Immediate-Head Parsing for Language Models*

Eugene Charniak
Brown Laboratory for Linguistic Information Processing
Department of Computer Science
Brown University, Box 1910, Providence RI
ec@cs.brown.edu

Abstract

We present two language models based upon an "immediate-head" parser — our name for a parser that conditions all events below a constituent c upon the head of c. While all of the most accurate statistical parsers are of the immediate-head variety, no previous grammatical language model uses this technology. The perplexity for both of these models significantly improve upon the trigram model baseline as well as the best previous grammar-based language model. For the better of our two models these improvements are 24% and 14% respectively. We also suggest that improvement of the underlying parser should significantly improve the model's perplexity and that even in the near term there is a lot of potential for improvement in immediate-head language models.

1 Introduction

All of the most accurate statistical parsers [1,3,6,7, 12,14] are *lexicalized* in that they condition probabilities on the lexical content of the sentences being parsed. Furthermore, all of these parsers are what we will call *immediate-head* parsers in that all of the properties of the immediate descendants of a constituent c are assigned probabilities that are conditioned on the lexical head of c. For example, in Figure 1 the probability that the vp expands into v np pp is conditioned on the head of the vp, "put", as are the choices of the sub-heads under the vp, i.e., "ball" (the head of the np) and "in" (the head of the pp). It is the experience of the statistical parsing community that immediate-head parsers are the most accurate we can design.

It is also worthy of note that many of these parsers [1,3,6,7] are *generative* — that is, for a sentence s they try to find the parse π defined by Equation 1:

$$\arg max_{\pi} p(\pi \mid s) = \arg max_{\pi} p(\pi, s) \tag{1}$$

This is interesting because insofar as they compute $p(\pi, s)$ these parsers define a language-model in that they can (in principle) assign a probability to all possible sentences in the language by computing the sum in Equation 2:

$$p(s) = \sum_{\pi} p(\pi, s) \tag{2}$$

where $p(\pi, s)$ is zero if the yield of $\pi \neq s$. Language models, of course, are of interest because speech-recognition systems require them. These systems determine the words that were spoken by solving Equation 3:

$$\arg max_{s} p(s \mid A) = \arg max_{s} p(s) p(A \mid s) \tag{3}$$

where A denotes the acoustic signal. The first term on the right, $p(s)$, is the language model, and is what we compute via parsing in Equation 2.

Virtually all current speech recognition systems use the so-called *trigram* language model in which the probability of a string is broken down into conditional probabilities on each word given the two previous words. E.g.,

$$p(w_{0,n}) = \prod_{i=0,n-1} p(w_i \mid w_{i-1}, w_{i-2}) \tag{4}$$

* This research was supported in part by NSF grant LIS SBR 9720368 and by NSF grant IIS0095940. The author would like to thank the members of the Brown Laboratory for Linguistic Information Processing (BLLIP) and particularly Brian Roark who gave very useful tips on conducting this research. Thanks also to Fred Jelinek and Ciprian Chelba for the use of their data and for detailed comments on earlier drafts of this paper.

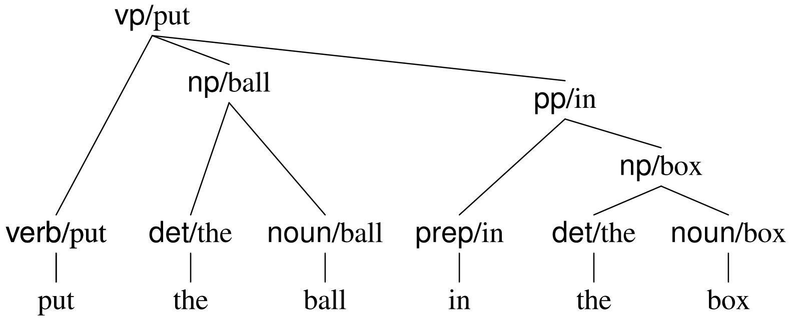

Figure 1: A tree showing head information

On the other hand, in the last few years there has been interest in designing language models based upon parsing and Equation 2. We now turn to this previous research.

2 Previous Work

There is, of course, a very large body of literature on language modeling (for an overview, see [10]) and even the literature on grammatical language models is becoming moderately large [4,9,15,16,17]. The research presented in this paper is most closely related to two previous efforts, that by Chelba and Jelinek [4] (C&J) and that by Roark [15], and this review concentrates on these two papers. While these two works differ in many particulars, we stress here the ways in which they are similar, and similar in ways that differ from the approach taken in this paper.

