Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Philadelphia, July 2002, pp. 289-295.
Association for Computational Linguistics.

Processing Comparable Corpora With Bilingual Suffix Trees

Dragos Stefan Munteanu and **Daniel Marcu**
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
{dragos, marcu}@isi.edu

Abstract

We introduce Bilingual Suffix Trees (BST), a data structure that is suitable for exploiting comparable corpora. We discuss algorithms that use BSTs in order to create parallel corpora and learn translations of unseen words from comparable corpora. Starting with a small bilingual dictionary that was derived automatically from a corpus of 5.000 parallel sentences, we have automatically extracted a corpus of 33.926 parallel phrases of size greater than 3, and learned 9 new word translations from a comparable corpus of 1.3M words (100.000 sentences).

1 Introduction

Current research in statistical machine translation based on the work of Brown et al. (1993) relies heavily on the existence of parallel corpora for the estimation of translation model parameters. Unfortunately, such corpora are available only in limited amounts and cover only specific genres (Canadian politics, Hong Kong laws, etc). However, monolingual texts exist in higher quantities and in many domains and languages. Methods for processing such resources can therefore greatly benefit the field.

Previous work on processing non-parallel, comparable corpora has focused mostly on learning word-level translations. Some researchers focused on discovering translations of new words (Rapp, 1995; Rapp, 1999; Fung and Yee, 1998; Diab and Finch, 2000). Others worked on choosing between several translation alternatives (Kikui, 1999; Koehn and Knight, 2000). In order to learn to discriminate between several word senses, these approaches model the contexts in which the words under consideration occur.

The context is usually represented as co-occurrence information. That is, for each word of interest one counts the occurrences, in its vicinity, of each element of a previously defined set of "context words". These counts define the context vector of the words under consideration. The common assumption in these approaches is that words which are translations of each other will have similar context vectors. The main differences between these approaches lie in the choice of context words and in the definition of similarity. For example, Rapp (1999) and Fung and Yee (1998) use as context words the entries from a small bilingual lexicon; Diab and Finch (2000) use the top N most frequent words in the corpus; and Kikui (1999) considers only "content bearing words".

The goal of our research goes beyond learning word-level translations from comparable corpora, as we not only learn translation of unseen words but also build parallel corpora. Our approach to context processing is not limited to co-occurrences but rather it takes into account the full literal context. A high-level description of our approach is shown in Figure 1.

Given a small bilingual lexicon and a comparable bilingual corpus we automatically construct a bilingual suffix tree (BST). The BST represents in linear space *all* substring alignments between the two corpora given in the input. From this representation we extract phrase and sentence alignments in order to produce a parallel corpus. We also present an algorithm that uses the BST in order to learn word translations of unknown words.

2 Bilingual Suffix Trees

2.1 Suffix Trees

A suffix tree is a data structure that stores in linear space all suffixes of a given string. Such

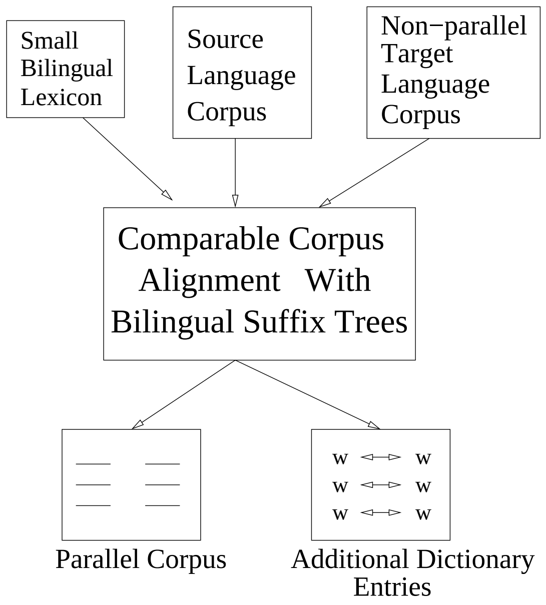
Small Bilingual Lexicon
Source Language Corpus
Non−parallel Target Language Corpus
Comparable Corpus Alignment With Bilingual Suffix Trees
Parallel Corpus
Additional Dictionary Entries
w w
w w
w w

Figure 1: Schema of the Corpus Alignment System

succinct encoding exposes the internal structure of the string, providing efficient (usually linear-time) solutions for many complex string problems, such as exact and approximate string matching, finding the longest common substring of multiple strings, and string compression. Suffix trees have originally been introduced by Weiner (1973) but since then have been rediscovered many times in the literature, under different names (Grossi and Italiano, 1993). A thorough discussion of this data structure and its applications is given by Gusfield (1997).

