Text Alignment in a Tool for Translating Revised Documents

Hadar Shemtov

Stanford University

Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304 USA
shemtov@parc.xerox.com

1 Introduction

Making use of previously translated texts is a very appealing idea that can be of considerable practical and economical benefit as a translation aid. There are different ways to exploit the potential of "re-translation" with different degrees of generality, complication and ambition. Example-based machine translation is probably the most ambitious end of the spectrum but there can be other points along it. In this paper I describe a simple tool which deals with a particular special case of the "re-translation" problem. It occurs when a new version of a previously translated document needs to be translated. The tool identifies the changes between the two versions of the source language (SL) text and retrieves appropriate sentences from the target language (TL) text. With that, it creates a bilingual draft which consists of sections in the TL text from the existing translation and update materials from the SL text, thereby reducing the effort required from the translator. This tool could substantially increase the productivity of translators which deal with technical documents of frequently modified products (software-based products are the best example of that). If this is true, it suggests that simple solutions can be very effective in addressing "real-life" translation problems.

The paper is structured as follows. The first section discusses some relevant properties of typical texts which are likely to be (re-)translated with this tool. The second section is about the alignment process – I will present a new length-based alignment algorithm, designed for dealing with texts that include additions and deletions. In the following section I will propose a quick procedure to find the differences between two versions of the same document. Then, I will show how the bilingual draft is constructed. The last section will discuss possible continuations of this research which will extend the applicability of the tool to more general translation situations.

2 The Problem of Nationalization

Situations where a document needs re-translation are usually associated with commercial products that undergo modifications and revisions and require accompanying literature in different languages. The process of accommodating such texts to different countries and languages does not stop at merely translating the exact content of the original document. Rather, it involves adaptation of the text to different norms and shared knowledge of a different audience. Sometimes, the products themselves are modified and sometimes the new market impose changes that need to be made in the technical documentation of the products. This probably arises most frequently in the user manuals of software products. Different countries use different keyboards, different languages often require adaptation of the software itself and also, users in different countries have different expectations and norms which the documentation (if not the product itself) needs to reflect. These factors, together with the actual translation, constitute the process usually referred to as "nationalization".

Nationalization often gives rise to a situation where some of the text has no corresponding translation. Since documentation of commercial products are the type of texts that usually require re-translation, this situation has to be recognized and handled by the translation tool. For that purpose, I developed a new alignment algorithm that will be presented in the next section.

3 Alignment

Length-based alignment algorithms [Gale and Church, 1991b; Brown *et al.*, 1991] are computationally efficient which makes them attractive for aligning large quantities of text. The main problem with them is that they expect that, by and large, every sentence in one language has a corresponding sentence in the other (there can be insertions and deletions but they must be minor). In the character-based algorithm, for example, this is implicit in the assumption that the number of characters of the SL text at each point (counting from the beginning of the text) is a predictor for the number of characters in the TL. This assumption may hold for some texts but it cannot be relied on. As a consequence of nationalization, one text may be substantially longer than the other and this makes the length correspondence assumption incorrect (if the additions and omission were not reflected in the length of the two texts, the situation would have been even worse). Simply, the cumulative length of the text is no longer a good predictor for the length of its translation. This problem affects the consideration of the text as a whole. However, locally, the length-correspondence assumption can still be maintained. Gale and Church hint that their method works well for aligning sentences within paragraphs and that they use different means to find the correspondence (or lack thereof) of paragraphs. A more detailed description of such an approach is given by Brown et al. that use structural information to drive the correspondence of larger quantities of text. However, such clues are not always available. In order to address this problem more generally I developed an algorithm that is more robust in detecting insertions and deletions which I use for aligning paragraphs.

3.1 Aligning Paragraphs

The paragraph alignment algorithm relies on the observation that long segments of text translate into long segments and short ones into short ones. Unlike the approach taken in Gale and Church, it does not assume that for each text segment in the SL version there is a corresponding segment in the TL. Instead, the algorithm calculates for each pair of text segments (paragraphs in this case) a score based on their lengths. For each potential pair of segments, several editing assumptions (one-to-one, one-to-many, etc.) are considered and the one with the best score is chosen. Dynamic programming is then used to collect the set of pairs which yields the maximum likelihood alignment. The score needs to favor pairing segments of roughly the same length but since there is more variability as the length of the segments increases, the score needs to be more tolerant with longer segments. This effect is achieved by the following formula which provides the basis for scoring:

