Enhanced and Portable Dependency Projection Algorithms Using Interlinear Glossed Text

Ryan Georgi
University of Washington
Seattle, WA 98195, USA
rgeorgi@uw.edu

Fei Xia
University of Washington
Seattle, WA 98195, USA
fxia@uw.edu

William D. Lewis
Microsoft Research
Redmond, WA 98052, USA
wilewis@microsoft.com

Abstract

As most of the world's languages are under-resourced, projection algorithms offer an enticing way to bootstrap the resources available for one resource-poor language from a resource-rich language by means of parallel text and word alignment. These algorithms, however, make the strong assumption that the language pairs share common structures and that the parse trees will resemble one another. This assumption is useful but often leads to errors in projection. In this paper, we will address this weakness by using trees created from instances of Interlinear Glossed Text (IGT) to discover patterns of divergence between the languages. We will show that this method improves the performance of projection algorithms significantly in some languages by accounting for divergence between languages using only the partial supervision of a few corrected trees.

1 Introduction

While thousands of languages are spoken in the world, most of them are considered *resource-poor* in the sense that they do not have a large number of electronic resources that can be used to build NLP systems. For instance, some languages may lack treebanks, thus making it difficult to build a high-quality statistical parser.

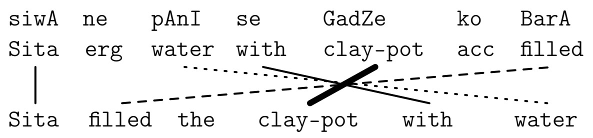
One common approach to address this problem is to take advantage of bitext between a resource-rich language (e.g., English) and a resource-poor language by projecting information from the former to the latter (Yarowsky and Ngai, 2001; Hwa et al., 2004). While projection methods can provide a great deal of information at minimal cost to the researchers, they do suffer from structural divergence between the language-poor language (aka target language) and the resource-rich language (aka source language).

In this paper, we propose a middle ground between manually creating a large-scale treebank (which is expensive and time-consuming) and relying on the syntactic structures produced by a projection algorithm alone (which are error-prone).

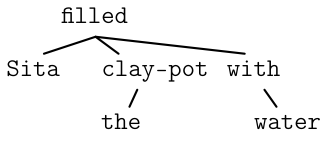
Our approach has several steps. First, we utilize instances of Interlinear Glossed Text (IGT) following Xia and Lewis (2007) as seen in Figure 1(a) to create a small set of parallel dependency trees through projection and then manually correct the dependency trees. Second, we automatically analyze this small set of parallel trees to find patterns where the corrected data differs from the projection. Third, those patterns are incorporated to the projection algorithm to improve the quality of projection. Finally, the features extracted from the projected trees are added to a statistical parser to improve parsing quality. The outcome of this work are both an enhanced projection algorithm and a better parser for resource-poor languages that require a minimal amount of manual effort.

2 Previous Work

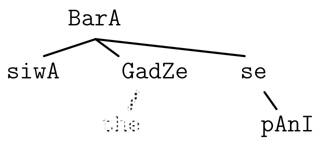
For this paper, we will be building upon the standard projection algorithm for dependency structures as outlined in Quirk et al. (2005) and illustrated in Figure 1. First, a sentence pair between resource-rich (source) and resource-poor (target) languages is word aligned [Fig 1(a)]. Second, the source sentence is parsed by a dependency parser for the source language [Fig 1(b)]. Third, sponta-


siwA ne pAnI se GadZe ko BarA
Sita erg water with clay-pot acc filled
Sita filled the clay-pot with water

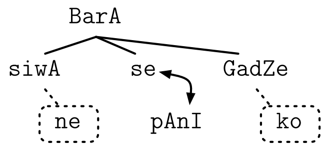
(a) An Interlinear Glossed Text (IGT) instance in Hindi and word alignment between the gloss line and the English translation.



(b) Dependency parse of English translation.



(c) English words are replaced with Hindi words and spontaneous word "the" are removed from the tree.



(d) Siblings in the tree are reordered based on the word order of the Hindi sentence and spontaneous Hindi words are attached as indicated by dotted lines. The words *pAnI* and *se* are incorrectly inverted, as indicated by the curved arrow.

Figure 1: An example of projecting a dependency tree from English to Hindi.

neous (unaligned) source words are removed, and the remaining words are replaced with corresponding words in the target side [Fig 1(c)]. Finally, spontaneous target words are re-attached heuristically and the children of a head are ordered based on the word order in the target sentence [Fig 1(d)]. The resulting tree may have errors (e.g., *pAni* should depend on *se* in Figure 1(d)), and the goal of this study is to reduce common types of projection errors.

