Automatically Learning Source-side Reordering Rules for Large Scale Machine Translation

Dmitriy Genzel
Google, Inc.
dmitriy@google.com

Abstract

We describe an approach to automatically learn reordering rules to be applied as a preprocessing step in phrase-based machine translation. We learn rules for 8 different language pairs, showing BLEU improvements for all of them, and demonstrate that many important order transformations (SVO to SOV or VSO, head-modifier, verb movement) can be captured by this approach.

1 Introduction

One of the major problems of modern statistical machine translation relates to its difficulties in producing the correct word order on the target side of the translation where the source side order is not the same as the target side. In many cases where the translation is spectacularly bad, if one only enters the source sentence in the word order of the target language the translation becomes near-perfect (largely because the language model can now make sense of it). The word order problems are especially extensive for languages that have major differences, such as SOV vs. SVO languages, but also cause insidious, but entirely avoidable errors for the language pairs where the word order is almost right, but not quite[1]. For practical reasons all phrase-based decoders limit the amount of reordering allowed and thus are completely unable to produce correct translations when the necessary movement is over a large distance. Furthermore, where the actual systematic reordering for the two languages is within the decoder's search space, it is penalized just as any other kind of reordering, whereas doing anything other than this systematic reordering should in fact be penalized.

It has been argued that this is a fundamental flaw in phrase-based decoding systems and hierarchical and syntax-based systems have been proposed to solve this problem. These systems can in principle resolve a part of this problem, but at a significant time cost during training, and even worse, during translation, making it less practical for realtime systems. Instead we propose a system for learning pre-ordering rules automatically from data and demonstrate that it can capture many different kinds of reordering phenomena and do so at no additional online cost.

2 Related Work

Many solutions to the reordering problem have been proposed, e.g. syntax-based models (Chiang, 2005), lexicalized reordering (Och et al., 2004), and tree-to-string methods (Zhang et al., 2006). All these methods try to solve the reordering problem in different ways, but have the following problems in common: word alignment is not affected by them and they tend to introduce significant additional work to be done at translation time. Most state of the art systems use HMM or IBM Model 4 word alignment, both of which have a penalty term associated with long distance jumps, and tend to misalign words which move far from their expected positions.

We are going to focus on the approaches where reordering is done as a preprocessing step (sometimes called pre-ordering). These approaches have the advantage that they are independent of the actual MT system used, are often fast to apply, and tend to decrease (due to improved quality of heuristic estimates) rather than dramatically increase the time spent in actual decoding, unlike

[1] For example of the latter kind, verb movement for English-German and similar language pairs often causes verbs to be aligned to nothing and to be altogether dropped in translation.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 376–384,
Beijing, August 2010

some of the previously mentioned approaches. The downside of these methods is that the reordering is fixed, and if it is wrong it can hurt the quality of translations. We will discuss solutions for this problem later.

Even in the relatively limited space of preprocessing-based reordering solutions, there has been a large amount of previous work, as far back as Brown et al. (1992). Most approaches focus on utilizing manually written rules for different languages. A common language pair for which rules were proposed is German-English (Nießen and Ney, 2001; Collins et al., 2005). There is similar work for Chinese-English (Wang et al., 2007) and quite a few other languages. Clearly, such methods work quite well, but require linguistic expertise to produce. Our goal, however, is to learn reordering from parallel data that is already available to an MT system in an entirely unsupervised manner.

We are not the first to attempt this task. In particular, Xia and McCord (2004) proposed a way to automatically learn reordering patterns for French-English. Their system parses parallel data both on the source and target side and then uses a variety of heuristics to extract reordering rules which are then applied during training. More recently, Li et al. (2007) use a maximum entropy system to learn reordering rules for binary trees (i.e., whether to keep or reorder for each node). An approach most similar to ours is that of Rottmann and Vogel (2007) where they learn reordering rules based on sequences of part-of-speech tags (but do not use parse trees). All of these approaches show improvements in translation quality, but are applied on a single language pair. Our goal is to find a method that works well for many language pairs, regardless of the word order transformations needed, and without language-specific tuning. Unlike our predecessors, we use a systematic search through the space of possible permutation rules to minimize a specific metric, related to the monotonicity of resulting alignments.

