Alignment of Shared Forests for Bilingual Corpora

Adam Meyers, Roman Yangarber, Ralph Grishman
New York University
715 Broadway, 7th Floor, NY, NY 10003, USA
meyers/roman/grishman@cs.nyu.edu

Abstract

Research in example-based machine translation (EBMT) has been hampered by the lack of efficient tree alignment algorithms for bilingual corpora. This paper describes an alignment algorithm for EBMT whose running time is quadratic in the size of the input parse trees. The algorithm uses dynamic programming to score all possible matching nodes between structure-sharing trees or forests. We describe the algorithm, various optimizations, and our implementation.

1 Introduction

The development of a machine translation (MT) system requires the lengthy manual preparation of bilingual lexicons and transfer rules. Research over the past few years using parallel sentence-aligned bilingual corpora suggests ways in which this manual effort can be partly replaced by corpus-based training. Some of this research has treated the sentences as *unstructured* word sequences to be aligned; this work has primarily involved the acquisition of bilingual lexical correspondences (Chen, 1993), although there has also been an attempt to create a full MT system based on such treatment (Brown et al., 1993). Recently, several groups have been exploring the possibility of aligning parallel *syntactically analyzed* sentences from the source and target languages (cf. (Sato and Nagao, 1990), (Klavans and Tzoukermann, 1990), (Grishman and Kosaka, 1992), (Kaji et al., 1992), (Matsumoto et al., 1993) and (Grishman, 1994)). This offers the potential for acquiring not just *lexical* but also *structural* correspondences between the two languages. The specific goal in aligning syntax trees is to identify the corresponding tree fragments in the source and target trees. By processing a substantial corpus, a large set of such corresponding fragments can be collected. These can then serve as the example base for a form of example-based MT (cf. (Nagao, 1984), (Sato and Nagao, 1990), (Kaji et al., 1992), (Matsumoto et al., 1993) and (Furuse and Iida, 1994)). This approach requires a fast tree alignment technique; research has been hampered by the lack of efficient algorithms. This paper describes an efficient algorithm for bilingual tree alignment.

2 Our Approach

For each input sentence our parser produces a set of trees, corresponding to each possible syntactic analysis. Our parse trees are transformed into a "regularized" format, to represent the Predicate-Argument structure. For each sentence, the output of the parser is a structure-sharing forest. An example of structure sharing between two parse trees of the same input sentence is shown in Figure 1. We apply the parser to the source and target sentences, using a Spanish and an English grammar, respectively. The resulting sets of structure-sharing parse trees form the input to the alignment procedure.

Our alignment program employs dynamic programming[1] algorithms, which are described in detail in later sections. The program begins at the roots of the source and target trees, and proceeds top-down recursively, filling a matrix of *scores*. Given N nodes in the source tree $T_s = T(V_s, E_s)$ [2] and M nodes in the target tree $T_t = T(V_t, E_t)$, the score matrix is an $N \times M$ matrix. For each pair of nodes $x_i, i = 1, ...N \in V_s$ and $y_j, j = 1, ...M \in V_t$, the corresponding entry in the score matrix is a measure of how well x_i matches y_j. The score for each pair of nodes depends only on the closeness of the lexical entries associated with the nodes and

[1] Cf. e.g. (Cormen et al., 1990), pp.299-328

[2] The expression $T(V_s, E_s)$ denotes a tree as a pair of sets: V_s is the set of vertices (nodes) in the tree, and E_s is the set of edges (arcs).

