LANGUAGE MODELING USING EFFICIENT BEST-FIRST BOTTOM-UP PARSING

Keith Hall

Brown University
Department of Computer Science
kh@cs.brown.edu

Mark Johnson

Brown University
Department of Cognitive and Linguistic Sciences
and Department of Computer Science
Mark_Johnson@brown.edu

ABSTRACT

In this paper we present a two-stage best-first bottom-up word-lattice parser which we use as a language model for speech recognition. The parser works by using a "Figure of Merit" that selects lattice paths while simultaneously selecting syntactic category edges for parsing. Additionally, we introduce a modified version of the Inside-Outside algorithm used as a pruning stage between syntactic context-free parsing and lexicalized context-dependent parsing. We report our results in terms of Word Error Rate on the HUB–1 word-lattices and compare these results to other syntactic language modeling techniques.

1. INTRODUCTION

This paper describes a best-first word-lattice chart parser and shows how it can be used for language modeling. Best-first parsing requires an estimate be used to select "good" syntactic edges for parsing. This estimate is known as the "Figure of Merit" (FOM). The FOM presented in this paper combines word-lattice and parse-chart information allowing the parser to simultaneously perform both path search in the word-lattice and parse search in the set of all possible parses. The parser is divided into two stages. First, we use a PCFG model to select a set of candidate edges which are then passed to the second stage that selects the best parse using a more sophisticated language model. The second stage model exploits lexical and other information in the way that Charniak's [1] and Collins' [2] parsers do. Dividing the parser into two stages is necessary for efficiency reasons; there are simply too many possible parses for the sophisticated model to examine.

In previous work on syntactic language models, the most successful techniques are based on n–best list rescoring[3, 4, 5]. These systems extract a set of strings from the word-lattice using a trigram model and then rescore the strings using the syntactic language model. By parsing directly from a word-lattice, we consolidate the work expended to process identical substrings. In other words, we parse all strings represented by the word-lattice simultaneously. Any common substrings are automatically processed as they occur in the lattice as a single sub-path.

The most closely related work to our word-lattice chart parsing can be found in [6]. As in our work, they derive a "Figure of Merit" that is used to direct a best-first chart-parser. Unlike the work discussed in this paper their parser is based on a pseudo-probabilistic unification-based grammar. Chappalier, et. al. [7] describe a word-lattice chart parsing scheme which is similar to the one we use here although they use a modified CKY parser rather than a best-first parsing solution. We are unaware of any previous results for word-lattice chart-parsing techniques used for speech-recognition language modeling.

Our work is also very closely related to methods for best-first PCFG parsing of strings devised by Caraballo and Charniak[8] and Goldwater et al.[9]; indeed, this paper can be seen largely as an attempt to extend the techniques these authors developed for string parsing to word-lattices. Caraballo and Charniak[8] presented the basic best-first bottom-up parsing architecture and introduced the idea of approximating the outside probability of an edge by a bitag model estimate that we use here. Goldwater et al.[9] refined this model by using a binarized "Markov" grammar rather the original PCFG, and by using the Viterbi inside score for the most likely subtree rooted in that edge in place of the traditional inside score (which sums the probability of all possible subtrees).

The model presented here differs from the one presented by Goldwater et al.[9] in two main respects. First, because we are parsing from a word-lattice rather than a string, we define the outside score approximation in terms of the paths from the lattice start and end vertices to the beginning and ending vertices of the edge respectively. And second, in calculating all edge scores, including the outside score approximation, we always use the Viterbi score, which considers only the most likely derivation rather than summing over all derivations. This speeds up calculation and is robust to the

This material is based upon work supported by the National Science Foundation under IGERT Award No. 9870676.

kinds of "weight movement" changes that occur when the speech lattice is preprocessed by standard weighted FSM tools.

The rest of the paper is structured as follows. In the following section, we present an overview of our word-lattice parser. Next, Section 3 describes in detail the input to the first-stage parser, the word-lattices. Section 4 describes the algorithms that implement the first-stage parser. In Section 5 we describe the HUB–1 experiments and present the results for our lattice-parser combined with Charniak's parser. Finally, Section 6 summarizes and concludes the paper.

