Representing a Bilingual Lexicon with Suffix Trees

Jorge Costa
FCT-UNL
Caparica, Portugal
jorge.costa@fct.unl.pt

Gabriel Pereira Lopes
FCT-UNL
Caparica, Portugal
gpl@fct.unl.pt

Luís Gomes
FCT-UNL
Caparica, Portugal
luismsgomes@gmail.com

Luís M.S. Russo
FCT-UNL
Caparica, Portugal
lsr@di.fct.unl.pt

ABSTRACT

This paper presents a system based on generalized suffix trees that efficiently implements a set of operations over a bilingual lexicon. Besides the basic operations of adding and removing translations from the lexicon, the system provides two unique query functions that we refer to as *monolingual* and *bilingual coverage*. These two functions lay the foundation for higher-level mining operations, such as identification of translation patterns, that are the subject of ongoing research. Nevertheless, the system presented here is interesting in and by itself, for the novelty of the coverage functions and the potential of the whole data structure. We compare the performance of two implementations, one based on suffix trees and the other on suffix arrays.

Keywords

Coverage, Correspondence, Translation Pairs, Lexicon

1. INTRODUCTION

A bilingual lexicon is a set of pairs of expressions that are translations. An expression may be a single word or several words in which case we call it a multi-word expression. We designed a system to store and query a bilingual lexicon efficiently. Using generalized suffix trees [4], most operations take optimal time.

We call monolingual coverage to a function that takes as input an expression and returns a list of segments that are not covered by the lexicon of the same language. If the expression is a word, then it is covered if it is already in the lexicon. If the expression has several words, then exists full coverage if all the segments are in the lexicon. In the case of a partial coverage, the segments not covered are returned.

In terms of bilingual coverage, the input is two expressions and the output is also a list of segments not covered. The bilingual coverage follows an idea similar to the monolingual, but for both lexicons, in which an expression is searched in the respective lexicon. Then, with the segments found, we determine if these segments are translations of another segments that exist in the other lexicon. Two segments have bilingual coverage, if they follow these restrictions.

SAC'11 March 21-25, 2011, TaiChung, Taiwan.

2. REPRESENTING THE LEXICON

Our lexicon representation uses two generalized suffix trees, built using the Ukkonnen's algorithm [6].

2.1 Correspondence

To represent a bilingual lexicon, it is necessary to define correspondence links between the expressions. A correspondence link marks two expressions, from two different languages, as a translation pair, meaning that one expression is the translation of the other. Therefore, the system receives only pairs of expressions, one for each language, to insert them and to create the respective correspondence link.

2.2 Data Management

Adding new information to the system can be done in two ways. We can add a single pair or several pairs, from an input file. After the new nodes are created, the system calculates Depth-First Search Timestamps. They are important to determine if a node is ancestor of another node, in constant time. Having a node X and a node Y, X is ancestor of Y, if $X.first \leq Y.first$ and $X.second \geq Y.second$, with first and second representing the two Timestamps.

The removal of a pair does not change the structure of the trees, it only eliminates the correspondence link between two terms, in order to avoid larger complexities.

2.3 Coverage

For the monolingual coverage, it is only necessary to search for the expression in one tree. For the bilingual coverage, we have to use both trees and then check for the existence of translations pairs, between the segments found.

2.3.1 Monolingual Coverage

To determine this coverage, we use two indexes. The index i is incremented every time we can descend in the tree, while j is incremented when it is not possible to descend in the tree. In this case, we follow a suffix link to another node.

The first thing to do is to preprocess the expression to search, adding a space in the beginning and a $ at the end. Then, a segment of the expression is covered, if i and j are in a space or $, or if the accumulated spaces between the two

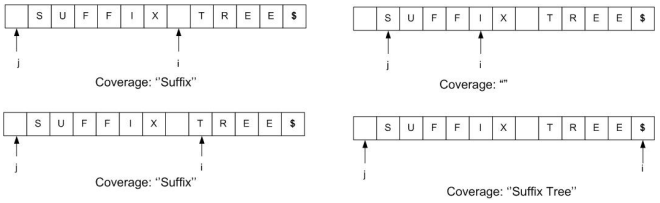

Figure 1: Examples of the use of indexes i and j

indexes are two, which happens if i was recently in a space or in \$ and j is in a space. Figure 1 shows an example.

