
Language Modeling with Tree Substitution Grammars

Matt Post and Daniel Gildea
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

We show that a tree substitution grammar (TSG) induced with a collapsed Gibbs sampler results in lower perplexity on test data than both a standard context-free grammar and other heuristically trained TSGs, suggesting that it is better suited to language modeling. Training a more complicated bilexical parsing model across TSG derivations shows further (though nuanced) improvement. We conduct analysis and point to future areas of research using TSGs as language models.

1 Introduction

Recently a number of groups have had success parsing with tree substitution grammars (TSGs) that were induced from the Penn Treebank with collapsed Gibbs samplers in a Bayesian framework (Cohn et al., 2009; Post and Gildea, 2009). Compared to past heuristic approaches, these grammars are compact and intuitive, have a more natural distribution over rule size, and perform well on parsing accuracy relative to the Treebank grammar. This sort of learning can be viewed as refining the structure of the Treebank grammar; while TSGs are weakly equivalent to CFGs and have the same distribution over strings, the rules of the grammar can be quite different. Extracting TSG subtrees from the Treebank allows the rules themselves to capture more complicated dependencies (as opposed to leaving this to the parsing model alone). For example, a lexicalized subtree can capture the predicate-argument structure of a verb.

In this paper, we show that, in addition to increasing parsing accuracy, these induced TSGs also have a significantly lower perplexity on test data under a standard context-free parsing model. With small modifications, this result also holds for a more complicated bilexical parsing model trained over a version of the Treebank that has been “flattened” with the induced TSG derivations. These observations are appealing from the perspective of language learning and representation, where we are more directly interested in how accurately and compactly a grammar encodes grammatical strings than in how well it can infer the derivation of a sentence whose grammaticality is assumed.

2 Experimental setup

Our training data was all sentences from sections 2 to 21 of the Wall Street Journal portion of the Penn Treebank, and for testing we used the 2,245 sentences with forty or fewer tokens in section 23. We took as our vocabulary the set of 23,767 case-sensitive tokens appearing more than once in the training data. All other tokens were converted to a set of eighty unknown word classes based on surface features (Petrov et al., 2006). Trace nodes and node annotations (e.g., temporal, subject, and locative markers) were removed from all of our data.

Underlying our experiments are three grammars:

grammar	size	F1	perplexity	rules used	
				tokens	types
CFG	46K	72.6	692.0	86,561	9,012
spinal+CFG	191K	79.2	544.0	61,750	14,090
DOP+CFG	2,567K	77.3	505.1	45,914	18,301
sampled+CFG	77K	80.1	523.9	65,311	12,367
sampled	63K	81.7	429.7	56,769	13,375
bigram	342K	-	238.4	-	-
bigram	430K	-	202.4	-	-

Table 1: Parsing scores and model perplexity. The size of a grammar is the number of subtrees in the grammar, whereas the size of an ngram model is the number of entries in its table, including the backoff tables.

1. *The Treebank grammar*. This is the grammar obtained by using all standard (height one) CFG rules from our training data.
2. A “*spinal*” *tree substitution grammar*. Similar to Chiang (2000), we heuristically define subtrees to be the sequence of adjacent rules in a parse tree that share a head, according to the Collins head-selection rules. This yields n distinct subtrees for each length n sentence, each having a single lexical item among the nodes of its frontier.
3. A *sampled tree substitution grammar*. We induced TSG derivations from the parse trees in the training data using a collapsed Gibbs sampler with a Dirichlet Process prior, as described in Post and Gildea (2009).¹, and formed the grammar from the subtrees from the derivations at the end of the 100th iteration.

Each grammar was built from derivations of the parse trees in the training corpus. We use the terms *rule* and *subtree* more or less interchangeably to denote the rewrites of nonterminals in CFGs and TSGs. In addition to parsing with these grammars alone, we also experiment with combining grammars, e.g., “spinal+CFG” means that we took the set of rules from two instances of the training corpus, one annotated with spinal grammar derivations and the other with standard Treebank derivations.