2. WORD-LATTICE PARSER LANGUAGE MODEL

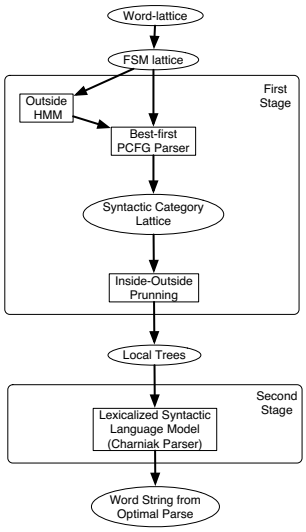

Fig. 1. Overview of the word-lattice parser

Figure 1 depicts the general flow of the word-lattice parser we present in this paper. Prior to the first parsing stage the word-lattices are converted into weighted Finite State Machines. This step is done to reduce the size of the lattice and therefore reduce the amount of duplicate effort the parser expends on identical substrings. We provide a description of this step in the next section.

This first-stage parser itself consists of several steps, the output of previous serving as the input of the next.

Step 1: A best-first, bottom-up PCFG chart parser identifies candidate category edges (i.e., a category label plus a beginning and ending vertex from the word-lattice). The bulk of this paper is concerned with this step.

Step 2: The edges produced by the bottom-up parser are filtered by a modified version of the inside-outside algorithm that selects just those edges that are contained in some parse tree constructible from the edge set whose probability is greater than a threshold. This ensures that all of the filtered edges are "useful".

Step 3: A set of "local trees" in the form required for the second stage of the language model/parser is assembled from the filtered category edges by a process very much like parsing. A local tree consists of a a parent plus its immediate children together with a pair of beginning and ending vertices for each node.

The second stage takes the output from Step 3, and applies a lexicalized context dependent syntactic parsing model. Each local tree is assigned a new probability by this model from which the standard Inside and Outside probabilities are computed. Once this is done, the second stage selects the most probable complete parse. Charniak's parser models long-distance lexical dependencies; for example, the probability of a word given the head-word of the grand-parent phrase (a phrase which dominates a phrase dominating the word). Due to the complexity of these models (e.g. fourth-order Markov Models), the number of local trees that can be processed is limited (by both size and time).

3. WORD LATTICES

A word-lattice is a compact representation of a set of string hypotheses. The word-lattices we consider in this paper have been extracted from speech-lattices where all recognizer-based time-frame information has been removed (following word start/end quantization). Specifically, we are using the NIST HUB–1(93) evaluation set which is encoded in the HTK Standard Lattice Format (SLF) (`http://htk.eng.cam.ac.uk/docs/docs.shtml`). For reasons of space efficiency, the SLF format assigns labels to nodes rather than the arcs to which they are associated. Each arc entering a node should carry the label that is on the node the arc enters. We describe the lattices as if the labels were on the arcs. A word-lattice $L = (V, E)$ is a labeled directed acyclic graph (DAG) where:

V a set of *nodes* (or vertices) which represent the transitions between words.

E a set of labeled, weighted *edges* which represent a word hypotheses. In the SLF format, each arc may contain multiple weights, each associated with a separate model. Two such weights are:

acoustic score is the score assigned by the acoustic model. This is an estimate of $p(a_i|w_i)$, the conditionally probability of the acoustic observation (for the time-frame loosely associated with the current word) a_i, given word w_i (from a closed vocabulary). These scores are computed through a series of HMMs that predict phonemes from acoustic measurements and then words from phonemes[10].

language model score is the score assigned by a language model (most commonly a trigram): $p(w_i|w_{i-1}, w_{i-2}, \ldots)$.

Each path through the word-lattice corresponds to a string hypothesis. The probability assigned to a path by the acoustic model is computed by multiplying the acoustic scores along the path (or in log-space, summing the scores followed by a conversion back to probability space). For example, the probability of the acoustic evidence, A, for the utterance given the lattice path $W = (w_1, \ldots, w_i, \ldots, w_n)$ is defined as:

$$p(A|W) = \prod_{w_i \in W} p(a_i|w_i) \quad (1)$$

In the current work, we are ignoring the trigram scores and instead using our parser as the language model.

