Attention Shifting for Parsing Speech [*]

Keith Hall
Department of Computer Science
Brown University
Providence, RI 02912
`kh@cs.brown.edu`

Mark Johnson
Department of Cognitive and Linguistic Science
Brown University
Providence, RI 02912
`Mark_Johnson@Brown.edu`

Abstract

We present a technique that improves the efficiency of word-lattice parsing as used in speech recognition language modeling. Our technique applies a probabilistic parser iteratively where on each iteration it focuses on a different subset of the word-lattice. The parser's attention is shifted towards word-lattice subsets for which there are few or no syntactic analyses posited. This attention-shifting technique provides a six-times increase in speed (measured as the number of parser analyses evaluated) while performing equivalently when used as the first-stage of a multi-stage parsing-based language model.

1 Introduction

Success in language modeling has been dominated by the linear n-gram for the past few decades. A number of syntactic language models have proven to be competitive with the n-gram and better than the most popular n-gram, the trigram (Roark, 2001; Xu et al., 2002; Charniak, 2001; Hall and Johnson, 2003). Language modeling for speech could well be the first *real* problem for which syntactic techniques are useful.

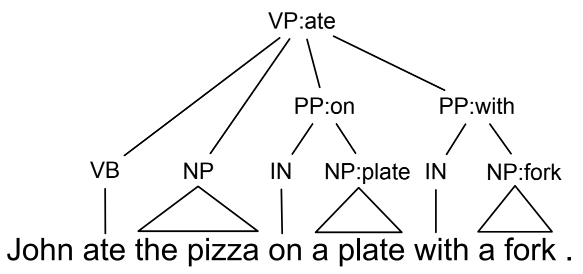


Figure 1: An incomplete parse tree with head-word annotations.

One reason that we expect syntactic models to perform well is that they are capable of modeling long-distance dependencies that simple n-gram models cannot. For example, the model presented by Chelba and Jelinek (Chelba and Jelinek, 1998; Xu et al., 2002) uses syntactic structure to identify lexical items in the left-context which are then modeled as an n-gram process. The model presented by Charniak (Charniak, 2001) identifies both syntactic structural and lexical dependencies that aid in language modeling. While there are n-gram models that attempt to extend the left-context window through the use of caching and skip models (Goodman, 2001), we believe that linguistically motivated models, such as these lexical-syntactic models, are more robust.

Figure 1 presents a simple example to illustrate the nature of long-distance dependencies. Using a syntactic model such as the the Structured Language Model (Chelba and Jelinek, 1998), we predict the word *fork* given the context $\{$*ate, with*$\}$ where a trigram model uses the context $\{$*with, a*$\}$. Consider the problem of disambiguating between *...plate with a fork* and *...plate with effort*. The syntactic model captures the semantic relationship between the words *ate* and *fork*. The syntactic structure allows us to find lexical contexts for which there is some semantic relationship (e.g., predicate-argument).

Unfortunately, syntactic language modeling techniques have proven to be extremely expensive in terms of computational effort. Many employ the use of string parsers; in order to utilize such techniques for language modeling one must preselect a set of strings from the word-lattice and parse each of them separately, an inherently inefficient procedure. Of the techniques that can process word-lattices directly, it takes significant computation to achieve the same levels of accuracy as the n–best reranking method. This computational cost is the result of increasing the search space evaluated with the syntactic model (parser); the larger space resulting from combining the search for syntactic structure with the search for paths in the word-lattice.

In this paper we propose a variation of a probabilistic word-lattice parsing technique that increases

[*] This research was supported in part by NSF grants 9870676 and 0085940.

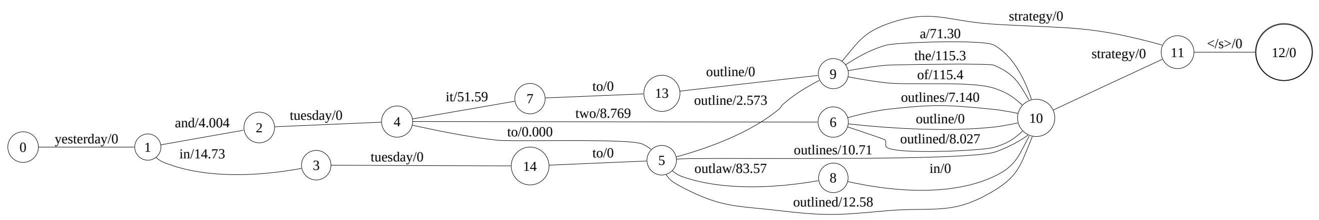


Figure 2: A partial word-lattice from the NIST HUB-1 dataset.

