Head-Driven Parsing for Word Lattices

Christopher Collins
Department of Computer Science
University of Toronto
Toronto, ON, Canada
ccollins@cs.utoronto.ca

Bob Carpenter
Alias I, Inc.
Brooklyn, NY, USA
carp@alias-i.com

Gerald Penn
Department of Computer Science
University of Toronto
Toronto, ON, Canada
gpenn@cs.utoronto.ca

Abstract

We present the first application of the head-driven statistical parsing model of Collins (1999) as a simultaneous language model and parser for large-vocabulary speech recognition. The model is adapted to an online left to right chart-parser for word lattices, integrating acoustic, *n*-gram, and parser probabilities. The parser uses structural and lexical dependencies not considered by *n*-gram models, conditioning recognition on more linguistically-grounded relationships. Experiments on the Wall Street Journal treebank and lattice corpora show word error rates competitive with the standard *n*-gram language model while extracting additional structural information useful for speech understanding.

1 Introduction

The question of how to integrate high-level knowledge representations of language with automatic speech recognition (ASR) is becoming more important as (1) speech recognition technology matures, (2) the rate of improvement of recognition accuracy decreases, and (3) the need for additional information (beyond simple transcriptions) becomes evident. Most of the currently best ASR systems use an *n*-gram language model of the type pioneered by Bahl et al. (1983). Recently, research has begun to show progress towards application of new and better models of spoken language (Hall and Johnson, 2003; Roark, 2001; Chelba and Jelinek, 2000).

Our goal is integration of head-driven lexicalized parsing with acoustic and *n*-gram models for speech recognition, extracting high-level structure from speech, while simultaneously selecting the best path in a word lattice. Parse trees generated by this process will be useful for automated speech understanding, such as in higher semantic parsing (Ng and Zelle, 1997).

Collins (1999) presents three lexicalized models which consider long-distance dependencies within a sentence. Grammar productions are conditioned on headwords. The conditioning context is thus more focused than that of a large *n*-gram covering the same span, so the sparse data problems arising from the sheer size of the parameter space are less pressing. However, sparse data problems arising from the limited availability of annotated training data become a problem.

We test the head-driven statistical lattice parser with word lattices from the NIST HUB-1 corpus, which has been used by others in related work (Hall and Johnson, 2003; Roark, 2001; Chelba and Jelinek, 2000). Parse accuracy and word error rates are reported. We present an analysis of the effects of pruning and heuristic search on efficiency and accuracy and note several simplifying assumptions common to other reported experiments in this area, which present challenges for scaling up to real-world applications.

This work shows the importance of careful algorithm and data structure design and choice of dynamic programming constraints to the efficiency and accuracy of a head-driven probabilistic parser for speech. We find that the parsing model of Collins (1999) can be successfully adapted as a language model for speech recognition.

In the following section, we present a review of recent works in high-level language modelling for speech recognition. We describe the word lattice parser developed in this work in Section 3. Section 4 is a description of current evaluation metrics, and suggestions for new metrics. Experiments on strings and word lattices are reported in Section 5, and conclusions and opportunities for future work are outlined in Section 6.

2 Previous Work

The largest improvements in word error rate (WER) have been seen with *n*-best list rescoring. The best *n* hypotheses of a simple speech recognizer are processed by a more sophisticated language model and re-ranked. This method is algorithmically simpler than parsing lattices, as one can use a model developed for strings, which need not operate strictly left to right. However, we confirm the observation of (Ravishankar, 1997; Hall and Johnson, 2003) that parsing word lattices saves computation time by only parsing common substrings once.

Chelba (2000) reports WER reduction by rescoring word lattices with scores of a structured language model (Chelba and Jelinek, 2000), interpolated with trigram scores. Word predictions of the structured language model are conditioned on the two previous phrasal heads not yet contained in a bigger constituent. This is a computationally intensive process, as the dependencies considered can be of arbitrarily long distances. All possible sentence prefixes are considered at each extension step.

Roark (2001) reports on the use of a lexicalized probabilistic top-down parser for word lattices, evaluated both on parse accuracy and WER. Our work is different from Roark (2001) in that we use a bottom-up parsing algorithm with dynamic programming based on the parsing model II of Collins (1999).