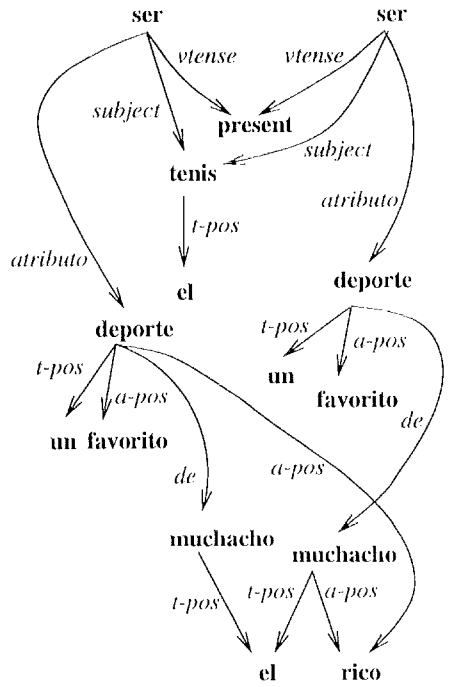

Figure 1: Two parses for: *El tenis es un deporte favorito del muchacho rico.*

on the scores of the best matching pairs of their descendants. Dynamic programming assures that each entry in the matrix, (i.e. the score for the corresponding pair of nodes) is computed only once.

We have implemented this approach for 185 sentences from two sources: *El Camino Real* (a Spanish textbook from which we used 73 sentences with their English translations), and *Curious George* or *Jorge el Curioso* (the English and Spanish versions of H. A. Ray's popular children's book, from which we used all 111 English sentences and 112 Spanish Sentences). Of the total 185 sentences, 57 from *El Camino Real* and 55 from *Curious George* produced at least one parse tree in *both* languages. The alignment procedure was applied only to these pairs of sentences.[3]

3 Data Structures

Regularized parses are similar to the F-structure of Lexical Functional Grammar (LFG), except that a dependency type structure is assumed.[4] Here a relation $Role(PRED, ARG)$ is represented by an arc, the $PRED$ of LFG at the tail of the arc, and the ARG (or the role recipient) at the head of the arc. For example, *el tenis* is the subject of the predicate *ser* in Figure 1. In a regularized parse, certain closed syntactic classes, such as prepositions and subordinate conjunctions, are represented as arc labels denoting roles, (e.g., the preposition *de* in Figure 1) rather than as nodes in F-structure.

Structure sharing among the trees in the parse forest allows us to reduce the number of computed scores. We compute the score for a given pair of subtrees only once, regardless of the number of trees which share these subtrees because the score of a pair of nodes depends only on the scores of their descendants (and not of their ancestors). Currently our parser records structure sharing only between NPs. Experiments in which all common structure is shared, as in Figure 1, suggest that extending structure sharing to other types of nodes would further improve performance. This structure-sharing approach is based on previous work in optimizing Feature Structure-based parsing. (See for example, (Karttunen, 1985) and (Pereira, 1985)).

4 The LCA-Preserving Algorithms

We first discuss the formal aspects of the alignment problem and introduce terminology.

4.1 The Maximal Tree Alignment Problem

The objective is to find a maximum-score correspondence between nodes in a pair of trees. The statement of the problem of aligning two trees[5] T_s and T_t, corresponds closely to that found in (Matsumoto et al., 1993). Our algorithms are based on those presented in (Steel and Warnow, 1993), (Farach et al., 1995b) and (1995a).

We say that a node x is a common ancestor of nodes a and b in a tree T if there exist paths of length ≥ 0 from x to a and from x to b. The least common ancestor (lca) of two nodes a and b is the node $x_0 = lca(a, b)$, such that

1. x_0 is a common ancestor of a and b, and
2. for any other common ancestor x of a and b, x_0 is a descendant of x.

An alignment between two trees $T = (V, E)$ and $T' = (V', E')$ is a correspondence (a one-to-one mapping) $f : S \mapsto S'$, where $S \subseteq V$ and $S' \subseteq V'$, which maintains the following relationship:

[3] By improving the asymptotic speed of alignment on these few sentences, we open the possibility for using much larger corpora in future work.

[4] (Sato and Nagao, 1990) and (Matsumoto et al., 1993) also assume dependency type structures in their example-based work.

[5] For simplicity of presentation we state the problem in terms of alignment of trees. In practice we are using an optimized variant of the algorithm, which aligns pairs of structure-sharing forests.

