Lang Resources & Evaluation (2014) 48:709–739
DOI 10.1007/s10579-014-9273-4

PROJECT NOTES

Capturing divergence in dependency trees to improve syntactic projection

Ryan Georgi · Fei Xia · William D. Lewis

Published online: 16 October 2014

Abstract Obtaining syntactic parses is an important step in many NLP pipelines. However, most of the world's languages do not have a large amount of syntactically annotated data available for building parsers. Syntactic projection techniques attempt to address this issue by using parallel corpora consisting of resource-poor and resource-rich language pairs, taking advantage of a parser for the resource-rich language and word alignment between the languages to project the parses onto the data for the resource-poor language. These projection methods can suffer, however, when syntactic structures for some sentence pairs in the two languages look quite different. In this paper, we investigate the use of small, parallel, annotated corpora to automatically detect divergent structural patterns between two languages. We then use these detected patterns to improve projection algorithms and dependency parsers, allowing for better performing NLP tools for resource-poor languages, particularly those that may not have large amounts of annotated data necessary for traditional, fully-supervised methods. While this detection process is not exhaustive, we demonstrate that common patterns of divergence can be identified automatically without prior knowledge of a given language pair, and the patterns can be used to improve performance of syntactic projection and parsing.

Keywords Multilingualism · Translation divergence · Syntactic projection

R. Georgi (✉) · F. Xia
Department of Linguistics, University of Washington, Box 352425, Seattle, WA 98195-2425, USA
e-mail: rgeorgi@uw.edu

F. Xia
e-mail: fxia@uw.edu

W. D. Lewis
Microsoft Research, Bldg 99, 14820 NE 36th St, Redmond, WA 98052-6399, USA
e-mail: wilewis@microsoft.com

1 Introduction

When it comes to resources for natural language processing, a small handful of languages account for the vast majority of available resources. Out of the resources listed by the Language Resource and Evaluation (LRE) Map (Calzolari et al. 2012), English accounts for 30 % of all recorded resources, and the ten most resourced languages account for 62 %. A broad variety of tools are available for these resource-rich languages, since the time and effort spent to annotate resources for them allows for state-of-the-art systems to be built utilizing supervised and semi-supervised methods.

The availability of such resources does not come at a low cost; they are the result of a large investment over many years on a per-language-basis. Because creating high-quality annotation is expensive and labor intensive, the vast majority of the world's languages lack such resources and, likewise, high-performance NLP tools. To address this issue, recent studies (Lewis and Xia 2008; Benajiba and Zitouni 2010; Georgi et al. 2012) have investigated using bitexts in which one half of the bitext is a resource-rich language. In this paradigm, existing tools for the resource-rich language can be used to process one side and project the information to the other (the resource-poor language) via word alignments.

While projecting annotation from one language to another is a promising method for adding annotation to languages using automated methods, it relies on the assumption that simple word alignments between languages are sufficient to represent analogous meanings and structures between the languages. For reasons we will discuss in the following sections, this assumption is useful, but often erroneous.

Finding out whether and when this assumption fails for a given language pair is not easy without knowledge about the two languages. It would be useful if, given a small set of seed data, a language pair could be analyzed to find where and in what ways the languages diverge, and use these detected patterns as corrective guidelines for improving projection on other sentences for the language pair.

In this paper, we propose a method for analyzing a language pair and determining the degree and types of divergence between two dependency trees in the two languages. We then use this systematic identification of divergence types to inform and correct the trees produced by syntactic projection. Using these improved trees, we are able to boost the performance of a dependency parser by adding new features extracted from projected trees.

2 Background

While there is a growing body of work on projection methods as a means to bootstrap resources for one language from another, there are not many studies on how to handle the issue of linguistic divergence between these languages. In this section, we provide a brief review of work on divergence and projection algorithms. We will also introduce interlinear glossed text (IGT), a common format used by linguists to represent language examples (Lewis 2006).

2.1 Projection methods

Projection algorithms have been the target of a fair amount of research in the last decade, as attempts have been made to utilize statistical alignment methods to match words between languages with parallel data and "project" annotations between them. Figure 1 shows an example bitext in the form of an IGT, while Fig. 2 shows how this data may be used to project a dependency tree from English to Hindi.

Some of the initial research on the subject of projecting word-level annotation from one language to another was published in Yarowsky and Ngai (2001). Here, the authors used IBM Model 3 (Brown et al. 1990) to align large parallel corpora in English–Chinese and English–French. A part-of-speech (POS) tagger was trained for French using projections from English, and noun phrase (NP) bracketers were trained similarly for both French and Chinese. The authors identified noisy statistical alignment and 1-to-many alignments as two main issues affecting the performance of projection. The first of these issues is a difficult problem for resource-poor languages, as high-quality statistical word alignment often requires much more bitext than might be available for the language. While it is not a full solution to the problem, many of the language pairs we use in this work are drawn from collection of IGT instances, as shown in Fig. 1, which provide unique shortcuts for obtaining word alignments with a small amount of data. IGT will be discussed further in Sect. 2.2.

The second issue, 1-to-many alignments, may be the result of linguistic divergence in a language pair where the language being projected from is morphologically richer than the other. In cases such as this, finding common patterns of conflation can be useful for generalizing a projection to new data. For instance, Fig. 3 shows a very simple but common case of conflation in the SMULTRON corpus (Volk et al. 2010), where a single German word aligns to multiple English words. Using this one-to-many alignment, the same POS tag would be projected to both English tokens. In this case, using a universal tagset such as those presented by Petrov et al. (2012), could help alleviate the problem, but for more complex cases, learning the pattern would be more critical.

Hwa et al. (2004) investigated the issues in using projection between languages in order to develop and train syntactic parsers, and described the Direct Correspondence Assumption (DCA), the assumption made in projection algorithms that the target language tree should be homomorphic with the source language tree. While useful, this assumption often does not hold, as the authors pointed out. In order to fix some of the errors made by faulty projections, Hwa et al. used an approach that applies post-projection correction rules. For projection from English to Spanish, the accuracy of the projected structures increased from 36.8 to 70.3 %. The accuracy of the English to Chinese projection increased from 38.1 to 67.3 %.

While these language-specific rewrite rules are promising, they still require language-specific knowledge. What we seek to accomplish in this paper is a general framework for automatically detecting this divergence, both in specific language pairs and its frequency throughout a large number of languages. With the use of this automated divergence detection, it may be possible to learn these rewrite rules from a small annotated corpus and use them to improve projection algorithms.

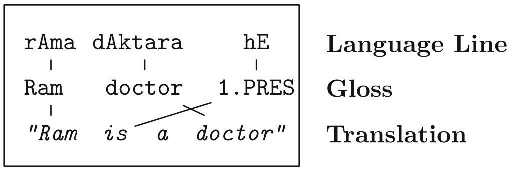


Fig. 1 An instance of interlinear glossed text (IGT) for Hindi, with a gloss line and translation in English. The word alignment between the gloss and translation lines is not part of the IGT instance, but it can be produced by a statistical word aligner trained on bitext or an aligner that uses heuristic rules

2.2 Interlinear glossed text

As mentioned in the preceding section, much of the data for our experiments was drawn from the unique IGT data type. IGT instances are a common way for linguists to give illustrative examples for a language being studied. Figure 1 shows an instance of IGT for Hindi. As with this example, an IGT instance typically has three lines: a language line, a word-to-word or morpheme-to-morpheme gloss line, and a translation line. The translation line is typically in English, the language of the research paper from which the IGT is extracted. Of special interest in IGT instances is the middle gloss line, which gives a word-by-word gloss of the original language. By design, the alignment between the language and gloss lines is monotonic and one-to-one, thus providing easy matches between these two lines. The matching of words between the gloss and translation can be utilized to obtain high-quality, automatic word alignment between the sentences in the language pair without the need for the much larger amounts of data typically required by statistical alignment algorithms.

In Lewis and Xia (2010), IGT data for seven language pairs was automatically aligned, projection performed, then finally hand-corrected to create gold standards with minimal manual intervention. They showed the potential for using IGT as a resource for languages for which finding resources would otherwise be extremely difficult or impossible to obtain. We will use this data for the current work. A breakdown of the language pairs can be seen in Sect. 4.1.

Lewis and Xia (2008) used projected phrase structures to determine the basic word order for 97 languages using a database of IGT instances. By using the alignment method described above and projecting phrase structures from English to the foreign language line, the word order in the foreign language could be inferred. For languages with just 10–39 IGT instances, the accuracy of predicting basic word order was 79 %; with more than 40 instances, the accuracy jumped to 99 %.

Figure 4 gives a high-level view of a basic syntactic projection system that uses IGT as the source of projection. Using the IGT as a source, we extract sentences in the foreign language (F) and English (E), as well as word alignment between F and E via the gloss line. Then the E sentence is parsed and the parsed tree is projected to the F side via word alignment and heuristics. Section 3.5 will discuss learning and applying corrections to projected trees, while Sect. 3.6 will describe a system that bootstraps a parser using the projected systems.

