Substring Alignment using Suffix Trees

Martin Kay

Stanford University

Abstract. Alignment of the sentences of an original text and a translation is considerably better understood than alignment of smaller units such as words and phrases. This paper makes some preliminary proposals for solving the problem of aligning substrings that should be treated as basic translation unites even though they may not begin and end at word boundaries. The proposals make crucial use of suffix trees as a way of identifying repeated substrings of the texts that occur significantly often.

It is fitting that one should take advantage of the few occasions on which one is invited to address an important and prestigious professional meeting like this to depart from the standard practice of reporting results and instead to seek adherents to a new enterprise, even if it is one the details of which one can only partially discern. In this case, I intend to take that opportunity to propose a new direction for a line of work that I first became involved in in the early 1990's [3]. It had to do with the automatic alignment of the sentences of a text with those in its translation into another language. The problem is non-trivial because translators frequently translate one sentence with two, or two with one. Sometimes the departure from the expected one-to-one alignment is even greater. We undertook this work not so much because we thought it was of great importance but because it seemed to us rather audacious to attempt to establish these alignments on the basis of no *a priori* assumptions about the languages involved or about correspondences between pairs of words.

As often happens, it turned out that what we thought of as new and audacious was already "in the air" and, while we a were at work, Gale and Church[2] published a solution to the problem that was arguably somewhat less accurate than ours, but was altogether simpler, computationally less complex, and entirely adequate for practical purposes. Whereas their approach was based on nothing more than the lengths of the sentences, in terms either of characters or words, ours made hypotheses about word alignments on the basis of which it circumscribed the space of possible sentence alignments. It then refined the initial set of word alignments, and proceeded back and forth in this manner until no further refinements were possible. Fortunately the process converged fast.

In the relatively short time since this work was done, sentence alignment has come to be seen as a central tool in work on machine translation. During this time the perception has grown that the rules that direct the operation of a machine-translation system should be derived automatically from existing translations rather than being composed one by one by linguists and system designers. The first step in just about any such learn-

ing procedure is to set the sentences in correspondence with one another because sentences continue to be seen as providing the framework within which translation is done.

The natural second step is to attempt an alignment of the parts of an aligned pair of sentences as a prelude to proposing some kind of translation rule. A great many approaches to this problem have been proposed, but they fall naturally into two classes depending on whether the aligned sentence sequences that they work on are analysed in some way, perhaps by associating a syntactic structure with them, or whether the work continues to be done entirely on the basis of strings. The former approach has the drawback that disambiguation of the output of a syntactic parser is still expensive, unreliable, or both. On the other hand, it suggests a much broader set of finer tools for alignment below the sentence level than are available for working simply on strings. Suppose that it has been established beyond reasonable doubt that a pair of nouns should be aligned with one another, and that each is modified by a single adjective which may not, however, occur next to it in the string. Clearly, this constitutes stronger evidence for the alignment of the adjectives than would be possible purely on the basis of the strings. More simply put, the hypothesis that phrases translate phrases is not only a strong and appealing one, but it in fact underlies a great deal of the work that has been done on machine translation.

It goes without saying that natural languages provide innumerable examples of translation units that almost certainly could not be reliably identified without recourse to the syntactic structure of the sentences in which they occurred. Discontinuities are perhaps the most obvious example. English *take ... into consideration* presumably often constitutes a unit for translation, but only when the gap is filled with a noun phrase. Separable verbs in German and their analogues present a similar problem, as do their analogues in English, namely verbs involving particles. Some less severe problems might be made less severe by working with a string of lemmas or with tagged text. In many languages, a sequence that should be treated as a translation unit, like French *carte orange* (‘subway pass’; literally ‘orange card’) supports changes in the middle of the string. Thus, the plural of *carte orange* is *cartes oranges*. This problem is exacerbated in other languages, such as German, where various different changes are possible, depending not only on number, but also on gender.

While phrases clearly do often translate phrases, it is equally clearly the case that substrings of a sentence that do not constitute phrases can also function as units for translation. The English preposition *as* often corresponds to *au four et à mesure* in French, but this is at best a compound preposition and no phrase. The words *one and the same* might be called fixed phrase in everyday parlance, but they do not constitute a phrase in linguistic terms. The French word *connaître* translates into English in many ways, including *know* and *get to know*, the second of which could hardly be regarded as phrase. The question of how to identify sequences like these quickly and easily, especially during the early phases of working on a new language, is one that is surely worthy of attention.