In order to reduce the size of a lattice while preserving the scores, we inherit a well-known observation: the weighted lattices can be represented by weighted Finite State Machines (FSM). A weighted Finite State Acceptor associated with a lattice accepts all strings that were encoded in the word-lattice and assigns each string the cumulative score associated with the path in the word-lattice containing that string.

After transforming the lattice into FSMs we apply standard techniques which reduce the size of the lattice. First, we determinize the weighted FSM, ensuring that there is only one path for any unique string. And second, we minimize the FSM, combining common sub-paths of the FSM. In order to perform these manipulations while preserving the scores associated with strings, the weights associated with each FSM transition must be shifted. We use the AT&T FSM toolkit to perform these manipulations[11]. More information on these tools is available at
`http://www.research.att.com/sw/tools/fsm`.

By shifting the scores within the weighted FSM, we no longer have accurate scores assigned to each arc. A greedy search through the FSM after weight shifting may produce a different result than the same search done on the original FSM. The parsing technique, in particular the "Figure of Merit", proposed in this paper is designed to be resilient to this weight shifting.

Finally, we note that the weighted DAG (associated with the weighted FSM) is a prefect representation for the chart as used in chart parsing. This follows from the insights made in[6] and [7].

4. PARSING FROM SPEECH-LATTICES

The parsing model we propose is similar to the string parsing models found in [8, 9]. The FOM is an estimate of the probability that a category edge exists in the complete parse chart, where a category edge is defined as a start node, an end node, and a syntactic category which dominates the span. The probability of an edge is the product of the inside probability times the outside probability divided by the total probability assigned to the word-lattice by the parsing model (we can ignore the denominator when comparing parses for the same word-lattice). During bottom-up parsing, we compute an approximation of the inside probability but are lacking the structure needed to compute the outside probability. The FOM we define here is the product of the approximated inside probability and an estimate of the outside probability.

Caraballo and Charniak suggest using a bitag model to approximate the outside probability[8], where the tags are the part-of-speech tags. We also make use of the bitag probability but must consider the many partial strings preceding and succeeding a particular edge. We describe these modifications later in this section.

In each iteration of the chart-parsing algorithm, we retrieve a category edge from the agenda. Best-first chart parsing replaces the FIFO queue agenda with a priority queue. The priority of an edge is simply the FOM of the edge.

4.1. Binarization and Markovization

The chart-parsing algorithm is typically defined in terms of active and passive edges. By requiring the grammar to be binary (each production expands to at most two constituents), we need only work with passive edges, simplifying the algorithm. In order to create a PCFG based on a binary grammar, we transform trees in the training set in the following manner. For any tree with greater than two children: A→B C D becomes A→B–C D, where B–C is a new category and a new subtree B–C→B C is inserted in the tree. The binarization shown in Fig. 2 is known as right-factoring, or left-binarization. This results in a left-branching tree. Similarly, we could left-factor the tree. Note that a PCFG trained on the binarized trees assigns the same probability mass to original tree as the binarized tree.

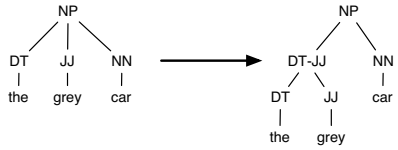

Fig. 2. Left binarization

Another technique that was used in [9] is Markovization. We will describe Markovization as another tree transformation, but it should be clear that Markovization, unlike binarization, does alter the probability model. Markovization is a form of smoothing in that the Markov grammar assigns probability to strings that the old grammar did not. The Markov grammar is also smaller than the original.

To Markovize a tree, we first binarize the tree. For each

binarized category that is made up of more than two subcategories (e.g. the category 'NP–VP–PP'), we retain only a subset of the categories. In this work we chose the most extreme variety of Markovization and removed all internal categories. This means that a category A–B–C–D–E becomes A–E in the Markovized trees. Alternatively, we could substitute the internal nodes with some more meaningful syntactic information, such as the category of the head constituent.

4.2. Viterbi Approximation

In order to compute the probability of an edge, we compute the inside and outside probabilities for that edge. The traditional inside (and outside) probabilities effectively sum over the probability assigned to all possible tree structures that include the category edge. An alternative that was used in [9], is to compute the Viterbi probability of an edge. The Viterbi probability of an edge is the max probability of any single tree in which that edge occurs. This greatly reduces the complexity of the parsing algorithm without a noticeable loss in accuracy[9].