In both cases the grammar based language model computes the probability of the next word based upon the previous words of the sentence. More specifically, these grammar-based models compute a subset of all possible grammatical relations for the prior words, and then compute

- the probability of the next grammatical situation, and
- the probability of seeing the next word given each of these grammatical situations.

Also, when computing the probability of the next word, both models condition on the two prior heads of constituents. Thus, like a trigram model, they use information about triples of words.

Neither of these models uses an immediate-head parser. Rather they are both what we will call *strict left-to-right parsers*. At each sentence position in strict left-to-right parsing one computes the probability of the next word given the previous words (and does not go back to modify such probabilities). This is not possible in immediate-head parsing. Sometimes the immediate head of a constituent occurs after it (e.g, in noun-phrases, where the head is typically the rightmost noun) and thus is not available for conditioning by a strict left-to-right parser.

Model	Perplexity		
	Trigram	Grammar	Interpolation
C&J	167.14	158.28	148.90
Roark	167.02	152.26	137.26

Table 1: Perplexity results for two previous grammar-based language models

There are two reasons why one might prefer strict left-to-right parsing for a language model (Roark [15] and Chelba, personal communication). First, the search procedures for guessing the words that correspond to the acoustic signal works left to right in the string. If the language model is to offer guidance to the search procedure it must do so as well.

The second benefit of strict left-to-right parsing is that it is easily combined with the standard trigram model. In both cases at every point in the sentence we compute the probability of the next word given the prior words. Thus one can interpolate the trigram and grammar probability estimates for each word to get a more robust estimate. It turns out that this is a good thing to do, as is clear from Table 1, which gives perplexity results for a trigram model of the data in column one, results for the grammar-model in column two, and results for a model in which the two are interpolated in column three.

Both the were trained and tested on the same training and testing corpora, to be described in Section 4.1. As indicated in the table, the trigram model achieved a perplexity of 167 for the testing corpus. The grammar models did slightly better (e.g., 158.28 for the Chelba and Jelinek (C&J) parser), but it is the interpolation of the two that is clearly the winner (e.g., 137.26 for the Roark parser/trigram combination). In both papers the interpolation constants were 0.36 for the trigram estimate and 0.64 for the grammar estimate.

While both of these reasons for strict-left-to-right parsing (search and trigram interpolation) are valid, they are not necessarily compelling. The ability to combine easily with trigram models is important only as long as trigram models can improve grammar models. A sufficiently good grammar model would obviate the need for trigrams. As for the search problem, we briefly return to this point at the end of the paper. Here we simply note that while search requires that a language model provide probabilities in a left to right fashion, one can easily imagine procedures where these probabilities are revised after new information is found (i.e., the head of the constituent). Note that already our search procedure needs to revise previous most-likely-word hypotheses when the original guess makes the subsequent words very unlikely. Revising the associated language-model probabilities complicates the search procedure, but not unimaginably so. Thus it seems to us that it is worth finding out whether the superior parsing performance of immediate-head parsers translates into improved language models.

3 The Immediate-Head Parsing Model

We have taken the immediate-head parser described in [3] as our starting point. This parsing model assigns a probability to a parse π by a top-down process of considering each constituent c in π and, for each c, first guessing the pre-terminal of c, $t(c)$ (t for "tag"), then the lexical head of c, $h(c)$, and then the expansion of c into further constituents $e(c)$. Thus the probability of a parse is given by the equation

$$\begin{aligned} p(\pi) &= \prod_{c\in\pi} p(t(c) \mid l(c), H(c)) \\ &\quad \cdot p(h(c) \mid t(c), l(c), H(c)) \\ &\quad \cdot p(e(c) \mid l(c), t(c), h(c), H(c)) \end{aligned}$$

where $l(c)$ is the label of c (e.g., whether it is a noun phrase (np), verb phrase, etc.) and $H(c)$ is the relevant history of c — information outside c that our probability model deems important in determining the probability in question. In [3] $H(c)$ approximately consists of the label, head, and head-part-of-speech for the parent of c: $m(c), i(c)$, and $u(c)$ respectively. One exception is the distribution $p(e(c) \mid l(c), t(c), h(c), H(c))$, where H only includes m and u.[1]

Whenever it is clear to which constituent we are referring we omit the (c) in, e.g., $h(c)$. In this notation the above equation takes the following form:

$$\begin{aligned} p(\pi) &= \prod_{c\in\pi} p(t \mid l, m, u, i) \cdot p(h \mid t, l, m, u, i) \\ &\quad \cdot p(e \mid l, t, h, m, u). \end{aligned} \tag{5}$$

Because this is a point of contrast with the parsers described in the previous section, note that all of the conditional distributions are conditioned on one lexical item (either i or h). Thus only $p(h \mid t, l, m, u, i)$, the distribution for the head of c, looks at two lexical items (i and h itself), and none of the distributions look at three lexical items as do the trigram distribution of Equation 4 and the previously discussed parsing language models [4,15].