The question of how to identify translation units that consist of several words is one side of a coin whose other side concerns translation units that are smaller than a word or whose end points are not marked by word boundaries. The most obvious examples of these are the components of compound nouns in a language like German in which

these components are not separated by spaces. In a language in which word boundaries are not routinely represented in text, the problem becomes acute.

The approach to these problems that is advocated, though only partially worked out, in this paper, is to place word boundaries on an equal footing with all other characters. In particular, it gives to no special status to translation units whose boundaries correspond to word boundaries.

If translation units are not to be sought between word boundaries, the first question that arises is, what substrings are considered as candidates for this status? The possibility of giving all substrings equal status can presumably be excluded on the grounds that there are simply too many of them. A text consisting of n characters contains $n(n+1)$ substrings. If the strings are confined within sentences and a string contains m sentences of k characters each, then there are $mk(k+1)$ substrings. This is more manageable. A 50-character sentence contains 1275 substrings, and a-200 character sentence 20,100 and a 500-character sentence 125,250. But the number of substrings that actually needs to be considered is a great deal less.

The substrings that need to be considered as candidates as translation units should surely have at least two properties. The first is that they should occur enough times in the text to make them interesting. The second is that, if some set of the substrings are distributionally indistinguishable, then they should be treated as the same.

The first of these properties is fairly straightforward and uncontroversial. The second can be made clear with one or two examples. Consider the substrings “uncontroversia” and “uncontroversial” of the first sentence of this paragraph. The first is clearly found wherever the second is because it is a prefix of the second. But the second will probably be found, in just about any text of English, wherever the first is, because there is, I suppose, no word of English containing the letters “uncontroversia” in which they are not followed by “l”. The distributions of the two strings will therefore be the same and each will therefore be equivalent to the other from the point of view of an investigation which, like the present one, gives not special status to word boundaries. Notice that, simply by engaging in a brief discussion of these example, we have made the present text an exception to the supposed rule. In particular, this text contains to instances of the sequence “uncontroversia” followed, not by “l”, but by quotation marks. Another member of this equivalence class, at least in most texts, will likely be the string “ncontroversial” because, at least outside this sentence, it will almost always follow the letter “u”. Other members of the class are “unconversi”, and “uncontrovers”, but not, of course “controversial”.

A fact that is not immediately obvious is that the number of equivalence classes exemplified in a text of n characters, even if sentence boundaries are ignored, can be no more than $2n$-1. This, as we shall see, is direct consequence of the fact that there are n terminal nodes, and at most n-1 branching nonterminal nodes in a *suffix tree* constructed from a string of n characters.

Suffix trees constitute a well understood, extremely elegant, but regrettably poorly appreciated data structure with potentially many applications in language processing. For a gentle introduction, see [4]. We shall have space here only for the briefest of introductions, which we begin by considering a related but somewhat simpler structure

called a *suffix trie*. A *trie* [1] is a very well-known data structure often used for storing words so that they can be looked up easily. It is a deterministic finite-state automaton whose transition diagram has the shape of a tree and in which the symbols encountered on a path from its initial state to a final state spell out a word. A suffix trie is a trie built, not from words in the usual lexicographic sense, but from the strings that are the suffixes of some text. If the entire text consists of the word "mississippi", the corresponding suffix tree is the one depicted in Figure 1. In this diagram, the labels on every path from the start state at the root of the tree, on the left of the diagram, to a terminal node on the right spells out some suffix of the text. Since every substring of the text is a prefix of some suffix, it follows that every substring is spelled out by the characters on the path from the root to some node, terminal or nonterminal. An extra character (*$*), know not to occur otherewise in the text, is added to the end for technical reasons.

