Deriving Transfer Rules from Dominance-Preserving Alignments

Adam Meyers, Roman Yangarber, Ralph Grishman,
Catherine Macleod, Antonio Moreno–Sandoval†
New York University
715 Broadway, 7th Floor, NY, NY 10003, USA
†Universidad Autónoma de Madrid
Cantoblanco, 28049-Madrid, SPAIN
meyers/roman/grishman/macleod@cs.nyu.edu
sandoval@lola.lllf.uam.es

1 Introduction

Automatic acquisition of translation rules from parallel sentence-aligned text takes a variety of forms. Some machine translation (MT) systems treat aligned sentences as unstructured word sequences. Other systems, including our own ((Grishman, 1994) and (Meyers et al., 1996)), syntactically analyze sentences (parse) before acquiring transfer rules (cf. (Kaji et al., 1992), (Matsumoto et al., 1993), and (Kitamura and Matsumoto, 1995)). This has the advantage of acquiring structural as well as lexical correspondences. A syntactically analyzed, aligned corpus may serve as an example base for a form of example-based MT (cf. (Sato and Nagao, 1990), (Kaji et al., 1992), and (Furuse and Iida, 1994)).

This paper[1] describes: (1) an efficient algorithm for aligning a pair of source/target language parse trees; and (2) a procedure for deriving transfer rules from this alignment. Each transfer rule consists of a pair of tree fragments derived by "cutting up" the source and target trees. A set of transfer rules whose left-hand sides match a source language parse tree is used to generate a target language parse tree from their set of right-hand sides, which is a translation of the source tree. This technique resembles work on MT using synchronous Tree-Adjoining Grammars (cf. (Abeille et al., 1990)).

The Proteus translation system learns transfer rules from pairs of aligned source and target *regularized parses*, Proteus's representation of predicate argument structure (cf. Figure 1).[2] Then it uses these transfer rules to map source language regularized parses generated by our source language parser into target language regularized parses. Finally a generator converts target regularized parses into target language sentences.

An alignment f is a 1-to-1 partial mapping from source nodes to target nodes. We consider only alignments which preserve the dominance relationship: If node a dominates node b in the source tree, then $f(a)$ dominates $f(b)$ in the target tree. In Figure 1, source nodes A, B, C and D map to the corresponding target nodes, marked with a prime, e.g., $f(A) = A'$. The alignment may be represented by the set $\{(A, A'), (B, B'), (C, C'), (D, D')\}$. We can assign a *score* to each alignment f, based on the (weighted) number of pairs in f; finding the best alignment translates into finding the alignment with the highest score. Our algorithms are based on (Farach et al., 1995) and related work.

We needed efficient alignment algorithms because: (1) Corpus-based training requires processing a lot of text; and (2) An exhaustive search of all alignments is too computationally expensive for realistically sized parse trees.

Eliminating dominance violations greatly reduced our search space. Similar work (e.g., (Matsumoto et al., 1993)) considers all possible matches. Although, our system cannot account for actual dominance violations in a given bitext, there are no such violations in our corpus and many hypothetical cases can be avoided by adopting the appropriate grammar. Cases of adjuncts aligning with heads and vice versa are not dominance violations if we replace our dependency analysis with one in which internal nodes have category labels and the head constituents are marked by $HEAD$ arcs and we assume the following Categorial Grammar (CG) style analyses. Suppose that verb (V1) maps to adverb (A'1) and adverb (A2) maps to verb (V'2), where

[1]We thank Cristina Olmeda Moreno for work on parsing our Spanish text. This research was supported by National Science Foundation Grant IRI-9303013.

[2]Regularized parses (henceforth, "parse trees") are like F-structures of Lexical Function Grammar (LFG), except that a dependency structure is used."