3 Our Approach

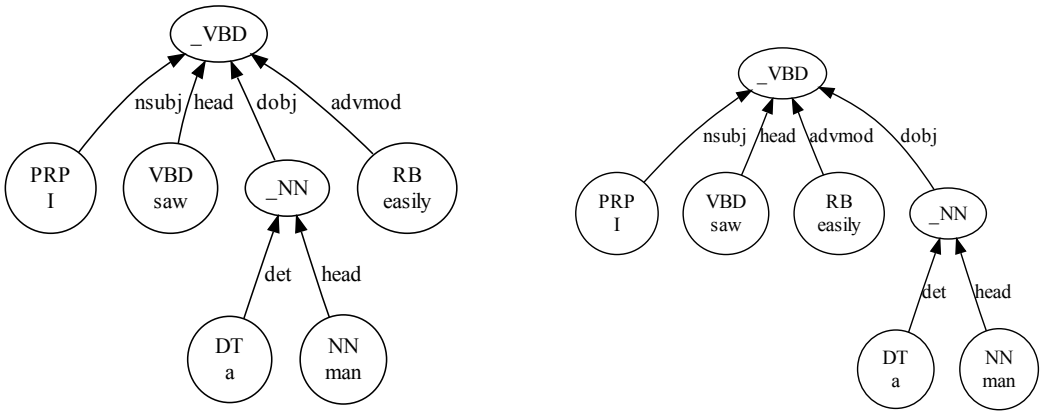
We limit ourselves to reorderings of the source side of training and test data. To constrain our reorderings, we first produce a parse tree, using a dependency parser similar to that of Nivre and Scholz (2004). The above parser is much faster than the time spent in translating the same sentence and thus creates almost no overhead. In our experiments where the source language is English the training data for the parser is the Penn Treebank (Marcus et al., 1993). For German, we use TIGER treebank (Brants et al., 2002). We then convert the dependency tree to a shallow constituent tree. The trees are annotated by both Penn Treebank part of speech tags and by Stanford dependency types (de Marneffe et al., 2006; de Marneffe and Manning, 2008). For an example, see Figure 1a.

Our reorderings are constrained by reordering of nodes in a parse tree of the source sentence. Thus, the full space of reorderings we consider consists of all reorderings that would produce a parse tree with the same set of child-parent relationships. For an example of a valid reordering, see Figure 1b.

Each reordering is described by a series of rules and we learn one such series for each language pair automatically. Each source sentence is parsed, and the tree is transformed sequentially, one rule at a time applying to the entire tree, top down. The reordered sentence is read off the leaves of the tree and training and evaluation proceeds as normal. We are using a state-of-the-art phrase-based statistical machine translation system to perform the actual translation. The system is itself capable of further local reordering during translation limited by the maximum distance of 4 words.

3.1 Rule Space

Each rule consists of two parts: conditioning context and action. For every internal node in the parse tree, traversed top-down, the node is matched against the conditioning context, and if a match is found, the associated action applies. All actions are limited to reordering children of the matching node. Furthermore, if a rule applies at a node, its descendants are not traversed for the purpose of matching to avoid modifying the same part of the sentence twice by the same rule. A different rule may apply on this node or its descendants



(a) A sample parse tree

(b) After reordering (moving RB over _NN)

Figure 1: Parse tree of a sentence and its reordering

Feature	Description
nT	POS tag of this node
nL	Syntactic label of this node
pT	POS tag of the parent of this node
pL	Syntactic label of the parent
1T	POS tag of the first child
1L	Label of the first child
2T	POS tag of the second child
2L	Label of the second child
...	...

Table 1: Set of features used as conditioning variables

later in the sequence.

A conditioning context is a conjunction of conditions. Each condition is a (feature, value) pair. List of features is given in table 1. In practice, we limit ourselves to no more than 4 conditions in a given context to avoid combinatorial explosion and sparsity as well as contexts that fail to generalize. However, we may exhaustively generate every possible conjunction of up to 5 conditions from this list that covers up to 4 children that we actually observe in training.

For example, the following contexts would be valid for transformation in Fig. 1:

- nT = _VBD
- 1T = PRP
- 1L = nsubj
- 3T = dobj
- etc.

or any conjunction of these. The action performed in this example is swapping children 3 and 4 of the _VBD node, and can be denoted as the permutation (1,2,4,3).

When processing a rule sequence, once a rule applies, the action is performed, and that rule is no longer applied on the same node or its descendants (but can be further applied elsewhere in the tree). Another rule (even an identical one) starts from the top and can apply to nodes modified by previous rules.

3.2 Reordering metrics

To evaluate the quality of a given reordering rule, we need to have reliable metrics that, for each sentence pair, can evaluate whether an improvement in monotonicity has been made.