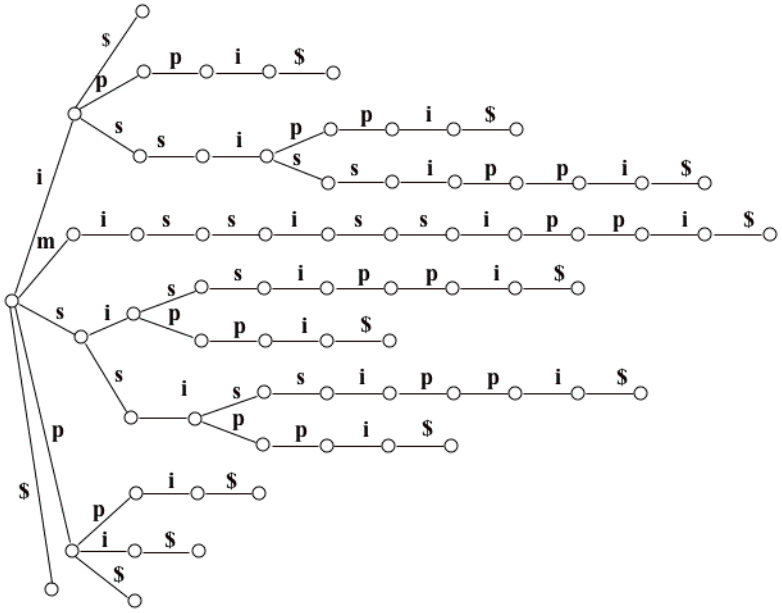

Figure 1 A Suffix Trie

Before moving from suffix tries to suffix trees, let us pause to verify that the tries already have one of the properties of "interesting" substrings of a text. A branching nonterminal node is one out of which there is more than one transition. Any substring that ends at such a node can be continued in more than one way, that is, by at least two different characters, thus meeting one of the requirements of an "interesting" substring. It follows that there can be no more "interesting" substrings of the text than there are branching nodes in the suffix tree constructed from it. As we shall see in a moment, the corresponding constraint can be verified at the left hand end of the string just as readily.

The branching nodes in the suffix trie have a special interest for us and it is worth noting that there must be strictly less of these than there are suffixes. This follows im-

mediately from the following pair of observations. First, when the initial suffix is placed in the tree, no branching nodes are created. Secondly, the entry into the trie of each succeeding suffix can give rise to at most one new branching node.

The trouble with suffix trees is that they are totally impractical for large texts because of their sheer size and the time it takes to build them. However, these things are suddenly brought under control in the passage from tries to trees. This is accomplished by replacing every sequence of transitions and nonbranching nodes by a single transition labelled with the corresponding sequence of characters. This sequence, however, is represented, not by the characters themselves, but by a pair of integers giving the end points of an occurrence of the string in the text. Nothing is lost by representing the tree in this way, because points represented by erased nodes can be reconstructed trivially, and pairs of integers constructed to represent the nodes in and out of them. The results of performing this operation on the trie in Figure 1 are shown in Figure 2.

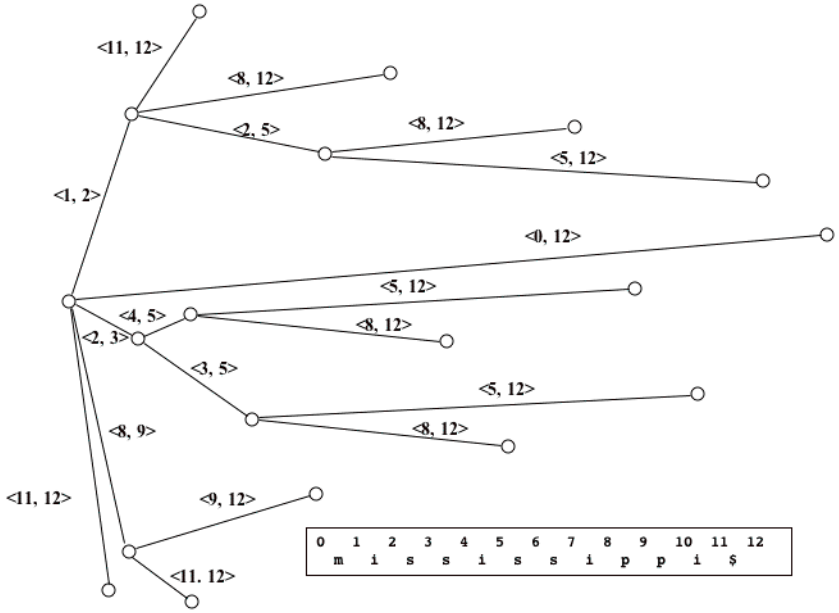

Figure 2 A Suffix Tree

The importance of this transformation cannot be overstated. The upper bound on the number of transitions in the original trie was $k(k+1)$ for a text of k characters. This has been reduced to k-1, and the size of the label on a transition remains constant. The size of the tree is therefore linear in the size of the text. A somewhat surprising fact that we shall not be able to go into here is that methods also exist for constructing these trees in linear time ([5] and [6]).