Next we describe how we assign a probability to the expansion e of a constituent. We break up a traditional probabilistic context-free grammar (PCFG) rule into a left-hand side with a label $l(c)$ drawn from the non-terminal symbols of our grammar, and a right-hand side that is a sequence of one or more such symbols. For each expansion we distinguish one of the right-hand side labels as the "middle" or "head" symbol $M(c)$. $M(c)$ is the constituent from which the head lexical item h is obtained according to deterministic rules that pick the head of a constituent from among the heads of its children. To the left of M is a sequence of one or more left labels $L_i(c)$ including the special termination symbol $\triangle$, which indicates that there are no more symbols to the left, and similarly for the labels to the right, $R_i(c)$. Thus an expansion $e(c)$ looks like:

$$l \rightarrow \triangle L_m \ldots L_1 M R_1 \ldots R_n \triangle. \tag{6}$$

The expansion is generated by guessing first M, then in order L_1 through L_{m+1} (= $\triangle$), and similarly for R_1 through R_{n+1}.

In anticipation of our discussion in Section 4.2, note that when we are expanding an L_i we do not know the lexical items to its left, but if we properly dovetail our "guesses" we can be sure of what word, if any, appears to its right and before M, and similarly for the word to the left of R_j. This makes such words available to be conditioned upon.

[1] We simplify slightly in this section. See [3] for all the details on the equations as well as the smoothing used.

Finally, the parser of [3] deviates in two places from the strict dictates of a language model. First, as explicitly noted in [3], the parser does not compute the partition function (normalization constant) for its distributions so the numbers it returns are not true probabilities. We noted there that if we replaced the "max-ent inspired" feature with standard deleted interpolation smoothing, we took a significant hit in performance. We have now found several ways to overcome this problem, including some very efficient ways to compute partition functions for this class of models. In the end, however, this was not necessary, as we found that we could obtain equally good performance by "hand-crafting" our interpolation smoothing rather than using the "obvious" method (which performs poorly).

Secondly, as noted in [2], the parser encourages right branching with a "bonus" multiplicative factor of 1.2 for constituents that end at the right boundary of the sentence, and a penalty of 0.8 for those that do not. This is replaced by explicitly conditioning the events in the expansion of Equation 6 on whether or not the constituent is at the right boundary (barring sentence-final punctuation). Again, with proper attention to details, this can be known at the time the expansion is taking place. This modification is much more complex than the multiplicative "hack," and it is not quite as good (we lose about 0.1% in precision/recall figures), but it does allow us to compute true probabilities.

The resulting parser strictly speaking defines a PCFG in that all of the extra conditioning information could be included in the non-terminal-node labels (as we did with the head information in Figure 1). When a PCFG probability distribution is estimated from training data (in our case the Penn tree-bank) PCFGs define a tight (summing to one) probability distribution over strings [5], thus making them appropriate for language models. We also empirically checked that our individual distributions ($p(t \mid l,m,u,i)$, and $p(h \mid t,l,m,u,i)$ from Equation 5 and $p(L \mid l,t,h,m,u)$, $p(M \mid l,t,h,m,u)$, and $p(R \mid l,t,h,m,u)$ from Equation 5) sum to one for a large, random, selection of conditioning events[2]

As with [3], a subset of parses is computed with a non-lexicalized PCFG, and the most probable edges (using an empirically established threshold) have their probabilities recomputed according to the complete probability model of Equation 5. Both searches are conducted using dynamic programming.

[2]They should sum to one. We are just checking that there are no bugs in the code.