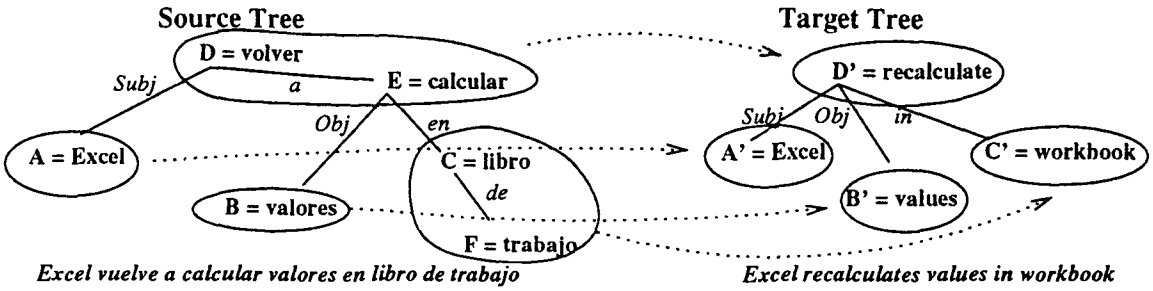

Figure 1: A Pair of Aligned Trees

A2 modifies V1 and A'1 modifies V'2. We assume the following structures: [VP [VP1 V1 ...] A2] and [VP [VP2 V'2 ...] A'1]. No dominance violation exists because no dominance relation holds between V1 and A2 or V'2 and A'1. Y. Matsumoto (p.c.) notes that the subordinate clause of a source sentence may align with the main clause of a target language and vice versa, e.g., *X after Y* aligns with *Y' before X'*, where X, X', Y and Y' are all clauses. Assuming a CG style analysis, [S X [after Y]] aligns with [S Y' [before X']] with no dominance violations.

2 The Least-Common-Ancestor Constraint

Our earlier tree alignment algorithms (cf. (Meyers et al., 1996)) were designed to produce alignments which preserve the least common ancestor relationship: If nodes a and b map into nodes $a' = f(a)$ and $b' = f(b)$, then $f(LCA(a,b)) = LCA(f(a), f(b)) = LCA(a', b')$. The least common ancestor (LCA) of a and b is the lowest node in the tree dominating both a and b. The LCA-preserving approach imposes limitations on the quality of the resulting alignments. In Figure 1, the LCA-preserving algorithm will match node E with node D' and report that as the best match overall. The score $S(D, D')$ would take into account only the match (E, D'), which in turn includes (B, B') and (C, C'). ($S(D, D')$ would be penalized for collapsing the arc from D to E.)

We seek a better alignment scheme, in which the score $S(D, D')$ could benefit from $S(A, A')$. We are willing to pay a small penalty to collapse the path from D to E, and align the resulting structure. This leads to new algorithms where the LCA-preserving restriction is replaced by the weaker, dominance-preserving constraint. The rationale behind allowing an edge, say (v, u) to be collapsed when matching two nodes v and v'. is that we may find some children of u which correspond well to some children of v', while other children of v correspond well to *other* children of v'. (This is not possible if LCA's are preserved.) The algorithm relies on the assumption that two different children of v will not match well with *the same* child of v'.

3 The Dominance-Preserving Algorithm

Let T and T' be the source and the target trees. We use a dynamic programming algorithm to compute, in a bottom-up fashion, the scores for matching each node in T against each node in T'. There are $O(n^2)$ such scores, $n = \max(|T|, |T'|)$ is number of nodes in the trees. Let the $d(v)$ be the degree of a node v. We denote children of v by v_i, $i = 1, \ldots, d(v)$, and arc (v, v_i) by $\vec{v_i}$.

For all pairs of nodes $v \in T$ and $v' \in T'$, the algorithm computes the score function $S(v, v')$. $S(v, v')$ corresponds to the best match found between the subtrees rooted at v in T and at v' in T'. The values of S are stored in a $|T| \times |T'|$ matrix, also denoted by S. Initially, we fill the matrix S with undefined values, and invoke the procedure $SCORE_{dom}$, described below, to compute $S(root(T), root(T'))$, the score for matching the root nodes of the trees. During the computation of the score for the roots, the procedure recursively finds the best-scoring matches for *all* the nodes in the trees. This yields the best alignment of the entire trees.