Formally, a suffix tree for a string S of length N has the following properties (Gusfield, 1997):

- Each edge of the tree is labeled by a non-empty substring of S.
- Each internal node other than the root has at least 2 children.
- No two edges out of a node can have edge-labels beginning with the same character/word.
- The key feature of the tree is that there is a one-to-one correspondence between *all* suffixes of S and paths in the tree from the root to the leaves.

Figure 2 shows the suffix tree of string

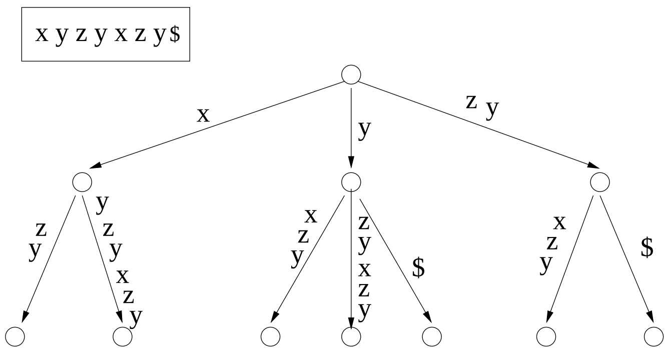

Figure 2: Suffix tree for string `xyzyxzy`

`xyzyxzy`.

Note that if a suffix of a string is also a prefix of another suffix (as would be the case for suffix `zy` of string `xyzyxzy`) we cannot build a proper suffix tree for the string. The problem is that the path corresponding to that suffix would not end at a leaf, so the tree cannot have the last property in the list above. To avoid this, we always append to our strings an end-of-string marker that appears nowhere else in the string, which we denote by $. For clarity, our examples only show the $ marker when necessary.

Since we are interested only in sentence level alignments, we divide each monolingual corpus given as input into a set of sentences. We then use a variant of suffix trees that works with sets of strings, namely Generalized Suffix Trees (GST). In a GST of a set of strings, each path from the root to a leaf represents a suffix in one *or more* strings from the set. A conceptually easy way to build such a tree is to start by building a regular suffix tree for the first sentence in the corpus, and then for each of the other sentences to take their suffixes one by one and add them to the tree (if they are not already in it). Figure 3 shows the GST for a corpus of two sentences. The numbers at the leaves of the tree show which sentences contain the suffix that ends there.

Building the suffix tree of a string takes time and space linear in the length of the string. (Ukkonen, 1995; Nelson, 1996). Building a GST for a set of strings takes time and space linear in the sum of the lengths of all strings in the set (Gusfield, 1997).

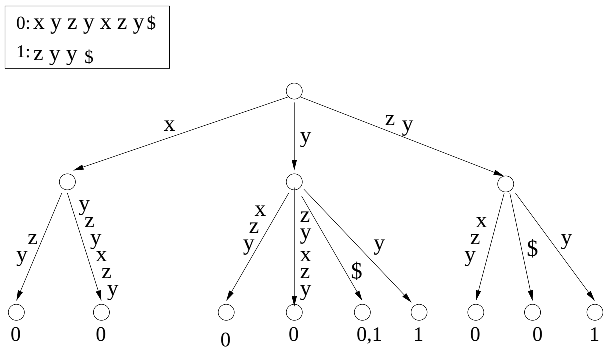

Figure 3: The GST for strings {xyzyxzy, zxy}.

2.2 Bilingual Suffix Trees

A Bilingual Suffix Tree is the result of matching a source language GST against a target language GST. Two strings (i.e. sequences of words) match if the corresponding words are translations of each other according to a bilingual lexicon. In order to perform the matching operation, all paths that correspond to an exhaustive traversal of one of the trees (the source tree) are traversed in the other (the target tree), until a mismatch occurs. In the process, the target tree is augmented with information about the alignments between its paths and those of the source, thus becoming a bilingual suffix tree. Figure 4 shows two corpora, a bilingual lexicon, and the corresponding BST. Edges drawn with dotted lines mark ends of alignment paths through the tree. Their labels are (unaligned) continuations of the source language substrings from the respective paths.

Since there is a one-to-one correspondence between the substrings in the text and the paths in the suffix trees, the operation described above will yield all pairs of substrings in the two corpora given as input and discover all partial monotone alignments defined by the lexicon.

If the lexicon is probabilistic, each matching between two words will be weighted by the corresponding translation probability. The paths in the resulting bilingual tree will also have weights associated with them, defined as the product of the matching probabilities of the words along the path.

