section 2.2 for paragraph marks.

0	1	1	1	1	1	1
1						
1						
1						
1				S		
1						
1						
1						

Figure 12-10. Filling the time warping matrix

The distance computation presented here gives an advantage to rare words, because in all texts the total number of rare words is very high compared to frequent words (Zipf's law). The chances of finding two hapaxes (words with frequency 1) that are not mutual translations at some arbitrary identical fraction of the text (for example 47.6%) are very high and these pairs will thus obtain a distance close to zero, corresponding to their distance in the text. Inversely, frequent words have a very low chance of occurring all the time at exactly identical positions on both sides and they will always have a distance greater than zero. Their high frequency, however, keeps this number quite small. In heuristic tests, we wanted to give an advantage to groupings of frequent words, e.g. by dividing the distance by the number of created couples, but this will favour too boldly frequent words and exclude all rare words from the list of best couples.

The couples we want to retain depend on the use we have for them. When aligning paragraphs, we are interested in words that allow us to find as many interesting paragraph alignments as possible. In other words, we want couples that appear in a few but not in too many paragraphs, the most discriminating distribution being close to half of the number of paragraphs. These maximum and minimum values remain parameterizable by the user. Words that appear between 5% and 50% of all the paragraphs are found to be good value, but further tests will have to determine the optimal values and whether these values differ considerably between languages.[9]

In our implementation of the algorithm, we use another heuristic that does not change the results, but speeds up computation considerably. We only compute the distance between pairs of words that have a similar frequency. We take 50% to 200%; in other words, for a given word W, we do not compute distances between the word and a putative translation T, if T appears more than double or less than half the number of occurrences of W.

4. Language internal cognates and example results

We assume in the introduction section that any sufficiently long text between any two languages will contain some "easy" pairs of word-to-word translation that can be discovered by time warping signal distance comparison. This may be true, but in order to enhance our chances of finding enough "good" pairs even for languages that refuse to take most proper nouns (like Slavic languages) as usually natural candidates for "good" pairs, we propose to include groups of "graphically similar" words. How can this be done?

The answer is simply to apply a cognate search internal to the text in one language. Instead of a simple Levenshtein distance (that equals the number of changes needed to pass from one word to the other, which is used, for example, in any spell checker's replacement options), we will go for a slightly more complex distance, i.e. the Jaro-Winkler distance, a measure that counts variations at the end of the word less than variations in the beginning of the word. This privileges the detection of word final inflection, and if languages should exist where the beginning of words is inflected more heavily than the end, this algorithm is not a good choice and could be seen as a linguistic parameterization, contrary to the stated goal.

For both texts in a bilingual pair, we compute the groups of words that are graphically similar (again using a heuristic minimum that speeds up computation) and add the discovered word groups to our list of words as if they were words with a unique form. Of course, many of the proposed word groups are not different forms of a common morpheme and have nothing in common but a similar form; but theoretically, this should not matter as a group of forms that have no common morpheme should have no counterpart in the other language with a sufficiently similar signal. To our surprise, this holds not completely true, and some of the discovered groups are slightly polluted, though the slight error does not destroy the overall advantage of using these groups.

Table 12-1 shows the 20 best pairings found for the French-German bitext *The Sorrows of Young Werther*. Word groups are based on the language internal Jaro-Winkler distance. Note that all but the second pairing are correct (sometimes partial) translation. The word ‘Daura’ is grouped with ‘Armar’ because they only appear together in a short specific section of the text, and thus this pair also helps to adjust the alignment. It is important to see this extraction of pairs not as a final goal of extraction of a translation vocabulary but exclusively as an extraction of useful anchors for the subsequent alignment process. Bilingual vocabulary extraction should be done on the basis of the final aligned corpus. Note also, that the proper nouns and the numbers in this list were not discovered by cognate matching but exclusively by their signal similarity.

Table 12-1. The 20 best pairings in *The Sorrows of Young Werther*

Distance	German words (or word groups)	French words (or word groups)	Gloss
0:0.00767	arindal	arindal	Arindal (name)
1:0.00773	daura	armar	Daura/Armar (names)
2:0.00863	daura dauras	daura	Daura

			(name)
3:0.01015	morars morar	morar	Morar (name)
4:0.01043	heide	bruyère bruyères	heath
5:0.01069	armins armin	armin	Armin (name)
6:0.01076	linden lindenbäume linde	tilleul tilleuls	linden (trees)
7:0.01090	bücher	livres	books
8:0.01118	paradiesisch paradies paradiese	paradis	paradise, paradisiac
9:0.01140	mai	mai	May
10:0.01144	gesandten gesandter gesandtschaft gesandte	ambassade ambassades ambassadeur	embassy, embassador
11:0.01145	schnee schneeglänzenden	neige	(sparkling) snow
12:0.01145	dezember	décembre	December
13:0.01179	krankheit	maladie	illness
14:0.01244	august	août	August
15:0.01282	buches buche buch	livre	book
16:0.01307	klaviere klavier	clavecin	piano
17:0.01324	september	septembre	September
18:0.01342	8	8	8
19:0.0141449775087	30	30	30