efficiency while incurring no loss of language modeling performance (measured as Word Error Rate – WER). In (Hall and Johnson, 2003) we presented a modular lattice parsing process that operates in two stages. The first stage is a PCFG word-lattice parser that generates a set of candidate parses over strings in a word-lattice, while the second stage rescores these candidate edges using a lexicalized syntactic language model (Charniak, 2001). Under this paradigm, the first stage is not only responsible for selecting candidate parses, but also for selecting paths in the word-lattice. Due to computational and memory requirements of the lexicalized model, the second stage parser is capable of rescoring only a small subset of all parser analyses. For this reason, the PCFG prunes the set of parser analyses, thereby indirectly pruning paths in the word lattice.

We propose adding a meta-process to the first-stage that effectively shifts the selection of word-lattice paths to the second stage (where lexical information is available). We achieve this by ensuring that for each path in the word-lattice the first-stage parser posits at least one parse.

2 Parsing speech word-lattices

$$P(A, W) = P(A|W)P(W) \tag{1}$$

The noisy channel model for speech is presented in Equation 1, where A represents the acoustic data extracted from a speech signal, and W represents a word string. The acoustic model $P(A|W)$ assigns probability mass to the acoustic data given a word string and the language model $P(W)$ defines a distribution over word strings. Typically the acoustic model is broken into a series of distributions conditioned on individual words (though these are based on false independence assumptions).

$$P(A|w_1 \ldots w_i \ldots w_n) = \prod_{i=1}^{n} P(A|w_i) \tag{2}$$

The result of the acoustic modeling process is a set of string hypotheses; each word of each hypothesis is assigned a probability by the acoustic model.

Word-lattices are a compact representation of output of the acoustic recognizer; an example is presented in Figure 2. The word-lattice is a weighted directed acyclic graph where a path in the graph corresponds to a string predicted by the acoustic recognizer. The (sum) product of the (log) weights on the graph (the acoustic probabilities) is the probability of the acoustic data given the string. Typically we want to know the most likely string given the acoustic data.

$$\begin{aligned} &\arg\max P(W|A) && (3)\\ &= \arg\max P(A, W) \\ &= \arg\max P(A|W)P(W) \end{aligned}$$

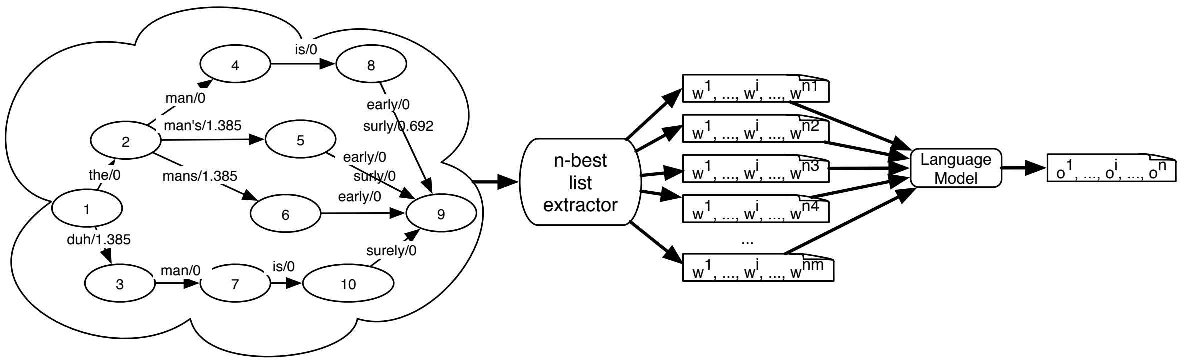
In Equation 3 we use Bayes' rule to find the optimal string given $P(A|W)$, the acoustic model, and $P(W)$, the language model. Although the language model can be used to *rescore*[1] the word-lattice, it is typically used to select a single hypothesis.

We focus our attention in this paper to syntactic language modeling techniques that perform complete parsing, meaning that parse trees are built upon the strings in the word-lattice.