In Georgi et al. (2012a), we proposed a method for analyzing parallel dependency corpora in which word alignment between trees was used to determine three types of edge configurations: **merged**, **swapped**, and **spontaneous**. Merged alignments were those in which multiple words in the target tree aligned to a single word in the source tree, as in Figure 2. Swapped alignments were those in which a parent node in the source tree aligned to a child in the target tree and vice-versa. Finally, spontaneous alignments were those for which a word did not align to any word on the other side. These edge configurations could be detected from simple parent–child edges and the alignment (or lack of) between words in the language pairs. Using these simple, language-agnostic measures allows one to look for divergence types such as those described by Dorr (1994).

Georgi et al. (2012b) described a method in which new features were extracted from the projected trees and added to the feature vectors for a statistical dependency parser. The rationale was that, although the projected trees were error-prone, the parsing model should be able to set appropriate weights of these features based on how reliable these features were in indicating the dependency structure. We started with the MSTParser (McDonald et al., 2005) and modified it so that the edges from the projected trees could be used as features at parse time. Experiments showed that adding new features improved parsing performance.

In this paper, we use the small training corpus built in Georgi et al. (2012b) to improve the projection algorithm itself. The improved projected trees are in turn fed to the statistical parser to further improve parsing results.

3 Enhancements to the projection algorithm

We propose to enhance the projection algorithm by addressing the three alignment types discussed earlier:

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

The detail of the enhancements are explained below.

3.1 Merge Correction

"Merged" words, or multiple words on the target side that align to a single source word, are problematic for the projection algorithm because it is not clear which target word should be the head and which word should be the

```
rAma buxXimAna    lagawA  hE
Ram  intelligent  seem    be-Pres
"Ram seems intelligent"
```

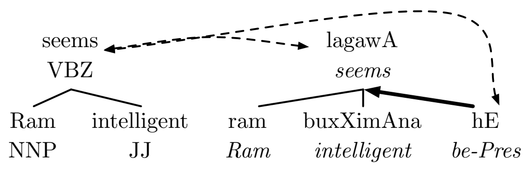


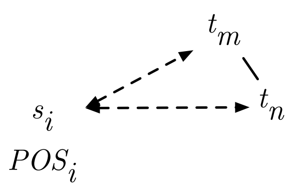
Figure 2: An example of merged alignment, where the English word *seems* align to two Hindi words *hE* and *lagawA*. Below the IGT are the dependency trees for English and Hindi. Dotted arrows indicate word alignment, and the solid arrow indicates that *hE* should depend on *lagawA*.

dependent. An example is given in Figure 2, where the English word *seems* align to two Hindi words *hE* and *lagawA*.

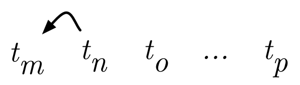
On the other hand, from the small amount of labeled training data (i.e., a set of hand-corrected tree pairs), we can learn what kind of source words are likely to align to multiple target words, and which target word is likely to the head. The process is illustrated in Figure 3. In this example, the target words t_m and t_n are both aligned with the source word s_i whose POS tag is POS_i, and t_m appears before t_n in the target sentence. Going through the examples of merged alignments in the training data, we keep a count for the POS tag of the source word and the position of the head on the target side.[1] Based on these counts, our system will generate rules such as the ones in Figure 3(c) which says if a source word whose POS is POS_i aligns to two target words, the probability of the right target word depending on the left one is 75%, and the probability of the left target word depending on the right one is 25%. We use maximum likelihood estimate (MLE) to calculate the probability.

The projection algorithm will use those rules to handle merged alignment; that is, when a source word aligns to multiple target words, the algorithm determines the direction of dependency edge based on the direction preference stored in the rules. In addition to rules for

[1]We use the position of the head, not the POS tag of the head, because the POS tags of the target words are not available when running the projection algorithm on the test data.



(a) Alignment between a source word and two target words, and one target word t_m is the parent of the other word t_n.



(b) Target sentence showing the "left" dependency between t_m and t_n.

POS_i	$\rightarrow$	left	0.75
POS_i	$\rightarrow$	right	0.25

(c) Rules for handling merged alignment

Figure 3: Example of merged alignment and rules derived from such an example

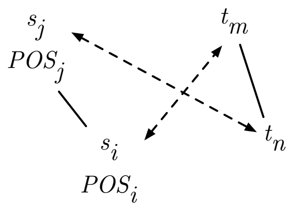
an individual source POS tag, our method also keeps track of the overall direction preference for all the merged examples in that language. For merges in which the source POS tag is unseen or there are no rules for that tag, this language-wide preference is used as a backoff.