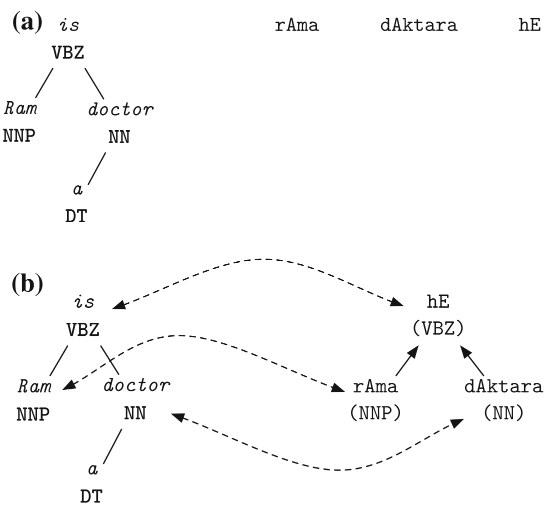


Fig. 2 A simple example of syntactic projection as performed on the IGT instance in Fig. 1. (**a**) Using the interlinear instance from Fig. 1, the English text is parsed. (**b**) Using the word alignments from Fig. 1, the tree structure and POS tags forthe English tree are "projected" onto the Hindi sentence

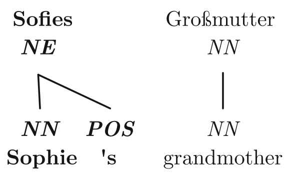


Fig. 3 Simple but frequent example of a 1-to-many German–English alignment found in the *Sophie's World* portion of the SMULTRON corpus (Volk et al. 2010)

2.3 Linguistic divergence

The previously mentioned studies illustrate the promise of projection for bootstrapping new tools for resource-poor languages, but one limitation is their reliance on the assumption that syntactic structures of the two sentences in a given sentence pair are similar. While Hwa et al. 's DCA describes the assumption made for projection, Dorr (1994) makes a deeper analysis of divergence in languages. Dorr outlined *lexical conceptual structures* (LCS) that provide a general framework to describe these exceptions to the DCA. This framework is capable of representing divergence stemming from syntactic, lexical, or semantic differences, but for the purposes of this paper we will focus primarily on those that are lexical and syntactic.

Our goal in this work is to create a methodology by which these common types of divergences can be detected automatically from bitexts in order to improve the performance of existing structural projection methods.

3 Methodology

In our approach to automatically detecting divergent structures between language pairs, we first propose a metric to measure the degree of *matched* edges between trees in a language pair (Sect. 3.2). Second, we define three operations on trees in order to capture three common types of divergence (Sect. 3.3). Third, we apply the operations on a tree pair and show how the operations could affect the degree of tree *match* (Sect. 3.4).

Next, we address how the detected patterns can be used to apply tree modification rules to improve the projection algorithm (Sect. 3.5) and help in training a dependency parser (Sect. 3.6). Finally, we will explain the relationship of our operations to Dorr's divergence types (Sect. 3.7).

3.1 Definitions

In the following sections, as we describe the projection algorithm and trees, we will assume that the resource-rich language being projected from is typically English, E (1), found in the translation line of IGT instances, and the resource-poor language is the foreign language F (2). Each sentence will be represented by a pair of words and tree edges (W, T). T_E (3) will refer to the English tree, and T_F (4) the foreign-language tree, with W_E (5) and W_F (6) referring to the individual words in the respective sentences. Each tree edge will be represented as a pair of words to be connected, (w_{child}, w_{parent}).

$$E = (W_E, T_E) \tag{1}$$

$$F = (W_F, T_F) \tag{2}$$

$$T_E = \{(e_i, e_k)\ldots(e_n, e_m)\} \tag{3}$$

$$T_F = \{(f_i, f_k)\ldots(f_n, f_m)\} \tag{4}$$

$$W_E = \{e_i\ldots e_m\} \tag{5}$$

$$W_F = \{f_j\ldots f_n\} \tag{6}$$

3.2 Comparing dependency trees

Defining a metric for comparing dependency trees cross-linguistically proved to be a crucial component of our method, as most existing tree similarity measures are

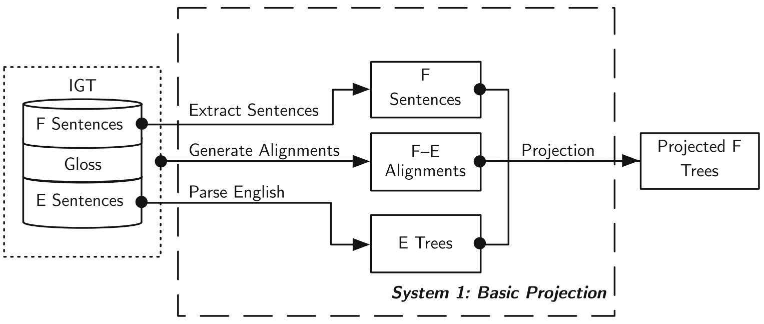


Fig. 4 Flowchart demonstrating the basic projection system using IGT

intended to compare tree representations with the same number of tokens. Comparing across languages, however, means that the number of tokens can vary. We instead look for a method to determine similarity by means of matched edges in the tree, as shown in Fig. 5.

Given an IGT instance, and the sentences F and E as defined above, A is the word alignment between the sentence pair, where f_i and f_j are words in W_F and e_k and e_l are words in W_E. E is defined similarly, except words in the translation line are denoted as e_i, not f_i. The alignment A is a set of word pairs:

$$A = \{(f_i, e_k)\ldots(f_j, e_l)\} \tag{7}$$

We call an (F, E, A) tuple an aligned tree pair. A corpus, C, in our experiments, is a set of (F, E, A) tuples. An edge (f_i, f_k) in F is said to match an edge (e_i, e_k) in E if f_i is aligned to e_i and f_k is aligned to e_k. Because the alignment between a sentence pair can be many-to-many, we define the following functions, which map a word from one sentence to the set of words in the other sentence.

$$R_{F\to E}(f_i, A) = \{e \mid (f_i, e) \in A\} \tag{8}$$

$$R_{E\to F}(e_i, A) = \{f \mid (f, e_i) \in A\} \tag{9}$$

We then define the boolean function *match*, as follows:

$$(f_i, f_j, T_E, A) = \begin{cases} 1 & \textbf{if } \exists e_a, e_b((e_a \in R_{F\to E}(f_i))\wedge \\ & (e_b \in R_{F\to E}(f_j)) \wedge ((e_a, e_b) \in T_E)) \\ 0 & \textit{otherwise} \end{cases} \tag{10}$$

That is, an edge (f_i, f_j) in F matches some edge in E according to A if there exists two words, e_a and e_b in E such that e_a aligns to f_i, e_b aligns to f_j, and (e_a, e_b) is an edge in E.

Given an aligned tree pair (F, E, A), we define $SentMatch(F, E, A)$ as the percentage of edges in F that match some edge in E. Given a corpus C, we define

$CorpusMatch_{F\rightarrow E}(C)$ as the percentage of edges in the T_F trees that match some edges in the corresponding T_E trees. Similarly, $CorpusMatch_{E\rightarrow F}(C)$ is the percentage of edges in the T_E trees that match some edges in the corresponding T_F trees.

Algorithm 1: Calculating the percentage of matched edges in corpus C.

```
input  : A corpus $C$
output: $CorpusMatch_{F\rightarrow E}(C)$
        $CorpusMatch_{E\rightarrow F}(C)$
begin
    Let $F \rightarrow E$ matches = 0 ;
    Let $E \rightarrow F$ matches = 0 ;
    Let all $F$ edges = 0 ;
    Let all $E$ edges = 0 ;
    foreach $(F, E, A) \in C$ do
        Let $F = (W_F, T_F)$ ;
        Let $E = (W_E, T_E)$ ;
        Let $A = \{(f_i, e_j), \ldots, (f_k, e_l)\}$ ;
        // Get matches for $F \rightarrow E$
        foreach $(f_c, f_p) \in T_F$ do
            /* If the child and parent in this edge align with the child→parent
               edge of the other tree...                                    */
            if $\exists\, e_c, e_p : e_p = parent(e_c, T_E)$
                and $(f_p, e_p) \in A$
                and $(f_c, e_c) \in A$
            then
                // Increase the match count.
                $(F \rightarrow E$ matches$)$++;
            (all $F$ edges )++;
        // Get matches for $E \rightarrow F$
        foreach $(e_c, e_p) \in T_E$ do
            /* If the child and parent in this edge align with the child→parent
               edge of the other tree...                                    */
            if $\exists\, f_c, f_p : f_p = parent(f_c, T_F)$
                and $(f_p, e_p) \in A$
                and $(f_c, e_c) \in A$
            then
                // Increase the match count.
                $(E \rightarrow F$ matches$)$++;
            (all $E$ edges )++;
    return $Match(F \rightarrow E) = 100 \times \frac{(F\rightarrow E matches)}{\text{all } F \text{ edges}}$ ;
    return $Match(E \rightarrow F) = 100 \times \frac{(E\rightarrow F matches)}{\text{all } E \text{ edges}}$ ;
```

$$CorpusMatch_{F\rightarrow E}(C) = \frac{\sum_{(F,E,A)\,\in\, C}\left(\sum_{(f_i,f_j)\,\in\, T_F} match(f_i,\, f_j,\, T_E, A)\right)}{\sum_{(F,E,A)\in C} |T_F|} \tag{11}$$

3.3 Defining Tree Operations

When an edge (f_i, f_k) in T_F does not match any edge in T_E, it may be caused by one of the following cases:

C1 f_i or f_k are spontaneous (they do not align with any words in the other tree).
C2 f_i and f_k are both aligned with the same node e_i in the other tree (Fig. 5b).

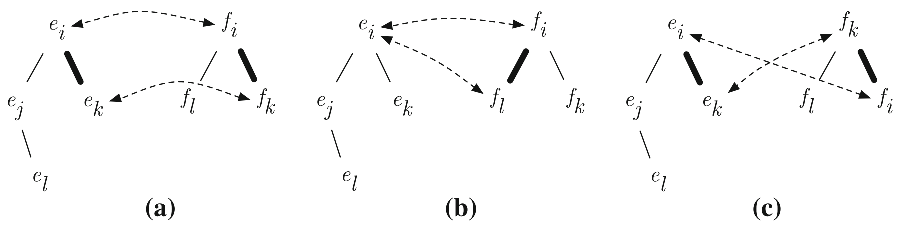


Fig. 5 Definition of a *match*, *merge*, and *swap* edges in a tree pair. (**a**) A *match* alignment. (**b**) A *merge* alignment. (**c**) A *swap* alignment

C3 f_i and f_k are both aligned with nodes in the other tree, e_k and e_i, but in a reversed parent-child relationship (Fig. 5c).

C4 There are some other structural differences not caused by C3.3–C3.3.

The first three cases are common. To capture them, we define three operations on a tree—*remove*, *merge*, and *swap*.

