Transforming Trees to Improve Syntactic Convergence

David Burkett and **Dan Klein**
Computer Science Division
University of California, Berkeley
{dburkett,klein}@cs.berkeley.edu

Abstract

We describe a transformation-based learning method for learning a sequence of monolingual tree transformations that improve the agreement between constituent trees and word alignments in bilingual corpora. Using the manually annotated English Chinese Translation Treebank, we show how our method automatically discovers transformations that accommodate differences in English and Chinese syntax. Furthermore, when transformations are learned on automatically generated trees and alignments from the same domain as the training data for a syntactic MT system, the transformed trees achieve a 0.9 BLEU improvement over baseline trees.

1 Introduction

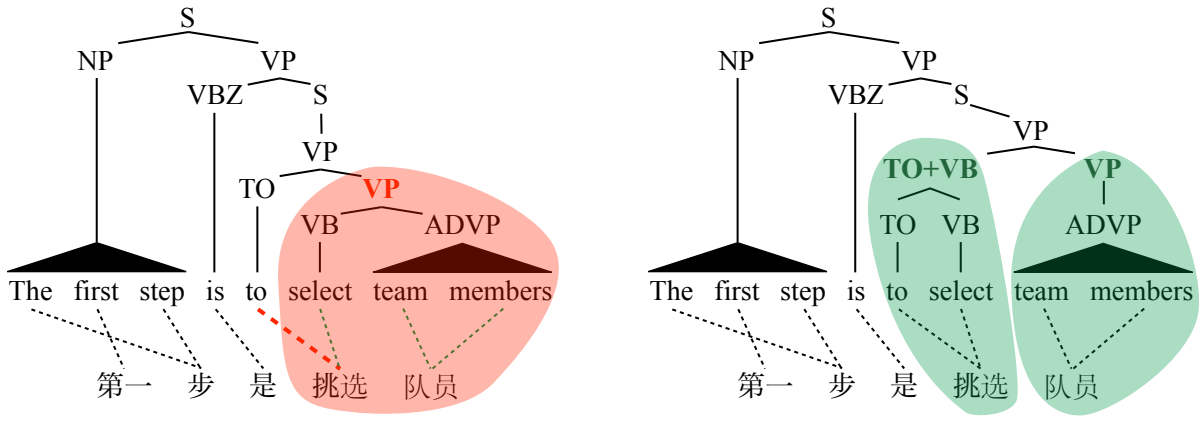
Monolingually, many Treebank conventions are more or less equally good. For example, the English WSJ treebank (Marcus et al., 1993) attaches verbs to objects rather than to subjects, and it attaches prepositional modifiers outside of all quantifiers and determiners. The former matches most linguistic theories while the latter does not, but to a monolingual parser, these conventions are equally learnable. However, once bilingual data is involved, such treebank conventions entail constraints on rule extraction that may not be borne out by semantic alignments. To the extent that there are simply divergences in the syntactic structure of the two languages, it will often be impossible to construct syntax trees that are simultaneously in full agreement with monolingual linguistic theories and with the alignments between sentences in both languages.

To see this, consider the English tree in Figure 1a, taken from the English side of the English Chinese Translation Treebank (Bies et al., 2007). The lowest VP in this tree is headed by 'select,' which aligns to the Chinese verb '挑选.' However, '挑选' also aligns to the other half of the English infinitive, 'to,' which, following common English linguistic theory, is outside the VP. Because of this violating alignment, many syntactic machine translation systems (Galley et al., 2004; Huang et al., 2006) won't extract any translation rules for this constituent. However, by applying a simple transformation to the English tree to set up the infinitive as its own constituent, we get the tree in Figure 1b, which may be less well-motivated linguistically, but which corresponds better to the Chinese-mediated semantics and permits the extraction of many more syntactic MT rules.

In this work, we develop a method based on *transformation-based learning* (Brill, 1995) for automatically acquiring a sequence of tree transformations of the sort in Figure 1. Once the transformation sequence has been learned, it can be deterministically applied to any parsed sentences, yielding new parse trees with constituency structures that agree better with the bilingual alignments yet remain consistent across the corpus. In particular, we use this method to learn a transformation sequence for the English trees in a set of English to Chinese MT training data. In experiments with a string-to-tree translation system, we show resulting improvements of up to 0.9 BLEU.

A great deal of research in syntactic machine translation has been devoted to handling the inherent syntactic divergence between source and target languages. Some systems attempt to model the differences directly (Yamada and Knight, 2001; Eisner, 2003), but most recent work focuses on reducing the sensitivity of the rule-extraction procedure to the constituency decisions made by 1-best syntactic parsers, either by using forest-based methods

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 863–872, Jeju Island, Korea, 12–14 July 2012.