The easiest metric to use is the number of crossing alignment links for a given aligned sentence pair. For instance, in Figure 2, there are 2 crossing links. This metric is trivial to compute and has some nice properties. For instance, moving a single word one position out of place causes one link

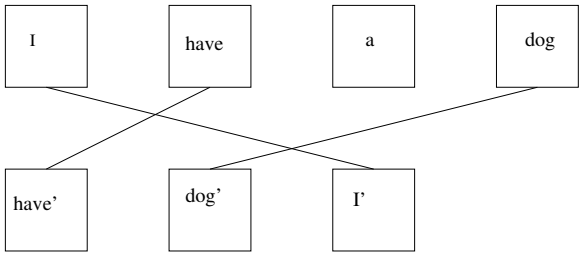


Figure 2: Counting crossing alignment links

to cross, moving it farther away from its correct position would cause more links to cross. We will refer to this metric as *crossing score*.

An ideal metric would be the actual BLEU score that the system would obtain under this reordering rule on the development set. However, since each rule affects word alignment, phrase extraction, optimal feature weights, and the actual translation, it would be necessary to retrain the entire phrase-based system for each possible rule, which is impractical. It is, however, practical, to retranslate the development set, keeping the phrase table and feature weights constant. Normally, however, phrase tables contain multi-word phrases, such as "a b" which may no longer match after the reordering, and this biases the system toward the original word order. To avoid this, for this computation only, we use a phrase table that only contains single words and is therefore independent of the source sentence word order. This lets us test whether a given reordering improves the search space for the phrase-based decoder at the relatively small computational cost of translating the development set. We obtain a difference of the BLEU scores with and without a given rule, which we hope to be a reasonable estimate of the true gain in BLEU score that one would obtain, by retraining the full system, including word alignment, full-length phrase extraction, and tuning the feature weights. We refer to this score as *estimated BLEU gain*.

Note that these two scores are used to obtain an estimate of utility of any given rule, and are not used for evaluation of the entire system. Those metrics are discussed in detail in the evaluation section.

3.3 Algorithm

We propose a straightforward algorithm to automatically learn reordering rules. The input data for all algorithms is word-aligned sentence pairs. We have found that sophisticated alignment models introduce a bias toward alignment between certain kinds of nodes (usually ones that are close), and this has undesirable effects. In practical terms this means that neither HMM nor Model 4 alignments are useful (even though they are better as alignments), but Model 1 alignments are. However, to compensate for poor quality of the alignments, we simply delete those alignment links that have posterior probabilities under 0.5[2] and remove sentence pairs which have very few alignments left. The crossing score works quite well even when only a portion of the words in a sentence are aligned.

The algorithm's outline is given as Alg. 1.

The algorithm proceeds by considering all rules after the best sequence of rules so far, and appends the best new rule (according to the metric) to the sequence. In practice, some changes are needed, and we describe some variations. Each of these variations produces a different sequence of rules, but they are interchangeable, and we can simply pick one that performs best on the development set, or to combine them through multi-source translation or consensus.

In all variations, we are unable to generate all possible rules for every sentence, as the number can easily be 10^4-10^6 per sentence. It is sufficient, however, to take a random sample of the input, extract top candidates, and reevaluate those on the entire set.

We also limit the kinds of rules we are allowed to generate. The number of possible actions on a node with n children is $n! - 1$ and our trees are quite shallow, often containing 5, 6, or even more children per node. To avoid dealing with explosion of rules and the resulting sparsity of the rule space, we modify the process slightly, so that instead of matching a node, we match a node and a consecutive subsequence of its children of a given size, as a sliding window. For example, in Figure 1a, node _VBD has 4 children. If we limit our-

[2]This guarantees only one alignment per word

Algorithm 1 Optimizing alignment links

```
input: A set of aligned sentence pairs
base = <empty sequence>;
for several iterations do
  candidate_rules = GenerateAllCandidateRules(input, base);
  base.append(MinCost(candidate_rules))
end for
```

selves to 3 children at a time we would attempt to match this node twice: with its children 1,2,3 and 2,3,4. In other words, we pretend to consider two nodes, one with the first set of children, and one with the second, proceeding left to right. If either one matches, we apply the action to the subset of children in the window and stop processing the node further.

It is also useful to produce more than one rule per iteration, although this can be problematic, since the rules may interfere with each other.