Table 12-2 gives the 20 best results for the Chinese-French bitext *Aube* from *Jean Christophe*. Note that the grouping of the misspelled ‘Gotttfried’ (three times the letter *t*) gives a smaller time warping distance than the correctly spelled ‘Gottfrieds’ alone. The system thus “discovered” the spelling error. Note also the greater distances than in the first examples. As expected, the second bitext offers fewer “easy” pairs than the first. Work is in progress to determine useful heuristics on the maximum distance values of useful word pairs, depending probably on the overall size of the corpus.

Table 12-2. The 20 best pairings in *Aube*

Distance	French words (or word groups)	Gloss	Chinese charac-ters	Gloss
0:0.06186	gotttried gottfried	Gottfried (name)	舅	part of the name *Gottfried*
1:0.14329	chairs chaises caisse chaise	chair, box	椅	chair
2:0.15069	table tablier tables	table, apron	桌	table
3:0.20146	lit	bed	床	bed
4:0.20734	melchior	Melchior (name)	沃	part of the name *Melchior*
5:0.21150	nuit	night	夜	night
6:0.21265	piano	piano	钢	part of the word *piano*
7:0.23058	louisa	Lousia (name)	莎	part of the name *Louisa*
8:0.26149	père	father	父	father
9:0.28664	oie voie joie	goose, voice, joy	忘	forget
10:0.29866	grands grains graisse grasses gras ras	big, grain, fat, crop	内	in(side)
11:0.29993	conseil consentait conservait conversation servait conserve	council, consented, kept, conversation, served, can	书	book, letter, write
12:0.30906	regarder	look	熟	cook
13:0.30921	craignait crainte criant craintif craintes	fear, fearful, shouting	德	virtue, Germany
14:0.30960	petits	small	久	small

15:0.31001	rêves rêveries rêver rêve	dream	梦	dream
16:0.31376	réveille réelle réveiller réveilla réveillé éveille veiller réveillait	wake, real	醒	wake
17:0.31705	soupir assoupir soupirail soupira soir souper	sigh, evening supper	晚	evening
18:0.31767	vieilles vieillots vieille vieillissait vieil vieillards vieillard	old (growing)	闷	melan-choly, bored
19:0.32201	connais connaisseur connaissance connaissent connaissait	know, expert, knowledge	立	stand, establish

5. The alignment algorithm

To conclude the description of the overall alignment process, we present in this section a simple algorithm of using anchors to align paragraphs (see Figure 12-11). These anchors can be other cognates than the new line character. If we find any, of course, we have to take them into account. This step integrates well in the overall process, although the goal of this chapter is precisely the description of a solution for aligning bitext where no cognates or insufficient numbers of cognates are available. To this list of cognates we can add a search for all Unicode names of all punctuation and numeral symbols. If their names are similar, they can also be added to the cognate list. This is particularly interesting for numeral symbols or "rare" punctuations (as colons or semicolons) as the more frequent symbols like commas will not help paragraph alignment because they appear in nearly every paragraph. All cognates must have a distance value compatible with the distances of the time warping measures. The easiest step is just to give "real" cognates the value of the lowest discovered time warping distance.

```
getAlignmentMatrix(goodCoupleList):
# takes a list of good couples
    alignmatrix = numberParagraphsText1 x numberParagraphsText2
    set all alignmatrix values to 1
    for each (word1, word2, distance) from goodCoupleList do:
                    parInd1 = getParagraphIndeces(word1)
                    parInd2 = getParagraphIndeces(word2)
                    for i=0 to length(parInd1) do:
                         for j=0 to length(parInd2) do:
                              alignmatrix[i,j]=alignmatrix[i,j]*distance
    return alignmatrix
```

Figure 12-11. Pseudo-code for the construction of the alignment matrix

We call the combined list of "real" cognates, Unicode cognates and time warping couples the "list of good couples". We create an alignment matrix crossing all paragraph positions of the two texts and we initialize the matrix with ones in all slots. For each couple (*word1, word2, distance*) in the list of good couples, we obtain the two lists of paragraph indices in which *word1* and *word2* appear respectively. Each value of the slot that corresponds to a pair of paragraphs (in which *word1* and *word2* appear respectively) will be multiplied by *distance*. In this way, pairs of paragraphs that occur for various couples will receive a particularly small value.[10]

Now we only have to compute the "cheapest" path crossing this alignment matrix. For this we can practically use the same algorithm that we use for time warping (see Figure 12-12); we only have to record at each step which of the three choices (left, top, diagonal) has the minimal value, in order to be able to trace back the way through the matrix. Once we are through, we have to follow these indications from the lower right corner back to the top left corner of the matrix. Each diagonal step will correspond to a separation of two paragraph blocks, while each vertical or horizontal step adds a new paragraph to the existing block.