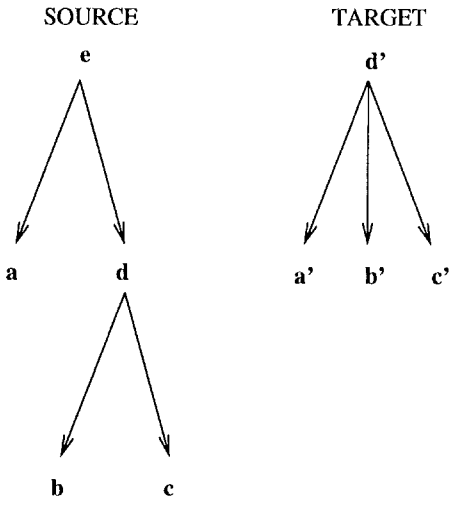

Figure 2: There is no lca-preserving alignment between a, b and c, and a', b' and c'.

> If nodes $a \in V$ and $b \in V$ map into nodes $a' = f(a) \in V'$ and $b' = f(b) \in V'$, then $f(lca(a,b)) = lca(f(a), f(b)) = lca(a', b')$

To illustrate, in Figure 2 there is no lca-preserving alignment of the two trees which maps all three of the leaf nodes a, b and c into the nodes a', b' and c'. Lca-preserving alignments are possible which map any two of the leaves.

The algorithm assumes that least common ancestors are preserved in the alignment. We assign a score to each alignment based on the labels of the corresponding nodes and the arcs from these nodes, as described below. The algorithm seeks an alignment with maximal score.

4.2 The Algorithm

Let T_s and T_t be the source and the target trees. The algorithm uses dynamic programming to build up, in a bottom-up fashion, the scores for matching each node in T_s against each node of T_t. There are $O(n^2)$ such scores, where $n = \max(|\ T_s\ |, |\ T_t\ |)$ is the number of nodes in the trees. Let $d(v)$ be the degree of a node v. We denote children of v by v_i, $i = 1, \ldots, d(v)$, and the arc (v, v_i) by $\vec{v}_i$.

Procedure $SCORE_{LCA}$: The dynamic programming builds up a score function $S(v, v')$ for all $v \in T_s$ and $v' \in T_t$, which is stored in a $|\ T_s\ | \times |\ T_t\ |$ matrix S. The value $S(v, v')$ is the score assigned to the best match between the two subtrees rooted at v in T_s and at v' in T_t. Initially, S is filled with undefined values. When a value for $S(v, v')$ is required, and the corresponding entry in the matrix is undefined, it is recursively computed by the following formula:

$$S(v,v') = \max \begin{cases} MATCH_{lca}(v,v') \\ \max_{i=1,\ldots,d(v)} S(v_i, v') - P(\vec{v}_i) \\ \max_{j=1,\ldots,d(v')} S(v, v'_j) - P(\vec{v}'_j) \end{cases} \quad (1)$$

The function $MATCH_{lca}(v, v')$ is a measure of how well the nodes v and v' align, and is computed as follows:

$$\begin{aligned} MATCH_{lca}(v,v') = Lex_{node}(v,v') + \\ + \max_{p \in \mathcal{P}(v,v')} \sum_{(i,j) \in p} Lex_{arc}(\vec{v}_i, \vec{v}'_j) + \\ + S(v_i, v'_j) \end{aligned} \quad (2)$$

where:

- $Lex_{node}(v, v') \geq 0$ is a measure of how closely the label on source node v corresponds to the label on target node v' in the bilingual dictionary. $Lex_{arc}(\vec{v}, \vec{v}')$ is the corresponding measure for arcs.
- $\mathcal{P}(v, v')$ is the set of all possible pairings of the children of v against the children of v'. There are $O(d!)$ such pairings, where d is the smaller of the degrees of v and v'.
- $P(\vec{v}_i) \geq 0$ is the penalty for collapsing the edge $\vec{v}_i$, which may depend on the label of that edge.