Our matching operation is similar to that described by Bieganski et al. (1994), differing only in the matching operator and the structure of the resulting tree. Bieganski et. al match trees that are defined over the same alphabet, therefore two strings match only if they are identical. The result is another suffix tree over the same alphabet, which encodes all the common subsequences. In contrast, our trees are defined over different alphabets, and the matching is defined by the bilingual lexicon; in particular, a given sequence in one tree can (and usually does) match with several sequences from the other tree, which increases the complexity of the operation.

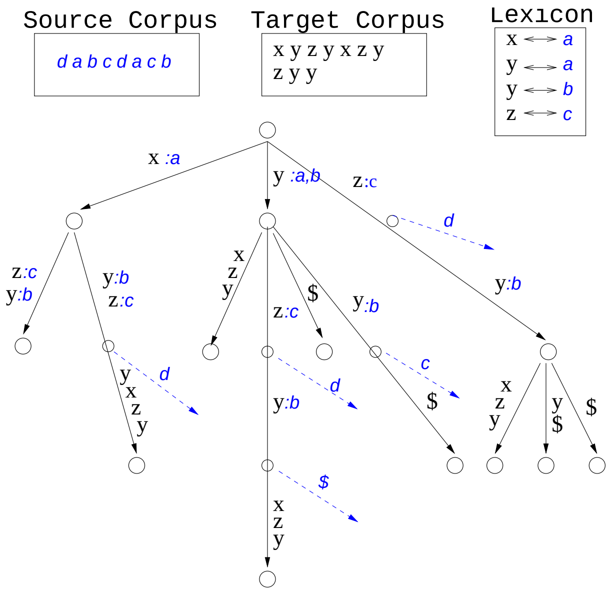

Figure 4: Bilingual Suffix Tree.

3 Extracting a parallel corpus from a comparable one

BSTs are constructed to encode alignment information, therefore the extraction of parallel phrases amounts to a simple depth-first traversal of the tree. Figure 5 shows some alignments we can extract from the BST in Figure 4, a portion of which is shown in Figure 5.

As can be seen in Figure 4, there are three types of edge labels in a BST: only target language sequences (e.g. `xzy`), pairs of target and source language sequences (`y:b` followed by `z:c`) and only source language words (`b` or `c`). For alignment extraction we are interested in edges of the third type, because they mark ends of alignments. Let `e` be an edge labeled only with a source language word, originating from node `n`. A path from the root to `n` will only tra-

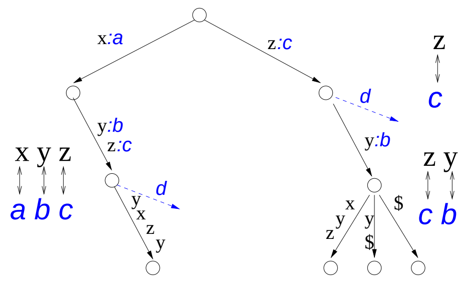
Figure 5: Example alignments

verse edges labeled with word pairs, defining two aligned sequences. The fact that `n` has outgoing edge `e` indicates there is a mismatch on the subsequent words of those two sequences.

Thus, in order to extract all aligned substrings, we traverse the BST on edges labeled with word pairs, and extract all paths that end either at the leaves or at nodes that have outgoing edges labeled only with source language words.

4 Learning translations

The heuristic by which we discover new word translations is shown graphically in Figure 6 and explained below. Figure 6.i shows a branch of the BST corresponding to the comparable corpus in the same figure. The path defined by the bold edges shows that sequences `xyz` and `abc` are aligned, and diverge (i.e. have a mismatch) at characters `y` and `d` respectively. We take this as a weak indication that `d` and `y` are translations of each other. This indication would become stronger if, for example, the sequences following `d` and `y` in the two corpora would also be aligned. One way to verify this is to reverse both strings, build a BST for the reversed corpora (a reverse BST), and look for a common path that diverges at the same `d` and `y`. Figure 6.ii shows the reverse BST, and in bold, the path we are interested in. When `d` and `y` are surrounded by aligned sequences, we hypothesize that they are translations of each other.

For a pair of words from the two corpora, we use the terms *right alignment* and *left alignment* to refer to the aligned sequences that precede and respectively succeed the two words in each corpus. The left and right alignments and the two words delimited by them make up a *context alignment*. For example, the left alignment `xyz-abc`, the right alignment `xzy-acb` and the words `y` and `d` in Figure 6.iii make up a context alignment.