Note that we do not have to apply any preference of the diagonal again, as this preference is already contained in the choice of good pairs (they are declared good because they have similar signals, i.e. the pair is close to the diagonal). In other words, this algorithm will stay on the diagonal unless a detour is “cheaper” for very good reasons, i.e. lots of good couples asking for it.

```
getAlignment(alignmentMatrix):
    # takes an alignment matrix
    # and computes the diagonal path through the matrix with the lowest overall values
    # the output is a matrix that contains ones at the aligned paragraphs
    lines = number of lines of alignmentMatrix
    colons = number of colons of alignmentMatrix
    warp  =  lines +1  x colons +1
    directions = lines x colons
    finalAlignment  = lines x colons
    f,g=lines-1,colons-1
    set all warp values to ∞
    warp[0,0]=0
    for i=0 to lines do:
        for j=0 to colons do:
                mini = min(warp[i,j+1], warp[i+1,j], warp[i,j])
                warp[i+1,j+1] = matrix[i,j] + mini
                if mini == warp[i,j]: directions[i,j] = 0
                elif mini == warp[i,j+1]: directions[i,j] = 1
                else : directions[i,j] = -1
    while f>=0 or g>=0:
        finalAlignment[f,g]=1
        if directions[f,g]==0:
                f-=1
                g-=1
        elif directions[f,g]==1: f-=1
        else: g-=1
    return finalAlignment
```

Figure 12-12. Pseudo-code for computing the final alignment from the alignment matrix

This algorithm gives satisfying results for insertions and deletions if sufficient good pairs have been found. At least for all concrete examples we have tested the system on, the results are always notably better than the simple paragraph length alignments.

Further work will have to test systematically the advantages and disadvantages of this system compared to other approaches and we will explore the usage of other cognate algorithms that allow for quality values of the cognates to be taken into account.

The system is implemented on a private web server (http://elizia.net/alignator/). Although the main system is programmed in Python, the computation of the time warping distance as well as the Jaro-Winkler distance between all possible couples of words remain very heavy on long texts, even with the "tricks" of restricting the analysis to words with interesting frequencies for the paragraph alignment and to couples that have a chance of being translation based only on their frequency. This part had to be written in C (thus enhancing the speed by a factor of nearly 50) to make the system usable in a few minutes even on long texts. The user interface uses Javascript. The use of a web server allows for a direct access on all computer systems without prior installation. The complete code will be distributed as free software under the GNU licence.

6. Conclusion

To sum up, here is a brief list of the steps of the algorithm of this alignment system:

1. Word detection – if *scriptua continua*, work on the character level;
2. Cognate detection, including punctuation cognates using Unicode names (it is not necessary to find any);
3. On languages with word spacing, add "intra-language cognates" to the word list, i.e. groups of words with similar forms using the Jaro-Winkler distance;
4. Apply DTW distance measures with the normalized text length on potentially useful word pairs (or word group pairs) and extract potential translation pairs;
5. Add distance of all potential translation pairs (including cognates, if any) to the paragraph matrix of both languages and compute a minimal diagonal matrix path, corresponding to the best paragraph alignment;
6. This alignment can be corrected manually, directly on the

web, and exported in different formats for further examination;

7. This approach is considerably better than naïve approaches to paragraph alignment like a purely length based alignment, but it is difficult to evaluate and compare in greater detail for two reasons:
    - on the one hand, other work is often language specific, focuses on sentence alignment or vocabulary extraction, and is often unavailable for testing;
    - on the other hand, while it is easy to construct artificial bitexts that will fool the system, the lack of large manually aligned bitexts for various non-European language pairs makes it impossible to give numbers on the reliability of the system on real texts in those languages.

In conclusion, we believe that the alignment system presented in this chapter can be of great help for researchers of translation studies working on rare language pairs when they create aligned parallel corpora; and if the automatically aligned corpora are eventually corrected manually, they can serve as control data for further systematic enhancement of the algorithm and its associated heuristic parameters. Moreover, the results obtained by this resourceless system as well as the problems encountered in its development have shed light on some universals of translation.