In the remainder of the paper, we choose to extend the use of the Viterbi approximation. The primary motivation, described immediately above, is to simplify and speed-up the parsing algorithm. When computing the estimate of the outside probability, we also use the Viterbi estimate in an attempt to be compatible with the Viterbi inside probabilities.

4.3. Computing the FOM

We now describe the "Figure of Merit" used by our parser which directs the search through the word-lattice and parses simultaneously. The recursion used to compute Viterbi inside probability is:

$$\beta^*(N^i_{j,l}) = \max_{r,s,k} \beta^*(N^r_{j,k})\beta^*(N^s_{k,l})p(N^i \rightarrow N^r N^s) \quad (2)$$

where N^i is the i^{th} category in the grammar, $N^i_{j,l}$ is an N^i category edge spanning from node j to node l, and $p(N^i \rightarrow N^r N^s)$ is the PCFG probability of the production. In the case of preterminal edges, the Viterbi inside probability is defined as:

$$\beta^*(N^i_{j,l}) = \max_{w_{j,l}} p(N^i \rightarrow w_{j,l})s(j,l) \quad (3)$$

where $w_{j,l}$ is a the word label on an arc between node j and node l, and $s(j,l)$ is the scaled lattice score associated with that arc, $s(j,l) = exp(\alpha \ln(a(j,l)))$, where α scales the raw acoustic score, $a(j,l)$, to account for incorrect independence assumptions in the acoustic model. The probability $p(N^i \rightarrow w_{j,l})$ is estimated by a relative frequency estimator. It is necessary to smooth this distribution due to unknown words (words that were not seen in the training data). We use Laplace smoothing where the number of unknown vocabulary items is determined from the vocabulary used for the speech-recognition task (a closed vocabulary).

In order to compute the Viterbi outside probability we follow Caraballo and Charniak[8] in using a combination of a bitag model and a tag-category boundary model. We define the recursion used to compute the Viterbi outside estimate as:

$$\alpha^*(N^i_{j,l}) = \max_{q,r,h,m} f^*(T^q_{h,j})p(N^i|T^q)p(T^r|N^i)b^*(T^r_{l,m}) \quad (4)$$

where T^i is the i^{th} part-of-speech tag, and $T^i_{a,b}$ is a part-of-speech arc extending from node a to node b. The functions $f^*(T^q_{h,j})$ and $b^*(T^r_{l,m})$ are the HMM Viterbi forward and backward bitag probabilities (described below). The boundary probabilities $p(N^i|T^q)$ and $p(T^r|N^i)$ are estimated using relative frequency estimators.

The bitag model for the lattice is created by adding new part-of-speech arcs to the graph. For each word arc we create part-of-speech arcs for each part-of-speech to which the word belongs. Since there may be multiple word arcs for any pair of nodes, we make sure not to create duplicate part-of-speech arcs. In this new graph (an acoustic/bitag HMM), we compute the Viterbi forward and Viterbi backward values as follows:

$$f^*(T^q_{h,j}) = \max_{p,g,w_{h,j}} f^*(T^p_{g,h})p(T^q|T^p)p(w_{h,j}|T^q)s(h,j) \quad (5)$$

$$b^*(T^r_{l,m}) = \max_{s,n,w_{l,m}} s(l,m)p(w_{l,m}|T^r)p(T^s|T^r)b^*(T^s_{m,n}) \quad (6)$$

The f^* and b^* probabilities represent the bitag probability of the path that maximizes both the bitag model and the acoustic model from the word-lattice.

Finally, the "Figure of Merit" for an edge $N^i_{j,k}$ is defined as the product of the Viterbi outside probability and the Viterbi inside probability.

$$FOM(N^i_{j,k}) = \alpha^*(N^i_{j,k})\beta^*(N^i_{j,k})\eta^{C(j,k)} \quad (7)$$

As in [9] we add a correction term, η to correct for the difference between the PCFG model and the bitag model. $C(j,k)$ is the number of words on the path associated with the max inside probability for edge $N^i_{j,k}$.

Given the FOM, the parser works in the normal manner of best-first parsing. Edges with a high FOM are added to the chart. Then, neighboring edges are combined with the new edges; the resulting new edges are scored (an FOM is computed for them) and inserted into the agenda. Parsing continues until a root edge (an edge labeled with the root category spanning the entire lattice) is popped off the agenda.