Model	Perplexity		
	Trigram	Grammar	Interpolation
C&J	167.14	158.28	148.90
Roark	167.02	152.26	137.26
Bihead	167.89	144.98	133.15

Table 2: Perplexity results for the immediate-bihead model

4 Experiments

4.1 The Immediate-Bihead Language Model

The parser as described in the previous section was trained and tested on the data used in the previously described grammar-based language modeling research [4,15]. This data is from the Penn Wall Street Journal tree-bank [13], but modified to make the text more "speech-like". In particular:

1. all punctuation is removed,
2. no capitalization is used,
3. all symbols and digits are replaced by the symbol N, and
4. all words except for the 10,000 most common are replaced by the symbol UNK.

As in previous work, files F0 to F20 are used for training, F21-F22 for development, and F23-F24 for testing.

The results are given in Table 2. We refer to the current model as the bihead model. "Bihead" here emphasizes the already noted fact that in this model probabilities involve at most two lexical heads. As seen in Table 2, the immediate-bihead model with a perplexity of 144.98 outperforms both previous models, even though they use trigrams of words in their probability estimates.

We also interpolated our parsing model with the trigram model (interpolation constant .36, as with the other models) and this model outperforms the other interpolation models. Note, however, that because our parser does *not* define probabilities for each word based upon previous words (as with trigram) it is not possible to do the integration at the word level. Rather we interpolate the probabilities of the entire sentences. This is a much less powerful technique than the word-level interpolation used by both C&J and Roark, but we still observe a significant gain in performance.

4.2 The Immediate-Trihead Model

While the performance of the grammatical model is good, a look at sentences for which the trigram model outperforms it makes its limitations apparent. The sentences in question have noun phrases like "monday night football" that trigram models eats up but on which our bihead parsing model performs less well. For example, consider the sentence "he watched monday night football". The trigram model assigns this a probability of $1.9 \cdot 10^{-5}$, while the grammar model gives it a probability of $2.77 \cdot 10^{-7}$. To a first approximation, this is entirely due to the difference in probability of the noun-phrase. For example, the trigram probability $p(\text{football} \mid \text{monday}, \text{night}) = 0.366$, and would have been 1.0 except that smoothing saved some of the probability for other things it might have seen but did not. Because the grammar model conditions in a different order, the closest equivalent probability would be that for "monday", but in our model this is only conditioned on "football" so the probability is much less biased, only 0.0306. (Penn treebank base noun-phrases are flat, thus the head above "monday" is "football".)

This immediately suggests creating a second model that captures some of the trigram-like probabilities that the immediate-bihead model misses. The most obvious extension would be to condition upon not just one's parent's head, but one's grandparent's as well. This does capture some of the information we would like, particularly the case heads of noun-phrases inside of prepositional phrases. For example, in "united states of america", the probability of "america" is now conditioned not just on "of" (the head of its parent) but also on "states".

Unfortunately, for most of the cases where trigram really cleans up this revision would do little. Thus, in "he watched monday night football" "monday" would now be conditioned upon "football" and "watched." The addition of "watched" is unlikely to make much difference, certainly compared to the boost trigram models get by, in effect, recognizing the complete name.

It is interesting to note, however, that virtually all linguists believe that a noun-phrase like "monday night football" has significant substructure — e.g., it would look something like Figure 2. If we assume this tree-structure the two heads above "monday" are "night" and "football" respectively, thus giving our trihead model the same power as the trigram for this case. Ignoring some of the conditioning events, we now get a probability $p(h = \text{monday} \mid i = \text{night}, j = \text{football})$, which is much higher than the corresponding bihead version $p(h = \text{monday} \mid i = \text{football})$. The reader may remember that h is the head of the current constituent, while i is the head of its parent. We now define j to be the grandparent head.

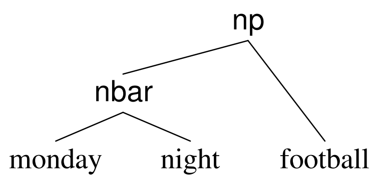

Figure 2: A noun-phrase with sub-structure

We decided to adopt this structure, but to keep things simple we only changed the definition of "head" for the distribution $p(h \mid t, l, m, u, i, j)$. Thus we adopted the following revised definition of head for constituents of base noun-phrases:

> For a pre-terminal (e.g., noun) constituent c of a base noun-phrase in which it is not the standard head (h) and which has as its right-sister another pre-terminal constituent d which is not itself h, the head of c is the head of d. The sole exceptions to this rule are phrase-initial determiners and numbers which retain h as their heads.

In effect this definition assumes that the substructure of all base noun-phrases is left branching, as in Figure 2. This is not true, but Lauer [11] shows that about two-thirds of all branching in base-noun-phrases is leftward. We believe we would get even better results if the parser could determine the true branching structure.