Table 1(a) shows the values of S for the trees in Figure 1. Whenever we compute a score for internal nodes, we also record the best way of pairing up their children in Table 1(b).[3] The

[3] *Children* pairings include child/child pairs and parent/child pairs: $(D.D')$'s pairing is $\{(A, A'), (E, D')\}$.

alignment, implicit in these children pairings, is used in a later phase (Section 4) to recover the alignment for the entire trees.

Procedure $SCORE_{dom}$: For a pair of nodes, (v, v'), recursively compute the score $S(v, v')$:

Construct an intermediate *child-scoring* matrix $M = M(v, v')$, for the children of v and v'; the dimensions of M are $(d(v)+1) \times (d(v')+1)$. That is, the number of rows in M is one more than the number of children of v, and the number of columns is one more than number of children of v'. We label row $d(v)+1$ and column $d(v')+1$ with a "*". Fill the matrix M:

1. $\forall i, j$, where $1 \leq i \leq d(v), 1 \leq j \leq d(v')$ compute the corresponding entry in M_{ij}:
   $M_{ij} = S(v_i, v'_j) + Lex_{arc}(\vec{v}_i, \vec{v}'_j)$
   The function $Lex_{node}(v, v') \geq 0$ (used below) is the quality of translation, i.e. the measure of how closely the label (word) at source node v corresponds to the label at target node v' in the bilingual dictionary, and $Lex_{arc}(\vec{v}, \vec{v}') \geq 0$ is the corresponding measure for arc labels.
2. Fill the last column as follows: $\forall i$, where $1 \leq i \leq d(v)$ compute the entries:
   $M_{i*} = S(v_i, v') - Pen(\vec{v}_i)$
   $Pen(\vec{v}_i) \geq 0$ is the penalty for collapsing the edge $\vec{v}_i$, which depends on the value of the label of that edge.
3. Symmetrically, $\forall j$ s.t. $1 \leq j \leq d(v')$ fill the last row with the entries:
   $M_{*j} = S(v, v'_j) - Pen(\vec{v}'_j)$
4. The entry M_{**} is disfavored: $M_{**} = -\infty$

For example, during the calculation of the scores $S(D, D')$ and $S(E, D')$ from Table 1, the corresponding matrices $M(D, D')$ and $M(E, D')$ are filled in as in Table 2. The proper values for the *parameter* functions used above, such as the penalty function Pen and the translation measures, are chosen empirically, and constitute the tunable parameters of the procedure. Normally, we will expect that the values of Lex_{node} will be much larger than the values of Lex_{arc} and Pen. In the example we used the following settings:
1. $Lex_{node} = 100$ for an exact translation, as for (A, A'), (B, B') and (C, C'), and 0 otherwise.
2. all values of Lex_{arc} are set to zero
3. all penalties Pen are set to 1

Now, using the values in M, compute the score for matching v and v':

$$S(v, v') = Lex_{node}(v, v') + \max_{P \in \mathcal{LP}} \sum_{(i,j) \in P} M_{ij} \quad (1)$$

Here P is a *legitimate* pairing of v and its children against v' and its children. A legitimate pairing P is a set of elements of the matrix M, that conform to the following conditions:

1. each row and each column of M may contribute at most one element to P, except that the row and the column labeled $*$ may contribute more than one element to P
2. if P contains an element M_{ij} corresponding to the node pair (w, w'), and some child node u appears in the Children-Pairing for (w, w'), then the row or column of u may not contribute any elements to P.

We use $\mathcal{LP} = \mathcal{LP}(v, v')$ to denote the set of all legitimate pairings. There are $O(d!)$ such pairings, where d is the greater of the degrees of v and v'. The summation in (1) ranges over all the pairs (i, j) that appear in a legitimate pairing $P \in \mathcal{LP}(v, v')$. We evaluate this summation for all $O(d!)$ legitimate pairings in $\mathcal{LP}$, and then select the pairing P_{best} with the maximum score. P_{best} is then stored in the Children-Pairing matrix entry for (v, v').