The probability of a sentence s when parsing with a grammar G is given as $\Pr_G(s) = \sum_{d \in D_s} \Pr(d)$, where D_s is the set of derivations whose yield is s and $\Pr(d)$ is defined by the parsing model. To compute perplexity, we follow standard practice and estimate the cross-entropy of each model G with respect to the true distribution p on the test corpus S as

$$H(p, G) \approx -\frac{1}{N} \sum_{s \in S} \log_2 \Pr_G(s)$$

(where N is the number of tokens in the sentences of S for which the parser produced an analysis) and report perplexity as $2^{H(p, G)}$. Under the assumption that a better model will assign higher probability to a test corpus of grammatical sentences than a worse model, lower perplexity is better.

The bigram and trigram language model baselines presented in Section 4 were trained and tested on the same data sets described above using SRILM version 1.5.2² with the default settings.

3 Context-free grammar parsing

We begin by presenting perplexity results for context-free grammar parsing. This standard model defines the probability of a derivation as the product of the probabilities of the fixed rules that constitute it, where each probability is conditioned only on the label of the nonterminal it is expanding. Subtree probabilities were assigned according to their relative frequency.

¹See Goldwater et al. (2009) for an excellent introduction to using this technique for segmentation tasks.

²<http://www.speech.sri.com/projects/srilm/>

grammar	failures	perplexity
CFG	122	1,325.0
spinal+CFG	124	1,902.9
sampled	133	2,118.1
sampled+CFG	123	1,803.4
DOP+CFG	123	3,174.3
bigram	0	1,274.9
trigram	0	1,277.1

Table 2: Model perplexity on a mildly ungrammatical corpus in which the children of all NPs were reversed. The “failures” column indicates the number of sentences for which the model could not find a parse.

Table 1 lists perplexity results for language modeling with these grammars along with F1 scores on the parsing task.³ In addition to the grammars mentioned above, we present results from a Data-Oriented Parsing (DOP) “all subtrees” grammar produced by randomly sampling 400,000 subtrees of heights two, three, and so on, up to a height of fourteen, as described in Bod (2001).

From this table we can see that, apart from the DOP model, parser accuracy and perplexity are correlated for these grammars under this simple parsing model. The lower perplexity results for the tree substitution grammars might appear obvious at first glance: TSGs use larger rules, so there will be fewer probabilities to multiply together in the derivation. The DOP grammar provides a useful counterexample; despite using many fewer rules, its perplexity is significantly higher than that of the sampled grammar. This can be explained by the fact that although the DOP grammar employed significantly fewer rule tokens, it is overfit to the training data, and its immense size means that the probability mass for each nonterminal rewrite is spread much more thinly across rule types. Compactness of a grammar is an important property for keeping perplexity low. The collapsed Gibbs sampling procedure yields a compact grammar that, relative to other CFGs, is both more accurate for parsing and better for language modeling.

Table 2 contains results of an additional experiment. A number of research groups have shown that PCFGs are not very helpful in improving BLEU scores for machine translation (Charniak et al., 2003; Och et al., 2004; Post and Gildea, 2008). Furthermore, they do not even appear to be very useful in distinguishing grammatical from ungrammatical text. Cherry and Quirk (2008) used model scores produced by a maximum-likelihood estimated parser with a Markovized, parent-annotated Treebank grammar to classify a corpus of 6,000 Wall Street Journal sentences and “pseudo-negative” ngram-sampled sentences (Okanohara and Tsujii, 2007). They reported a development set accuracy of only 65% when using sentence length and model score as the only features of an SVM-trained classifier. A good language model should have a large gap between model scores on good and bad text, so comparing a model’s scores on both kinds of text ensures that it is not simply happy with any kind of input.⁴ To produce ungrammatical text that did not result in too many parse failures with these grammars, we took the parse trees associated with the sentences in the test data and reversed the order of the children of all NP nodes. This produces mildly ungrammatical sentences whose meaning is still mostly clear, as can be seen in this example (with some constituents marked for clarity):

[banks investment Big]_{NP} refused to step up to [plate the]_{NP} to support [traders floor beleaguered the]_{NP} by buying [[of stock]_{PP} [blocks big]_{NP}]_{NP} , traders say .