3.2 Swap Correction

An example of swapped alignment is in Figure 4(a), where (s_j, s_i) is an edge in the source tree, (t_m, t_n) is an edge in the target tree, and s_j aligns to t_n and s_i aligns to t_m. Figure 1(d) shows an error made by the projection algorithm due to swapped alignment. In order to correct such errors, we count the number of $(\text{POS}_{child}, \text{POS}_{parent})$ dependency edges in the source trees, and the number of times that the directions of the edges are reversed on the target side. Figure 4(b) shows a possible set of counts resulting from this approach. Based on the counts, we keep only the POS pairs that appear in at least 10% of training sentences and the percentage of swap for the pairs are no less than 70%.[2] We say that those pairs trigger a swap operation.

At the test time, swap rules are applied as a post-processing step to the projected tree. After the projected tree is completed, our swap handling step checks each edge in the source tree. If the POS tag pair for the edge triggers

[2]These thresholds are set empirically.



(a) A swapped alignment between source words s_j and s_i and target words t_m and t_n.

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

(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 source side, and `total` is the number of times said pair occurs overall.

Figure 4: Example swap configuration and collected statistics.

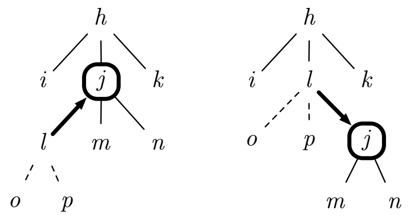


Figure 5: Swap operation: on the left is the original tree; on the right is the tree after swapping node l with its parent j.

a swap operation, the corresponding nodes in the projected tree will be swapped, as illustrated in Figure 5.

3.3 Spontaneous Reattachment

Target spontaneous words are difficult to handle because they do not align to any source word and thus there is nothing to project to them. To address this problem, we collect two types of information from the training data. First, we keep track of all the lexical items that appear in the training trees, and the relative position of their head. This lexical approach may be useful in handling closed-class words which account for a large percentage of spontaneous words. Second, we use the training trees to determine the favored attachment direction for the language as a whole.

At the test time, for each spontaneous word in the target sentence, if it is one of the words for which we have gathered statistics from the training data, we attach it to the next word in the preferred direction for that word. If the word is unseen, we attach it using the overall language preference as a backoff.

3.4 Parser Enhancements

In addition to above enhancements to the projection algorithm itself, we train a dependency parser on the training data, with new features from the projected trees following Georgi et al. (2012b). Furthermore, we add features that indicate whether the current word appears in a merge or swap configuration. The results of this combination of additional features and improved projection is shown in Table 1(b).

4 Results

For evaluation, we use the same data sets as in Georgi et al. (2012b), where there is a small number (ranging from 46 to 147) of tree pairs for each of the eight languages. The IGT instances for those tree pairs come from the Hindi Treebank (Bhatt et al., 2009) and the Online Database of Interlinear Text (ODIN) (Lewis and Xia, 2010).

We ran 10-fold cross validation and reported the average of 10 runs in Table 1. The top table shows the accuracy of the projection algorithm, and the bottom table shows parsing accuracy of MSTParser with or without adding features from the projected trees. In both tables, the *Best* row uses the enhanced projection algorithm. The *Baseline* rows use the original projection algorithm in Quirk et al. (2005) where the word in the parentheses indicates the direction of merge. The *Error Reduction* row shows the error reduction of the *Best* system over the best performing baseline for each language. The *No Projection* row in the second table shows parsing results when no features from the projected trees are added to the parser, and the last row in that table shows the error reduction of the *Best* row over the *No Projection* row.

Table 1 shows that using features from the projected trees provides a big boost to the quality of the statistical parser. Furthermore, the enhancements laid out in Section 3 improve the performance of both the projection algorithm and the parser that uses features from projected trees. The degree of improvement may depend on the properties of a particular language pair and the labeled data we

(a) The accuracies of the original projection algorithm (the *Baselin* rows) and the enhanced algorithm (the *Best* row) on eight language pairs. For each language, the best performing baseline is in italic. The last row shows the error reduction of the Best row over the best performing baseline, which is calculated by the formula $ErrorRate = \frac{Best - BestBaseline}{100 - BestBaseline} \times 100$

	YAQ	WLS	HIN	KKN	GLI	HUA	GER	MEX
Best	88.03	94.90	77.44	91.75	87.70	90.11	88.71	93.05
Baseline (Right)	*87.28*	*89.80*	57.48	*90.34*	*86.90*	79.31	*88.03*	*89.57*
Baseline (Left)	84.29	*89.80*	*68.11*	88.93	76.98	*79.54*	*88.03*	*89.57*
Error Reduction	5.90	50.00	29.26	14.60	6.11	51.66	5.68	33.37

(b) The parsing accuracies of the MSTParser with or without new features extracted from projected trees. There are two error reduction rows: one is with respect to the best performing baseline for each language, the other is with respect to *No Projection* where the parser does not use features from projected trees.