4.4. Connecting to the second-stage parser

In the first-stage PCFG parsing, our goal is to generate a set of candidate edges which we then pass on to a more sophisticated parser. In this paper we use a modified version of the Charniak parser which accepts a packed forest representation of a parse chart (local trees).

Due to the complexity of the language model in the Charniak parser, the number of local trees that can be processed is limited; this is largely due to memory constraints. Therefore, we must be judicious in choosing which local trees are passed on to the second stage. In the first-stage parsing, we use an estimate of the edge probability to choose which edges are added to the chart. It is possible for this estimate to direct the parser to populate the chart with some edges that are not in the most likely PCFG parse. Additionally, we are not only looking for the edges that are in the most likely PCFG parse, but a rich set of edges that are likely according to the PCFG model. We accomplish this by over-parsing; continuing to parse after we have found the first complete parse for the word-lattice. As in previous work, the parser continues generating edges until it has popped many more edges off the agenda than it took to reach the first parse[9, 12]. The threshold for the number of popped edges is dynamically defined to be a multiple of the number of agenda pops needed to reach the first parse; typically this multiplier is between 10 and 100.

Having populated the syntactic category chart, we compute the exact Viterbi outside probabilities for each edge in the given chart. Computing the Viterbi outside is much like computing the traditional outside probability[13], replacing the sum operators with max operators. Following, we compute the probability of each edge occurring in a parse, given the current chart. We now have a probability for each edge which we use to prune edges that are less likely.

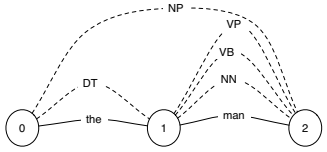

Fig. 3. A section of the syntactic category chart

Local trees are generated by combining category edges with children that span the same categories. For example, the NP edge in Fig. 3 will expand to: ($NP_{0,2} \rightarrow DT_{0,1}$ $NN_{1,2}$), ($NP_{0,2} \rightarrow DT_{0,1}$ $VB_{1,2}$), etc. The probability for these local trees are computed and pruned in the same manner as the chart edges. Finally, we pass these local trees to the the second-stage parser. The second-stage parser returns the most likely parse according to its language model. The most likely string is the string associated with the most likely parse.

5. EXPERIMENTS

We test the performance of our parser/language model on the word-lattices from the NIST HUB–1 evaluation task from 1993. This corpus was chosen so that we can make a direct comparison between our work and previous work on syntactic language modeling[4, 3, 14]. The HUB–1 lattices are a set of 213 word-lattices in HTK SLF (described in Section 3). These lattices are derived from a set of utterances produced by professional readers reading excerpts from the Wall Street Journal newspaper.

We trained the PCFG for our first-stage parser on a modified version of sections 2–24 of the Penn Treebank Wall Street Journal. In the modified version, newspaper text is converted to speech-like text; punctuation is removed, numbers are converted to words, etc. Brian Roark at AT&T provided a utility to perform this conversion. A detailed description of the conversion process can be found in [15, 4].

As mentioned above, the second-stage parser we use is the Charniak parser[1]. We obtained a version of the Charniak parser (from Charniak) which accepts a packed forest representation of the parse chart. This parser was trained on a version of the BLLIP99 corpus that was transformed to speech-like text in the same manner as described above. The sentences used for the HUB–1 speech evaluation originated from Wall Street Journal articles which are contained in the BLLIP99 corpus. Prior to training the parser, we removed all occurrences of the HUB–1 strings from the BLLIP99 corpus.

Model	WER
40m-word trigram	13.7
Chelba02 SLM (n–best list)	12.3
Roark01 (n–best list)	12.7
Charniak Parser (n–best list)	11.9
Lattice Parser (n–best lattices)	13.0
Lattice Parser (acoustic lattices)	15.5

Table 1. Word Error Rate (WER) for various language models on the HUB–1 lattices.