We then adopt the following definition of a grandparent-head feature j.

1. if c is a noun phrase under a prepositional phrase, or is a pre-terminal which takes a revised head as defined above, then j is the grandparent head of c, else
2. if c is a pre-terminal and is not next (in the production generating c) to the head of its parent (i) then $j(c)$ is the head of the constituent next to c in the production in the direction of the head of that production, else

Model	Perplexity		
	Trigram	Grammar	Interpolation
C&J	167.14	158.28	148.90
Roark	167.02	152.26	137.26
Bihead	167.89	144.98	133.15
Trihead	167.89	130.20	126.07

Table 3: Perplexity results for the immediate-trihead model

3. j is a "none-of-the-above" symbol.

Case 1 now covers both "united states of america" and "monday night football" examples. Case 2 handles other flat constituents in Penn tree-bank style (e.g., quantifier-phrases) for which we do not have a good analysis. Case three says that this feature is a no-op in all other situations.

The results for this model, again trained on F0-F20 and tested on F23-24, are given in Figure 3 under the heading "Immediate-trihead model". We see that the grammar perplexity is reduced to 130.20, a reduction of 10% over our first model, 14% over the previous best grammar model (152.26%), and 22% over the best of the above trigram models for the task (167.02). When we run the trigram and new grammar model in tandem we get a perplexity of 126.07, a reduction of 8% over the best previous tandem model and 24% over the best trigram model.

4.3 Discussion

One interesting fact about the immediate-trihead model is that of the 3761 sentences in the test corpus, on 2934, or about 75%, the grammar model assigns a higher probability to the sentence than does the trigram model. One might well ask what went "wrong" with the remaining 25%? Why should the grammar model ever get beaten? Three possible reasons come to mind:

1. The grammar model is better but only by a small amount, and due to sparse data problems occasionally the worse model will luck out and beat the better one.

2. The grammar model and the trigram model capture different facts about the distribution of words in the language, and for some set of sentences one distribution will perform better than the other.

3. The grammar model is, in some sense, always better than the trigram model, but if the parser bungles the parse, then the grammar model is impacted very badly. Obviously the trigram model has no such Achilles' heel.

Sentence Group	Num.	Labeled Precision	Labeled Recall
All Sentences	3761	84.6%	83.7%
Grammar High	2934	85.7%	84.9%
Trigram High	827	80.1%	79.0%

Table 4: Precision/recall for sentences in which trigram/grammar models performed best

We ask this question because what we should do to improve performance of our grammar-based language models depends critically on which of these explanations is correct: if (1) we should collect more data, if (2) we should just live with the tandem grammar-trigram models, and if (3) we should create better parsers.

Based upon a few observations on sentences from the development corpus for which the trigram model gave higher probabilities we hypothesized that reason (3), bungled parses, is primary. To test this we performed the following experiment. We divide the sentences from the test corpus into two groups, ones for which the trigram model performs better, and the ones for which the grammar model does better. We then collect labeled precision and recall statistics (the standard parsing performance measures) separately for each group. If our hypothesis is correct we expect the "grammar higher" group to have more accurate parses than the trigram-higher group as the poor parse would cause poor grammar perplexity for the sentence, which would then be worse than the trigram perplexity. If either of the other two explanations were correct one would not expect much difference between the two groups. The results are shown in Table 4. We see there that, for example, sentences for which the grammar model has the superior perplexity have average recall 5.9 ($= 84.9 - 79.0$) percentage points higher than the sentences for which the trigram model performed better. The gap for precision is 5.6. This seems to support our hypothesis.

5 Conclusion and Future Work

We have presented two grammar-based language models, both of which significantly improve upon both the trigram model baseline for the task (by 24% for the better of the two) and the best previous grammar-based language model (by 14%). Furthermore we have suggested that improvement of the underlying parser should improve the model's perplexity still further.

We should note, however, that if we were dealing with standard Penn Tree-bank Wall-Street-Journal text, asking for better parsers would be easier said than done. While there is still some progress, it is our opinion that substantial improvement in the state-of-the-art precision/recall figures (around 90%) is unlikely in the near future.[3] However, we are *not* dealing with standard tree-bank text. As pointed out above, the text in question has been "speechified" by removing punctuation and capitalization, and "simplified" by allowing only a fixed vocabulary of 10,000 words (replacing all the rest by the symbol "UNK"), and replacing all digits and symbols by the symbol "N".