A few words are in order on the question of how to read substrings out of the suffix tree represented in this particular way. It is generally necessary, when tracing a path

from the root to a particular other node in the tree, to keep track of the length of the string traversed. This is simply a matter of adding the difference between each pair of text pointers encountered to a running count. The starting point of the string ending in a particular transition is the second of the numbers labelling that transition minus the current count. The various occurrences of the substring ending at a particular node, whether branching, and therefore "actual" or non branching, and therefore "virtual", can be enumerated straightforwardly and in a time that depends on the number of them that there are in the text and not on the actual length of the text. The method of doing this depends on the following observation: Just as the beginning of the substring corresponding to a sequence of arcs starting at the root can be determined by keeping track of the lengths of the segments covered by each transition, so the end points of the various occurrence of the substring represented by a particular node can be determined by keeping track of the lengths of the segments from the node to the various terminal nodes reachable from it.

Consider the string "si" which traces out a path through the tree shown in Figure 2 over the edges <2, 3> and <4, 5>. The string is of length 2 and its first occurrence in the text is at position 5 -(5-4+3-2)=3. The complete set of its occurrences are found by continuing the search to all possible—in this case 2—terminal nodes. The lengths of the suffixes of the text covered in tracing out these two paths, including the substring itself are (3-2)+(5-4)+(12-5)=9 and (3-2)+(5-4)+(12-8)=6. These are the lengths of the suffixes of the text that begin with the string "si" so that their locations in the text are 12-9=3 and 12-6=6. Now suppose that the substring of interest is "s", reachable from the root over the single transition <2, 3> and with paths to terminal nodes of lengths (3-2)+(5-4)+(5-12)=9, (3-2)+(5-4)+(12-8)=4, (3-2)+(5-3)+(12-5)=10 and (3-2)+(5-3)+(12-8)=7. This shows that it has four locations in the text at positions 12-9=3, 12-4=8, 12-10=2 and 12-7=5. Notice that, if a substring occurs in k locations, finding them in this way involves visiting no more than $2k$-1 nodes, once the first occurrence has been located.

Since it is computationally inexpensive to locate all the instances of a string in a text using a suffix tree, it is also inexpensive to determine whether these instances are all preceded by the same character, thus making the "uninteresting" for present purposes. In fact let us assume that the process of building a suffix tree for those purposes includes a step in which a bit is associated with every node to show whether the corresponding substring of the text has this property. We are now in a strong position to locate, in linear time, all substrings of an arbitrary text which, for given integers L and R,

1. consist of at least L characters,
2. are repeated at least R times,
3. are not always preceded by the same character,
4. are not always followed by the same character, and
5. that do not cross a sentence boundary.

These will be our initial candidate translation units.

If we were to abandon all further concern for computational complexity at this point, the plan for completing at least the first version of the project would be fairly straightforward. It would consist in aligning the sentences of a text and its translation in

another language, locating “interesting” substrings of the two text and then evaluating the similarity of pairs “interesting” strings in terms of the overlap in the sets of aligned sentences in which they occur. Let A be the set of sentences in which one of the strings occurs, and B the set in which the other string occurs. A reasonable measure of similarity might by $|A \cap B|/|A \cup B|$ which has values v in the range $0 \leq v \leq 1$, the value 1 representing identity, and 0 representing disjointness. The only problem with the plan, at least in this simple form, lies in the fact that its time complexity is $O(mn)$, if m and n are the lengths of the two texts. Unfortunately, this represents not simply the cost of an extreme worst case. It is also a reasonable estimate of the average situation, in the absence of any *a priori* knowledge of which pairs of strings are likely to align well.

While we have no proposal reducing the worst-case complexity, there is some hope of substantially reducing the constants involved in determining the computational cost. With this in mind, we propose to increase the amount of information carried by a node in the suffix tree. We will add a new pointer from each node to its parent in the tree and a field, which we call the *count* field, capable of accommodating an integer. We will also construct an index with an entry for every (aligned) sentence, containing pointers to the terminal nodes of the suffix tree corresponding to suffixes of the text that begin in a given sentence. Using this index, and the parent pointers, we can quickly visit all the substrings of the text that occur in a given sentence. In particular, given a particular set of sentences, we can cheaply populate the counter fields of the nodes to show how many of them the corresponding string occurs in. Crucially, we can do this while visiting only nodes for strings that occur in at least one of the sentences. Notice that, if this procedure assigns to non-zero value to the count at a given node, it will also assign a non-zero value, and indeed at least as high a value, to all nodes on paths from the root to that node. Once the values have been assigned, it will therefore be possible to traverse the tree, visiting only nodes for strings that occur in the current set of sentences.