3.3.1 Variant 1: Optimizing crossing score

We start with the initially empty base sequence. As described above, we generate every possible rule from a subset of sentences, and evaluate them on the entire input, with the base sequence always applied first. We use crossing score as a metric. However, instead of extracting only one best-scoring rule, we extract K best. Now we need to obtain a decorrelated set: for every pair of rules, we count the number of sentences where they both apply. For every rule we consider all rules that are ranked higher, and if the percentage of matches between these two rules is high, the rules may interfere with each other, and the current rule is dropped. We thus obtain a small ordered set of rules that tend to apply on different sentences, and should not interfere with each other. From this ordered set we produce all candidate rule subsequences and evaluate them, to ensure there really is no interference. The one with the best score is then appended to the base sequence. The process is then repeated with a new base sequence.

3.3.2 Variant 2: Optimizing Estimated BLEU gain

We proceed as in the previous variant, but final evaluation of potential sequences to be appended is done differently. Instead of using a crossing score, we reorder the development set with each candidate rule sequence and score it using a translation system with a fixed phrase table with single word phrases only (to avoid bias for a specific word order). The sequence with the highest BLEU is then appended to base sequence, and the process is repeated.

3.3.3 Variant 3: Optimizing Estimated BLEU gain in sequence

In this variant, once we obtain a set of decorrelated candidate rules $\{a_1, a_2, \ldots a_n\}$ ordered by crossing score, we evaluate the following rule sequences (where b is base sequence): $(b), (b, a_1), (b, a_1, a_2) \ldots (b, a_1, \ldots a_n)$ using estimated BLEU gain, as above. If we find that for some k, $score(b, a_1, \ldots a_{k-1}) > score(b, a_1, \ldots a_{k-1}, a_k)$, that means that a_k interferes with preceding rules. We remove all such a_k, and retranslate/rescore until the score sequence is monotonically non-decreasing. At this point, we append all surviving rules to the base sequence, and repeat the process.

4 Evaluation

As described above, our base system is a phrase-based statistical MT system, similar to that of Och and Ney (2004). The baseline decoder is capable of local reordering of up to 4 words. Our training data is extracted by mining from the Web, as well as from other published sources. We train systems from English to 7 other languages, as well as German-English. We chose them as follows: SOV languages (Japanese, Korean, Hindi), VSO language (Welsh), long distance verb movement (German), noun-modifier issues (Russian and Czech). The amount of training data varies from 28 million words (for Hindi) to 260 million (for German). The baseline sys-

tem is a production-quality system used by a large number of users.

For the first set of experiments for German-English and English-German we use WMT-09 data sets for development and testing (Callison-Burch et al., 2009). We report BLEU scores for each of the algorithms along with the best score from the WMT-09 workshop for reference in Table 2.

Unfortunately, there is no standard data set for most of the languages we would like to experiment with. For the second set of experiments, we use an unpublished data set, containing data in English and 7 languages mentioned above. Our test data comes from two sources: news articles from WikiNews[3] (996 sentences) and a set of random sentences from the web (9000 sentences). From these, we create 3 sets: *dev1*: 3000 sentences from *web* and 486 sentences from *wiki*; *dev2*: 1000 sentences from *web*; and *test*: the remainder of *web* (5000 sentences) and *wiki* (510 sentences). The *dev1* set is used for tuning the system, both *dev1* and *dev2* for tuning consensus, and the *test* set for evaluation. These sets are the same for all 7 languages.

Discriminative minimum error rate training (Macherey et al., 2008) was applied to optimize the feature weights for each system.

We evaluate the three variants of the algorithm mentioned above. Each algorithm outputs a reordering rule sequence (40-50 rules long) which is applied to all the training and test data, and a complete system is trained from scratch.

There is no need for us to pick a single algorithm for all language pairs, since each algorithm produces rules that are compatible with each other. We are able to pick the algorithm that works best on the development set for each language pair.

In addition, we can use a decoder that is capable of performing a multi-input translation which is given the unreordered input as well as the three reordered inputs produced by the above algorithm. This decoder is able to learn separate feature weights for each feature/algorithm combination.

Finally, we can use consensus translation (Macherey and Och, 2007) to produce the best possible translation for each sentence.

[3]http://en.wikinews.org

Table 4: Manual vs. automatic reordering. *Automatic* score is the combined score from Table 3.