	YAQ	WLS	HIN	KKN	GLI	HUA	GER	MEX
Best	89.28	94.90	81.35	92.96	81.35	88.74	92.93	93.05
Baseline (Right)	*88.28*	*94.22*	78.03	*92.35*	*80.95*	87.59	90.48	*92.43*
Baseline (Left)	87.88	94.22	*79.64*	90.95	*80.95*	*89.20*	*90.48*	*92.43*
No Projection	66.08	91.32	65.16	80.75	55.16	72.22	62.72	73.03
Error Reduction (BestBaseline)	8.53	11.76	8.40	7.97	2.10	-4.26	25.74	8.19
Error Reduction (No Projection)	68.39	41.24	46.47	63.43	58.41	59.47	81.04	74.23

Table 1: System performance on eight languages: Yaqui (YAQ), Welsh (WLS), Hindi (HIN), Korean (KKN), Gaelic (GLI), Hausa (HUA), German (GER), and Malagasy (MEX).

have for that language pair. For instance, swap is quite common for the Hindi-English pair because postpositions depend on nouns in Hindi whereas nouns depend on prepositions in English. As a result, the enhancement for the swapped alignment alone results in a large error reduction, as in Table 2. This table shows the projection accuracy on the Hindi data when each of the three enhancements is turned on or off. The rows are sorted by descending overall accuracy, and the row that corresponds to the system labeled "Best" in Table 1 is in bold.

5 Conclusion

Existing projection algorithms suffer from the effects of structural divergence between language pairs. We propose to learn common divergence types from a small number of tree pairs and use the learned rules to improve projection accuracy. Our experiments show notable gains for both projection and parsing when tested on eight language pairs. As IGT data is available for hundreds of languages through the ODIN database and other sources, one could produce a small parallel treebank for a language pair after spending a few hours manually correcting the output of a projection algorithm. From the treebank, a better projection algorithm and a better parser can be built automatically using our approach.

Spont	Swap	Merge Direction	Accuracy
✓	✓	Left	78.07
✓	✓	**Informed**	**77.44**
	✓	Left	76.69
	✓	Informed	76.06
✓		Left	69.49
✓		Informed	68.96
		Left	68.11
		Informed	67.58
✓	✓	Right	66.32
	✓	Right	64.97
✓		Right	58.84
		Right	57.48

Table 2: Projection accuracy on the Hindi data, with the three enhancements turning on or off. The "spont" and "swap" columns show a checkmark when the enhancements are turned on. The merge direction indicates whether a left or right choice was made as a baseline, or whether the choice was *informed* by the rules learned from the training data.

While the improvements for some languages are incremental, the scope of coverage for this method is potentially enormous, enabling the rapid creation of tools for under-resourced languages of all kinds at a minimal cost.

Acknowledgment

This work is supported by the National Science Foundation Grant BCS-0748919. We would also like to thank the reviewers for helpful comments.

References

Rajesh Bhatt, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Misra Sharma, and Fei Xia. A multi-representational and multi-layered treebank for Hindi/Urdu. In *ACL-IJCNLP '09: Proceedings of the Third Linguistic Annotation Workshop*. Association for Computational Linguistics, August 2009.

Bonnie Jean Dorr. Machine translation divergences: a formal description and proposed solution. *Computational Linguistics*, 20:597–633, December 1994.

R Georgi, F Xia, and W D Lewis. Measuring the Divergence of Dependency Structures Cross-Linguistically to Improve Syntactic Projection Algorithms. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2012)*, Istanbul, Turkey, May 2012a.

Ryan Georgi, Fei Xia, and William D Lewis. Improving Dependency Parsing with Interlinear Glossed Text and Syntactic Projection. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, Mumbai, India, December 2012b.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 1(1):1–15, 2004.

William D Lewis and Fei Xia. Developing ODIN: A Multilingual Repository of Annotated Language Data for Hundreds of the World's Languages. 2010.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. Non-projective dependency parsing using spanning tree algorithms. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530, 2005.

Chris Quirk, Arul Menezes, and Colin Cherry. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*. Microsoft Research, 2005.

Fei Xia and William D Lewis. Multilingual Structural Projection across Interlinear Text. In *Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2007.

David Yarowsky and Grace Ngai. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Second meeting of the North American Association for Computational Linguistics (NAACL)*, Stroudsburg, PA, 2001. Johns Hopkins University.