(a) Before

(b) After

Figure 1: An example tree transformation merging a VB node with the TO sibling of its parent VP. Before the transformation (a), the bolded VP cannot be extracted as a translation rule, but afterwards (b), both this VP and the newly created TO+VB node are extractable.

for learning translation rules (Mi and Huang, 2008; Zhang et al., 2009), or by learning rules that encode syntactic information but do not strictly adhere to constituency boundaries (Zollmann et al., 2006; Marton and Resnik, 2008; Chiang, 2010). The most closely related MT system is that of Zhao et al. (2011), who train a rule extraction system to transform the subtrees that make up individual translation rules using a manually constructed set of transformations similar to those learned by our system.

Instead of modifying the MT system to work around the input annotations, our system modifies the input itself in order to improve downstream translation. Most systems of this sort learn how to modify word alignments to agree better with the syntactic parse trees (DeNero and Klein, 2007; Fossum et al., 2008), but there has also been other work directly related to improving agreement by modifying the trees. Burkett et al. (2010) train a bilingual parsing model that uses bilingual agreement features to improve parsing accuracy. More closely related to the present work, Katz-Brown et al. (2011) retrain a parser to directly optimize a word reordering metric in order to improve a downstream machine translation system that uses dependency parses in a preprocessing reordering step. Our system is in the same basic spirit, using a proxy evaluation metric (agreement with alignments; see Section 2 for details) to improve performance on a downstream translation task. However, we are concerned more generally with the goal of creating trees that are more compatible with a wide range of syntactically-informed translation systems, particularly those that extract translation rules based on syntactic constituents.

2 Agreement

Our primary goal in adapting parse trees is to improve their agreement with a set of external word alignments. Thus, our first step is to define an *agreement score* metric to operationalize this concept.

Central to the definition of our agreement score is the notion of an *extractable node*. Intuitively, an extractable English[1] tree node (also often called a "frontier node" in the literature), is one whose span aligns to a contiguous span in the foreign sentence.

Formally, we assume a fixed word alignment $a = \{(i, j)\}$, where $(i, j) \in a$ means that English word i is aligned to foreign word j. For an English span $[k, \ell]$ (inclusive), the set of aligned foreign words is:

$$\mathit{fset}([k, \ell]) = \{j \mid \exists\, i : k \leq i \leq \ell; (i, j) \in a\}$$

We then define the aligned foreign span as:

$$\mathit{fspan}([k, \ell]) = [\min(\mathit{fset}([k, \ell])), \max(\mathit{fset}([k, \ell]))]$$

[1]For expositional clarity, we will refer to "English" and "foreign" sentences/trees, but our definitions are in no way language dependent and apply equally well to any language pair.

The aligned English span for a given foreign span $[s,t]$ is defined analogously:

$$\textit{eset}([s,t]) = \{i \mid \exists\, j : s \leq j \leq t; (i,j) \in a\}$$
$$\textit{espan}([s,t]) = [\min(\textit{eset}([s,t])), \max(\textit{eset}([s,t]))]$$

Finally, we define $[k,\ell]$ to be extractable if and only if it has at least one word alignment and its aligned foreign span aligns back to a subspan of $[k,\ell]$:

$$\textit{fset}([k,\ell]) \neq \emptyset \wedge \textit{espan}(\textit{fspan}([k,\ell])) \subseteq [k,\ell]$$

With this definition of an extractable span, we can now define the agreement score $g_a(t)$ for an English tree t, conditioned on an alignment a:[2]

$$g_a(t) = \sum_{\substack{[k,\ell] \in t: \\ |[k,\ell]|>1}} sign([k,\ell]) \qquad (1)$$

Where

$$sign([k,\ell]) = \begin{cases} 1 & [k,\ell] \text{ is extractable} \\ -1 & \text{otherwise} \end{cases}$$

Importantly, the sum in Equation 1 ranges over all *unique* spans in t. This is simply to make the metric less gameable, preventing degenerate solutions such as an arbitrarily long chain of unary productions over an extractable span. Also, since all individual words are generated by preterminal part-of-speech nodes, the sum skips over all length 1 spans.

As a concrete example of agreement score, we can return to Figure 1. The tree in Figure 1a has 6 unique spans, but only 5 are extractable, so the total agreement score is 5 - 1 = 4. After the transformation, though, the tree in Figure 1b has 6 extractable spans, so the agreement score is 6.

3 Transformation-Based Learning

Transformation-based learning (TBL) was originally introduced via the Brill part-of-speech tagger (Brill, 1992) and has since been applied to a wide variety of NLP tasks, including binary phrase structure bracketing (Brill, 1993), PP-attachment disambiguation (Brill and Resnik, 1994), base NP chunking (Ramshaw and Marcus, 1995), dialogue act tagging (Samuel et al., 1998), and named entity recognition (Black and Vasilakopoulos, 2002).