Bottom-up chart parsing, through various forms of extensions to the CKY algorithm, has been applied to word lattices for speech recognition (Hall and Johnson, 2003; Chappelier and Rajman, 1998; Chelba and Jelinek, 2000). Full acoustic and *n*-best lattices filtered by trigram scores have been parsed. Hall and Johnson (2003) use a best-first probabilistic context free grammar (PCFG) to parse the input lattice, pruning to a set of *local trees* (candidate partial parse trees), which are then passed to a version of the parser of Charniak (2001) for more refined parsing. Unlike (Roark, 2001; Chelba, 2000), Hall and Johnson (2003) achieve improvement in WER over the trigram model without interpolating its lattice parser probabilities directly with trigram probabilities.

3 Word Lattice Parser

Parsing models based on headword dependency relationships have been reported, such as the structured language model of Chelba and Jelinek (2000). These models use much less conditioning information than the parsing models of Collins (1999), and do not provide Penn Treebank format parse trees as output. In this section we outline the adaptation of the Collins (1999) parsing model to word lattices.

The intended action of the parser is illustrated in Figure 1, which shows parse trees built directly upon a word lattice.

3.1 Parameterization

The parameterization of model II of Collins (1999) is used in our word lattice parser. Parameters are

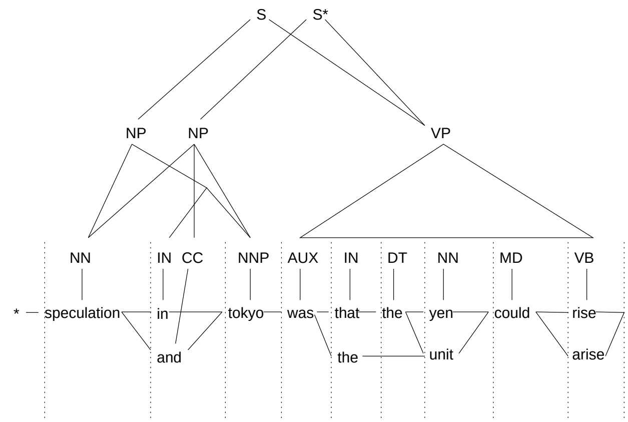

Figure 1: Example of a partially-parsed word lattice. Different paths through the lattice are simultaneously parsed. The example shows two final parses, one of low probability (S^*) and one of high probability (S).

maximum likelihood estimates of conditional probabilities — the probability of some event of interest (*e.g.*, a left-modifier attachment) given a context (*e.g.*, parent non-terminal, distance, headword). One notable difference between the word lattice parser and the original implementation of Collins (1999) is the handling of part-of-speech (POS) tagging of unknown words (words seen fewer than 5 times in training). The conditioning context of the parsing model parameters includes POS tagging. Collins (1999) falls back to the POS tagging of Ratnaparkhi (1996) for words seen fewer than 5 times in the training corpus. As the tagger of Ratnaparkhi (1996) cannot tag a word lattice, we cannot back off to this tagging. We rely on the tag assigned by the parsing model in all cases.

Edges created by the bottom-up parsing are assigned a score which is the product of the inside and outside probabilities of the Collins (1999) model.

3.2 Parsing Algorithm

The algorithm is a variation of probabilistic online, bottom-up, left-to-right Cocke-Kasami-Younger parsing similar to Chappelier and Rajman (1998).

Our parser produces trees (bottom-up) in a right-branching manner, using unary extension and binary adjunction. Starting with a proposed headword, left modifiers are added first using right-branching, then right modifiers using left-branching.

Word lattice edges are iteratively added to the agenda. Complete closure is carried out, and the next word edge is added to the agenda. This process is repeated until all word edges are read from the lattice, and at least one complete parse is found.

Edges are each assigned a score, used to rank parse candidates. For parsing of strings, the score for a chart edge is the product of the scores of any child edges and the score for the creation of the new edge, as given by the model parameters. This score, defined solely by the parsing model, will be referred to as the *parser score*. The total score for chart edges for the lattice parsing task is a combination of the parser score, an acoustic model score, and a trigram model score. Scaling factors follow those of (Chelba and Jelinek, 2000; Roark, 2001).

3.3 Smoothing and Pruning

The parameter estimation techniques (smoothing and back-off) of Collins (1999) are reimplemented.