Language	Base	Manual	Automatic	Diff
Hindi	16.85	19.25	19.36	0.11
Japanese	25.91	28.78	29.12	0.34
Korean	23.61	27.99	27.91	-0.08

Results using BLEU score (character-level for Japanese and Korean, word-level for other languages) for English to X systems are given in Table 3, along with the score of Google Translate as of Feb 15, 2010, for expected quality reference. All gains in the combined and consensus columns are statistically significant using a bootstrap resampling test (Noreen, 1989).

We should also note that the parsing and reordering overhead was an average of 10msec per sentence, and had no appreciable impact on the speed of the system.

4.1 Comparison with manual reordering

We also compared our automatic method with a manually written reordering rule set for SOV languages (Xu et al., 2009) (rules initially written for Korean) for comparison with our approach. The results are given in Table 4. The results are mostly comparable, with automatic rules being better for two of the three languages.

4.2 Turning off decoder reordering

All of the above experiments allowed the decoder to further reorder the sentence as needed. Reordering in the decoder creates an exponential increase in the search space, and for a typical decoding strategy can lead to increase in decoding time, search errors, or both. Since we already preorder the sentence, it should be possible to avoid reordering in the decoder altogether.

Results for the combined decoder are given in Table 5. It contains the gain of the combined decoder against the baseline from Table 3, and the gain when decoder reordering is turned off against the same baseline (which has decoder reordering on). For many languages it is indeed now possi-

Table 2: Results for 3 algorithms on WMT-09 data with best individual system score from the workshop: for EN to DE, Edinburgh, for DE to EN, Google

Language	Base	Var. 1	Var. 2	Var. 3	Best workshop
EN to DE	16.09	16.30	16.35	**16.40**	*14.76*
DE to EN	21.00	**22.45**	22.13	22.05	*20.23*

Table 3: Results on internal test set for 3 systems (Variant 1,2,3), the variant which performed best on the development set, the combined system, and the consensus run, along with Google Translate scores (Feb 15, 2010) for reference

Language	Google	Base	Var. 1	Var. 2	Var. 3	Best on dev	Combined	Consensus
	%BLEU	%BLEU	gain	gain	gain	gain	gain	gain
Czech	16.68	15.35	-0.08	0.13	**0.19**	0.19	0.21	0.21
German	20.34	18.65	**0.47**	0.30	0.39	0.39	0.72	0.73
Hindi	19.15	16.85	**2.25**	2.08	0.15	2.08	2.51	2.47
Japanese	30.74	25.91	**3.05**	2.60	**3.05**	3.05	3.21	3.03
Korean	27.99	23.61	3.34	3.77	**4.16**	4.16	4.30	4.30
Russian	16.80	15.33	0.08	**0.10**	**0.10**	0.08	0.14	0.23
Welsh	27.38	25.48	1.25	0.77	**1.43**	1.43	1.34	1.63

Table 5: Disallowing decoder reordering: difference against baseline in %BLEU gain

Language	Decoder reordering	No decoder reordering
Czech	0.21	0.08
German	0.72	0.55
Hindi	2.51	2.27
Japanese	3.21	3.21
Korean	4.30	4.15
Russian	0.14	-0.10
Welsh	1.34	0.98

ble to avoid decoder reordering altogether which leads to a significant speedup.

5 Analysis

We looked at the rules being learned as well as at the differences in the output to see if the gains in BLEU are in fact due to the reordering phenomena being resolved. The top rules for each language are given in Table 6.

One can observe that the top rules for German and Slavic languages are as expected: verb movement and noun modifier reordering. Other top rules for German cover other specific cases of verb movement, other rules for Czech include, for example, movement of the subject of the passive sentence to the right and movement of the possessive (which is similar to the noun compound case).

The rules for Welsh include movement of the adjective modifier over its head (given in the table above) and other rules moving noun modifiers, moving a modal verb left over its subject, moving determiners to the right of nouns, etc.

For Japanese and Korean, there are many rules with dramatic impact, such as a rule moving all heads to the right, reversing a sequence of three nodes starting with a modal (e.g. *can do something* to *something do can*), moving numerical modifiers to the right of their heads, and many others.

Hindi is also an SOV language, but its grammar is not as similar to Japanese or Korean as they are to each other. Still, Hindi also has some similar rules, but there are many more involving verb movement, such as a rule directly moving the verb to the final position.