We draw attention to a few interesting aspects of this table. The DOP model is again an outlier; its perplexity score is much higher than that of any other model. Taken alone, this result is a positive finding for DOP, but together with the result in Table 1, it seems to corroborate the earlier suggestion that it is overfit to the training data. Just as it generalizes less well in finding parse structures for unseen (grammatical) sentences, it is also unable to find satisfactory explanations for ungrammatical ones. Second, the overall trend of the perplexity scores here correlate with those of the grammatical

³Computing the best parse with tree substitution grammars is NP-hard (Sima’an, 1996), so we approximate it with the Viterbi derivation.

⁴It also prevents a model from winning a perplexity competition by ignoring the test data entirely and reporting a very small perplexity.

text: the sampled grammar alone is best, followed by the (roughly equal) spinal+CFG and sampled+CFG, which are all much improved over the plain CFG model. Finally, note that the plain CFG assigns a perplexity score that is in the neighborhood of those assigned by the ngram models, with the CFG’s slight lead perhaps explained by the fact that the permutation of the test-data was performed at the constituent (and not the word) level. Together, the results in Tables 1 and 2 provide evidence that the sampled TSG is the best grammatical language model.

4 Bilexicalized parsing

The previous section showed that sampled grammars outperform other grammars in terms of perplexity with a standard context-free parsing model that rewrites nonterminals in a single act conditioned solely on the identity of the current node. This is significant, because it suggests that PCFGs in general may not be as poor of language models as often thought, and that we have mechanisms for producing context-free grammars that do a much better job of modeling language than the Treebank grammar. The fact remains, however, that the context-free model is quite constrained, ignoring salient elements of the structural history that should surely be conditioned upon. A natural question that arises is whether the perplexity improvements seen in the previous section would carry over to more complicated generative models with fewer (and more realistic) independence assumptions. If so, this would provide some evidence that the underlying grammars being learned are finding something closer to the “real” structures behind the text.

In this section, we explore this question, using the same grammars from the previous section to train a bilexicalized, Markovized parser. This model is based on Collins Model 1 (Collins, 1999) and is similar to Charniak’s bihead model (Charniak, 2001).⁵ The generative model proceeds as follows: given nonterminal P (initially the top-level symbol), we

1. generate the head word and tag (h, t)
2. generate the head child H conditioned on $P, h,$ and $t)$
3. generate the siblings of H and their head tags, conditioned on $P, H, h, t,$ the direction from the head (left or right), and the distance from the head child (adjacent or not).
4. generate each sibling head word conditioned again on $P, H, h, t,$ direction, adjacency, plus the sibling label C and its head’s tag, c_t .

This process recurses until an entire parse tree has been generated. We employed the three-level backoff scheme presented in Table 7.1 of Collins (1999).

This model is called a lexicalized model because (a) the expansion of a node into all of its children is conditioned on the parent’s head word and (b) heads of the siblings of the head child are also conditioned on the parent’s head word. This latter conditioning is also called bilexicalization. We will take care not to confuse this with the notion of lexicalization in tree substitution grammars, where the term denotes the fact that subtrees are permitted to have lexical items among their frontier nodes.

The Collins model is trained by reading events from the Treebank. In order to do this with our TSGs, we had to preprocess the corpus in two ways. First, we flattened TSG subtrees to equivalent height-one CFG trees. This does not affect the end result, because internal nodes of TSG subtrees contribute nothing to language modeling under the inference models we are considering.⁶ Second, the Collins parsing model training procedures expect (tag,word) pairs in the training data, but flattening the lexicalized TSG rules removes many of the tags from the training corpus. To correct for this, we reintroduce dummy preterminals above such words that expand deterministically. We also modified the head-finding rules to select the same lexical head that would have been selected if the interior nodes were present. These preprocessing steps are illustrated in Figure 1. With this in place, the correct statistics can be collected for the Collins parsing model.