Notes

1. See also the classic debate on the existence of discontinuous morphemes (Harris 1945).
2. The most serious methodological problem concerning Japanese is that only texts using many foreign words can be aligned. A "pure" Japanese text, for example with its English translation, cannot be aligned in this way. In this latter case we would need a complex pronunciation lexicon, just as for Chinese texts.
3. In tenths of the whole text for simplification. We refer to the paragraphs by the fraction of the text that indicate the starting points of the paragraphs in the text.
4. The hyphen indicates here the association of two groups of paragraphs.

5. The electronic versions of these texts were graciously provided by Jun Miao from the ESIT, Sorbonne Nouvelle.
6. As the Chinese writing system does not give easy indications on the beginning or ending of words (contrary to Japanese, for example, where certain simple heuristics on the changes of the types of characters can go a long way), it is natural to use extensive lists of words. The only alternative could be a search for repeated sequences in very large corpora. This, however, will not easily give linguistically relevant results (because the definition of "word" is much more semantic than statistical – one would consider as words, in English deprived of spaces for example, nouns that are always followed by a specific preposition).
7. Fung and McKeown (1994) give an astonishing example: 一氧化碳 'carbon monoxide' is listed twice as a word, once translated as 'carbon' and a second time translated as 'monoxide'. We can thus conclude that in their corpus, the segmentation does not separate compound nouns.
8. Knowing the word pairs does not imply knowing how to align the occurrences of the words. See the extensive literature on cognate alignment and section 2.2 of this chapter where we show a possible alignment procedure for the known "pair" of the new line character.
9. It is possible to enhance the algorithm further by also taking account of high frequency word couples (or symbols like punctuations), for which we believe that they are mutual translations. However, they will have to be taken into account differently in the subsequent alignment computation (where for the moment we only count a binary absent/present feature).
10. Note that we enter all possible alignments of the couple into the matrix, not just the "best" alignment (i.e. the closest one to the diagonal). This makes it possible to get longer distances from the diagonal, as soon as a large number of pairs point to the same paragraph pairing.

References

Fung P. and McKeown, K. (1994), "Aligning noisy parallel corpora across language groups: Word pair feature matching by dynamic time warping", in *Proceedings of the First Conference of the Association for Machine Translation in the Americas (AMTA-94)*, 81-88, Columbia, Maryland.

Gale W. and Church, K. W. (1991), "A program for aligning sentences in bilingual corpora", in *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*. Berkeley, CA.

Harris, Z. S. (1945), "Discontinuous morphemes". *Language* 21(3): 121-127.

Haruno M. and Yamazaki, T. (1997), "High performance bilingual text alignment using statistical and dictionary information". *Natural Language Engineering* 3(1): 1-14.

Jelinek F. (1997), *Statistical Methods for Speech Recognition*. Cambridge, MA: MIT Press.

Kay, M. and Roscheisen, M. (1993), "Text-translation alignment". *Computational Linguistics* 19(1): 121-142.

Knight K. and Graehl J. (1998), "Machine transliteration". *Computational Linguistics* 24(4): 599-612.

Lebart L. and Salem A. (1994), *Statistique Textuelle*. Paris: Dunod.

Meng H., Lo, W. K., Chen, B. and Tang, K. (2001), "Generating phonetic cognates to handle named entities in English-Chinese cross-language spoken document retrieval", in *Proceedings of the Automatic Speech Recognition and Understanding Workshop*. Trento, Italy.

Ratanamahatana, C. A. and Keogh, E. (2004), "Everything you know about Dynamic Time Warping is wrong", in *Third Workshop on Mining Temporal and Sequential Data*. Seattle, WA.

Ribeiro, A., Dias, G., Lopes, G., and Mexia, J. (2001), "Cognates alignment", in B. Maegaard (ed.) *Proceedings of the Machine Translation Summit VIII*. Santiago de Compostela, Spain.

Simard, M., Foster, G. and Isabelle, P. (1992), "Using cognates to align sentences in bilingual corpora", in *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation TMI-92*, 67-81. Montréal, Canada.

Somers, H. (1998), "Further experiments in bilingual text alignment". *International Journal of Corpus Linguistics* 3: 115-150.

Wagner, R. A. and Fischer, M. J. (1974), "The string-to-string correction problem". *Journal of the ACM* 21(1): 168-173.

Yamada, K., and Knight, K. (2001), "A syntax-based statistical translation model", in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 523-529. Toulouse, France.

Yarowsky, D., Nag, G. and Wicentowski, R. (2001), "Inducing multilingual text analysis tools via robust projection across aligned corpora", in *First International Conference on Human Language Technologies*. San Diego.

Zhou, D. (2003), *Waiguo Diming Yiming Shouce* (Translation Dictionary of Proper Nouns and Foreign Places (modified edition). Beijing: Commercial Press.