[2]Unextractable spans are penalized in order to ensure that space is saved for the formation of extractable ones.

The generic procedure is simple, and requires only four basic inputs: a set of training sentences, an initial state annotator, an inventory of atomic transformations, and an evaluation metric. First, you apply the initial state annotator (here, the source of original trees) to your training sentences to ensure that they all begin with a legal annotation. Then, you test each transformation in your inventory to see which one will yield the greatest improvement in the evaluation metric if applied to the training data. You greedily apply this transformation to the full training set and then repeat the procedure, applying transformations until some stopping criterion is met (usually either a maximum number of transformations, or a threshold on the marginal improvement in the evaluation metric).

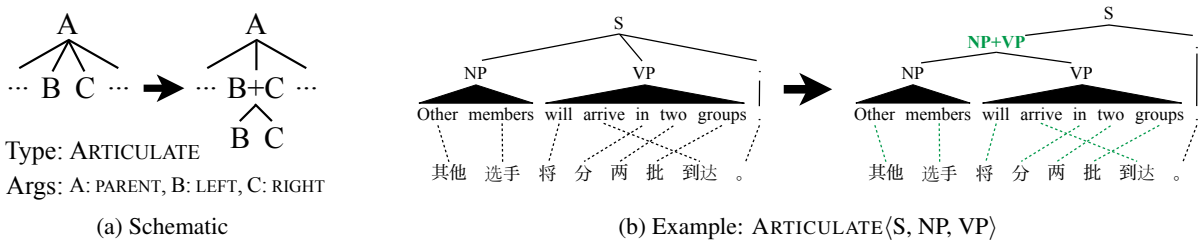
The output of the training procedure is an ordered set of transformations. To annotate new data, you simply label it with the same initial state annotator and then apply each of the learned transformations in order. This process has the advantage of being quite fast (usually linear in the number of transformations and the length of the sentence; for parsing, the cost will typically be dominated by the cost of the initial state annotator), and, unlike the learned parameters of a statistical model, the set of learned transformations itself can often be of intrinsic linguistic interest.

For our task, we have already defined the evaluation metric (Section 2) and the initial state annotator will either be the gold Treebank trees or a Treebank-trained PCFG parser. Thus, to fully describe our system, it only remains to define the set of possible tree transformations.

4 Tree Transformations

The definition of an atomic transformation consists of two parts: a rewrite rule and the triggering environment (Brill, 1995). Tree transformations are best illustrated visually, and so for each of our transformation types, both parts of the definition are represented schematically in Figures 2-7. We have also included a real-world example of each type of transformation, taken from the English Chinese Translation Treebank.

Altogether, we define six types of tree transformations. Each class of transformation takes be-

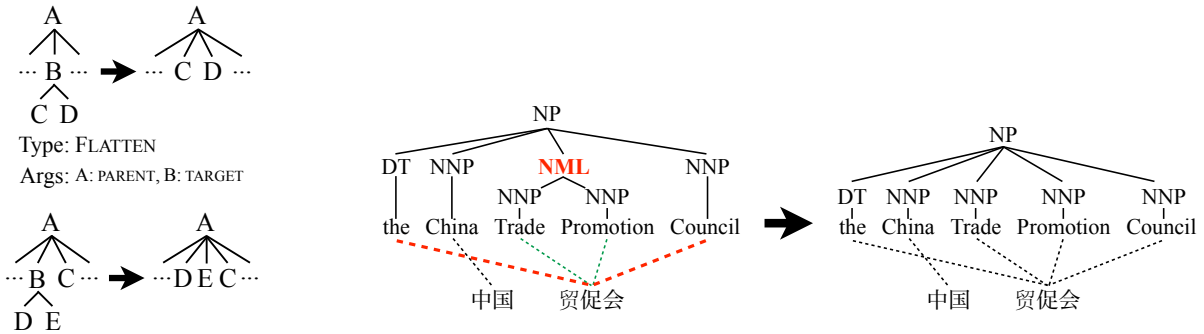


Type: ARTICULATE
Args: A: PARENT, B: LEFT, C: RIGHT

(a) Schematic

(b) Example: ARTICULATE⟨S, NP, VP⟩

Figure 2: ARTICULATE transformations.