⁵Our implementation of Collins Model 1 differs in that we parse punctuation as regular symbols and do not make special provisions for commas and colons, in order to compute comparable perplexity scores.

⁶This is true for CFG parsing, as well, but not for parser evaluation. To evaluate for parsing accuracy, the original structure can be restored by retaining a mapping between the original subtrees and the flattened representation, as described by Bod (2001).

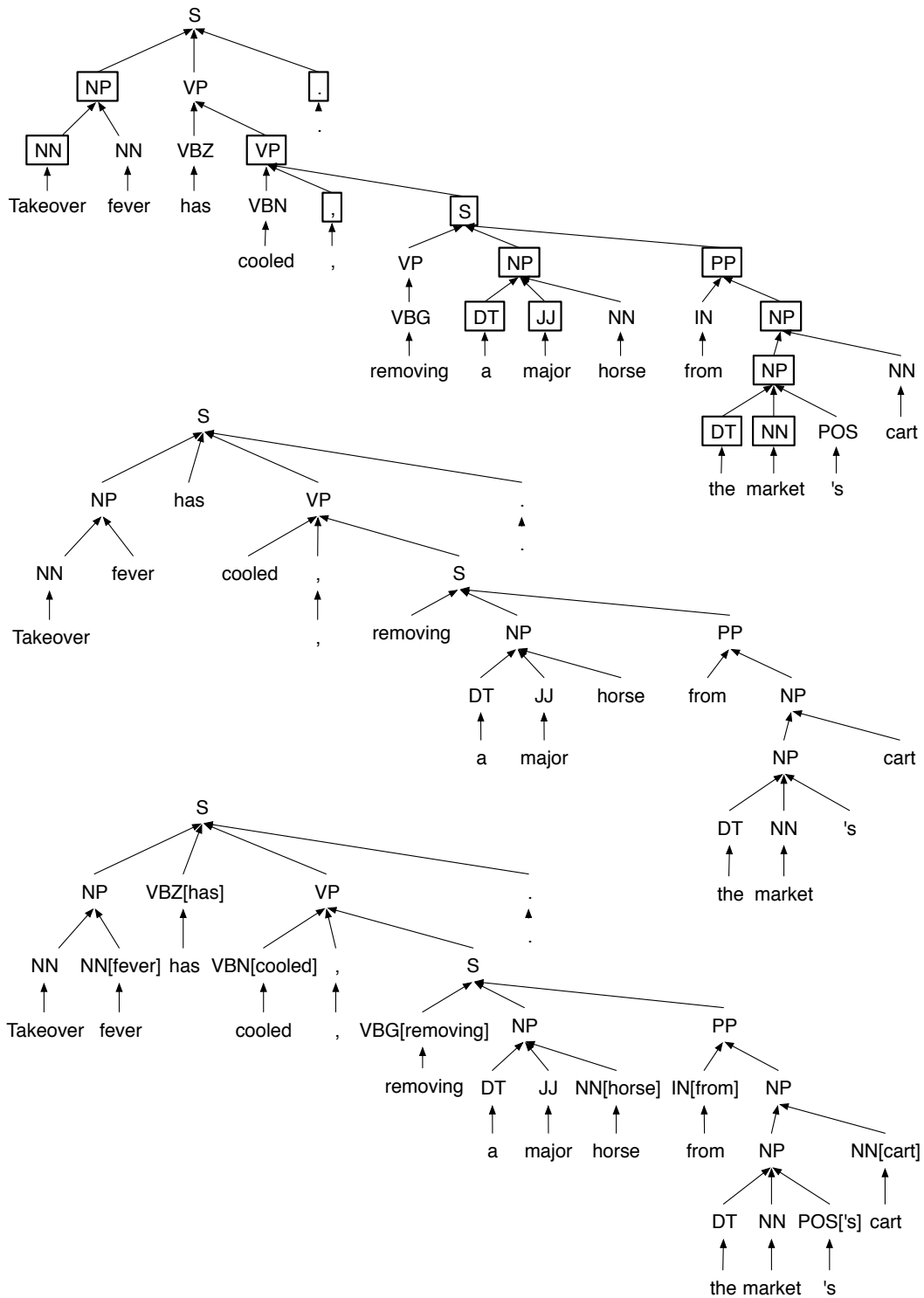


Figure 1: Preprocessing steps applied to TSG derivations in the Treeank to ensure compatibility with the Collins parsing model. (top) A TSG derivation tree in the training data for the “spinal” grammar. Boxed nodes denote boundaries between TSG subtrees (the root and frontier nodes of each subtree). (middle) The same derivation, collapsed to an equivalent height-one CFG. (bottom) The final tree with dummy preterminals inserted.

grammar	failures	perplexity
CFG	12	305.2
spinal+CFG	15	318.8
sampled	138	327.7
sampled+CFG	4	305.0
bigram	0	238.4
trigram	0	202.4

Table 3: Perplexity for the bilexicalized parsing model constructed from different training corpus derivations. The failures column denotes the number of sentences for which no parse could be found.

Table 3 presents perplexity results obtained by training this model over the flattened corpora. We begin by noting that perplexity scores for all the grammars are well above the ngram baselines. This is in contrast to previous work on syntax-based language modeling which has improved upon a trigram baseline (Chelba and Jelinek, 1998; Roark, 2001; Charniak, 2001). It is difficult to compare directly to this body of work: the vocabularies used were much smaller (10K), punctuation was removed and numbers were collapsed to a single token, and few details were given about the backoff scheme employed by the baseline trigram model. We note, too, that because of pruning, the string probabilities assigned by our parser are underestimates, and we did not experiment with beam width. But irrespective of these arguments, for purposes of this exploration we take the desirability of syntax as a given and focus on improving those models relative to the syntactic baseline.