2.1 n–best list reranking

Much effort has been put forth in developing efficient probabilistic models for parsing strings (Caraballo and Charniak, 1998; Goldwater et al., 1998; Blaheta and Charniak, 1999; Charniak, 2000; Charniak, 2001); an obvious solution to parsing word-lattices is to use n–best list reranking. The n–best list reranking procedure, depicted in Figure 3, utilizes an external language model that selects a set of strings from the word-lattice. These strings are analyzed by the parser which computes a language model probability. This probability is combined

[1]To rescore a word-lattice, each arch is assigned a new score (probability) defined by a new model (in combination with the acoustic model).



is/0
4
8
man/0
early/0
man's/1.385
surly/0.692
2
5
the/0
mans/1.385
early/0
surly/0
early/0
1
6
9
duh/1.385
surely/0
3
man/0
7
is/0
10
n-best list extractor
$w^1, ..., w^i, ..., w^{n1}$
$w^1, ..., w^i, ..., w^{n2}$
$w^1, ..., w^i, ..., w^{n3}$
$w^1, ..., w^i, ..., w^{n4}$
...
$w^1, ..., w^i, ..., w^{nm}$
Language Model
$o^1, ..., o^i, ..., o^n$

Figure 3: n–best list reranking

with the acoustic model probability to reranked the strings according to the joint probability $P(A, W)$.

There are two significant disadvantages to this approach. First, we are limited by the performance of the language model used to select the n–best lists. Usually, the trigram model is used to select n paths through the lattice generating at most n unique strings. The maximum performance that can be achieved is limited by the performance of this extractor model. Second, of the strings that are analyzed by the parser, many will share common substrings. Much of the work performed by the parser is duplicated for these substrings. This second point is the primary motivation behind parsing word-lattices (Hall and Johnson, 2003).

2.2 Multi-stage parsing

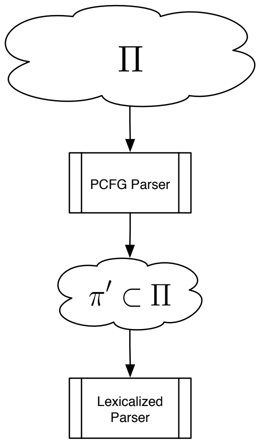


Figure 4: Coarse-to-fine lattice parsing.

In Figure 4 we present the general overview of a multi-stage parsing technique (Goodman, 1997; Charniak, 2000; Charniak, 2001). This process is know as coarse-to-fine modeling, where coarse models are more efficient but less accurate than fine models, which are robust but computationally expensive. In this particular parsing model a PCFG best-first parser (Bobrow, 1990; Caraballo and Charniak, 1998) is used to search the unconstrained space of parses Π over a string. This first stage performs *overparsing* which effectively allows it to generate a set of high probability candidate parses π'. These parses are then rescored using a lexicalized syntactic model (Charniak, 2001). Although the coarse-to-fine model may include any number of intermediary stages, in this paper we consider this two-stage model.

1. Parse word-lattice with PCFG parser
2. Overparse, generating additional candidates
3. Compute inside-outside probabilities
4. Prune candidates with probability threshold

Table 1: First stage word-lattice parser

There is no guarantee that parses favored by the second stage will be generated by the first stage. In other words, because the first stage model prunes the space of parses from which the second stage rescores, the first stage model may remove solutions that the second stage would have assigned a high probability.

In (Hall and Johnson, 2003), we extended the multi-stage parsing model to work on word-lattices. The first-stage parser, Table 1, is responsible for positing a set of candidate parses over the word-lattice. Were we to run the parser to completion it would generate all parses for all strings described by the word-lattice. As with string parsing, we stop the first stage parser early, generating a subset of all parses. Only the strings covered by complete parses are passed on to the second stage parser. This indirectly prunes the word-lattice of all word-arcs that were not covered by complete parses in the first stage.

We use a first stage PCFG parser that performs a best-first search over the space of parses, which means that it depends on a heuristic “figure-of-merit” (FOM) (Caraballo and Charniak, 1998). A good FOM attempts to model the true probability of a chart edge[2] $P(N^i_{j,k})$. Generally, this probability is impossible to compute during the parsing process as it requires knowing both the inside and outside probabilities (Charniak, 1993; Manning and Schütze, 1999). The FOM we describe is an approximation to the edge probability and is computed using an estimate of the inside probability times an approximation to the outside probability[3].