The summation in (2) ranges over all pairs, denoted by (i, j), which appear in a given pairing $p \in \mathcal{P}(v, v')$. The summation is evaluated for all $O(d!)$ possible pairings. The pairing with the maximum score is then selected.

The total running time for computing the scores of all of the $O(n^2)$ node pairs v and v', is $O(d!n^2)$, where d is the lesser of the degrees of the source and target trees. Computing the max term in (2) can be mapped into the Maximum-Weight Clique problem (which is NP-complete), cf. (Farach et al., 1995b). However, in the NLP domain, the running time is contained because $d < 6$ for most trees encountered in practice. Next we describe a heuristic which achieves a time bound quadratic in the size of the tree.

5 A Greedy Heuristic

We can reduce the computation time of the max term in (2), if we do not consider *all of the* $O(d!)$ *pairings* of the children of v and v'. Instead we

use a greedy approach and choose the d highest-scoring, mutually disjoint pairs from among the d^2 possible pairs of children of v and v'. The justification for this heuristic is that we expect that the high-scoring pairs will dominate, and will be *a priori* mutually disjoint.

The following is an alternative, greedy procedure for computing $S(v, v')$:

Procedure $GREEDY_{LCA}$:

1. $\forall i, j$ s.t. $1 \leq i \leq d(v), 1 \leq j \leq d(v')$ compute the corresponding entry in a $d(v) \times d(v')$ matrix M:

$$M_{ij} = Lex_{arc}(\vec{v}_i, \vec{v}'_j) + S(v_i, v'_j)$$

   The entry M_{ij} of $M = M(v, v')$ is the score of matching the ith child of v with jth child of v'.[6]

2. Let $TOP \leftarrow \{\}$ be the set of highest scoring pairs.

3. Find the largest entry $M_{i_0 j_0}$ in the matrix, such that neither its row nor its column is already occupied by some pair in TOP:

$$TOP \leftarrow TOP \cup \{(i_0, j_0)\}$$

   where the coordinates (i_0, j_0) are such that

$$M_{i_0 j_0} = \max_{i,j}\{M_{ij} \mid \forall (i', j') \in TOP, i \neq i', j \neq j'\} \quad (3)$$

4. Repeat the above step d times, where $d = \min(d(v), d(v'))$.

5. Compute the result:

$$MATCH_{lca}(v, v') = = Lex_{node}(v, v') + \sum_{(i,j) \in TOP} M_{ij} \quad (4)$$

With sorting, this can be done in $O(d \log d + d^2)$ time.

The validity of this heuristic can be tested by comparing the performance of the procedures using the computation in (2) and in (4).

[6] Note: if we disregard the arc labels for simplicity, and set $Lex_{arc}(\cdot, \cdot) = 0$, then we do not need to build M, and may simply use $M_{ij} = S(v_i, v'_j)$.

6 Strict Lexical Matching Heuristic

(Grishman, 1994) employed an optimization heuristic which favored lexical matches. For each source node v with label $L(v)$, the procedure using this heuristic would first attempt to find a target node v' with label $L(v')$ such that $L(v)$ translated as $L(v')$ in the bilingual dictionary (a perfect lexical match). If such a lexical match was found, the procedure did not attempt to match v with any other target node.

A similar heuristic (Lex-Match) was incorporated into our program as the following preprocessing steps:

1 For each source node v, all possible lexical matches are identified in the target tree.[7] If v has at least one possible lexical match, all of those positions in the score matrix S which do not correspond to a lexical match of v are set to zero.

2 For each target node v' which has at least one lexical match, all of those positions in the score matrix which do not correspond to a lexical match of v' are set to zero.

By setting to zero those positions in the score matrix which represent unlikely matches, this heuristic prevents these scores from ever being calculated, substantially reducing the running time. Lex-Match, unlike the (Grishman, 1994) heuristic, allows one source node to match lexically with more than one node in the target tree.

7 Implementation

We have implemented the greedy LCA-preserving algorithm with the following features:

Penalties: The penalties for collapsing edges were set to 0.[8]

Scores: A Lex_{node} score of 100 and a Lex_{arc} score of 21 was awarded for each match using our bilingual dictionary. These functions have the value 0 if there is no lexical match.