Additional techniques are required to prune the search space of possible parses, due to the complexity of the parsing algorithm and the size of the word lattices. The main technique we employ is a variation of the beam search of Collins (1999) to restrict the chart size by excluding low probability edges. The total score (combined acoustic and language model scores) of candidate edges are compared against edge with the same span and category. Proposed edges with score outside the beam are not added to the chart. The drawback to this process is that we can no longer guarantee that a model-optimal solution will be found. In practice, these heuristics have a negative effect on parse accuracy, but the amount of pruning can be tuned to balance relative time and space savings against precision and recall degradation (Collins, 1999). Collins (1999) uses a fixed size beam (10,000). We experiment with several variable beam ($\hat{b}$) sizes, where the beam is some function of a base beam (b) and the edge *width* (the number of terminals dominated by an edge). The base beam starts at a low beam size and increases iteratively by a specified increment if no parse is found. This allows parsing to operate quickly (with a minimal number of edges added to the chart). However, if many iterations are required to obtain a parse, the utility of starting with a low beam and iterating becomes questionable (Goodman, 1997). The base beam is limited to control the increase in the chart size. The selection of the base beam, beam increment, and variable beam function is governed by the familiar speed/accuracy trade-off.[1] The variable beam function found to allow fast convergence with minimal loss of accuracy is:

$$\hat{b} = \frac{b}{\log((w+2)/2)} \tag{1}$$

[1]Details of the optimization can be found in Collins (2004).

Charniak et al. (1998) introduce *overparsing* as a technique to improve parse accuracy by continuing parsing after the first complete parse tree is found. The technique is employed by Hall and Johnson (2003) to ensure that early stages of parsing do not strongly bias later stages. We adapt this idea to a single stage process. Due to the restrictions of beam search and thresholds, the first parse found by the model may not be the model optimal parse (*i.e.*, we cannot guarantee best-first search). We therefore employ a form of overparsing — once a complete parse tree is found, we further extend the base beam by the beam increment and parse again. We continue this process as long as extending the beam results in an improved best parse score.

4 Expanding the Measures of Success

Given the task of simply generating a transcription of speech, WER is a useful and direct way to measure language model quality for ASR. WER is the count of incorrect words in hypothesis $\hat{W}$ per word in the true string W. For measurement, we must assume prior knowledge of W and the best *alignment* of the reference and hypothesis strings.[2] Errors are categorized as insertions, deletions, or substitutions.

$$\text{Word Error Rate} = 100\frac{\text{Insertions}+\text{Substitutions}+\text{Deletions}}{\text{Total Words in Correct Transcript}} \tag{2}$$

It is important to note that most models — Mangu et al. (2000) is an innovative exception — minimize *sentence error*. Sentence error rate is the percentage of sentences for which the proposed utterance has at least one error. Models (such as ours) which optimize prediction of test sentences W_t, generated by the source, minimize the sentence error. Thus even though WER is useful practically, it is formally not the appropriate measure for the commonly used language models. Unfortunately, as a practical measure, sentence error rate is not as useful — it is not as fine-grained as WER.

Perplexity is another measure of language model quality, measurable independent of ASR performance (Jelinek, 1997). Perplexity is related to the entropy of the source model which the language model attempts to estimate.

These measures, while informative, do not capture success of extraction of high-level information from speech. Task-specific measures should be used in tandem with extensional measures such as perplexity and WER. Roark (2002), when reviewing

[2]SCLITE (`http://www.nist.gov/speech/tools/`) by NIST is the most commonly used alignment tool.

parsing for speech recognition, discusses a modelling trade-off between producing parse trees and producing strings. Most models are evaluated either with measures of success for parsing or for word recognition, but rarely both. Parsing models are difficult to implement as word-predictive language models due to their complexity. Generative random sampling is equally challenging, so the parsing correlate of perplexity is not easy to measure. Traditional (*i.e.*, *n*-gram) language models do not produce parse trees, so parsing metrics are not useful. However, Roark (2001) argues for using parsing metrics, such as labelled precision and recall,[3] along with WER, for parsing applications in ASR. Weighted WER (Weber et al., 1997) is also a useful measurement, as the most often ill-recognized words are short, closed-class words, which are not as important to speech understanding as phrasal head words. We will adopt the testing strategy of Roark (2001), but find that measurement of parse accuracy and WER on the same data set is not possible given currently available corpora. Use of weighted WER and development of methods to simultaneously measure WER and parse accuracy remain a topic for future research.

5 Experiments

The word lattice parser was evaluated with several metrics — WER, labelled precision and recall, crossing brackets, and time and space resource usage. Following Roark (2001), we conducted evaluations using two experimental sets — strings and word lattices. We optimized settings (thresholds, variable beam function, base beam value) for parsing using development test data consisting of strings for which we have annotated parse trees.