3.3.1 O1: Remove

The *remove* operation is used to remove spontaneous words. As shown in Fig. 6a, removal of the node *l* is accomplished by removing the link between node *l* and its parent *j*, and adding links between the parent and the removed node's children.

This result of this operation can be seen in Fig. 6a, using the relation *Children*, which maps a word to the set of all its children in the tree.

3.3.2 O2: Merge

The *merge* operation is used when a node and some or all of its children in one tree align to the same node(s) in the other tree, as can be seen in Fig. 5b. The parent *j* and child *l* are collapsed into a merged node, as indicated by *l*+*j* in Fig. 6b, and the children of *l* are promoted to become children of the new node *l*+*j*. The result can be seen in Fig. 6b.

Algorithm 2: Remove a token w from the tree T.

```
1 Algorithm: $Remove(w, T)$
  Input  : $T = \{(w_i, w_j) \dots (w_m, w_n)\}$ ;          // Input tree
  Input  : $w$ ;                                            // Word to remove.
  Output: $T'$ ;                                            // Modified tree
2 begin
3     $T' = T - \{(w, w_i) | w_i = parent(w, T)\}$       // Remove edge between $w$ and parent $w_i$
4             $-\{(w_j, w) | w = parent(w_j, T)\}$         // Remove edges for children of $w$
5             $+\{(w_j, w_i) | w_i = parent(w, T), w = parent(w_j, T)\}$ ;
      /* Finish by "promoting" former children of $w$ to now attach to $w$'s
         parent, $w_i$.                                                     */
6 return $T'$
```

Algorithm 3: Merge a child w_c and parent w_p in the tree T, and “promote” the children of w_c to be children of w_p.

```
1 Algorithm: $Merge(w_c, w_p, T)$
  Input : $T = \{(w_i, w_j) \dots (w_m, w_n)\}$ ;            // Input tree
  Input : $w_c$ ;                                          // Child word to merge.
  Input : $w_p$ ;                                          // Parent word to merge.
  Output: $T'$ ;                                           // Modified tree
2 begin
3     $T' = T - \{(w_c, w_p)\}$
4              $-\{(w_i, w_c) | w_c = parent(w_i, T)\}$
5              $+\{(w_i, w_p) | w_c = parent(w_i, T)\}$;
6 return $T'$
```

Algorithm 4: Swap a child w_c and parent w_p in the tree T.

```
1 Algorithm: $Swap(w_c, w_p, T)$
  Input : $T_L = \{(w_i, w_j) \dots (w_m, w_n)\}$ ;          // Input tree
  Input : $w_c$ ;                                          // Child word to swap.
  Input : $w_p$ ;                                          // Parent word to swap.
  Output: $T'$ ;                                           // Modified tree
2 begin
3     $T' = T - \{(w_c, w_p)\} + \{(w_p, w_c)\}$      // Swap the order in the $(w_c, w_p)$ edge
4              $-\{(w_p, w_i) | w_i = parent(w_p, T)\}$     // Remove edges for parent $w_p$
5              $+\{(w_c, w_i) | w_i = parent(w_p, T)\}$ ;   // Add edges from $w_p$'s parent to $w_c$
6 return $T'_L$
```

3.3.3 O3: Swap

The *swap* operation is used when two nodes in one tree are aligned to two nodes in the other tree, but in a reciprocal relationship, as shown in Fig. 5c. This operation can be used to handle certain divergence types such as *demotional* and *promotional* divergence, which will be discussed in more detail in Sect. 3.7.

Figure 6c illustrates how the swap operation takes place by swapping nodes *l* and *j*. Node *j*, the former parent, is *demoted*, keeping its attachment to its children. Node *l*, the former child, is *promoted*, and its children become siblings of node *j*, the result of which can be seen in Fig. 6c. Note that the *swap* operation does affect multiple edges simultaneously, and thus can create a mismatch on one edge while fixing that of another. We allow for this possibility since trees that exhibit such behavior are rare, and will not be easily reconciled.

3.4 Calculating tree matches after applying operations

The operations O1–O3 are proposed to handle common divergence cases in C1–C3. To measure how common C1–C3 is in a language pair, we designed an algorithm that transforms a tree pair based on a word alignment.

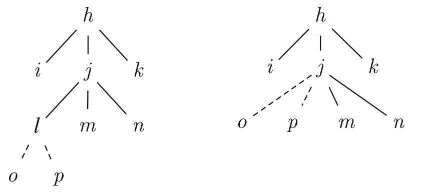
Algorithm 5: Algorithm for altering an aligned tree pair.

```
   input : $c = (F, E, A)$ ;                      // A parallel sentence with alignment
   output: $c' = (F', E', A')$ ;                  // modified output sentence.
1  Let $F = (W_F, T_F)$ ;
2  Let $E = (W_E, T_E)$ ;
3  Let $A = \{(f_i, e_j), \ldots, (f_k, e_l)\}$ ;
4  begin
       // Step 1(a): Remove spontaneous nodes from $F$
5      foreach $f_i \in W_F$ do
6          if $\nexists e_j : (f_i, e_j) \in A$ then
7              $T_F = Remove(f_i, T_F)$ ;                  // See Algorithm 2
       // Step 1(b): Remove spontaneous nodes from $E$
8      foreach $e_j \in W_E$ do
9          if $\nexists f_i : (f_i, e_j) \in A$ then
10             $T_E = Remove(e_i, T_E)$ ;                  // See Algorithm 2
       // Step 2(a): Find nodes to merge in $F$ and merge them
11     foreach $(f_i, e_j) \in A$ do
12         Let $f_p = parent(f_i, T_F)$ ;
13         if $(f_p, e_j) \in A$ then
14             $T_F = Merge(f_i, f_p, T_F)$ ;              // See Algorithm 3
15             $A = A - \{(f_i, e_j)\}$ ;
       // Step 2(b): Find nodes to merge in $E$ and merge them
16     foreach $(f_i, e_j) \in A$ do
17         Let $e_p = parent(e_j, T_E)$ ;
18         if $(f_i, e_p) \in A$ then
19             $T_E = Merge(e_j, e_p, T_E)$ ;              // See Algorithm 3
20             $A = A - \{(f_i, e_j)\}$ ;
       // Step 3: Find nodes to swap in $F$ and swap them
21     foreach $(f_i, e_j) \in A$ do
22         Let $f_p = parent(f_i, T_F)$ ;
23         if $\exists e_c : e_j = parent(e_c, T_E)$ and $(f_p, e_c) \in A$ then
24             $T_F = Swap(f_i, f_p, T_F)$ ;               // See Algorithm 4
25     return $(F', E', A')$ ;
```

The algorithm takes a tree pair (F, E) and a word alignment A as input and creates a modified tree pair (F', E') and an updated word alignment A' as output. It has several steps. First, spontaneous nodes (nodes that do not align to any node on the other tree) are removed from each tree. Next, if a node and its parent align to the same node on the other tree, they are merged and the word alignment is changed accordingly. Finally, the swap operation is applied to a node f_i and its parent f_p in one tree if they align to e_i and e_p respectively and e_p is a child of e_i in the other tree. The pseudocode of the algorithm is shown in Algorithm 5.

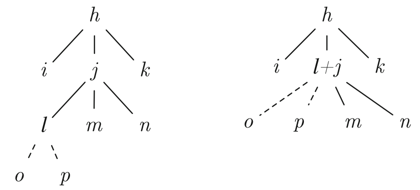
Now given a corpus C and word alignment between each sentence pair, we can measure the impact of C1–C3 by comparing $CorpusMatch_{E \to F}(C)$ scores before and after applying operations O1–O3. This process can also reveal some patterns of divergence (e.g., what types of nodes are often merged), and the patterns can later be used to enhance existing projection algorithms.

$$Remove(l, T) : \left\{ Children(j) = \{o, p, m, n\} \right.$$



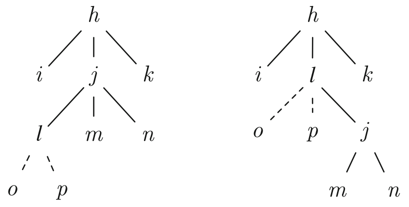
(a)

$$Merge(j, l, T) : \begin{cases} Children(j + l) = \{o, p, m, n\} \\ Children(h) = \{i, j + l, k\} \end{cases}$$



(b)

$$Swap(j, l, T) : \begin{cases} Children(j) = \{m, n\} \\ Children(h) = \{i, l, k\} \\ Children(l) = \{o, p, j\} \end{cases}$$



(c)

Fig. 6 Trees showing the results of the operations defined in O1–O3. *Children*(w) returns the set of words that depend on w. Here we show the value of *Children*(*node*) after the operations only if its value is changed by the operations. (**a**) Before and after the node l has been removed (O1). (**b**) Before and after the nodes l and j have been merged (O2). (**c**) Before and after the nodes l and j have been swapped (O3)

3.5 Improving projection algorithms

With the tree operations described above, we can detect potential post-processing rules automatically. In Georgi et al. (2013), we showed that by coupling the divergent *remove*, *merge*, and *swap* cases C1–C3 with corresponding operations O1–O3, we are able to keep statistics on the affected nodes, and then use these statistics to make the following corrections to the projection algorithm:

1. Spontaneous: better informed attachment of spontaneous words.
2. Merge: better informed choice for head for multiply-aligned words.
3. Swap: post-projection correction of frequently swapped word pairs.