[7] A match $M(v, v')$ is also a lexical match if either $M(v, w')$ or $M(w, v')$ is a lexical match, where w and w' are children of v and v', respectively.

[8] When penalties are set to zero and an empty bilingual dictionary is used, the alignment algorithm fills the scoring matrix with zeros. When we introduce non-zero penalties, the alignment procedure prefers matches between nodes dominating similar structures, since nodes dominating dissimilar structures receive negative scores. We expect that non-zero penalties will improve precision with a nonempty bilingual dictionary, because they will favor similar structures. In preliminary testing, penalty values of 20 and 30 yielded improvements in precision.

Text	*Baseline*	*Struc-Share*	*Lex-Match*	*Struc-Share and Lex-Match*
El Camino Real	11.5 sec	11.3 sec	8.3 sec	7.7 sec
Curious George	98.0 sec	48.8 sec	87.4 sec	44.7 sec
Total	109.5 sec	60.1 sec	95.7 sec	52.4 sec

Table 1: Time Improvements Due to Optimizations

Text	*Lex-Match Off*	*Lex-Match On*
El Camino Real	47 out of 57 (82%)	47.5 out of 57 (83%)
Curious George	44.6 out of 55 (81%)	44.6 out of 55 (81%)
Total	91.6 out of 112 (82%)	92.1 out of 112 (82%)

Table 2: Changes in Accuracy due to Lex-Match Heuristic

Optimization Variables: We experimented with variants of the procedures which included Structure Sharing (Struc-Share) and the Lexical Match Optimization (Lex-Match), as well as with those that did not.

Table 1 shows the time consumed by our program to align sentences under different conditions. The baseline refers to our program without any optimizations (which is at least 6 times faster than before using this algorithm.) The optimization variables have different effects on the different texts. We believe that structure sharing has a much stronger effect on *Curious George* than on *El Camino Real* because the former has longer sentences which produced more parses. The Lex-Match optimization has a greater effect on *El Camino Real* than on *Curious George* because all of the words contained in *El Camino Real* are included in our bilingual dictionary, but only a small portion of the words in *Curious George* are included. We expect that as the size of our dictionary increases, the Lex-Match optimization will have a greater effect.

The precision for each aligned pair of sentences is computed according to the formula:

$$\frac{|ResultSet \cap AnswerKey|}{|ResultSet|}$$

where $ResultSet$ is the set of source parses to which the alignment procedure assigned the highest score, and $AnswerKey$ is the set of best source parses as judged by one of the experimenters.[9] This precision measure was previously used in (Matsumoto et al., 1993) and (Grishman, 1994). Table 2 compares the precision of the alignment procedure with and without the Lex-Match heuristic (structure sharing had no effect on the scores.) The slight increase in precision observed with the Lex-Match optimization, may be an indication that we should raise the score for lexical matches of node labels.

8 Results and Future Directions

The current implementation aligns trees 63 times faster than our previous program (Grishman, 1994), with a 2.3% improvement in precision.[10] We expect fine-tuning of the parameters in our procedures to improve our performance. We expect to gain greater efficiency if all common nodes between forests are shared, rather than just the NPs. Another efficiency improvement will be achieved by factoring all ambiguity into the parse tree, as in (Matsumoto et al., 1993). In our current approach, disjunctions are represented only at the root level.

In order to improve the precision of alignment, we plan to experiment with varying the values of the *Lex* functions and penalties in our scoring algorithm and expanding our bilingual dictionary. We will also experiment with the non-greedy algorithm discussed above and a dominance-preserving algorithm (a less constrained version of the algorithm which we have omitted due to space limitations). In the dominance-preserving algorithm we relax the requirement of lca-preservation, and require the preservation of the dominance relationship between nodes:

> If, for two nodes $a \in T'$ and $b \in T'$, a dominates b (denoted as $a \prec b$), then for $f(a) \in T''$ and $f(b) \in T''$, $f(a) \prec f(b)$.