By looking at the sentences produced by the system we can see that the differences are dramatic for SOV and VSO languages, as expected,

Table 6: Examples of top rules and their application

Languages	Context	Order	Example
Hindi	1L:head 3L:none	2,1,3	*I see him → I him see*
Japanese, Korean	2L:prep	2,1	*eat with a spoon → eat a spoon with*
German	1T:VBN 2L:prep	2,1	*struck with a ball → with a ball struck*
Russian, Czech	1L:nn 2L:head	2,1	*a building entrance → a entrance building*
Welsh	1L:amod 2L:head	2,1	*blue ball → ball blue*

but more interestingly, most German sentences now have a verb where the baseline had none. Another profound effect can be observed for Russian: the baseline almost invariably translated noun compounds incorrectly: e.g. *group leaders* may be translated as *group of-leaders* since this requires no reordering and no preposition insertion. This is especially problematic, since the user of the translation system often cannot detect this: the resulting sentence is not ungrammatical and can even make sense. Our algorithm learns a rule that prevents this from happening. Now the decoder must pay a cost to keep the order the same as in English.

6 Discussion and Future Work

We have demonstrated a general technique which requires only access to a parser for the source language (in addition to parallel data which already exists for an MT system) and is capable of reducing reordering problems endemic in a phrase-based system. No linguists or even native speakers of any of these languages were needed to write the rules. The algorithm is quite robust and performs well on noisy web data, much of it being ungrammatical.

All variants turned out to perform well, although variants 1 and 3 were better most of the time. We consider all variants to be useful, since they find different local maxima under different objective functions, and in practice use all of them and pick a rule sequence that performs best on the development set for any specific language pair.

We plan to explore this research area further in several ways. First, it would be interesting to experiment with applying rules learned for one language to a related language, e.g. Portuguese for Spanish or German for Dutch. This would let us use rules learned from a major language for a minor one with less available training data.

We have only used English and German as source languages. There is training data for parsers in other languages, and this approach should work well for most source languages. Where a source language parser is not available, we can still improve quality, by learning rules from the target side and applying them only for the purpose of improving word alignment. Improving word alignment alone would not help as much as also using the reordering in the decoder, but it will probably help in extracting better phrases. We also plan to use parser projection to induce a reasonable quality parser for other languages.

References

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *In Proceedings of the Workshop on Treebanks and Linguistic Theories*, pages 24–41.

Peter F. Brown, Stephen A. Della, Pietra Vincent, J. Della Pietra, John D. Lafferty Robert, and L. Mercer. 1992. Analysis, statistical transfer, and synthesis in machine translation. In *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 83–100.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece, March. Association for Computational Linguistics.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the ACL'05*, pages 263–270, Ann Arbor, Michigan, June.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine

translation. In *Proceedings of the ACL'05*, pages 531–540, Ann Arbor, Michigan, June.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representations. In *COLING'08 Workshop on Cross-framework and Cross-domain Parser Evaluation*, Manchester, England, August.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure trees. In *LREC*.

Chi-Ho Li, Minghui Li, Dongdong Zhang, Mu Li, Ming Zhou, and Yi Guan. 2007. A probabilistic approach to syntax-based reordering for statistical machine translation. In *Proceedings of the ACL-07*, pages 720–727, Prague, Czech Republic, June.

Wolfgang Macherey and Franz J. Och. 2007. An empirical study on computing consensus translations from multiple machine translation systems. In *Proceedings of the EMNLP-CoNLL'07*, pages 986–995, Prague, Czech Republic, June.

Wolfgang Macherey, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the EMNLP-2008*, pages 725–734, Honolulu, Hawaii, October.

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

Sonja Nießen and Hermann Ney. 2001. Morpho-syntactic analysis for reordering in statistical machine translation. In *Machine Translation Summit*, pages 247–252, Santiago de Compostela, Spain, September.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of Coling 2004*, pages 64–70, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley & Sons, Canada.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *HLT-NAACL 2004: Main Proceedings*, pages 161–168, Boston, Massachusetts, USA, May 2 - May 7.

Kay Rottmann and Stephan Vogel. 2007. Word reordering in statistical machine translation with a pos-based distortion model. In *Proceedings of TMI*, Skovde, Sweden.

Chao Wang, Michael Collins, and Philipp Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proceedings of the EMNLP-CoNLL'2007*, pages 737–745, Prague, Czech Republic, June.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of Coling 2004*, pages 508–514, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve SMT for subject-object-verb languages. In *Proceedings of NAACL-HLT'09*, Boulder, Colorado.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 256–263, New York City, USA, June. Association for Computational Linguistics.