2.3.2 Bilingual Coverage

To determine if two expressions have bilingual coverage, it is necessary to make a search, much like in the monolingual coverage, in both trees. With the nodes from the searches result, we check for the correspondences. A pair has bilingual coverage, if a correspondent of a node representing one of the expressions, is ancestor of a node from the other tree.

This is the only operation without linear time consumption, due to its complexity. We can avoid quadratic times, by checking for an ancestor in constant time and by using binary searches.

3. CONTEXT

To obtain the bilingual lexicon[1] needed for Phrase-Based Statistical Machine Translation we apply an iterative process with three steps: (1) alignment of parallel texts, (2) extraction of word and multi-word translations from aligned texts, and (3) human verification of the correctness of the extracted translations. As a result of this process we have a bilingual lexicon that grows at each iteration.

We take advantage of this growing lexicon to produce alignments with increased precision at each iteration [3], which in turn enable the extraction [1] of word and multi-word translations that were not extracted in earlier iterations, due to alignment errors. The human verification of word and multi-word translations allows us to work around the usual tradeoff between precision and recall, that is inherent to automatic translation extraction, and keeps the lexicon growing with high-quality translations.

As we see, each of these steps involves the lexicon in one way or another, thus it is important to have appropriate data structures and algorithms for storing and querying the bilingual lexicon.

4. SUFFIX TREES VS. SUFFIX ARRAYS

We tested our system, against the alternative of suffix arrays. To simulate the suffix arrays behaviour, we defined an implementation of both coverage operations, without following suffix links, leading to a constant return to the root. The expression used for tests has fifteen words, including punctuation marks and numbers.

Despite the increase of the number of terms, the operations done do not vary too much, as we can see in Table 1. This happens because the searches in suffix trees, have larger dependency on the size of the string searched.

The difference between both implementations is significant. That comes from the main difference between them, the suffix links. With these links, we follow "shortcuts" through the edges, avoiding the constant return to the root of the tree, each it is not possible to descend.

[1]usually called phrase tables in MT literature

Table 1: Bilingual Coverage Operations by Tree Size

Number of Terms	Suffix Trees	Suffix Arrays
10000	244326	25534
50000	251884	22182
100000	249987	20241
200000	242471	19302

5. CONTRIBUTIONS

The representation of the bilingual lexicon using suffix trees, permits a more flexibility in terms of defining some query functions, that are important to the entire iterative process mentioned earlier. With the coverage queries, it is possible to determine in an aligned segment, which are the parts still missing in the lexicon. Knowing the terms covered and having the segments aligned, it is easier to determine the pairs unknown. An application to infer new translations at a higher level, is an interesting possibility to develop.

Using the linear characteristics of the trees, it is easier to implement new efficient queries and to improve the ones presented, due to the suffix links.

6. CONCLUSIONS AND FUTURE WORK

The system implemented accomplishes our goals in terms of finding a way to representing a bilingual lexicon, using suffix trees, a structure not used so often in translation, due to the memory constraints that do not exist in this project. We could also prove the benefits of the structure against the suffix arrays, in the queries implementation.

In terms of future work, the system can be expanded for using more languages, leading to the need of creating more suffix trees and some auxiliary structures. To avoid the memory problem, it would be interesting to use a form of compression which would consume less memory, without losing the characteristics of the suffix trees. The compressed suffix trees [5] could be a good solution.

The definition of new operations, like the inferring of new translations, using the information stored in the lexicon and from the alignment, is also an interesting theme to explore.

7. REFERENCES

[1] Aires, G. P. Lopes, and L. Gomes. Phrase translation extraction from aligned parallel corpora using suffix arrays and related structures. In *Progress in Artificial Intelligence*, LNCS, pages 588–597, 2009.

[2] L. Gomes. Parallel texts alignment. 2009.

[3] L. Gomes, Aires, and G. P. Lopes. Parallel texts alignment. In *EPIA 2009, Aveiro*, pages 513–524. Universidade de Aveiro.

[4] D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology.* Cambridge Univ Pr, 1997.

[5] L. Russo, G. Navarro, and A. Oliveira. Fully-compressed suffix trees. *LATIN 2008: Theoretical Informatics*, pages 362–373, 2008.

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