Table 2 shows how scores are calculated. The best score for $S(E, D')$ is 200, the sum of the scores for (B, B') and (C, C'). $S(D, D') = 299 = S(A, A') + S(E, D') - 1$, a penalty of 1 for collapsing the edge from D to E.

We can reduce the computation time of the max term in (1), if we do not consider *all* $O(d!)$ pairings of the children of v and v'. Instead of exhaustively computing the maximal-scoring pairing P_{best} in (1), we can build it in a *greedy* fashion: successively choos the d highest-scoring, mutually disjoint pairs from the $O(d^2)$ possible pairs of children of v and v'.

1. Initialize the set of highest scoring pairs $P_{best} \leftarrow \emptyset$
2. $P_{best} \leftarrow P_{best} \cup \{(i, j)\}$ where M_{ij} is the next largest entry in the matrix, which that satisfies both conditions 1 and 2 of legitimate pairings

Source Nodes \ Target Nodes	A'	B'	C'	D'
A	100	0	0	0
B	0	100	0	0
C	0	0	100	0
D	0	0	0	299
E	0	0	0	200
F	0	0	0	0

Source Nodes \ Target Nodes	A'	B'	C'	D'
A	-	-	-	-
B	-	-	-	-
C	-	-	-	-
D	-	-	-	$(A, A')(E, D')$
E	-	-	-	$(B, B')(C, C')$
F	-	-	-	-

Table 1: (a) A Final Score Matrix: (b) Children-Pairing Matrix

Source Children \ Target Children	1: A'	2: B'	3: C'	*: D'
1: B	0	**100**	0	99
2: C	0	0	**100**	99
*: E	0	99	99	$-\infty$

The Score $S(E, D') = 100 + 100 = 200$

Source Children \ Target Children	1: A'	2: B'	3: C'	*: D'
1: A	**100**	0	0	99
2: E	0	99	99	**199**
*: D	99	98	98	$-\infty$

The Score $S(D, D') = 199 + 100 = 299$

Table 2: Computing Child-Scoring Matrices

3. Repeat the above step until no more pairs can be added to P_{best}, at most d times. where $d = \min(d(v), d(v'))$.
4. Compute the result: $S(v, v') = Lex_{node}(v, v') + \sum_{(i,j) \in P_{best}} M_{ij}$

The greedy algorithm aligns trees with n nodes and maximal degree d in $O(n^2 d^2)$ time.

4 Acquiring Transfer Rules

This section describes the procedure for deriving transfer rules from aligned parse trees.

First, the best-scoring alignment is recovered from the Children-Pairing matrix, (Table 1(b)).[4] Start by including the root node-pair in the alignment, (here (D, D')). Then, for each pair (v, v') already in the alignment, repeat the following steps, until no more pairs can be added to the alignment: (1) look up the Children-Pairing for (v, v'); (2) for each pair in the children-pairing, if it does not include either v or v', add the pair to the alignment, (e.g. (A, A'), etc.).

[4]When sentences in the bitext have multiple parses, we align structure sharing forests of trees. If one pair of trees has the highest scoring alignment, we acquire transfer rules from that alignment. When more than one pair of trees tie for the highest score, we acquire transfer rules from the set of pairs of aligned subtrees which are shared by each of these high scoring alignments.

In the running example, the final alignment (FA) is $\{(D, D'), (A, A'), (B, B'), (C, C')\}$. Based on this alignment we can "chop up" the trees into fragments, or *substructures* ((Matsumoto et al., 1993)), where each substructure of a tree is a connected group of nodes in the tree, together with their joining arcs. In Figure 1, dashed arrows connect aligned pairs of source and target substructures. These correspondences become our transfer rules.

For each pair of aligned nodes (v, v') in FA, there is a pair of substructures in Figure 1 such that v and v' are the roots of the source and target substructures. These substructures include all unaligned source and target nodes v_u and v'_u below v and v', which have no intervening aligned nodes y or y' dominating v_u or v'_u.