Figure 7 shows flowcharts for the learning and applying of the correction patterns described in the following sections. Compared to the basic projection algorithm (System 1) illustrated in Fig. 4, the improved projection algorithm has two stages: in the training stage, correction rules are learned by comparing the projected trees

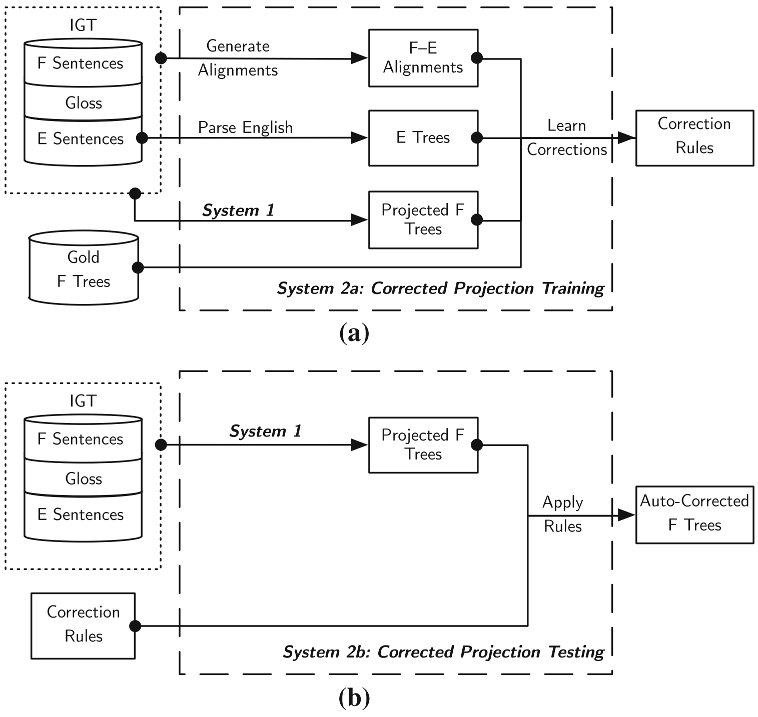
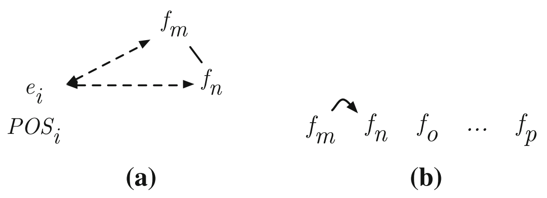


Fig. 7 Flowcharts for the learning and applying of correction rules for a basic projection algorithm. "System 1" refers to the basic projection algorithm illustrated in Fig. 4. (**a**) Flowchart for the training phase of corrected projection, in which the projections are produced, and then the correction rules are learned from a set of gold trees. (**b**) Flowchart for the testing phase of the corrected projection, in which the projection rules learned in **a** are applied to the projected trees from System 1

produced by System 1 with the gold standard trees for the F sentences. In the test stage, those rules are applied to the projected trees produced by System 1.

3.5.1 Spontaneous reattachment

The analog to the *Remove* operation in modifying the projection algorithm is determining how to reattach spontaneous (unaligned) words. Given that they are unaligned, no information from *E* is projected to them, so a lexicalized approach is used. First, we note all lexical items in the training trees and the relative position of their head (left/right). Second, we select the attachment direction for every word in the training data as noted and the attachment direction for the language as a whole. At test time, if the spontaneous word appears in the training data, we use either the word's preference based on the training data to make a left or right local attachment, otherwise we use the language's overall attachment direction as a backoff.



$\mathtt{POS}_i$	$\rightarrow$	left	0.75
$\mathtt{POS}_i$	$\rightarrow$	right	0.25

(c)

Fig. 8 Example of merged alignment and rules derived from the merged alignment. (**a**) Alignment between an English word e_i and two foreign-language words $\{f_m, f_n\}$, where f_m is the parent of the other word f_n. (**b**) Words in sentence F showing the "*left*" dependency between f_m and f_n. (**c**) Rules for handling merged alignment

3.5.2 Merge correction

As shown in Fig. 5b, "merged" alignments are those for which there are multiple words in W_F aligned to a single word in W_E. The difficulty facing projection algorithms in this instance is that it is not clear which of these multiply-aligned words should be made the head, and which the dependent.

In order to correct the projection at runtime, we would like to be able to know which of the multiply aligned words should be selected as the head. By keeping statistics on whether the multiply aligned words for a given POS tag tend toward the left or the right, we can then use the POS tag that is present at runtime to select the headedness for attachment.

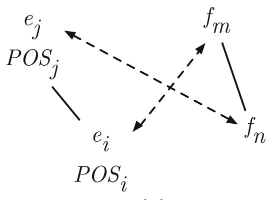
Figure 8a illustrates a detected merge case, while Figure 8b demonstrates the "left" direction of the multiply aligned dependency between the two words. Finally, Figure 8c shows an example set of rules for a given English POS tag learned by this method. At projection time, the direction of the merge is chosen by the most likely rule learned by the analysis, or by the language's overall headedness preference as a backoff.

These preferences can easily be learned by examining the attachments for each word in the corpus, and finding the proportion of those tokens that attach to words to the left versus those to the right, then using the majority preference at testing time.

3.5.3 Swap correction

Swapped alignments, as illustrated in Fig. 9a, are not patterns which would be able to be corrected in projection without some previous training, since it would require the foreign-language tree T_F to already exist. Unfortunately, as we will discuss later with Hindi, these swaps can be frequent enough to cause serious performance problems in projection.

In order to correct for the swapped elements, we analyze the edges for which the swap operation was triggered, similar to the merge operation above. However, rather than keeping track of only a single part of speech tag, we instead keep statistics on the (POS_{child}, POS_{parent}) edge in T_E, and the number of times that corresponds to a swap operation in T_F. Based on the collected counts, we keep only



(a)

POS Pair		Swaps	Total	%
$(\mathrm{POS}_i, \mathrm{POS}_j)$	$\rightarrow$	16	21	76
$(\mathrm{POS}_k, \mathrm{POS}_l)$	$\rightarrow$	1	1	100
$(\mathrm{POS}_n, \mathrm{POS}_o)$	$\rightarrow$	1	10	10

(b)

Fig. 9 Example swap configuration and collected statistics. (**a**) A swapped alignment between words e_j and e_i and words fm and f_n. (**b**) Example set of learned swap rules. Swaps counts the number of times the given (*child*, *parent*) pair is seen in a swap configuration in the English side, and total is the number of times said pair occurs overall. The % column lists the frequency % of the swap

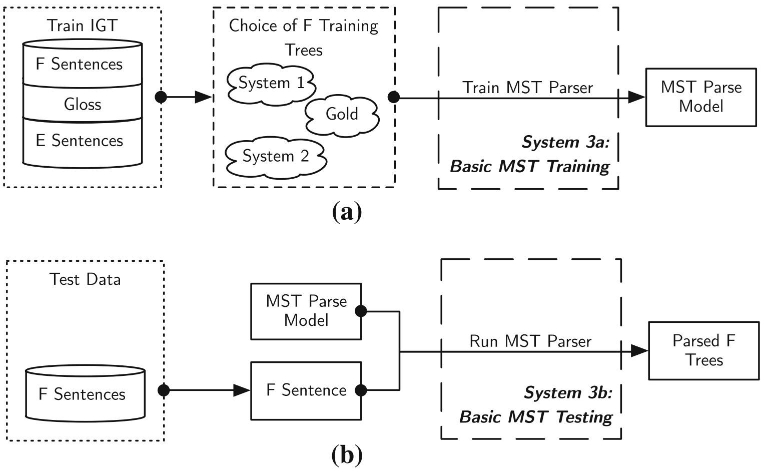


Fig. 10 Flowcharts for simple parser trained on projected trees alone. (**a**) Flowchart for the training phase of the basic parser. The training trees used to train the parser can be produced by System 1 (see Fig. 4), System 2 (See Fig. 7), or gold standard trees. (**b**) Flowchart for the testing phase of the basic parser, which requires only an F sentence and Parse model trained in **a**

the pairs that occur in at least of 10 % of the training sentences, and a frequency of at least 70 %[1].

To apply the rules, after projection, the POS tag pairs that meet the given requirements are swapped using the *Swap* operation defined by O3 in Sect. 3.3. The results of applying these post-processing rules will be discussed in Sect. 4.5.

3.6 Bootstrapping a parser

Given that the analysis described here uses a small amount of training data to build these rules, one can also train a parser using the same data. Figure 10 illustrates a

[1] These thresholds are set empirically to filter for rules that occur in multiple sentences with high regularity.

dependency parser trained on the small amount of monolingual data available for the given language, produced either by projection or by gold standard data. However, with such little data such a system is often outperformed by even the basic projection method, as noted in Georgi et al. (2012), cf. S3-1 in Table 5 versus S1-R/L in Table 4. In this previous paper, we used the edges of the projected trees as a feature to extend the MST Dependency Parser (McDonald et al. 2006) and the experiments showed an increase in performance over both the baseline parser and the basic projection algorithm in many cases. Here, we take the baseline parser and add a number of features based on projected trees provided at testing time (Fig. 11).

As previously defined, let $E = (W_E, T_E)$ and $F = (W_F, T_F)$, the latter being the tree built by the projection algorithm. $R_{F\to E}(f_i, A)$ and $R_{E\to F}(f_i, A)$ are defined the same way as Eqs. (8) and (9) in Sect. 3.2. Now we can define the projection-based features in Eqs. (12) and (13).