$$s(i,j) = \frac{|l_i - l_j|}{\sqrt{l_i + l_j}}$$

It approaches zero as the lengths get closer but it does so faster as the absolute length of the segments gets longer. So, for example $s_{10,20} = 1.8257$, but $s_{110,220} = .5504$ (the square root of the sum is used instead of simply the sum so that $s_{10,20}$ would be different from $s_{100,200}$). This simple heuristic seems to work well for the purpose of distinguishing correlated text segments. However, since paragraphs can be quite long and the degree of variability between them grows proportionally, this score is not always sufficient to put things in order. To augment it, more information is considered. The actual score for deciding that two paragraphs are matched also takes into consideration a sequence of paragraphs immediately preceding and following them (see figure 1 for an illustration). This is based on the observation that the potential for aligning a pair of segments also depends on the potential of them being in a context of alignable pairs of segments. According to this scheme, a pair with a relatively low score can still be taken as a correspondence if there are segments of text preceding and following it which are likely to form correspondences.

This scheme lends itself to calculating a score for the assumption that a given paragraph is an insertion (or deletion). So, if segment i is an insertion, the context for considering it will consist of the following pairs $\ldots i-2/j-2,\ i-1/j-1,\ i+1/j,\ i+2/j+1\ldots$ This way, a score is being assigned to the assumption that a certain segment in one text has no corresponding segment in the other text. Likewise, if j and $j+1$ are insertions to the other text the score considers $\ldots i-2/j-2,\ i-1/j-1,\ i/j+2,\ i+1/j+3\ldots$ as the appropriate context for calculation the score.

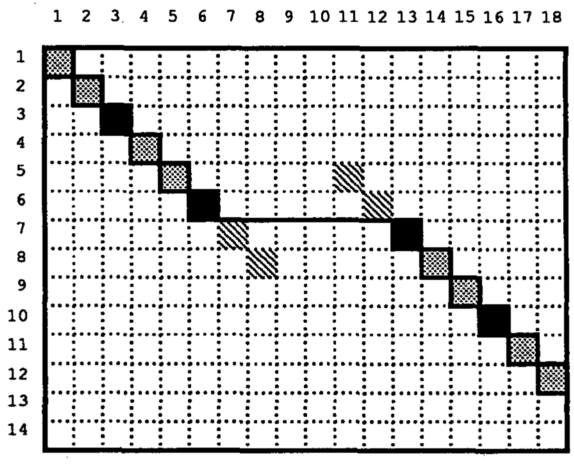

Figure 1: Paragraph Alignment

It is easy to see how this works for insertions of short sequences but it remains to be explained how arbitrarily long sequences are handled. In principle, it would be best if for each n (the length of a sequence of insertions), the following context would consist of $i+n/j, i+n+1/j+1$ etc. but obviously, this is not practical. This is related to another potential problem which has to do with the contexts calculated near insertions or deletions. Figure 1 depicts this situation (the gray squares identify the context for aligning the pairs denoted by the black squares; the marked path stands for the correct alignment).

The alignment score of a segment previous to an insertion is based on appropriate preceding context but irrelevant following context (the reverse holds for a segment following an insertion)[1]. To minimize the effect of this situation, a threshold is introduced so that when the score of one side of the context is good, the effect of very bad score in the other side of the context is kept below a certain value. Note also that

[1]This is an important factor for selecting the amount of context. It could be assumed that the wider the window of segments around each pair is, the more accurate the determination of its alignment will be. However, this is not the case exactly because of the fact that occasionally the algorithm has to consider some "noise". Empirical experimentation revealed that a window of 6 segments (3 to each side) provides the best compromise between beneficial information and noise.

although some noise is being introduced into the calculation of these scores, other editing assumptions are likely to be considered even worse. Occasionally this has an effect on the exact placement of the insertion but in most cases, the dynamic programming approach, by seeking a global maximum, picks up the correct alignment.

Now, let me return to the issue of long sequences of insertions. The situation is that in one location there is a sequence of high-quality alignment, then there is a disruption with scores calculated for arbitrary pairs of text segments, and then another sequence of high quality alignment begins. What happens in most cases is that between these two points, the scores for insertions or deletions are better than the scores assigned to random pairs of segments. Here too, the effect of global maximization forces the algorithm to pass through the points where the insertion begins, resume synchronization where it ends and consider the points in between as a long sequence of unpaired segments of texts. In other words, once the edges are set correctly, the remainder of the chain is almost always also correct, even though it is not based on appropriate contexts.