We believe that the resulting text grossly underrepresents the useful grammatical information available to speech-recognition systems. First, we believe that information about rare or even truly unknown words would be useful. For example, when run on standard text, the parser uses ending information to guess parts of speech [3]. Even if we had never encountered the word "showboating", the "ing" ending tells us that this is almost certainly a progressive verb. It is much harder to determine this about UNK.[4] Secondly, while punctuation is not to be found in speech, prosody should give us something like equivalent information, perhaps even better. Thus significantly better parser performance on speech-derived data seems possible, suggesting that high-performance trigram-less language models may be within reach. We believe that the adaptation of prosodic information to parsing use is a worthy topic for future research.

Finally, we have noted two objections to immediate-head language models: first, they complicate left-to-right search (since heads are often to the right of their children) and second, they cannot be tightly integrated with trigram models.

The possibility of trigram-less language models makes the second of these objections without force. Nor do we believe the first to be a permanent disability. If one is willing to provide sub-optimal probability estimates as one proceeds left-to-right and then amend them upon seeing the true head, left-to-right processing and immediate-head parsing might be joined. Note that one of the cases where this might be worrisome, early words in a base noun-phrase could be conditioned upon a head which comes several words later, has been made significantly less problematic by our revised definition of heads inside noun-phrases. We believe that other such situations can be brought into line as well, thus again taming the search problem. However, this too is a topic for future research.

[3] Furthermore, some of the newest wrinkles [8] use discriminative methods and thus do not define language models at all, seemingly making them ineligible for the competition on a priori grounds.

[4] To give the reader some taste for the difficulties presented by UNKs, we encourage you to try parsing the following real example: "its supposedly unk unk unk a unk that makes one unk the unk of unk unk the unk radical unk of unk and unk and what in unk even seems like unk in unk".

References

1. BOD, R. *What is the minimal set of fragments that achieves maximal parse accuracy.* In *Proceedings of Association for Computational Linguistics 2001*. 2001.
2. CHARNIAK, E. *Tree-bank grammars.* In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. AAAI Press/MIT Press, Menlo Park, 1996, 1031–1036.
3. CHARNIAK, E. *A maximum-entropy-inspired parser.* In *Proceedings of the 2000 Conference of the North American Chapter of the Association for Computational Linguistics*. ACL, New Brunswick NJ, 2000.
4. CHELBA, C. AND JELINEK, F. *Exploiting syntactic structure for language modeling.* In *Proceedings for COLING-ACL 98*. ACL, New Brunswick NJ, 1998, 225–231.
5. CHI, Z. AND GEMAN, S. Estimation of probabilistic context-free grammars. *Computational Linguistics 24* 2 (1998), 299–306.
6. COLLINS, M. J. *Three generative lexicalized models for statistical parsing.* In *Proceedings of the 35th Annual Meeting of the ACL*. 1997, 16–23.
7. COLLINS, M. J. Head-Driven Statistical Models for Natural Language Parsing. University of Pennsylvania, Ph.D. Dissertation, 1999.
8. COLLINS, M. J. *Discriminative reranking for natural language parsing.* In *Proceedings of the International Conference on Machine Learning (ICML 2000)*. 2000.

9. GODDEAU, D. *Using probabilistic shift-reduce parsing in speech recognition systems.* In *Proceedings of the 2nd International Conference on Spoken Language Processing*. 1992, 321–324.

10. GOODMAN, J. *Putting it all together: language model combination.* In *ICASSP-2000*. 2000.

11. LAUER, M. *Corpus statistics meet the noun compound: some empirical results.* In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*. 1995, 47–55.

12. MAGERMAN, D. M. *Statistical decision-tree models for parsing.* In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*. 1995, 276–283.

13. MARCUS, M. P., SANTORINI, B. AND MARCINKIEWICZ, M. A. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics 19* (1993), 313–330.

14. RATNAPARKHI, A. Learning to parse natural language with maximum entropy models. *Machine Learning 34* 1/2/3 (1999), 151–176.

15. ROARK, B. Probabilistic top-down parsing and language modeling. *Computational Linguistics* (forthcoming).

16. STOLCKE, A. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics 21* (1995), 165–202.

17. STOLCKE, A. AND SEGAL, J. *Precise n-gram probabilities from stochastic context-free grammars.* In *Proceedings of the 32th Annual Meeting of the Association for Computational Linguistics*. 1994, 74–79.