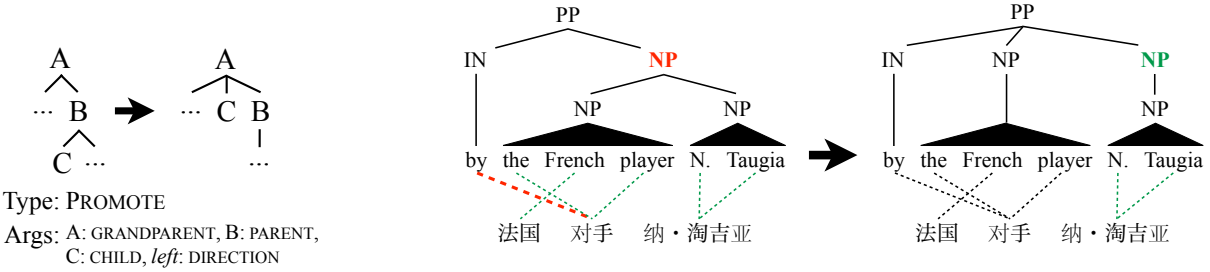
Type: FLATTEN
Args: A: PARENT, B: TARGET

Type: FLATTENINCONTEXT
Args: A: PARENT, B: TARGET, C: SIBLING, *left*: DIRECTION

(a) Schematic

(b) Example: FLATTENINCONTEXT⟨NP, NML, NNP, *left*⟩

Figure 3: FLATTEN transformations.

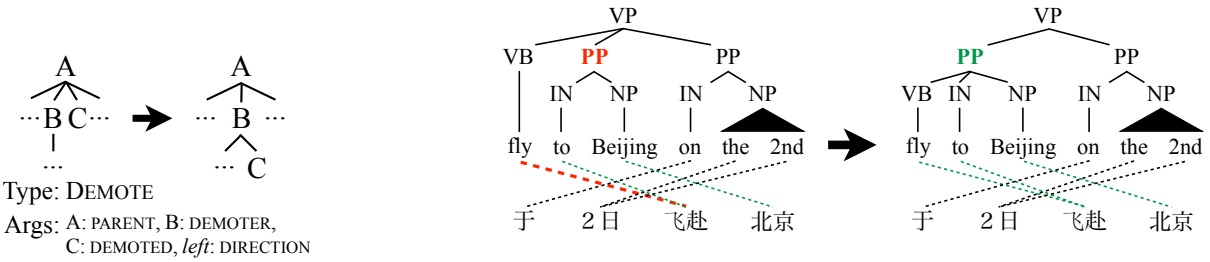


Type: PROMOTE
Args: A: GRANDPARENT, B: PARENT, C: CHILD, *left*: DIRECTION

(a) Schematic

(b) Example: PROMOTE⟨PP, NP, NP, *left*⟩

Figure 4: PROMOTE transformations.

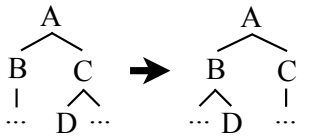


Type: DEMOTE
Args: A: PARENT, B: DEMOTER, C: DEMOTED, *left*: DIRECTION

(a) Schematic

(b) Example: DEMOTE⟨VP, PP, VB, *right*⟩

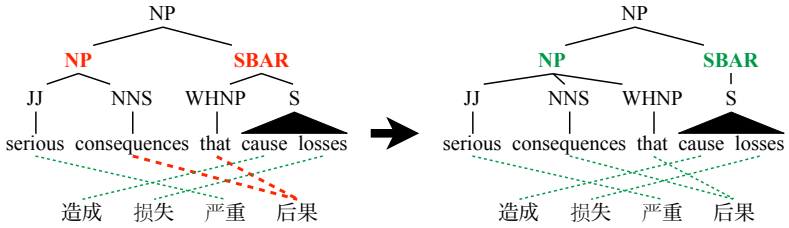
Figure 5: DEMOTE transformations.



Type: TRANSFER

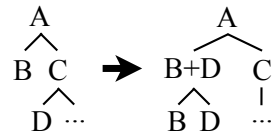
Args: A: GRANDPARENT, B: AUNT, C: PARENT, D: TARGET, *left*: DIRECTION

(a) Schematic



(b) Example: TRANSFER⟨NP, NP, SBAR, WHNP, *left*⟩

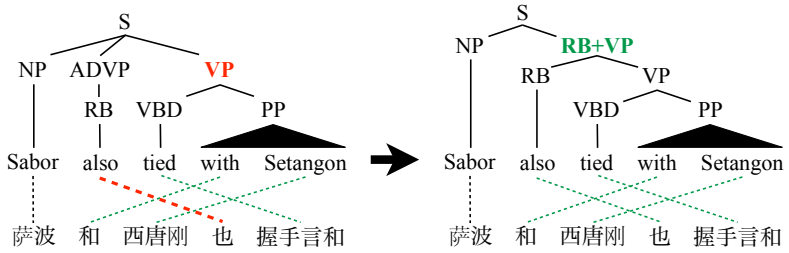
Figure 6: TRANSFER transformations.