With these mechanisms in place, we are in a position to proceed as follows. Conduct traversal of the suffix tree of the first text to locate “interesting” strings and obtain for each of these, the list of the sentences in which it occurs. Using the sentence index and the parent pointers of the other tree, assign value to the count fields of the strings that occur in the corresponding sentences of the other language. We now conduct a search for “interesting” strings in the suffix tree for the second text, limiting the search to nodes whose count fields contain non-zero values. Not surprisingly, this limitation is very significant if the string in the first language is indeed “interesting”.

The techniques are manifestly still in need of considerable refinement. Experiments are still in a very early stage so that it would be fruitless to give statistics. But there is some good news, both for engineers who want results, and scientists who want more problems to solve. Much of the bad news can be ascribed to the fact that the only texts that have been involved in the experiments consist of some 600 sentences from a single automobile maintenance manual and its French and German translations.

A consequence of working with too little data is that strings are allowed to qualify as “interesting” when they are in fact too long. This is because, in this particular text, for example, all instances of the word *warning* are followed by *light* and there are no instances of *warn* in any but the present-participle form. But light remains “interesting” because it is not always preceded by *warning*. The English *warning* was set in corre-

spondence with the German *Kontroll* in the compound *Kontrolleuchte.* Indeed there was an encouraging measure of success in properly aligning the parts of German compounds. When words are chosen as "interesting" substrings, they generally have an initial space attached. Whether they also have a final space depends of whether they have been observed at the end of a sentence, or otherwise before a punctuation mark. The word *coolant* is not "interesting" because it occurs only in *coolant in the engine* which is aligned with the French string *du liquide de refroidissment dans le moteur*. The inclusion of the initial *du* is an artifact of the very small number of contexts in which the string was found.

When the parameters were set so as to allow very short words to count as "interesting", correspondences were recognized even among function words and inflexions. The English suffixes "ic ", and "ly " (with the trailing space) were aligned with French "ique" and "ment ". The French sequence " les " (with spaces on both ends) was aligned with English "s " (with a trailing space). This last alignment presumable reflects that fact the French grammar requires more frequent use of the definite article than English does so that " les " more readily contracts an alignment with the plural ending of the noun. It may also have something to do with the fact that *les* occurred nowhere in this text as an object clitic.

Encouraging though these preliminary results are, they also reveal important shortcomings of the technique and challenges for the future. The most important of these will almost certainly involve additional procedures with unappealing complexity properties. As usual with ordinary language, ambiguity, or at least *relative ambiguity*, is at the heart of the severest of the problems.

A string in one language is ambiguous relative to another language if it is routinely translated differently in that other language, depending on the context. Simplifying somewhat, we can say that French *haut* translates into English as either or *high* or *tall.* If one of these adjectives were somehow replaced by the other in a English text before the alignment process began, there would be good reason to hope that a strong alignment with French *haut* would be recognized. But there appears to be no basis to make any such conflation of words, especially within a system in which the very notion of a word is so carefully deemphasized. The general problem is to be able to recognize situations in which a distribution that is abundantly endowed with "interesting" properties, aligns with the (disjoint) union of two or more distributions in the other language.

Another way to think about this problem involves abandoning the notion of similarity, which is inherently symmetrical, in favor of a notion of inclusion of one distribution in another. Continuing the above simplified example, this would allow an alignment to be established between French *haut* and English *high* on the grounds that the distribution of *haut* largely contains that of the English *high*. Suppose this were successful, it might then be possible to seek other alignments for just those instances *haut* that did not participate in the first alignment.

References

[1] Fredkin, E. Trie memory. Informal Memorandum. *Communications of the ACM*, 3,(9), 490--500 (1960).

[2] William A. Gale and Kenneth W.Church A program for Aligning Sentences in Bilingual Corpora. *Computational Linguistics* 19 (1) pp. 61-74, 1993

[3] M. Kay & M. Röscheisen Text-Translation Alignment, *Computational Linguistics* 19:1, 1995

[4] Nelson, Mark. Fast String Searching With Suffix Trees, *Dr. Dobb's Journal* August, 1996.

[5] McCreight, E.M. . A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23:262-272, 1976.

[6] Ukkonen, E. On-line construction of suffix trees. Algorithmica, 14(3):249-260, September 1995.