The TSG models do not lead to any significant improvement over the baseline CFG, either. Analysis of the failures of the spinal grammar gives us a clue as to why this is the case. The reader may have observed that neither Table 1 nor 3 contain row entries for the spinal grammar alone. This is due to the fact that the spinal grammar, each rule of which has a lexical item among the nodes of its frontier, is too rigid for either parsing model, and nearly half the sentences resulted in a parse failure. The standard CFG parsing model has no notion of backing off, so if the fixed rules cannot be assembled into a parse tree, the parser fails. The same problem is present in the Collins parsing model, even with its backed-off smoothing structure. With lexical items at the leaves of PCFG rules, the generation of sibling nodes contains bilexical statistics even at the third level of backoff structure. To see this, note that these third-level statistics generate the sibling label C and sibling head tag t_c conditioned on the parent label P , head tag t , and direction Δ :

$$\Pr(C, t_c \mid P, t, \Delta)$$

In the collapsed structure depicted in Figure 1, this amounts to, e.g.,

$$\Pr(\text{NP}, \text{NN}[\text{fever}] \mid \text{S}, \text{VBZ}[\text{has}], \leftarrow)$$

It seems that the Collins backoff model is less useful when using a grammar in which all subtrees contain lexical items; a more appropriate model would employ a different backoff structure that would allow an analysis to be produced in light of these sparse parameters.

In light of this problem, one might hope that the other TSGs, which contain both lexicalized and unlexicalized rules, would outperform the CFG baseline. We do see a small improvement with the sampled+CFG grammar (fewer parse failures), but the lexicalization problem described above led us to a solution which showed real improvement: we forcibly detach lexical items from all subtrees in the training corpus at the preterminal node, and retrain the bilexicalized model on this new corpus. This can be seen as smoothing the TSGs in a manner that is more appropriate for the bilexical parsing model. We denote a corpus that has been modified in this way as corpus_D , and present perplexity results for these new grammars in Table 4.