This potential problem is the weakest aspect of the algorithm but essentially, it does not have an impact on the quality of the alignment. Note also that even if the exact locus of insertion (or deletion) is not known, the fact that the algorithm detects the presence of text with no corresponding translation is the crucial matter. This way, the synchronization of the text segments can be maintained and alignment errors, even when they happen, can only have a very local effect. To demonstrate this, let us consider a concrete example. An English and a French versions of a software manual contain 628 and 640 paragraphs, respectively. In all, there are 30 paragraphs embedded in them which do not have a translation (some in fact do, but due to reordering of the text, these were considered as deletion from one location and then insertion in another location). The algorithm matched 618 pairs of paragraphs, only 11 of which were actually wrong. Note that between the two texts there were 13 different insertions and deletions of sequences varying from 1 to 6 paragraphs in length. The algorithm has proven to be extremely reliable in detecting segments of text that do not have a translation and this makes it very useful in dealing with what I have called "real-life" texts.

To summarize, this algorithm relies on the general assumption that the length of a segment of text is correlated with the length of its translation. It uses a sliding window for determining for each segment the likelihood of it being in a sequence of aligned text. This technique considers the correspondence as a local phenomenon, thereby allowing segments of text to appear in one text without a corresponding segments in its translation.

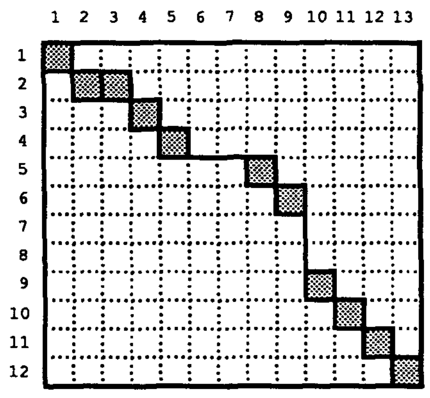

Figure 2: Minimizing alignment errors

3.2 Aligning Sentences

Sentences within paragraphs are aligned with the character-based probabilistic algorithm [Gale and Church, 1991b]. I used their algorithm since, compared to the algorithm described in the previous section, it is based on more firm theoretical grounds and within paragraphs, the assumptions it is based on are usually met.

However, there can be cases where it will be advantageous to use the new algorithm even at the sentence level. In texts where paragraphs are very long and contain sequences of inserted sentences, the character-based alignment will not perform well, because of the same considerations discussed above. Even a small amount of additions or omissions from one of the texts completely throws off alignment algorithms that do not entertain this possibility. In this respect, the new algorithm is more general than previous length-based approaches to alignment.

3.3 Minimizing alignment errors

An inherent property of the dynamic programming technique is that the effect of errors is kept at the local level; a single wrong pairing of two segments does not force all the following pairs to be also incorrect. This behavior is achieved by forcing another error, close to the first one, which compensates for the mistake and restore synchronization. As a result, errors in the alignment usually occur in pairs of opposite directionality (if the first error is to insert a sentence to one of the texts, the second is to insert a sentence into the other text). This situation is depicted in figure 2.

This, of course, can be a perfectly legitimate alignment but it is more likely to be a result of an error. These cases are easy to detect with a simple algorithm, which at the expense of losing some information can yield much better overall accuracy.

Each pair in the alignment is assigned one of 3 values: α if it is many-to-one (or one-to-zero) alignment, β if it is one-to-one alignment and γ if it is one-to-many (or zero-to-one) alignment. Intuitively, these values correspond to which text grows faster as a result of each pair of aligned segments. Having done that, the algorithm is simply a finite-state automaton that detects sequences of the form $\alpha\beta^k\gamma$ (or $\gamma\beta^k\alpha$) where k ranges from 0 to n (a predefined window size). The effect is that when an error occurs in one position and there is another "error" (with opposite contribution to the relative length of the text) within a certain number of segments, it is interpreted as a case of compensation; if it occurs farther away the situation is interpreted as involving two independent editing operations. The window is set to 4, since the dynamic programming approach is very fast in recovering from local errors.

When such a sequence is found, all the segments included in it are marked as insertions so the resulting alignment contains two contiguous sequences of inserted material, one to each one of the texts. This prevents wrong pairings to occur between the two identified alignment errors. For example, in figure 2, the pairing of segments 5/8 and 6/9 is undone, as it is likely to be incorrect.