Type: ADOPT

Args: A: GRANDPARENT, B: AUNT, C: PARENT, D: TARGET, *left*: DIRECTION

(a) Schematic



(b) Example: ADOPT⟨S, VP, ADVP, RB, *right*⟩

Figure 7: ADOPT transformations.

tween two and four syntactic category arguments, and most also take a DIRECTION argument that can have the value *left* or *right*.[3] We refer to the nodes in the schematics whose categories are arguments of the transformation definition as *participating* nodes. Basically, a particular transformation is triggered anywhere in a parse tree where all participating nodes appear in the configuration shown. The exact rules for the triggering environment are:

1. Each participating node must appear in the schematically illustrated relationship to the others. The non-participating nodes in the schematic do not have to appear. Similarly, any number of additional nodes can appear as siblings, parents, or children of the explicitly illustrated nodes.

2. Any node that will gain a new child as a result of the transformation must already have at least one nonterminal child. We have drawn the schematics to reflect this, so this condition is equivalent to saying that any participating node that is drawn with children must have a phrasal syntactic category (i.e. it cannot be a POS).

3. Repeated mergings are not allowed. That is, the newly created nodes that result from an ARTICULATE or ADOPT transformation cannot then participate as the LEFT or RIGHT argument of a subsequent ARTICULATE transformation or as the AUNT or TARGET argument of a subsequent ADOPT transformation. This is simply to prevent the unrestrained proliferation of new syntactic categories.

The rewrite rule for a transformation is essentially captured in the corresponding schematic. Additional nodes that do not appear in the schematic are generally handled in the obvious way: unillustrated children or parents of illustrated nodes remain in place, while unillustrated siblings of illustrated nodes are handled identically to their illustrated siblings. The only additional part of the rewrite that is not shown explicitly in the schematics is that if the node in the PARENT position of a TRANSFER or ADOPT transformation is left childless by the transformation (be-

[3]To save space, the schematic for each of these transformations is only shown for the *left* direction, but the *right* version is simply the mirror image.

cause the TARGET node was its only child), then it is deleted from the parse tree. In the case of a transformation whose triggering environment appears multiple times in a single tree, transformations are always applied leftmost/bottom-up and exhaustively.[4]

In principle, our transformation inventory consists of all possible assignments of syntactic categories to the arguments of each of the transformation types (subject to the triggering environment constraints). In practice, though, we only ever consider transformations whose triggering environments appear in the training corpus (including new triggering environments that appear as the result of earlier transformations). While the theoretical space of possible transformations is exponentially large, the set of transformations we actually have to consider is quite manageable, and empirically grows substantially sublinearly in the size of the training set.

5 Results and Analysis

There are two ways to use this procedure. One is to apply it to the entire data set, with no separate training phase. Given that the optimization has no notion of gold transformations, this procedure is roughly like an unsupervised learner that clusters its entire data. Another way is to learn annotations on a subset of data and apply it to new data. We choose the latter primarily for reasons of efficiency and simplicity: many common use cases are easiest to manage when annotation systems can be trained once offline and then applied to new data as it comes in.

Since we intend for our system to be used as a pre-trained annotator, it is important to ensure that the learned transformation sequence achieves agreement score gains that generalize well to unseen data. To minimize errors that might be introduced by the noise in automatically generated parses and word alignments, and to maximize reproducibility, we conducted our initial experiments on the English Chinese Translation Treebank. For this dataset, the initial state annotations (parse trees) were manually created by trained human annotators, as were the word alignments used to compute the agreement score.[5] The data was divided into training/dev/test using the standard Chinese parsing split; we trained the system on the training set (2261 sentences after filtering out sentences with missing annotations), and evaluated on the development set (223 sentences after filtering).

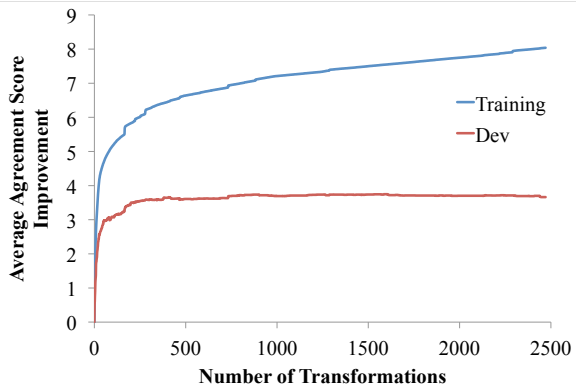


Figure 8: Transformation results on the English Chinese Translation Treebank. The value plotted is the average (per-sentence) improvement in agreement score over the baseline trees.

Transformations	Total Spans	Extractable Spans	Agreement Score
0	13.15	9.78	6.40
10	12.57	10.36	8.15
50	13.41	11.38	9.35
200	14.03	11.96	9.89
1584	14.58	12.36	10.15
2471	14.65	12.35	10.06

Table 1: Average span counts and agreement scores on the English Chinese Translation Treebank development set. The highest agreement score was attained at 1584 transformations, but most of the improvement happened much earlier.