In this way we obtain a larger improvement in perplexity. Table 4 shows that a combination of heuristic and sampled TSGs can beat the CFG baseline even though they are using a parsing model optimized for that baseline. Forcing the delexicalization of these grammars frees up the parsing model’s bilexical parameters, and computing those parameters over the revised structures produced by the flattened TSG subtrees helps produce a better language model. The results for parsing the mildly ungrammatical “reversed NP” version of the test corpus are less informative under this parsing model. The perplexity scores are all lower than the ngram models, and are very roughly similar; the CFG and spinal models give better (higher) overall scores, but they are also responsible for many more parse failures. Without further analysis of the characteristics of the sentences that produced failures and the lower scores, it is difficult to say what the different numbers mean.

grammar	failures	perplexity	reversed NPs	
			failures	perplexity
CFG	12	305.2	184	1,769.6
spinal _D	13	301.0	230	1,758.7
spinal _D +CFG	6	304.9	209	1,977.1
sampled _D	2	309.7	40	1,414.7
sampled _D +CFG	2	299.8	39	1,628.1
sampled _D +spinal _D	2	290.6	41	1,584.5
sampled _D +spinal _D +CFG	2	291.8	40	1,774.0
sampled _D +sampled	4	315.3	40	1,553.3
bigram	0	238.4	0	1,274.9
trigram	0	202.4	0	1,277.1

Table 4: Model perplexity for the bilexicalized parsing model after detaching lexical items from TSG subtrees in the training corpus.

5 Conclusion

Computational complexity is a significant barrier to the use of rich syntax-based models in practical applications. The large reduction in perplexity of induced tree substitution grammars in the standard CFG model is encouraging, because parsing in that model is cubic in the size of the input, as opposed to being $\mathcal{O}(n^4)$ for bilexical parsing (Eisner and Satta, 1999). With the perplexity scores of these TSG grammars under the simple parsing model approaching those of the more complicated bilexicalized parsing models, this kind of modeling could be feasible for applications like machine translation.

Beyond this, a growing intuition within the community is that the sorts of structures and models that have been useful for parsing are not necessarily the same as those that will be useful for language modeling. The improvements in perplexity gained from the use of automatically induced tree substitution grammars under two different parsing models suggest a research direction for future language modeling efforts, and the analysis presented here of the complicated interaction between grammar lexicalization and parsing model lexicalization should help inform smoothing methods more suited to language modeling with TSGs. In the future, we plan further experiments with this family of grammars, including the investigation of models and smoothing procedures best-suited to language modeling, and the optimal set of structures over which to train those models.

Acknowledgments This work was supported by NSF grants IIS-0546554 and ITR-0428020.

References

- Rens Bod. 2001. What is the minimal set of fragments that achieves maximal parse accuracy? In *Proc. ACL*, Toulouse, France.
- Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for machine translation. *MT Summit*.
- Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proc. ACL*, Toulouse, France.
- Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proc. COLING-ACL*, Montreal, Quebec.
- Colin Cherry and Chris Quirk. 2008. Discriminative, syntactic language modeling through latent SVMs. In *Proc. AMTA*, Honolulu, Hawaii. AMTA.
- David Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proc. ACL*, Hong Kong.

- Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *Proc. NAACL*, Boulder, Colorado.
- Michael John Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia.
- Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proc. ACL*, Morristown, New Jersey.
- Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*.
- 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 *Proc. NAACL*, Boston, Massachusetts.
- Daisuke Okanohara and Jun'ichi Tsujii. 2007. A discriminative language model with pseudo-negative samples. In *Proc. ACL*, Prague, Czech Republic.
- Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. ACL*, Sydney, Australia.
- Matt Post and Daniel Gildea. 2008. Parsers as language models for statistical machine translation. In *Proc. AMTA*, Honolulu, Hawaii.
- Matt Post and Daniel Gildea. 2009. Bayesian learning of a tree substitution grammar. In *Proc. ACL*, Singapore.
- Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.
- Khalil Sima'an. 1996. Computational complexity of probabilistic disambiguation by means of tree grammars. In *COLING*, Copenhagen, Denmark.