The inside probability $\beta(N^i_{j,k})$ can be computed incrementally during bottom-up parsing. The normalized acoustic probabilities from the acoustic recognizer are included in this calculation.

$$\begin{aligned}&\hat{\alpha}(N^i_{j,k}) \qquad\qquad (4)\\ &= \sum_{i,l,q,r} fwd(T^q_{i,j})p(N^i|T^q)p(T_r|N^i)bkwd(T^r_{k,l})\end{aligned}$$

The outside probability is approximated with a bitag model and the standard tag/category boundary model (Caraballo and Charniak, 1998; Hall and Johnson, 2003). Equation 4 presents the approximation to the outside probability. Part-of-speech tags T^q and T^r are the candidate tags to the left and right of the constituent $N^i_{j,k}$. The $fwd()$ and $bkwd()$ functions are the HMM forward and backward probabilities calculated over a lattice containing the part-of-speech tag, the word, and the acoustic scores from the word-lattice to the left and right of the constituent, respectively. $p(N^i|T^q)$ and $p(T_r|N^i)$ are the boundary statistics which are estimated from training data (details of this model can be found in (Hall and Johnson, 2003)).

$$\text{FOM}(N^i_{j,k}) = \hat{\alpha}(N^i_{j,k})\beta(N^i_{j,k})\eta^C(j,k) \qquad (5)$$

The best-first search employed by the first stage parser uses the FOM defined in Equation 5, where η is a normalization factor based on path length $C(j,k)$. The normalization factor prevents small constituents from consistently being assigned a higher probability than larger constituents (Goldwater et al., 1998).

Although this heuristic works well for directing the parser towards likely parses over a string, it is not an ideal model for pruning the word-lattice. First, the outside approximation of this FOM is based on a linear part-of-speech tag model (the bitag). Such a simple syntactic model is unlikely to provide realistic information when choosing a word-lattice path to consider. Second, the model is prone to favoring subsets of the word-lattice causing it to posit additional parse trees for the favored sublattice rather than exploring the remainder of the word-lattice. This second point is the primary motivation for the attention shifting technique presented in the next section.

3 Attention shifting[4]

We explore a modification to the multi-stage parsing algorithm that ensures the first stage parser posits at least one parse for each path in the word-lattice. The idea behind this is to intermittently shift the attention of the parser to unexplored parts of the word lattice.

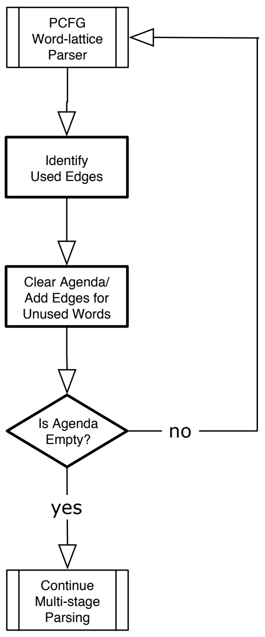

PCFG Word-lattice Parser
Identify Used Edges
Clear Agenda/ Add Edges for Unused Words
Is Agenda Empty?
no
yes
Continue Multi-stage Parsing

Figure 5: Attention shifting parser.

Figure 5 depicts the attention shifting first stage parsing procedure. A *used edge* is a parse edge that has non-zero outside probability. By definition of

[2]A chart edge $N^i_{j,k}$ indicates a grammar category N^i can be constructed from nodes j to k.

[3]An alternative to the inside and outside probabilities are the Viterbi inside and outside probabilities (Goldwater et al., 1998; Hall and Johnson, 2003).

[4]The notion of attention shifting is motivated by the work on parser FOM compensation presented in (Blaheta and Charniak, 1999).

the outside probability, used edges are constituents that are part of a complete parse; a parse is complete if there is a root category label (e.g., **S** for sentence) that spans the entire word-lattice. In order to identify used edges, we compute the outside probabilities for each parse edge (efficiently computing the outside probability of an edge requires that the inside probabilities have already been computed).

In the third step of this algorithm we clear the agenda, removing all partial analyses evaluated by the parser. This forces the parser to abandon analyses of parts of the word-lattice for which complete parses exist. Following this, the agenda is populated with edges corresponding to the unused words, priming the parser to consider these words. To ensure the parser builds upon at least one of these unused edges, we further modify the parsing algorithm:

- Only unused edges are added to the agenda.
- When building parses from the bottom up, a parse is considered complete if it connects to a used edge.

These modifications ensure that the parser focuses on edges built upon the unused words. The second modification ensures the parser is able to determine when it has connected an unused word with a previously completed parse. The application of these constraints directs the attention of the parser towards new edges that contribute to parse analyses covering unused words. We are guaranteed that each iteration of the attention shifting algorithm adds a parse for at least one unused word, meaning that it will take at most $|A|$ iterations to cover the entire lattice, where A is the set of word-lattice arcs. This guarantee is trivially provided through the constraints just described. The attention-shifting parser continues until there are no unused words remaining and each parsing iteration runs until it has found a complete parse using at least one of the unused words.