The idea which makes it possible to align sentences quickly is that we place restrictions on the ways in which we align the parse trees. We

[9] If there was no correct parse, the parses with the fewest errors were used for purposes of alignment.

[10] The dynamic programming algorithm accounts for an approximately 600% increase in speed of alignment – a rough estimate since much of the program has been re-implemented.

disallow alignments which violate the LCA constraint or the dominance requirement, and permit only one-to-one alignments between nodes. Some cases where one might posit a correspondence between a single node x and a group of nodes $G = \{y_1 \ldots y_n\}$, can be interpreted as an alignment between x and y_j, for some $j, 1 \leq j \leq n$, where y_j dominates the remaining nodes in G. We do not consider other types of one-to-many alignments.

Acknowledgements

We wish to thank Antonio Moreno Sandoval and Cristina Olmeda Moreno for preparation of the Spanish analyses for *Jorge el Curioso*. We also thank Catherine Macleod for preparation of the English parses.

This research was supported by the National Science Foundation under Grant IRI-9303013.

References

Peter Brown, Stephen A. Della Pietra, Vincent J. Della Pietra and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. In *"Computational Linguistics*, 19: 263–312.

S. Chen. 1993. Aligning Sentences in Bilingual Corpora using lexical information. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 9–16, Columbus Ohio. Association for Computational Linguistics, Morristown, New Jersey.

T. H. Cormen, C. E. Leiserson and R. L. Rivest. 1990. *Introduction to Algorithms*, The MIT Press, Cambridge, Mass.

Martin Farach, Teresa M. Przytycka and Mikkel Thorup. 1995. The maximum agreement subtree problem for binary trees. Unpublished manuscript, Rutgers University, Odense University, and University of Copenhagen.

Martin Farach, Teresa M. Przytycka and Mikkel Thorup. 1995. On the agreement of many trees. Unpublished manuscript, Rutgers University, Odense University, and University of Copenhagen.

Osamu Furuse and Hitoshi Iida. 1994. Constituent Boundary Parsing for Example-Based Machine Translation. In *COLING 94 Proceedings*, Volume 1, pages 105–111, Kyoto Japan.

Ralph Grishman. 1994. Iterative Alignment of Syntactic Structures for a Bilingual Corpus. In *Proceedings of the Second Annual Workshop for Very Large Corpora*, Tokyo, Japan.

Ralph Grishman and Michiko Kosaka. 1992. Combining Rationalist and Empiricist Approaches to Machine Translation. In *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation*, Montreal, Canada.

Hiroyuki Kaji, Yuuko Kida and Yasututsugo Morimoto. 1992. Learning Translation Templates from Bilingual Text. In *COLING 92 Proceedings*.

Lauri Karttunen. 1985. Structure-Sharing with Binary Trees. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*.

Judith Klavans and Evelyne Tzoukermann. 1990. The BICORD System. In *COLING 90 Proceedings*, Volume 3, pages 174–179.

Y. Matsumoto, H. Ishimoto, T. Utsuro and M. Nagao. 1993. Structural Matching of Parallel Texts. In *31st Annual Meeting of the Association for Computational Linguistics: "Proceedings of the Conference"*.

Makao Nagao. 1984. A Framework of a Mechanical Translation between Japanese and English by Analogy Principle. In Alick Elithorn and Ranan Banerji, editors, *Artificial and Human Intelligence*. Elsevier Science Publishers B.V., Amsterdam, The Netherlands.

Fernando C. N. Pereira. 1985. A Structure-Sharing Representation for Unification-Based Grammatical Formalisms. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*.

Satoshi Sato and Makoto Nagao. 1990. Toward Memory-based Translation. In *COLING 90 Proceedings*, Volume 3, pages 247–252.

Mike Steel and Tandy Warnow. 1993. Kaiokura Tree Theorems: Computing the Maximum Agreement Subtree. In *Information Processing Letters*, 48: 77–82.