The transfer rules derived from Figure 1 may be written as follows:

1. $< root : Excel > \rightarrow < root : Excel >$
2. $< root : valores > \rightarrow < root : values >$
3. $< root : libro, de : trabajo > \rightarrow < root : workbook >$
4. $< root : volver, subj : x_1, a :< root : calcular, obj : x_2, en : x_3 >> \rightarrow$ $< root : recalculate, subj : Tr(x_1), obj : Tr(x_2), in : Tr(x_3) >$

Each substructure is represented as a list con-

taining a root lexical item, and a set of arc-value pairs. An arc (role) a_1 with head (value) h is written as $a_1 : h$, where h is a fixed label (word), a substructure or a variable. If the source substructure has n of the leaves labeled with variables $x_1, \ldots, x_n$, the target will have n of the leaves labeled with $Tr(x_1), \ldots, Tr(x_n)$, where $Tr(x)$ is the lexical translation function. This general structure allows us to capture relations between multi-word expressions in the source and target languages.

5 Translation

The described procedure for acquisition of transfer rules from corpora is the basis for our translation system. A large collection of transfer rules are collected from a training corpus. When new text is to be translated, it is first parsed. The source tree is matched against the left hand sides of the transfer rules which have been collected. If a set of transfer rules whose left-hand sides match the parse tree is found, the corresponding target structure is generated from the right hand sides of these transfer rules. Typically, several sets of transfer rules meet this criterion. They are ranked by their frequency in the training corpus. Once a target tree has been produced, it is converted to a word sequence by a target language generator. We have applied this approach to the translation of Microsoft Help files in English and Spanish. The sentences are moderately simple and quite parallel in structure, which has made the corpus suitable for our initial system development. To date, we have been using a training corpus of about 1,000 sentences, and a test corpus of about 100 sentences.

6 Evaluation

Real evaluation of performance of MT systems is time consuming and subjective. Nevertheless, some evaluation system is needed to insure that incremental changes are for the better, or at least, are not detrimental. We measured the success of our translation by how closely we reproduced Microsoft's English (target language) text. Our evaluation procedure computes the ratio between (a) the complement of the intersection set of words in our translation and the actual Microsoft sentence; and (b) the combined lengths of these two sentences. An exact translation gives a score of 0. If the system generates the sentence "A B C D E" and the actual sentence is "A B C F", the score is 3/9 (the length of D E F divided by the combined lengths of A B C D E and A B C F.) The dominance-preserving version of the program produced output for 88 out of 91 test sentences. The average score for these 88 sentences was 0.29: 0.21 due to incorrect word matches and 0.08 due to failure to translate because insufficient confidence levels were reached. The LCA-preserving version produced output for only 83 sentences with an average score of over 0.30: about 0.23 due to incorrect word matches and about 0.08 due to insufficient confidence levels. This crude scoring technique suggests that the dominance-preserving algorithm improved our results: more sentences were translated with higher quality. One limitation of this scoring technique is that paraphrases are penalized. An imperfect score (even .20) may signify an adequate translation.

References

A. Abeille, Y. Schabes, and A. K. Joshi. 1990. Using Lexicalized Tags for Machine Translation. In *COLING90*.

M. Farach, T. M. Przytycka, and M. Thorup. 1995. On the agreement of many trees. *Information Processing Letters*, 55:297–301.

O. Furuse and H. Iida. 1994. Constituent Boundary Parsing for Example-Based Machine Translation. In *COLING94*.

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

H. Kaji, Y. Kida, and Y. Morimoto. 1992. Learning Translation Templates from Bilingual Text. In *COLING92*.

M. Kitamura and Y. Matsumoto. 1995. A Machine Translation System based on Translation Rules Acquired from Parallel Corpora. In *RANLP95*.

Y. Matsumoto, H. Ishimoto, T. Utsuro, and M. Nagao. 1993. Structural Matching of Parallel Texts. In *ACL93*.

A. Meyers, R. Yangarber, and R. Grishman. 1996. Alignment of Shared Forests for Bilingual Corpora. In *COLING96*, pages 460–465.

S. Sato and M. Nagao. 1990. Toward Memory-based Translation. In *COLING90*, volume 3, pages 247–252.