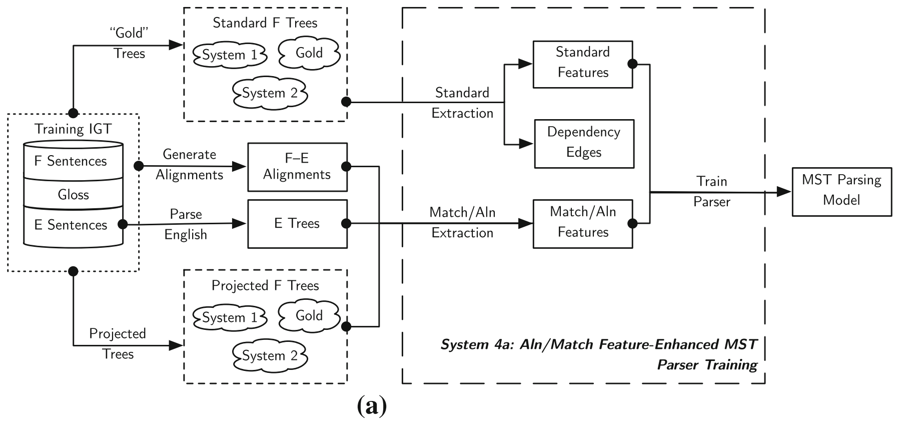
$$ProjBool(f_i, f_j) = \begin{cases} \texttt{TRUE} & \textbf{if } (f_i, f_j) \in T_F \\ \texttt{FALSE} & \text{otherwise} \end{cases} \tag{12}$$

$$AlignType(f_i, f_j) = \begin{cases} \texttt{IS\_SINGLE} & \textbf{if} \quad \left|R_{F\to E}(f_i, A)\right| = 1 \\ \hline \texttt{IS\_UNALIGNED} & \textbf{if} \quad \left|R_{F\to E}(f_i, A)\right| = 0 \\ \hline \texttt{IS\_MATCH} & \textbf{if} \quad \begin{aligned} &\exists e_i, e_j \Big(e_i \in R_{F\to E}(f_i, A) \\ &\quad \wedge e_j \in R_{F\to E}(f_j, A) \\ &\quad \wedge (e_i, e_j) \in T_E\Big) \end{aligned} \\ \hline \texttt{IS\_MERGE} & \textbf{if} \quad \begin{aligned} &\exists e_i \Big(e_i \in R_{F\to E}(f_i, A) \\ &\quad \wedge f_j \in R_{E\to F}(e_i, A)\Big) \end{aligned} \end{cases} \tag{13}$$

The *ProjBool* feature, as defined in (12) is the basic feature used for marking agreement between an edge in the projected tree and an edge being considered by the parser. This feature simply takes a TRUE value if the edge being considered by the parser also occurs in the projection. The *AlignType* (13), on the other hand, is actually a group of binary features used to subdivide agreement with the projection based upon the type of alignment exhibited by the word e_i to words in T_F. *AlignType* has four sub-features based on possible alignment types, which are illustrated in Fig. 12. IS_SINGLE (Fig. 12a) is TRUE when a token f_i aligns to only one word in T_E. IS_UNALIGNED (Fig. 12b) is triggered when f_i is a spontaneous word; that is, it does not align to any word in T_E.

IS_MATCH is TRUE in the case where the edge (f_i, f_j) is being considered by the parser for the edge and the parent and child align with words e_i, e_j in T_E, and have the same parent/child relationship, as seen in Fig. 12c. IS_MERGE is TRUE when the foreign language word f_i is one of multiple foreign words aligned with a single English word e_i (Fig. 12d).

Adding these features to the original feature set used by the MST Parser results the System 4, illustrated in Fig. 11. In the training stage, Match/Aln features are



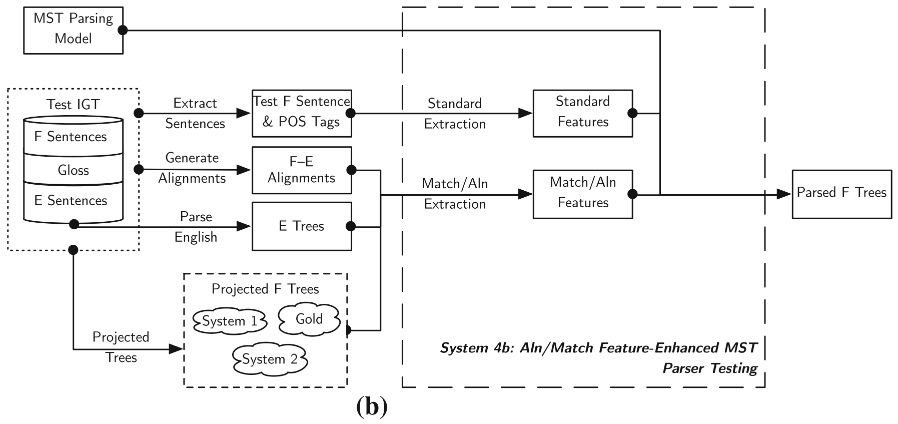


Fig. 11 Flowcharts for the improved parser parser described in Sect. 3.6. (**a**) Flowchart showing the training, using both match and alignment features. Note that the "gold" trees used to extract the standard features and dependency edges for training the parser are separate from the additional trees provided to the parser to extract match and alignment features. These additional trees are typically the projected trees, which will be available to the parser at testing time, but gold trees can be used for an oracle experiment. (**b**) Flowchart showing the testing phase, using the parse model produced in **a** as well as additional trees provided to the parser, using the system chosen for this task in **a**. Using the match and alignment features with these trees, and the standard features from the training phase, the parser produces an improved set of parsed F trees

extracted from the E trees and projected trees. Those features are added to the standard feature set used by the MST parser, and the expanded feature set is used to train the MST parser.

In the test stage, the Match/Aln features are extracted from projected F trees, and they are added to the standard features. Because of the addition of the Match/Aln features, unlike System 3, System 4 requires the test F sentences to have an aligned English sentence.

Note that for the projected F trees used to extract Match/Aln features can come from System 1, System 2, or the gold standard. The same is true for the standard

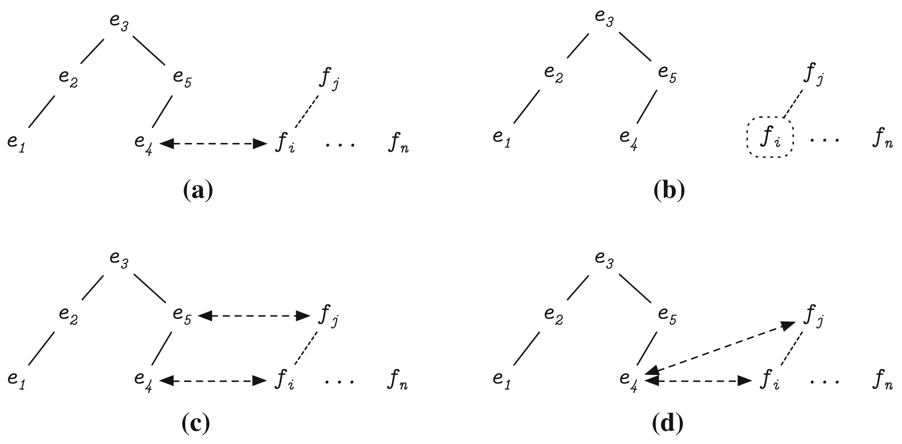


Fig. 12 Different alignment configurations that trigger the *AlignType* feature. (**a**) The IS_SINGLE alignment feature is *red* for token f_i in this conguration. (**b**) The IS_UNALIGNED alignment feature is *red* for token f_i in this conguration. (**c**) The IS_MATCH alignment feature is *red* for tokens (f_i; f_j) in this conguration. (**d**) The IS_MERGE alignment feature is *red* for tokens (f_i; f_j) in this conguration

features. In Sect. 4.6 we will compare the parsing performance when different combinations of F trees are used (see Table 5a, b).

3.7 Relationship to Dorr (1994)

Dorr (1994) lists seven types of divergence for language pairs. While our analysis method is more coarse-grained than the LCS that Dorr proposes, it is nonetheless able to capture some of the same cases.

For instance, Fig. 13a illustrates an example of what Dorr identified as "promotional" divergence, where *usually*, a dependent of the verb *goes* in English, is "promoted" to become the main verb, *suele* in Spanish. In this case, the direction of the dependency between *usually* and *goes* is reversed in Spanish, and thus the *swap* operation can be applied to the English tree and result in a tree that looks very much like the Spanish tree.

A similar operation is performed for *demotional* divergence cases, such as aligning "I like eating" with the German translation *"Ich esse gern"* ("I eat likingly"). Here, the main verb in English ("like") is *demoted* to an adverbial modifier in German ("*gern*"). The *swap* operation is applicable to both types of divergence and treats them equivalently, and so it essentially can handle a superset of promotional and demotional divergence, namely,"head-swapping."

Another type of divergence that can be captured by our approach is Dorr 's "structural" divergence type, as illustrated in Fig. 13b. The difference between the English and Spanish structures in this case is the form of the argument that the verb takes. In English, it is a NP; in Spanish, it is a prepositional phrase. While the tree operations defined previously do not explicitly recognize this difference in syntactic labels, the divergence can be handled by the *remove* operation, where the spontaneous *"en"* in the Spanish side is removed.

```
Juan suele      ir a casa
John tends-to   go home
"John usually goes home"
```

```
Juan entró    en la   casa
John entered  in the  house
"John entered the house"
```

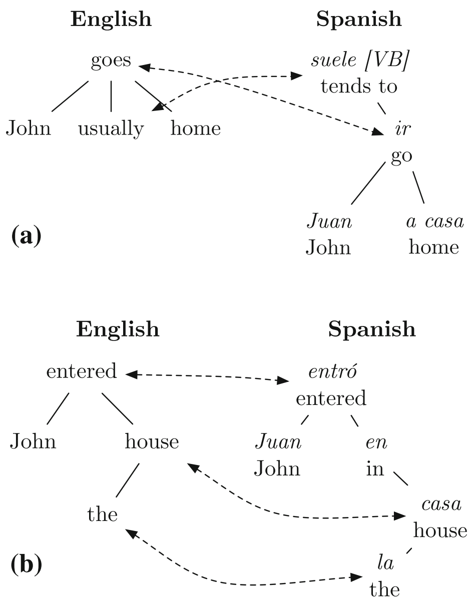


Fig. 13 Two examples of divergence types described in Dorr (1994). (**a**) An example of promotional divergence from Dorr (1994). The reverse in parent-child relation is handled by the Swap operation. (**b**) Example of structural divergence, which is handled by the remove operation

Next, Dorr's description of *conflational* divergence lines up well with the *merge* operation (see Fig. 6b). Figure 14 illustrates an example for English and Hindi, where both sides have spontaneous words (e.g., *to* and *a* in English) and a causative verb in Hindi corresponds to multiple verbs in English. Figure 14b shows the original tree pair, Fig. 14c demonstrates the altered tree pair after removing spontaneous words from both sides, while Fig. 14d shows the tree pairs after the English verbs are merged into a single node. It is clear that the *remove* and *merge* operations make the Hindi and English trees much more similar to each other.