Another possibility for minimizing the effect of alignment error has to do with the fact that occasionally, the exact location of an insertion of text cannot be determined completely accurately. I found that by disregarding a very small region around each instance of an insertion or deletion, the number of alignment mistakes can be reduced even farther. At the moment I found that to be unnecessary but it may be advantageous for other applications, such as obtaining even higher-quality pairs for the purpose of extraction of word correspondences.

4 Identifying the Revisions

On a par with identifying which portions of the SL text were omitted and which portion of the TL were added in the process of translation, the tool needs to identify the differences between the two releases of the SL text. It needs to know which parts of the text remain the same and which parts are revisions. To do that, what is needed is an algorithm that can match segments of equivalent texts which knows how to handle insertions and deletions. The algorithm that was developed for aligning paragraphs is a natural choice. It handles insertions and deletions successfully and it has certain other properties which make it extremely useful. Since it is based on length correspondence (rather than exact string comparison) it can align the two texts even when there are irrelevant structural differences between them. The idea is that since the two text are written at different times and presumably by different writers, there can be formatting differences which can complicate the task of identifying the changes. For this reason, a simple utility like 'diff' cannot be used. I found that by treating this problem as a special case of alignment, a much cleaner and simpler solution is obtained.

5 Constructing the Bilingual Draft

Once the correspondences between the old and the new versions and between the old version and its translation are obtained, the tool can construct the bilingual draft. In general, this is a very simple procedure. New text that appears only in the new version of the document is copied to the draft as is (in the SL). For text that has not been changed, the corresponding TL text is fetched from the translation and copied into the proper places in the draft. The final result is a bilingual version of the revised document that can be transformed into a full translation with minimal effort. Some complications may occur in this stage as a result of a conspiracy between certain specific factors. For example, if two SL sentences are translated by a single TL sentence and one of them is modified in the new release, probably it is not safe to use any of the translated materials in the draft. In such cases, in addition to the revised text, the tool copies into the draft both the relevant text from the old version and the relevant translation and marks them appropriately. The translator then can decide whether there is a point in using any of the existing TL text in the final translation of the document.

6 Conclusions and Future Directions

I hope to have shown in this paper that simple solutions can be quite useful when applied to specific and well-defined problems. In the process of developing this tool, a solution to a more general problem has been explored, namely, a more general text alignment algorithm. The algorithm described in section 3 has proven to be robust and efficient in aligning different types of bilingual texts.

The accuracy of the alignment process is the most important factor in the performance of this tool. One way to enhance the accuracy of the alignment, which I intend to pursue in the future, is to apply some form of the algorithm described in [Kay and Röscheisen, 1988] as a final stage of the processing. This will obtain the high accuracy of the computationally intensive algorithm while maintaining the benefits of the efficient length-based approach.

In addition to improving the current tool, I intend to explore other ideas that can apply in more general translation situations. For example, suppose that a new document needs to be translated and there exist a collection of bilingual documents in the same domain. It would be interesting to see how many sentences of the new document can be found, with their translation, in this collection. Probably, exact matches will not be so common, but one can think about ways to benefit from inexact matches as well. For instance, let us assume that two sentences have a a long sequence of words in common and one of them has already been translated. It is not unconceivable that obtaining the translation of the common sequence of words will facilitate the translation of the new sentence. To exploit this possibility, word-level correspondences [Gale and Church, 1991a] and phrase level correspondences will be required.

If this approach will be successful, it will enable more complicated and ambitious solutions to increasingly more general instances of the "re-translation" problem.

Acknowledgements

I would like to thank Martin Kay and Jan Pedersen for helpful comments and fruitful discussions relating to this paper.

References

[Brown *et al.*, 1991] Peter F. Brown, Jennifer C. Lai, and Robert L. Mercer. Alinging sentences in parallel corpora. In *Proceedings of the 29th Meeting of the ACL*, pages 169–176. Association for Computational Linguistics, 1991.

[Gale and Church, 1991a] William A. Gale and Kenneth W. Church. Identifying word correspondences in parallel texts. In *Proceedings of the 4th DARPA Speech and Natural Language Workshop*, pages 152–157, Pacific Grove, CA., 1991. Morgan Kaufmann.

[Gale and Church, 1991b] William A. Gale and Kenneth W. Church. A program for alinging sentences in bilingual corpora. In *Proceedings of the 29th Meeting of the ACL*, pages 177–184. Association for Computational Linguistics, 1991.

[Kay and Röscheisen, 1988] Martin Kay and Martin Röscheisen. Text-translation alignment. Xerox Palo-Alto Reserach Center, 1988.