As with multi-stage parsing, an adjustable parameter determines how much overparsing to perform on the initial parse. In the attention shifting algorithm an additional parameter specifies the amount of overparsing for each iteration after the first. The new parameter allows for independent control of the attention shifting iterations.

After the attention shifting parser populates a parse chart with parses covering all paths in the lattice, the multi-stage parsing algorithm performs additional pruning based on the probability of the parse edges (the product of the inside and outside probabilities). This is necessary in order to constrain the size of the hypothesis set passed on to the second stage parsing model.

The Charniak lexicalized syntactic language model effectively splits the number of parse states (an edges in a PCFG parser) by the number of unique contexts in which the state is found. These contexts include syntactic structure such as parent and grandparent category labels as well as lexical items such as the head of the parent or the head of a sibling constituent (Charniak, 2001). State splitting on this level causes the memory requirement of the lexicalized parser to grow rapidly.

Ideally, we would pass all edges on to the second stage, but due to memory limitations, pruning is necessary. It is likely that edges recently discovered by the attention shifting procedure are pruned. However, the true PCFG probability model is used to prune these edges rather than the approximation used in the FOM. We believe that by considering parses which have a relatively high probability according to the combined PCFG and acoustic models that we will include most of the analyses for which the lexicalized parser assigns a high probability.

4 Experiments

The purpose of attention shifting is to reduce the amount of work exerted by the first stage PCFG parser while maintaining the same quality of language modeling (in the multi-stage system). We have performed a set of experiments on the NIST ’93 HUB–1 word-lattices. The HUB–1 is a collection of 213 word-lattices resulting from an acoustic recognizer’s analysis of speech utterances. Professional readers reading Wall Street Journal articles generated the utterances.

The first stage parser is a best-first PCFG parser trained on sections 2 through 22, and 24 of the Penn WSJ treebank (Marcus et al., 1993). Prior to training, the treebank is transformed into speech-like text, removing punctuation and expanding numerals, etc.[5] Overparsing is performed using an *edge pop*[6] multiplicative factor. The parser records the number of edge pops required to reach the first complete parse. The parser continues to parse a until multiple of the number of edge pops required for the first parse are popped off the agenda.

The second stage parser used is a modified version of the Charniak language modeling parser described in (Charniak, 2001). We trained this parser

[5]Brian Roark of AT&T provided a tool to perform the speech normalization.

[6]An edge pop is the process of the parser removing an edge from the agenda and placing it in the parse chart.

on the BLLIP99 corpus (Charniak et al., 1999); a corpus of 30million words automatically parsed using the Charniak parser (Charniak, 2000).

In order to compare the work done by the n–best reranking technique to the word-lattice parser, we generated a set of n–best lattices. 50–best lists were extracted using the Chelba A* decoder[7]. A 50–best lattice is a sublattice of the acoustic lattice that generates only the strings found in the 50–best list. Additionally, we provide the results for parsing the full acoustic lattices (although these work measurements should not be compared to those of n–best reranking).

We report the amount of work, shown as the cumulative # edge pops, the oracle WER for the word-lattices after first stage pruning, and the WER of the complete multi-stage parser. In all of the word-lattice parsing experiments, we pruned the set of posited hypothesis so that no more than 30,000 local-trees are generated[8]. We chose this threshold due to the memory requirements of the second stage parser. Performing pruning at the end of the first stage prevents the attention shifting parser from reaching the minimum oracle WER (most notable in the full acoustic word-lattice experiments). While the attention-shifting algorithm ensures all word-lattice arcs are included in complete parses, forward-backward pruning, as used here, will eliminate some of these parses, indirectly eliminating some of the word-lattice arcs.

To illustrate the need for pruning, we computed the number of states used by the Charniak lexicalized syntactic language model for 30,000 local trees. An average of 215 lexicalized states were generated for each of the 30,000 local trees. This means that the lexicalized language model, on average, computes probabilities for over 6.5 million states when provided with 30,000 local trees.

Model	# edge pops	O-WER	WER
n–best (Charniak)	2.5 million	7.75	11.8
100x LatParse	3.4 million	8.18	12.0
10x AttShift	**564,895**	**7.78**	11.9

Table 2: Results for n–best lists and n–best lattices.