In addition to the four divergence types mentioned above, additional operations could be added to handle other divergence types. For instance, if dependency types (e.g., patient, agent) are given in the dependency structure, we can define a new operation that changes the dependency type of an edge to account for *thematic* divergence, where thematic roles are switched as in "I like Mary" in English vs. *"María me gusta a mí"* (Mary pleases me) in Spanish. Similarly, an operation that changes the POS tag of a word can be added to cover *categorial* divergence where words representing the same semantic content have different word categories in the two languages, such as in "I am hungry" in English versus *"Ich habe Hunger"* (I have hunger) in German.

Compared to Dorr's divergence types, whose identification requires knowledge about the language pairs, our operations on the dependency structure relies on word alignment and tree pairs and can be applied automatically.

mohana ne	kala	Arif se	mInA ko	kiwAba	xilavAyI
Mohan [erg]	yesterday	Arif [inst]	Mina [dat]	book	give-caus

"Mohan caused Mina to be given a book through Arif yesterday."

(a)

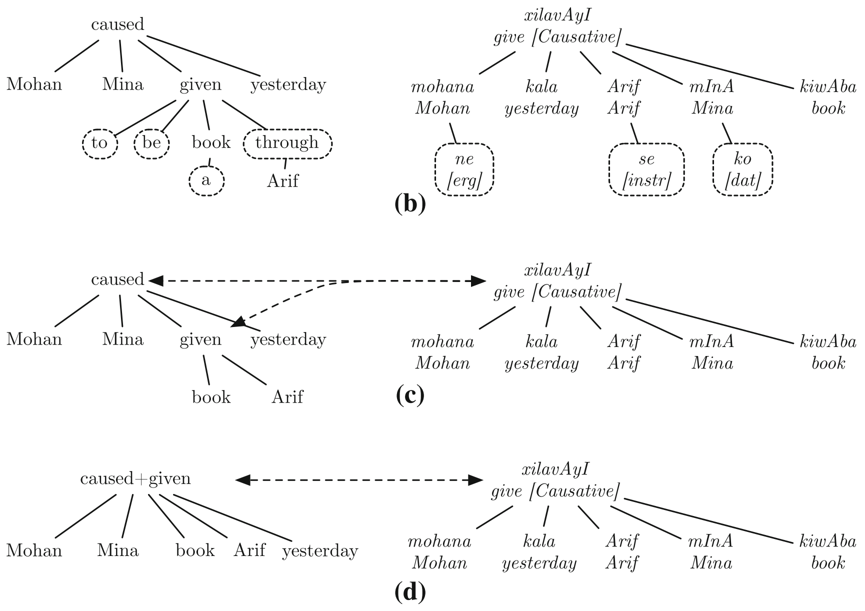


Fig. 14 Case of conflational divergence, handled by remove and merge operations. (**a**) Interlinear text of a sentence pair. *(**b**) Initial trees showing spontaneous words on both sides. (**c**) Altered trees after removing spontaneous words from both sides, and showing conflational divergence between multiple English words and a single Hindi word. (**d**) Altered trees after merging multiple words on the English side

4 Experiments

For evaluation, we ran our systems on a total of eleven language pairs, using the corpora described in Table 1. We will describe the data used for our experiments in Sect. 4.1. Section 4.2 details how the numbers of matches across corpus pairs are counted, using the *CorpusMatch* metric described in Algorithm 1 in Sect. 3.2. Section 4.3 will look at the cases in which the match percentage still doesn't reach 100 % after applying all the tree operations. Section 4.4 will show some of the patterns discovered by breaking down the analysis by POS. Finally, Sect. 4.5 will discuss the results of using the automatically discovered patterns to improve the baseline projection algorithm, while Sect. 4.6 shows the result of using this improved projection algorithm to bootstrap a dependency parser.

4.1 Data

Our work utilizes three corpora for a total of eleven language pairs. The three corpora used are the SMULTRON treebank (Volk et al. 2010), the guideline

Table 1 Data set sizes for all languages

Corpus	Language	# Instances	# Words (F/E)
Hindi Treebank	Hindi	147	963/945
ODIN	German	105	747/774
	Irish	46	252/278
	Hausa	77	424/520
	Korean	103	518/731
	Malagasy	87	489/646
	Welsh	53	312/329
	Yaqui	68	350/544
SMULTRON	German	281	6,829/7,236
	Swedish	281	8,402/9,377
	Ger-Swe†	281	6,829/8,402

All language pairs have English for the second element of the pair, except for the Ger-Swe row marked with †, where the language pair is German/Swedish . The last column shows the number of words in the first language of the language pair (F), followed by the number of words in the second language of the pair (E)

sentences in IGT form from the Hindi treebank (Bhatt et al. 2009), and several sets of IGT data as used in Lewis and Xia (2010). The statistics of the corpora are shown in Table 1. Ten of the language pairs use English as one side of the language, while the eleventh uses the pair of German and Swedish from the SMULTRON corpus.

In the SMULTRON Treebank, the German and Swedish phrase trees are marked for head children, allowing for the automatic extraction of dependency trees. The English side of the phrase structures do not contain edge labels and we converted the phrase structures into dependency trees using a head percolation table (Collins 1999).

From the Hindi Treebank guidelines, we extracted example sentences in the form of IGT (i.e., Hindi sentences, English gloss, and English translation) and the Hindi dependency structures manually created by the guideline designers. We obtained dependency structures for the English translation by running the Stanford dependency parser (Marneffe et al. 2006) and then we hand corrected the structures. Word alignment is initially derived from the IGT instances using heuristic alignment following Lewis and Xia (2010), and later hand-corrected. The IGT data from Lewis and Xia (2010) was obtained in the manually corrected dependency forms as described in Sect. 2.2.

4.2 Match results

By running Algorithm 1, we can calculate the $CorpusMatch_{E \rightarrow F}$ and $CorpusMatch_{F \rightarrow E}$ before and after each operation and see how the operation affects the percentage of matched edges in the corpus. As the operations are applied, the percentage of matches between the trees should increase until all the divergence cases that can be handled by operations O1–O3 have been resolved. At this point, the final match percentage can be

Table 2 Match results in detail for hindi, and in overview for the other 7 languages

	English → Hindi					Hindi → English				
	Match	UnAln	Merge	Swap	Edges	Match	UnAln	Merge	Swap	Edges
(a)										
Initial baseline	47.7	20.9	1.6	9.1	794	46.3	20.7	1.7	8.8	816
After remove	66.1	0.0	2.1	11.7	622	63.4	0.0	2.2	11.3	647
After merge	69.5	0.0	0.0	12.3	586	69.2	0.0	0.0	12.3	587
After swap	90.3	0.0	0.0	0.3	586	89.9	0.0	0.0	0.3	587

	ODIN DATA							SMULTRON DATA		
	YAQ	WLS	KKN	GLI	HUA	GER	MEX	GER	SWE	GER-SWE
(b)										
Initial baseline	75.4	75.4	56.0	72.0	54.4	76.7	57.4	40.7	37.5	43.3
After remove	95.1	95.1	88.1	87.8	95.7	93.9	88.9	63.6	62.2	73.5
After merge	97.2	97.2	95.4	92.5	97.5	95.4	97.4	71.8	73.9	82.8
After swap	98.2	98.2	96.1	94.1	97.5	96.8	98.0	83.0	84.2	87.2

(a) Breakdown of edges as operations are applied to the English ↔ Hindi language pair, given in both directions since the comparison is asymmetrical. The baseline is given, then operations are applied to create new trees. The "Edges" column represents the number of total edges in the trees of the left hand of the language pair. The numbers given in the other columns are the percentages of those edges that are either in a *match*, *swap*, or *merge* alignment, or the edges for which the child is *unaligned* (indicated in the table by 'UnAln')

(b) Summary of the match percentages for the remaining ten language pairs, Yaqui (YAQ), Welsh (WLS), Korean (KKN), Scots Gaelic (GLI), Hausa (HUA), German (GER), Swedish (SWE) and GER-SWE. Except for GER-SWE, English is the first language of the pair

seen as an estimate of the upper-bound on performance of a simple projection algorithm, if C1-C3 can be identified and handled by O1-O3. Table 2a shows the full results of this process for the Hindi-English pair, while Table 2b shows a summary for the results in the remaining ten languages.

The results in Table 2a show that the trees start out very dissimilar between English and Hindi, having only 47.7 % of the edges in the English trees matching those in the Hindi trees initially. After removing words that are not aligned between English and Hindi, still only 66.1 % are aligned. While merging multiply-aligned words improves this match by 3.4 %, applying swaps in the English trees results by increasing matches by a large 20.8 %. The reason for this large increase in this language pair can be attributed to the way in which prepositions and postpositions are represented in Hindi, which is explained further in Sect. 4.4.

Between Hindi in Table 2a and the other languages in Table 2b, the application of the operations increases the match percentage, but never reaches 100 %. The match percentage after all operations are applied can be seen as an upper bound on the tree similarities between the language pair for a given corpus.

```
merA mana amarIkA meM hE
my   mind America in  is
"I am mentally in America"
```

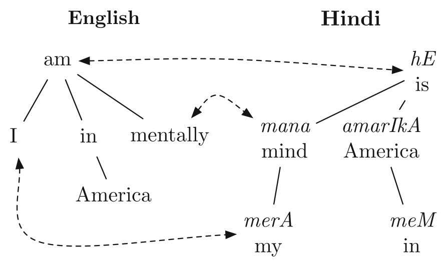


Fig. 15 A tree pair that still has unmatched edges after applying the algorithm in Table 5. The dotted line indicates word alignment that would be needed to resolve the divergence with the *extended* merge operation

4.3 Remaining cases

After applying three operations, there may still be unmatched edges. An example is given in Fig. 15.[2] The dependency edge *(in, America)* can be reversed by the *swap* operation to match the Hindi counterpart. The mismatch in this sentence is that the adverb *mentally* in English corresponds to the noun *mana (mind)* in Hindi. If the word alignment includes the three word pairs as indicated by the dotted lines, one potential way to handle this kind of divergence is to extend the definition of *merge* to allow edges to be merged on both sides simultaneously—in this case, merging *am* and *mentally* in the English side, and *hE (is)* and *mana (mind)* on the Hindi side.