The improvements in agreement score are shown in Figure 8, with a slightly more detailed breakdown at a few fixed points in Table 1. While the system was able to find up to 2471 transformations that improved the training set agreement score, the majority of the improvement, and especially the majority of the improvement that generalized to the test set,

[4]The transformation is repeatedly applied at the lowest, leftmost location of the parse tree where the triggering environment appears, until the triggering environment no longer appears anywhere in the tree.

[5]The annotation guidelines for the English side of this Treebank are similar, though not identical, to those for the WSJ Treebank.

1	ARTICULATE⟨S,NP,VP⟩
2	FLATTENINCONTEXT⟨PP,NP,IN,*right*⟩
3	PROMOTE⟨VP,VP,VBN,*left*⟩
4	ADOPT⟨VP,TO,VP,VB,*left*⟩
5	ADOPT⟨PP,VBG,PP,IN,*left*⟩
6	FLATTEN⟨VP,VP⟩
7	ARTICULATE⟨VP,VBD,NP⟩
8	FLATTENINCONTEXT⟨PP,NML,NNP,*left*⟩
9	ARTICULATE⟨NP,NNP,NNS⟩
10	ARTICULATE⟨S,NP,ADVP⟩
11	TRANSFER⟨NP,NP,SBAR,WHNP,*left*⟩
12	FLATTENINCONTEXT⟨NP,NML,NNP,*left*⟩
13	ARTICULATE⟨NP,NN,NNS⟩
14	TRANSFER⟨NP,NP+,,SBAR,WHNP,*left*⟩
15	ADOPT⟨PP,IN,PP,IN,*left*⟩
16	PROMOTE⟨S,VP,CC+VP,*right*⟩
17	ARTICULATE⟨VP,VBZ,VBN⟩
18	ARTICULATE⟨VP,VBD,PP⟩
19	ARTICULATE⟨VP,MD,ADVP⟩
20	ADOPT⟨PP,SYM,QP,CD,*right*⟩

Table 2: The first 20 learned transformations, excluding those that only merged punctuation or conjunctions with adjacent phrases. The first 5 are illustrated in Figure 9.

was achieved within the first 200 or so transformations. We also see from Table 1 that, though the first few transformations deleted many non-extractable spans, the overall trend was to produce more finely articulated trees, with the full transformation sequence increasing the number of spans by more than 10%.

As discussed in Section 3, one advantage of TBL is that the learned transformations can themselves often be interesting. For this task, some of the highest scoring transformations did uninteresting things like conjoining conjunctions or punctuation, which are often either unaligned or aligned monotonically with adjacent phrases. However, by filtering out all ARTICULATE transformations where either the LEFT or RIGHT argument is "CC", "-RRB-", ",", or "." and taking the top 20 remaining transformations, we get the list in Table 2, the first 5 of which are also illustrated in Figure 9. Some of these (e.g. #1, #7, #10) are additional ways of creating new spans when English and Chinese phrase structures roughly agree, but many others do recover known differences in English and Chinese syntax. For example, many of these transformations directly address compound verb forms in English, which tend to align to single words in Chinese: #3 (past participle constructions), #4 (infinitive), #6 (all), and #17 (present perfect). We also see differences between English and Chinese internal NP structure (e.g. #9, #12, #13).

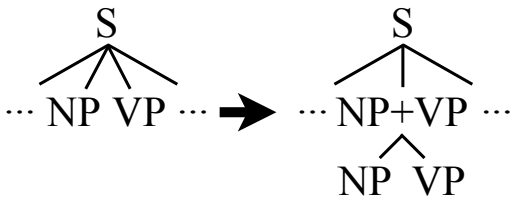
6 Machine Translation

The ultimate goal of our system is to improve the agreement between the automatically generated parse trees and word alignments that are used as training data for syntactic machine translation systems. Given the amount of variability between the outputs of different parsers and word aligners (or even the same systems with different settings), the best way to improve agreement is to learn a transformation sequence that is specifically tuned for the same annotators (parsers and word aligners) we are evaluating with. In particular, we found that though training on the English Chinese Translation Treebank produces clean, interpretable rules, preliminary experiments showed little to no improvement from using these rules for MT, primarily because actual alignments are not only noisier but also systematically different from gold ones. Thus, all rules used for MT experiments were learned from automatically annotated text.

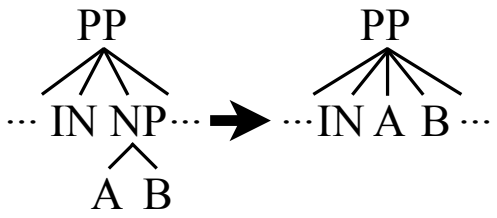
For our Chinese to English translation experiments, we generated word alignments using the Berkeley Aligner (Liang et al., 2006) with default settings. We used an MT pipeline that conditions on target-side syntax, so our initial state annotator was the Berkeley Parser (Petrov and Klein, 2007), trained on a modified English treebank that has been adapted to match standard MT tokenization and capitalization schemes.