The parsing accuracy for parsing word lattices was not directly evaluated as we did not have annotated parse trees for comparison. Furthermore, standard parsing measures such as labelled precision and recall are not directly applicable in cases where the number of words differs between the proposed parse tree and the gold standard. Results show scores for parsing strings which are lower than the original implementation of Collins (1999). The WER scores for this, the first application of the Collins (1999) model to parsing word lattices, are comparable to other recent work in syntactic language modelling, and better than a simple trigram model trained on the same data.

[3]Parse trees are commonly scored with the PARSEVAL set of metrics (Black et al., 1991).

5.1 Parsing Strings

The lattice parser can parse strings by creating a single-path lattice from the input (all word transitions are assigned an input score of 1.0). The lattice parser was trained on sections 02-21 of the Wall Street Journal portion of the Penn Treebank (Taylor et al., 2003) Development testing was carried out on section 23 in order to select model thresholds and variable beam functions. Final testing was carried out on section 00, and the PARSEVAL measures (Black et al., 1991) were used to evaluate the performance.

The scores for our experiments are lower than the scores of the original implementation of model II (Collins, 1999). This difference is likely due in part to differences in POS tagging. Tag accuracy for our model was 93.2%, whereas for the original implementation of Collins (1999), model II achieved tag accuracy of 96.75%. In addition to different tagging strategies for unknown words, mentioned above, we restrict the tag-set considered by the parser for each word to those suggested by a simple first-stage tagger.[4] By reducing the tag-set considered by the parsing model, we reduce the search space and increase the speed. However, the simple tagger used to narrow the search also introduces tagging error.

The utility of the overparsing extension can be seen in Table 1. Each of the PARSEVAL measures improves when overparsing is used.

5.2 Parsing Lattices

The success of the parsing model as a language model for speech recognition was measured both by parsing accuracy (parsing strings with annotated reference parses), and by WER. WER is measured by parsing word lattices and comparing the sentence yield of the highest scoring parse tree to the reference transcription (using NIST SCLITE for alignment and error calculation).[5] We assume the parsing performance achieved by parsing strings carries over approximately to parsing word lattices.

Two different corpora were used in training the parsing model on word lattices:

- sections 02-21 of the WSJ Penn Treebank (the same sections as used to train the model for parsing strings) [1 million words]

[4]The original implementation (Collins, 1999) of this model considered all tags for all words.

[5]To properly model language using a parser, one should sum parse tree scores for each sentence hypothesis, and choose the sentence with the best sum of parse tree scores. We choose the yield of the parse tree with the highest score. Summation is too computationally expensive given the model — we do not even generate all possible parse trees, but instead restrict generation using dynamic programming.

Exp.	OP	LP (%)	LR (%)	CB	0 CB (%)	≤2 CB (%)
Ref	N	88.7	89.0	0.95	65.7	85.6
1	N	79.4	80.6	1.89	46.2	74.5
2	Y	80.8	81.4	1.70	44.3	80.4

Table 1: Results for parsing section 0 ($\leq$ 40 words) of the WSJ Penn Treebank: OP = overparsing, LP/LR = labelled precision/recall. CB is the average number of crossing brackets per sentence. 0 CB, $\leq$ 2 CB are the percentage of sentences with 0 or $\leq$ 2 crossing brackets respectively. *Ref* is model II of (Collins, 1999).

- section "1987" of the BLLIP corpus (Charniak et al., 1999) [20 million words]

The BLLIP corpus is a collection of Penn Treebank-style parses of the three-year (1987-1989) Wall Street Journal collection from the ACL/DCI corpus (approximately 30 million words).[6] The parses were automatically produced by the parser of Charniak (2001). As the memory usage of our model corresponds directly to the amount of training data used, we were restricted by available memory to use only one section (1987) of the total corpus. Using the BLLIP corpus, we expected to get lower quality parse results due to the higher parse error of the corpus, when compared to the manually annotated Penn Treebank. The WER was expected to improve, as the BLLIP corpus has much greater lexical coverage.

The training corpora were modified using a utility by Brian Roark to convert newspaper text to speech-like text, before being used as training input to the model. Specifically, all numbers were converted to words (`60` $\rightarrow$ `sixty`) and all punctuation was removed.