4.4 Operation breakdown by POS

After performing the operations as seen in Sect. 4.2, we can get further insight into what precisely is happening within each language by breaking down the operations by the POS tags on which the operations apply. Table 3 shows some of these POS tag breakdowns for a number of languages, and the frequency with which the given operation applies to the POS tag or POS tag pair out of all the times it is seen in that language. For example, the results in Row 1 shows that when a modal (MD) depends on a verb (VB) in English, 43.9 % of the time in the training data, the two words align to the same word in Hindi. Table 3 shows expected phenomena from the language pairs. For instance, Rows 5 and 6 show the English→German pair merging many nouns as multiple English words are expressed as compounds in German. In another case, Row 8 shows that all Hindi nouns undergo *swap* with prepositions, as Hindi uses postpositions. Noticing the regularity with which NN and IN swap leads us to the next experiment, where we examine how such regularly-occurring rules might be harnessed to improve projection.

4.5 Analyzing trees for post-processing rules

Table 4 compares the projection accuracy of the basic projection algorithm (System 1 as in Fig. 4) and the improved projection algorithm (System 2 as in Fig. 7). For all the experiments, we use tenfold cross validation with a 9:1 training/test split.

[2] It is a topic of debate whether *mentally* in English should depend on *in* or *am*. If it depends on *in*, handling the divergence would be more difficult.

Table 3 Breakdown of significant merge and swap statistics for various language pairs, where the language to the left of the arrow is the one being altered

Lang pair	Row #	Child POS	Parent POS	% All cases
		Merges		
Eng→Hin	1	MD	VB	42.9
	2	NN	NN	14.3
Hin→Eng	3	VAUX	VM	45.4
	4	NN	VM	5.5
Eng→Ger	5	NN	NNS	66.7
	6	NN	NN	65.4
	7	NNS	NN	4.2
		Swaps		
Hin→Eng	8	IN	NN	100
	9	NNP	IN	20.0
Ger→Eng	10	APPRART	NN	72.7
	11	NN	CC	61.5
		Removals		
Eng→Hin	12	DT		86.4
	13	TO		75.6
Hin→Eng	14	PSP		69.8
	15	VAUX		18.6
Eng→Ger	16	POS		57.1
	17	DT		20.2
Ger→Eng	18	PRF		85.2
	19	ADV		43.9

In this table, System 1 (S1) serves as a baseline. In System 1, when a merge alignment was detected or a spontaneous word needed reattaching, the algorithm simply attached rightward (S1-R) or leftward (S1-L). The results are in the last two rows of Table 4b. The numbers in the table are Unlabeled Attachment Scores (UAS). In contrast, System 2 (S2) uses the training data to learn which direction the attachment is more common. In addition to the merge rules, System 2 can also apply rules for swap, or remove. These results are shown in the first three rows of Table 4b. The table shows that applying automatically learned rules improves projection accuracy significantly, the average accuracy increasing from 83.22 % (S1-L) or 83.58 % (S1-R) to 88.95 % (S2-1).

4.6 Parsing experiments

In Sect. 3.6, we described two parsing approaches: System 3 and System 4. In System 3 (see Fig. 10 in Sect. 3.6), the MST parser uses the standard feature set without using Match/Aln features. It is trained with the projected trees produced by System 1, System 2, or gold standard F trees (As shown in the “Train Source” Column in Table 5a. For

Table 4 Settings and unaligned attachment scores for the baseline and improved projection algorithms

	(a) Projection Algorithm Settings		
System Name	Merge	Swap	Remove
S2-1	X	✓	✓
S2-2	X	✓	
S2-3	X		
S1-R	R		
S1-L	L		

	(b) Projection Algorithm Results								
System Name	YAQ	WLS	HIN	KKN	GLI	HUA	GER	MEX	**AVG**
S2-1	88.03	94.90	77.38	91.75	87.70	90.11	88.71	93.05	**88.95**
S2-2	88.03	94.90	69.02	91.55	87.70	90.11	88.71	93.05	**87.86**
S2-3	87.28	89.80	68.60	90.34	86.90	79.54	88.03	89.57	**84.68**
S1-R	*87.28*	*89.80*	57.41	*90.34*	*86.90*	79.31	*88.03*	*89.57*	**83.58**
S1-L	84.29	*89.80*	*68.60*	88.93	76.98	*79.54*	*88.03*	*89.57*	**83.22**

(a) Settings for the different projection algorithms. The S1 systems are the baseline projection algorithm as illustrated in Figure 4 in Sect. 2.2, preferring either right-branching (R) or left-branching (L) for attachment and merging defaults. The S2 systems refer to the one illustrated in Figure 7 in Sect. 3.5. iteratively add the swap, remove, and merge correction. The merge correction in the S2 systems prefers the attachment directions determined by the discovery process in Sect. 3.5

(b) The results of the baseline (S1) and improved (S2) projection algorithms among eight language pairs: Yaqui (YAQ), Welsh (WLS), Hindi (HIN), Korean (KKN), Scots Gaelic (GLI), Hausa (HUA), German (GER), and Malagasy (MEX). Among the S1 systems, the best performing baseline is shown in italics

the feature set, we can use either the word features only or add POS tag features in the F sentences (as indicated by the checkmark in the "Projected POS Feat" column). If only the word features are used, the input are F sents with words only. If POS tag features are used, for both training and test data, we use the POS tags projected from the English side. The parsing results are shown rows labeled S3-1 to S3-5 in Table 5.

In System 4 (see Fig. 11 in Sect. 3.6), Match/Aln features are added to the standard feature set used by the MST parser. Standard features can be extracted from System 1, System 2, or gold F trees. The same is true for the Match/Aln features. The sources of the trees are indicated in the "Train Sources" column in Table 5a. At the test stage, standard features are extracted from test F sentences with projected POS tags, and the source of the Match/Aln features is indicated in the "Test Source" column. The "Oracle" system is the system in which Gold F trees are used both for training and testing. S4-1 through S4-3 are a few variants of System 4 when different combinations of train and test sources are used.

There are several observations: First, adding the Match/Aln features improves performance significantly, from 89.32 % in S4-1 versus 79.16 % in S3-1. Second, using the improved projection algorithm improves parsing results compared to using the basic projection algorithm for both System 3 (row S3-2 vs. S3-3 and S3-4) and

Table 5 Settings and results for MST parser systems 3 & 4

(a) MST parser settings

System Name	Match/Aln Feats	Projected POS feats	Train Sources	Test Source
Oracle	✓	✓	Gold, Gold	Gold
S4-1	✓	✓	Gold, S2-1	S2-1
S4-2	✓	✓	Gold, S1-R	S1-R
S4-3	✓	✓	Gold, S1-L	S1-L
S3-1		✓	Gold	—
S3-2		✓	S2-1	—
S3-3		✓	S1-R	—
S3-4		✓	S1-L	—
S3-5			Gold	—

(b) MST parser results

System name	YAQ	WLS	HIN	KKN	GLI	HUA	GER	MEX	AVG
Oracle	96.51	98.26	98.03	99.17	95.63	99.31	98.07	97.93	**97.86**
S4-1	89.28	94.90	81.35	92.96	81.35	88.74	92.93	93.05	**89.32**
S4-2	*88.28*	*94.22*	78.03	*92.35*	*80.95*	87.59	*90.48*	*92.43*	**88.04**
S4-3	87.88	*94.22*	*79.64*	90.95	*80.95*	*89.20*	*90.48*	*92.43*	**88.22**
S3-1	84.54	71.43	76.76	82.49	66.67	86.44	81.09	83.84	**79.16**
S3-2	81.30	74.49	59.73	83.30	59.92	85.29	78.78	82.82	**75.70**
S3-3	83.04	69.05	38.59	82.49	61.51	68.05	77.14	79.35	**69.90**
S3-4	79.05	69.05	49.42	81.09	56.75	73.33	77.69	79.75	**70.77**
S3-5	66.58	32.99	48.90	77.06	51.98	48.51	56.05	58.69	**55.10**

(a) Matrix of settings for the MST parser experiments. The Oracle system is a version of the S4 system that gold trees for both the training, additional features, and at test time. The S4-1 through 3 systems use a portion of the gold trees at training time, in conjunction with projected trees from the S2 system, or a version of the S1 system where attachment is left-default (S1-L) or right-default (S1-R). Finally, the S3-1 through 5 systems train the parser using only a set of monolingual trees, from the sources noted by "Train Sources."

(b) Results for the systems described in (a) across the 8 language pairs

System 4 (row S4-1 vs. S4-2 and S4-3). A further discussion and comparison of the average scores of the system can be found in Sect. 5.3.

5 Discussion of results

The results of the experiments above show that the match scoring that we have introduced here has the potential to address many interesting issues arising between language pairs. In this section, we highlight some observations based on the experimental results.

5.1 Match scores

The results of Table 2a and b compare similarity both across languages and across corpora. For instance, in the scores for the baseline ODIN data, we see that the baseline for matches between English and German is the highest out of all the pairs at 76.7 %. Scots Gaelic and Welsh are 72 and 75.4 %, respectively. Hausa, Malagasy, Korean, and Yaqui all show baseline scores between 54–57 %. This seems in line with what we would expect, with German and the Celtic languages being closely related to English, and the others being unrelated.