As mentioned in Section 5, we could, in principle train on all 500k sentences of our MT training data. However, this would be quite slow: each iteration of the training procedure requires iterating through all n training sentences[6] once for each of the m candidate transformations, for a total cost of $O(nm)$ where m grows (albeit sublinearly) with n. Since the

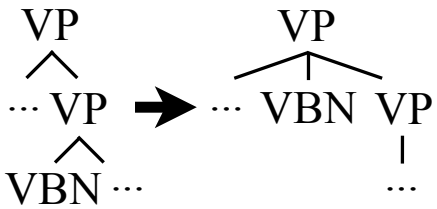
[6]By using a simple hashing scheme to keep track of triggering environments, this cost can be reduced greatly but is still linear in the number of training sentences.



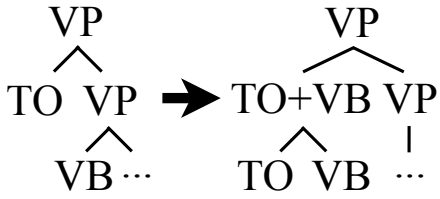
(a) ARTICULATE⟨S,NP,VP⟩



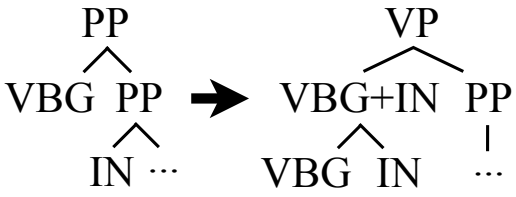
(b) FLATTENINCONTEXT⟨PP,NP,IN,*right*⟩



(c) PROMOTE⟨VP,VP,VBN,*left*⟩



(d) ADOPT⟨VP,TO,VP,VB,*left*⟩



(e) ADOPT⟨PP,VBG,PP,IN,*left*⟩

Figure 9: Illustrations of the top 5 transformations from Table 2.

most useful transformations almost by definition are ones that are triggered the most frequently, any reasonably sized training set is likely to contain them, and so it is not actually likely that dramatically increasing the size of the training set will yield particularly large gains.

Thus, to train our TBL system, we extracted a random subset of 3000 sentences to serve as a training set.[7] We also extracted an additional 1000 sentence test set to use for rapidly evaluating agreement score generalization. Figure 10 illustrates the improvements in agreement score for the automatically annotated data, analogous to Figure 8. The same general patterns hold, although we do see that the automatically annotated data is more idiosyncratic and so more than twice as many transformations are learned before training set agreement stops improving, even though the training set sizes are roughly the same.[8] Furthermore, test set generalization in the automatic annotation setting is a little bit worse, with later transformations tending to actually hurt test set agreement.

For our machine translation experiments, we used the string-to-tree syntactic pipeline included in the current version of Moses (Koehn et al., 2007). Our training bitext was approximately 21.8 million words, and the sentences and word alignments were the same for all experiments; the only difference between each experiment was the English trees, for which we tested a range of transformation sequence prefixes (including a 0-length prefix, which just yields the original trees, as a baseline). Since the transformed trees tended to be more finely articulated, and increasing the number of unique spans often helps with rule extraction (Wang et al., 2007), we equalized the span count by also testing binarized versions of each set of trees, using the left-branching and right-branching binarization scripts included with Moses.[9]

We tuned on 1000 sentence pairs and tested on

[7] The sentences were shorter on average than those in the English Chinese Translation Treebank, so this training set contains roughly the same number of words as that used in the experiments from Section 5.

[8] Note that the training set improvement curves don't actually flatten out because training halts once no improving transformation exists.

[9] Binarized trees are guaranteed to have $k-1$ unique spans for sentences of length k.

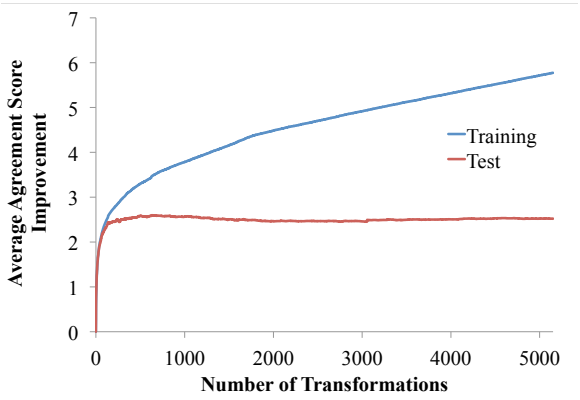


Figure 10: Transformation results on a subset of the MT training data. The training and test sets are disjoint in order to measure how well the learned transformation sequence generalizes. Once again, we plot the average improvement over the baseline trees. Though 5151 transformations were learned from the training set, the maximum test set agreement was achieved at 630 transformations, with an average improvement of 2.60.