In Table 1 we report the results of various language models. All of these models, with the exclusion of the trigram, use syntactic/parsing information to guide the search through the lattice. The Chelba02[5] and Roark01[3] models are left-to-right processing models. These models work with a list of n candidate strings extracted from the word-lattice(n–best lists). Each string is parsed and the n–best list is rescored according to the probability assigned to each parsed string. The n–best list for these two models was generated using a 40 million word trigram.

The results presented for parsing the n–best lists with the Charniak parser have not previously been reported[1]. Much like Chelba02 and Roark01, we parsed the n–best strings individually. To do this we created a lattice for each of the

[1]We have verified these results with Eugene Charniak.

strings (more like a chain than a lattice). We ran our PCFG parsed with 10–times over-parsing and passed the generated local-trees to the Charniak second-stage parser. We selected the string to which the Charniak parser assigned the highest probability.

The results of our lattice parser are reported in the last two rows of Table 1. These experiments are identical except for the word-lattices. In both cases we used 70–times over-parsing for the PCFG parser. The n–best lattices are lattices created by taking the union of the n–best strings. This experiment is similar to the n–best list experiments, except that we parse the n–best strings concurrently. The final result reported is for parsing on the original word-lattices without the trigram pruning. It appears that the first-stage parser is pruning correct word arcs from the lattice. This is likely due to to the outside probability model being based on bitag probabilities.

Model	Chart Edges
Charniak Parser (n–best list)	2,950,730
Lattice Parser (n–best lattices)	880,976

Table 2. Number of edges passed after first-stage pruning.

Table 2 shows the number of edges remaining in the syntactic category chart after first-stage pruning. Even with 70–times over-parsing, the number of edges used to generate local-trees for all 213 n–best lattices is only around 880k. On the other hand, if we add the count of edges that passed the pruning stage when parsing the n–best string individually, we consider close to three million edges; many of these three million edges are identical for strings generated from the same lattice. Although the current word-lattice parser takes a dramatic performance hit when parsing lattices verses the individual sentences, it is doing far less work than the individual string parsing effort.

6. CONCLUSION

We have presented an efficient best-first bottom-up word-lattice parser used for language modeling. The "Figure of Merit" used to drive this parser combines the search for a path through the lattice and the search for a good parse. Although this initial version of the word-lattice parser does not perform better than the current n–best list syntactic language-models, we have shown that by parsing the word-lattice in one pass, we greatly reduce the amount of work required. In future work, we will augment the PCFG model and the estimate the outside probability model to improve the quality of the edges generated from the first-stage parser.

7. REFERENCES

[1] Eugene Charniak, "Immediate-head parsing for language models," in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 2001.

[2] Michael Collins, "Three generative, lexicalised models for statistical parsing," in *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, 1997, pp. 16–23.

[3] Brian Roark, "Probabilistic top-down parsing and language modeling," *Computational Linguistics*, vol. 27, no. 3, pp. 249–276, 2001.

[4] Ciprian Chelba and Frederick Jelinek, "Structured language modeling," *Computer Speech and Language*, vol. 14, no. 4, pp. 283–332, 2000.

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

[6] Hans Weber, "Time synchronous chart parsing of speech integrating unification grammars with statistics," in *Proceedings of the eighth Twente Workshop on Language Technologies*, December 1994.

[7] J. Chappelier, M. Rajman, R. Aragues, and A. Rozenknop, "Lattice parsing for speech recognition," in *Proceedings of 6me confrence sur le Traitement Automatique du Langage Naturel (TALN'99)*, 1999, pp. 95–104.

[8] Sharon Caraballo and Eugene Charniak, "New figures of merit for best-first probabilistic chart parsing," *Computational Linguistics*, vol. 24, no. 2, pp. 275–298, June 1998.

[9] Sharon Goldwater, Eugene Charniak, and Mark Johnson, "Best-first edge-based chart parsing," in *6th Annual Workshop for Very Large Corpora*, 1998, pp. 127–133.

[10] Frederick Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, 1997.

[11] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16, no. 1, pp. 69–88, 2002.

[12] Don Blaheta and Eugene Charniak, "Automatic compensation for parser figure-of-merit flaws," in *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, 1999, pp. 513–518.

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

[14] Eugene Charniak, "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.

[15] Brian Edward Roark, *Robust Probabilistic Predictive Syntactic Processing: Motivations, Models, and Applications*, Ph.D. thesis, Brown University, May 2001.