Another stark contrast can be seen between all the languages in the ODIN data and the languages in the SMULTRON corpus. While the ODIN sentences tend to be short sentences used primarily for illustrative purposes, the SMULTRON corpus consists of economic, literary, and weather domains. As Table 1 shows, the SMULTRON sentences are much longer on average. A closer look at the SMULTRON translations also shows them to be much freer translations than those found in the ODIN data. While the size of the data sets used here are very small, and the ODIN IGT data may be biased towards illustrative purposes (described as the "IGT Bias" in Lewis and Xia (2010)), it would appear that these results illustrate that the match detection highlights two types of differences among the corpora. First, by comparing baselines match results among comparable corpora, basic similarities between languages appear to pattern as expected. Second, the freer translations in the SMULTRON data appear with lower *match* scores across all instances.

One final item of interest from the match results can be seen in the Hindi data in Table 2a. Here, there appears to be a large increase in match percentage after the *swap* operation has been performed. As previously noted, knowing this is the inspiration for automatically inferring the swap rules in Sect. 4.5.

5.2 POS breakdowns

The breakdown of the operations by language and POS in Table 3 provides a good opportunity to check that the defined operations conform with expectations for specific languages.

For instance, Row 1 in Table 3 shows Modals (MD) merging with a parent (VB). This is in line with instances such as Fig. 14c where Hindi combines aspect with a verb that is typically expressed as a separate word in English. This does not appear to be a very frequent occurrence, however, as it only occurs for 42.9 % of MD→VB dependencies.

Row 3, going from Hindi to English shows the case where auxiliary verbs VAUX merge with main verbs VM. These cases typically represent those where Hindi expresses tense as an auxiliary verb, whereas English tense is expressed by inflection on the verb.

With regard to spontaneous words in English and Hindi, Row 14 shows that 69.8 % of case markers (PSP) were removed from Hindi that were either absent in English or applied as inflections to the noun, while 86 % of determiners in English were removed, as Hindi does not have definite or indefinite determiners (Row 12).

Examining the English and German data in Table 3, we first see in Row 5 that 66.7 % of NN-NNS dependencies in English merge. This, along with the 65.4 % of NN-NN dependencies merging, is something we would expect to see in German, as it compounds nouns with far more frequency than English. Interestingly, as Row 7 shows, a plural noun child never merges with a parent noun.

Finally, looking more closely at the swaps, we see a 100 % of NN→IN dependencies are swapped in Hindi, giving further impetus for the rules as described in Sect. 4.5.

5.3 Performance summary

In this study, we proposed four systems for parsing F sentences: basic projection (System 1), improved projection (System 2), original MST parser trained with projected F trees (System 3), and MST parser with additional Match/Aln features (System 4). The results are in Tables 4 and 5. A summary of the most important results of these tables and the comparison of the systems are in Table 6a, and the error reduction of some system pairs are in Table 6b.

As Table 6b shows, in average across the eight languages, the reduction in error in using the parser trained with match and alignment features (S4-1) has a 48.76 % reduction in error over the parser trained on the gold standard trees with projected POS tags (S3-1). In the projection systems alone, the improved projection system S2-1 reduces error over the baseline system S1-R by 32.73 %. Finally, even without the Match/Aln features, a monolingual parser trained on projected dependency trees (S3-2) shows 19.27 % fewer errors than the system trained on the basic projections (S3-3).

While the improvement over the modified projection algorithm is modest, a dependency parser such as S4-1 does have the advantage of being more noise-robust. For instance, given an English sentence where the head of the English sentence is not aligned to any of the words in the language line, projection algorithms will not produce a tree structure, whereas the MST Parser will produce a tree structure based on other features when the alignment is not available. This advantage makes such an approach appealing for extending to larger corpora, which is something we will address in Sect. 6.

5.4 Remaining issues

Two large issues that our methodology faces are data sparsity and translation quality of the sentence pairs in the data sets. The former is somewhat inevitable given the task—a reasonable amount of annotated data is not always likely to exist for languages with scarce electronic resources, and guaranteeing coverage is difficult. As with the Hindi data, however, using IGT as a resource has convenience in both covering wide varieties of phenomena in a language, and providing a gloss that assists in creating word-level alignments. Creating dependency annotation on a small set of data from a source like ODIN (Lewis and Xia 2010) can get a lot of mileage with a small amount of investment.

Table 6 Summary of results from projection and parser systems

(a) Summary of Average Accuracies Across Languages							
	Parser Used	Proj. Used	Match/Aln Feats Used	Proj. Corrected	IGT/Bitext Req'd at Test Time	Avg. Parse Accuracy	Detail In
S4-1	✓	✓	✓	✓	✓	**89.32**	Table 5
S2-1		✓		✓	✓	**88.95**	Table 4
S1-R		✓			✓	**83.58**	
S3-1	✓					**79.16**	
S3-2	✓	✓		✓		**75.70**	Table 5
S3-3	✓	✓				**69.90**	
S3-5	✓					**55.10**	

(b) Error reduction between systems

Systems	Description	Avg % error reduction
S4-1 versus S3-1	Use of match/Aln features versus parser	48.76
S2-1 versus S1-R	Improved projections versus baseline projection	32.73
S3-2 versus S3-3	Parser trained on improved projections versus baseline projections	19.27

(a) Summary of select results from Tables 4 and 5 in decreasing order of performance. For full explanation of individual system parameters, see the referenced table
(b) Average percent reduction in error in comparisons between the systems from (a)

Perhaps the more challenging issue is determining whether divergence in a language pair is caused by fundamental differences between the languages, or simply stylistic choices in translation. The latter of these scenarios appeared to be common in portions of the SMULTRON data, where translations appeared to be geared toward naturalness in the target language; in contrast, the translations in the Hindi guideline sentences were intended to be as literal as possible. Again, IGT provides a good possible solution, as such examples are often intended specifically for illustrative purposes.

6 Conclusion and future work

In this paper, we have demonstrated a generalizable approach to detecting patterns of structural divergence across language pairs using simple tree operations based on word alignment. We have shown that this methodology can be used to detect similarities between languages on a coarse level, as well as serve as a general measure of similarity between dependency corpora. Finally, we establish that harnessing these detection methods improves standard projection algorithms and informs dependency parsing with little to no expert involvement.

For future work, we plan to focus on two areas. The first is that of adapting these techniques to larger data sets. In particular, use of the high-quality alignments derived from IGT to bootstrap a statistical aligner may allow for reasonable performance on languages for which the amount of parallel data may not be sufficient for building a high-quality statistical word aligner. Secondly, while this paper explores the utility of IGT in terms of word alignment and projection, we are currently looking into the ways in which the additional morphemic and lexicosyntactic information in the gloss lines may be used to perform more complex automated linguistic analysis.

The techniques described here are promising for maximizing the effectiveness of existing resources such as IGT for languages where such resources are limited. While access to electronic resources continues to increase globally, many of these resource-poor languages are still left behind in terms of NLP tools. Though projection techniques may not ultimately be full replacements for large treebank projects, the ability of these techniques to be rapidly deployed is extremely useful for researchers seeking to experiment with new languages at minimal cost.

References

Benajiba., Y. & Zitouni, I. (2010). Enhancing mention detection using projection via aligned corpora. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing at Cambridge, MA* (pp. 993–1001). Stroudsburg, PA, USA: Association for Computational Linguistics.

Bhatt, R., Narasimhan, B., Palmer, M., Rambow, O., Sharma, D. M., & Xia, F. (2009). A multi-representational and multi-layered treebank for hindi/urdu. In *The Third Linguistic Annotation Workshop (The LAW III) in conjunction with ACL/IJCNLP 2009*. Association for Computational Linguistics.

Brown, P. F., Cock, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., et al. (1990). A statistical approach to machine translation. *Computational Linguistics*, *16*(2), 79–85.

Calzolari, N., Del Gratta, R., Francopoulo, G., Mariani, J., Rubino, F., Russo, I. & Soria, C. (2012). The LRE Map. Harmonising Community Descriptions of Resources. In *LREC (International Conference on Language Resources and Evaluation*), Istanbul.

Collins, M. (1999). *Head-driven statistical models for natural language parsing*. PhD thesis, University of Pennsylvania.

de Marneffe, M. C., MacCartney, B., & Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

Dorr, B. J. (1994). Machine translation divergences: A formal description and proposed solution. *Computational Linguistics*, *20*, 597–633.

Georgi, R., Xia, F., & Lewis, W. D. (2012). Improving dependency parsing with interlinear glossed text and syntactic projection. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, Mumbai, India.

Georgi, R., Xia, F., & Lewis, W. D. (2013). Enhanced and portable dependency projection algorithms using interlinear glossed text. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics* (vol 2, Short Papers, pp. 306–311), Sofia, Bulgaria, August 2013. Association for Computational Linguistics. http://www.aclweb.org/anthology/P13-2055.

Hwa, R., Resnik, P., Weinberg, A., Cabezas, C., & Kolak, O. (2004). Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, *1*(1), 1–15.

Hwa, R., Resnik, P., Weinberg, A., & Kolak, O. (2002). Evaluating translational correspondence using annotation projection. In *Proceedings of ACL 2002*, July (2002).

Lewis, W. D. (2006). ODIN: A model for adapting and enriching legacy infrastructure. In *Proceedings of the E-Humanities Workshop*, p. 137.

Lewis, W. D. & Xia, F. (2008). Automatically identifying computationally relevant typological features. In *Proceedings of IJCNLP*.

Lewis, W. D., & Xia, F. (2010). Developing ODIN: A multilingual repository of annotated language data for hundreds of the world's languages. *Journal of Literary and Linguistic Computing (LLC)*, *25*(3), 303–319.

McDonald, R., Lerman, K., & Pereira, F. (2006). Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pp 216–220. Association for Computational Linguistics.

Petrov, S., Das, D. & McDonald, R. (2012). A universal part-of-speech tagset. In *Proceedings of LREC*.

Volk, M., Göhring, A., Marek, T., & Samuelsson, Y. (2010). SMULTRON (version 3.0)—The Stockholm MULtilingual parallel TReebank. http://www.cl.uzh.ch/research/paralleltreebanks/smultron_en.html. An English-French-German-Spanish-Swedish parallel treebank with sub-sentential alignments.

Yarowsky, D., & Ngai, G. (2001). Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of NAACL, Stroudsburg, PA*. Johns Hopkins University.