642 sentence pairs from the NIST MT04 and MT05 data sets, using the BLEU metric (Papineni et al., 2001). As discussed by Clark et al. (2011), the optimizer included with Moses (MERT, Och, 2003) is not always particularly stable, and results (even on the tuning set) can vary dramatically across tuning runs. To mitigate this effect, we first used the Moses training scripts to extract a table of translation rules for each set of English trees. Then, for each rule table, we ran MERT 11 times and selected the parameters that achieved the maximum tuning BLEU to use for decoding the test set.

Table 3 shows the results of our translation experiments. The best translation results are achieved by using the first 139 transformations, giving a BLEU improvement of more than 0.9 over the strongest baseline.

7 Conclusion

We have demonstrated a simple but effective procedure for learning a tree transformation sequence that improves agreement between parse trees and word alignments. This method yields clear improvements in the quality of Chinese to English translation, showing that by manipulating English syntax to converge with Chinese phrasal structure, we improve our ability to explicitly model the types of structural relationships between languages that syntactic MT systems are designed to exploit, even if we lose some fidelity to the original monolingual annotation standards in the process.

Transformations	Agrmnt Score	BLEU		
		None	Left	Right
0	5.36	31.66	31.81	31.84
32	7.17	32.41	32.17	32.06
58	7.42	32.18	32.68*	32.37
139	7.81	32.20	32.60*	**32.77***
630	7.96	32.48	32.06	32.22
5151	7.89	32.13	31.84	32.12

Table 3: Machine translation results. Agreement scores are taken from the test data used to generate Figure 10. Note that using 0 transformations just yields the original baseline trees. The transformation sequence cutoffs at 32, 58, and 139 were chosen to correspond to marginal training (total) agreement gain thresholds of 50, 25, and 10, respectively. The cutoff at 630 was chosen to maximize test agreement score and the cutoff at 5151 maximized training agreement score. Column headings for BLEU scores ("None," "Left," "Right") refer to the type of binarization used after transformations. Entries marked with a '*' show a statistically significant difference ($p < 0.05$) from the strongest (right-binarized) baseline, according to the paired bootstrap (Efron and Tibshirani, 1994).

Acknowledgements

This project is funded by an NSF graduate research fellowship to the first author and by BBN under DARPA contract HR0011-12-C-0014.

References

Ann Bies, Martha Palmer, Justin Mott, and Colin Warner. 2007. English Chinese translation treebank v 1.0. Web download. LDC2007T02.

William J. Black and Argyrios Vasilakopoulos. 2002. Language independent named entity classification by modified transformation-based learning and by decision tree induction. In *COLING*.

Eric Brill and Philip Resnik. 1994. A transformation-based approach to prepositional phrase attachment disambiguation. In *COLING*.

Eric Brill. 1992. A simple rule-based part of speech tagger. In *Proceedings of the workshop on Speech and Natural Language*.

Eric Brill. 1993. Automatic grammar induction and parsing free text: A transformation-based approach. In *ACL*.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

David Burkett, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *NAACL:HLT*.

David Chiang. 2010. Learning to translate with source and target syntax. In *ACL*.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: controlling for optimizer instability. In *ACL:HLT*.

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *ACL*.

Bradley Efron and R. J. Tibshirani. 1994. *An Introduction to the Bootstrap (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*. Chapman and Hall/CRC.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *ACL*.

Victoria Fossum, Kevin Knight, and Steven Abney. 2008. Using syntax to improve word alignment for syntax-based statistical machine translation. In *ACL MT Workshop*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *HLT-NAACL*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *HLT-NAACL*.

Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno, and Hideto Kazawa. 2011. Training a parser for machine translation reordering. In *EMNLP*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *HLT-NAACL*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrase-based translation. In *ACL:HLT*.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *EMNLP*.

Franz Josef Och. 2003. Miminal error rate training in statistical machine translation. In *ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. Research report, IBM. RC22176.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *ACL Workshop on Very Large Corpora*.

Ken Samuel, Sandra Carberry, and K. Vijay-Shanker. 1998. Dialogue act tagging with transformation-based learning. In *COLING*.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *EMNLP*.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *ACL*.

Hui Zhang, Min Zhang, Haizhou Li, Aiti Aw, and Chew Lim Tan. 2009. Forest-based tree sequence to string translation model. In *ACL-IJCNLP*.

Bing Zhao, Young-Suk Lee, Xiaoqiang Luo, and Liu Li. 2011. Learning to transform and select elementary trees for improved syntax-based machine translations. In *ACL:HLT*.

Andreas Zollmann, Ashish Venugopal, Stephan Vogel, and Alex Waibel. 2006. The CMU-AKA syntax augmented machine translation system for IWSLT-06. In *IWSLT*.