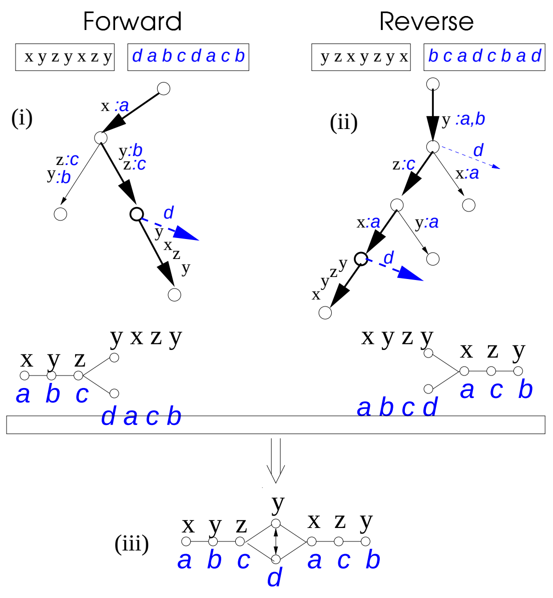

Figure 6: Discovering new word translations

Given a comparable corpus, this procedure will yield many context alignments which correspond to incorrect translations, such as that between the words *canadien* and *previous*:

tout *canadien* serieux
any *previous* serious

In order to filter out such cases, we use two simple heuristics: length and word content. Thus, for a context alignment to be *valid*, the left and right context together must contain at least 3 words, one of which must be an open-class word. The translation candidate must also be an open-class word.

The algorithm for learning translations of unknown words, which we explained in this section, is summarized in Figure 7. A major advantage of our algorithm over previous approaches is that we do not provide as input to the algorithm a list of unknown words. Instead, we automatically learn from the corpus both the unknown words and their translation, upon discovery of appropriate context alignments.

5 Experiment

We tested our system on an English-French comparable corpus, of approximately 1.3 million

```
1. Build the forward and backward BSTs.
2. Traverse each BST and extract left
   and right alignments for every node
   that represents a divergence
   For each word pair from the divergence set:
      a. create context alignments out of
         appropriate left and right alignments
      b. filter out invalid context alignments
      c. extract valid translation candidates
         from the context alignments
```

Figure 7: Algorithm for learning translations of unknown words

words — 50.000 sentences for each language. We obtained it by taking two non-parallel, non-aligned segments from the Hansard corpus. We also used GIZA[1] to automatically build a small bilingual lexicon of 6.900 entries using 5.000 sentences pairs (150.000 words for each language). The parallel corpus was taken from the Proceedings of the European Parliament (EuroParl). Note that the parallel corpus belongs to a different domain than the comparable corpus. Also, the parallel corpus is extremely small. For low-density languages, such a corpus can be built manually.

When given as input the comparable corpus described above and the bilingual lexicon of 6.900 entries, the algorithm described in Section 3 found 33.926 parallel sequences, with lengths between 3 and 7 words (we do not report here aligned sequences of less than 3 words). Out of 100 randomly selected sequences, 95% were judged to be correct. Some examples of the discovered alignments are shown in Figure 8.

The system also found translations for 30 unknown French words. Of these, 9 are correct, which means a precision of 30%. Figure 9 shows some proposed translations and the context alignments on which they are based.

For each of the two corpora, building the monolingual GST took only 1.5 minutes. The matching operation that yields the BST is the most time-consuming: it lasted 38 hours for the forward BST and 60 hours for the reverse BST. The extractions of all parallel phrases and of the translations took about 2 hours each. We ran the experiments on a Linux system with a Pentium 3 processor of 866Mhz.

[1]http://www.clsp.jhu.edu/ws99/projects/mt/

```
même   temps   ,  le   gouvernement   réduit   le
same   time    ,  the  government     reduced  the

mon   avis     ,  ce    est   très   important
my    opinion  ,  this  is    very   important

et    que    nous   sommes   disposés  à    prendre
and   that   we     are      prepared  to   take

,  en   fait  ,  le
,  in   fact  ,  the

(  1  )  ,  le
(  1  )  ,  the

par   mes   collègues   et    moi
by    my    colleagues  and   myself

partout     dans  le    monde
everywhere  in    the   world

pression  sur  le    gouvernement
pressure  on   the   government

que   ce    qui
that  for   which

toutes  les   personnes
all     the   people

doute      ,  le
certainly  ,  the
```