Table 2 shows the results for n–best list reranking and word-lattice parsing of n–best lattices. We recreated the results of the Charniak language model parser used for reranking in order to measure the amount of work required. We ran the first stage parser with 4-times overparsing for each string in the n–best list. The LatParse result represents running the word-lattice parser on the n–best lattices performing 100–times overparsing in the first stage. The AttShift model is the attention shifting parser described in this paper. We used 10–times overparsing for both the initial parse and each of the attention shifting iterations. When run on the n–best lattice, this model achieves a comparable WER, while reducing the amount of parser work sixfold (as compared to the regular word-lattice parser).

Model	# edge pops	O-WER	WER
acoustic lats	N/A	3.26	N/A
100x LatParse	3.4 million	5.45	13.1
10x AttShift	**1.6 million**	**4.17**	13.1

Table 3: Results for acoustic lattices.

In Table 3 we present the results of the word-lattice parser and the attention shifting parser when run on full acoustic lattices. While the oracle WER is reduced, we are considering almost half as many edges as the standard word-lattice parser. The increased size of the acoustic lattices suggests that it may not be computationally efficient to consider the entire lattice and that an additional pruning phase is necessary.

The most significant constraint of this multi-stage lattice parsing technique is that the second stage process has a large memory requirement. While the attention shifting technique does allow the parser to propose constituents for every path in the lattice, we prune some of these constituents prior to performing analysis by the second stage parser. Currently, pruning is accomplished using the PCFG model. One solution is to incorporate an intermediate pruning stage (e.g., lexicalized PCFG) between the PCFG parser and the full lexicalized model. Doing so will relax the requirement for aggressive PCFG pruning and allows for a lexicalized model to influence the selection of word-lattice paths.

5 Conclusion

We presented a parsing technique that shifts the attention of a word-lattice parser in order to ensure syntactic analyses for all lattice paths. Attention shifting can be thought of as a meta-process around the first stage of a multi-stage word-lattice parser. We show that this technique reduces the amount of work exerted by the first stage PCFG parser while maintaining comparable language modeling performance.

Attention shifting is a simple technique that attempts to make word-lattice parsing more efficient. As suggested by the results for the acoustic lattice experiments, this technique alone is not sufficient.

[7]The n–best lists were provided by Brian Roark (Roark, 2001)

[8]A local-tree is an explicit expansion of an edge and its children. An example local tree is $\mathrm{NP}_{3,8} \rightarrow \mathrm{DT}_{3,4}\ \mathrm{NN}_{4,8}$.

Solutions to improve these results include modifying the first-stage grammar by annotating the category labels with local syntactic features as suggested in (Johnson, 1998) and (Klein and Manning, 2003) as well as incorporating some level of lexicalization. Improving the quality of the parses selected by the first stage should reduce the need for generating such a large number of candidates prior to pruning, improving efficiency as well as overall accuracy. We believe that attention shifting, or some variety of this technique, will be an integral part of efficient solutions for word-lattice parsing.

References

Don Blaheta and Eugene Charniak. 1999. Automatic compensation for parser figure-of-merit flaws. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 513–518.

Robert J. Bobrow. 1990. Statistical agenda parsing. In *DARPA Speech and Language Workshop*, pages 222–224.

Sharon Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–298, June.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 1999. BLLIP 1987–89 wsj corpus release 1. LDC corpus LDC2000T43.

Eugene Charniak. 1993. *Statistical Language Learning*. MIT Press.

Eugene Charniak. 2000. 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.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*.

Ciprian Chelba and Frederick Jelinek. 1998. A study on richer syntactic dependencies for structured language modeling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 225–231.

Sharon Goldwater, Eugene Charniak, and Mark Johnson. 1998. Best-first edge-based chart parsing. In *6th Annual Workshop for Very Large Corpora*, pages 127–133.

Joshua Goodman. 1997. Global thresholding and multiple-pass parsing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 11–25.

Joshua Goodman. 2001. A bit of progress in language modeling, extendend version. In *Microsoft Research Technical Report MSR-TR-2001-72*.

Keith Hall and Mark Johnson. 2003. Language modeling using efficient best-first bottom-up parsing. In *Proceedings of IEEE Automated Speech Recognition and Understanding Workshop*.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24:617–636.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics (ACL-03)*.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT Press.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(3):249–276.

Peng Xu, Ciprian Chelba, and Frederick Jelinek. 2002. A study on richer syntactic dependencies for structured language modeling. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 191–198.