Figure 8: Examples of parallel phrases

CORRECT	
communément = commonly	il est *communément* accepté it is *commonly* accepted
bien = also	subventions ont *bien* été subsidies have *also* been
immensément = particularly	il est *immensément* important it is *particularly* significant
INCORRECT	
plaintes - meetings	ces *plaintes* ont déjà these *meetings* have already
renseignements - units	nouveaux *renseignements* sont disponibles new *units* are available
exemptions - documents	ces *exemptions* ont été these *documents* have been
exemptions - laws	ces *exemptions* ont été these *laws* have been

Figure 9: Proposed translations

6 Discussion and Future Work

GST and BST construction

The most important limitation of our method is that it can find and exploit only word alignments that are monotonic. This makes the system applicable primarily on language pairs which have similar word order, such as English-French and English-Chinese.

A second, less severe limitation concerns the scalability of the algorithms. The GST and BST derivation algorithms that we implemented to date are not the most efficient. For example, Kurtz (1999) shows how to reduce space requirements of suffix trees. And Farach (1997) and Andersson et al. (1999) present construction algorithms which, for trees defined over large alphabets such as words, are more efficient than Ukkonen's algorithm (which we used in our implementation). The matching operation can be parallelized with linear speedup, since matching of a pair of branches is independent of the matching of any other pair. We plan to incorporate these improvements in future versions of our system.

Learning new translations

The translation precision that we obtained is lower than that reported in previous approaches. However, we attempted to solve a much harder problem: our algorithm does not take as input the list of unknown words, but learns automatically from the corpus both the unknown words and their translations. Therefore our results depend both on the degree of parallelism of the two corpora, and on their size. We expect that as we scale the algorithm to process corpora of billions of words, our precision will improve.

As mentioned in section 4, our algorithm for learning new translations yields many context alignments, most of them corresponding to incorrect translations. Filtering out these incorrect translations is an important part of the algorithm. Our current filtering method, which is based on the length and content of the context alignment, is rather simple. The co-occurence methods used in the previous approaches could be of help here, by providing additional sources of evidence about the translations defined by our context alignments.

Bootstrapping

It is clear that our algorithm can bootstrap itself using the learned word translations. The additional word alignments could allow us to find more and longer parallel sequences, and thus better context alignments out of which will come yet more new translations. At the time of submission we have not yet implemented this bootstrapping procedure.

In general, we find BSTs to be an extremely useful data structure that we believe will be of great use to other natural language researchers interested in aligning sequences defined over different alphabets.

References

A. Andersson, N.J. Larsson, and K. Swanson. 1999. Suffix trees on words. *Algorithmica*, 23(3), September.

Paul Bieganski, John Riedl, John Carlis, and Ernest Retzel. 1994. Generalized suffix trees for biological sequence data: Applications and implementation. In *Proceedings of the 27th Annual Hawaii International Conference on System Sciences*, volume 5, pages 35–44. IEEE.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, June.

Mona Diab and Steve Finch. 2000. A statistical word-level translation model for comparable corpora. In *Proceedings of the Conference on Content-Based Multimedia Information Access.*

Martin Farach. 1997. Optimal suffix tree construction with large alphabets. *38th Annual Symposium on Foundations of Computer Science*, pages 137–143, October.

Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *"Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics*, pages 414–420.

Roberto Grossi and Giuseppe F. Italiano. 1993. Suffix trees and their applications in string algorithms. In *Proc. 1st South American Workshop on String Processing*, pages 57–76, September.

Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences.* Cambridge University Press, New York.

Genichiro Kikui. 1999. Resolving translation ambiguity using non-parallel bilingual corpora. In *Proceedings of ACL99 Workshop on Unsupervised Learning in Natural Language Processing.*

Philipp Koehn and Kevin Knight. 2000. Estimating word translation probabilities from unrelated monolingual corpora using the EM algorithm. In *Proceedings of the National*

Conference on Artificial Intelligence, pages 711–715.

Stefan Kurtz. 1999. Reducing the space requirement of suffix trees. *Software - Practice and Experience*, 29(13):1149–1171.

Mark Nelson. 1996. Fast string searching with suffix trees. *Dr. Dobb's Journal*, August.

Reinhard Rapp. 1995. Identifying word translation in non-parallel texts. In *Proceedings of the Conference of the Association for Computational Linguistics*, pages 320–322.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of the Conference of the Association for Computational Linguistics*, pages 519–526.

Esko Ukkonen. 1995. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, September.

P. Weiner. 1973. Linear pattern matching algorithm. In *Proc. 14 IEEE Symposium on Switching and Automata Theory*, pages 1–11.