We tested the performance of our parser on the word lattices from the NIST HUB-1 evaluation task of 1993. The lattices are derived from a set of utterances produced from Wall Street Journal text — the same domain as the Penn Treebank and the BLLIP training data. The word lattices were previously pruned to the 50-best paths by Brian Roark, using the A* decoding of Chelba (2000). The word lattices of the HUB-1 corpus are directed acyclic graphs in the HTK Standard Lattice Format (SLF), consisting of a set of vertices and a set of edges. Vertices, or nodes, are defined by a time-stamp and labelled with a word. The set of labelled, weighted edges, represents the word utterances. A word w is hypothesized over edge e if e ends at a vertex v labelled w. Edges are associated with transition probabilities and are labelled with an acoustic score and a language model score. The lattices of the HUB-1 corpus are annotated with trigram scores trained using a 20 thousand word vocabulary and 40 million word training sample. The word lattices have a unique start and end point, and each complete path through a lattice represents an utterance hypothesis. As the parser operates in a left-to-right manner, and closure is performed at each node, the input lattice edges must be processed in topological order. Input lattices were sorted before parsing. This corpus has been used in other work on syntactic language modelling (Chelba, 2000; Roark, 2001; Hall and Johnson, 2003).

[6]The sentences of the HUB-1 corpus are a subset of those in BLLIP. We removed all HUB-1 sentences from the BLLIP corpus used in training.

The word lattices of the HUB-1 corpus are annotated with an acoustic score, a, and a trigram probability, lm, for each edge. The input edge score stored in the word lattice is:

$$\log(P_{input}) = \alpha \log(a) + \beta \log(lm) \tag{3}$$

where a is the acoustic score and lm is the trigram score stored in the lattice. The total edge weight in the parser is a scaled combination of these scores with the parser score derived with the model parameters:

$$\log(w) = \alpha \log(a) + \beta \log(lm) + s \tag{4}$$

where w is the edge weight, and s is the score assigned by the parameters of the parsing model. We optimized performance on a development subset of test data, yielding $\alpha = 1/16$ and $\beta = 1$.

There is an important difference in the tokenization of the HUB-1 corpus and the Penn Treebank format. Clitics (*i.e.*, `he's, wasn't`) are split from their hosts in the Penn Treebank (*i.e.*, `he 's, was n't`), but not in the word lattices. The Treebank format cannot easily be converted into the lattice format, as often the two parts fall into different parse constituents. We used the lattices modified by Chelba (2000) in dealing with this problem — contracted words are split into two parts and the edge scores redistributed. We followed Hall and Johnson (2003) and used the Treebank tokenization for measuring the WER. The model was tested with and without overparsing.

We see from Table 2 that overparsing has little effect on the WER. The word sequence most easily parsed by the model (*i.e.*, generating the first complete parse tree) is likely also the word sequence found by overparsing. Although overparsing may have little effect on WER, we know from the experiments on strings that overparsing increases parse accuracy. This introduces a speed-accuracy tradeoff: depending on what type of output is required from the model (parse trees or strings), the additional time and resource requirements of overparsing may or may not be warranted.

5.3 Parsing *N*-Best Lattices vs. *N*-Best Lists

The application of the model to 50-best word lattices was compared to rescoring the 50-best paths individually (50-best list parsing). The results are presented in Table 2.

The cumulative number of edges added to the chart per word for *n*-best lists is an order of magnitude larger than for corresponding *n*-best lattices, in all cases. As the WERs are similar, we conclude that parsing *n*-best lists requires more work than parsing *n*-best lattices, for the same result. Therefore, parsing lattices is more efficient. This is because common substrings are only considered once per lattice. The amount of computational savings is dependent on the density of the lattices — for very dense lattices, the equivalent *n*-best list parsing will parse common substrings up to *n* times. In the limit of lowest density, a lattice may have paths without overlap, and the number of edges per word would be the same for the lattice and lists.

5.4 Time and Space Requirements

The algorithms and data structures were designed to minimize parameter lookup times and memory usage by the chart and parameter set (Collins, 2004). To increase parameter lookup speed, all parameter values are calculated for all levels of back-off at training time. By contrast, (Collins, 1999) calculates parameter values by looking up event counts at run-time. The implementation was then optimized using a memory and processor profiler and debugger. Parsing the complete set of HUB-1 lattices (213 sentences, a total of 3,446 words) on average takes approximately 8 hours, on an Intel Pentium 4 (1.6GHz) Linux system, using 1GB memory. Memory requirements for parsing lattices is vastly greater than equivalent parsing of a single sentence, as chart size increases with the number of divergent paths in a lattice. Additional analysis of resource issues can be found in Collins (2004).

5.5 Comparison to Previous Work

The results of our best experiments for lattice- and list-parsing are compared with previous results in Table 3. The oracle WER[7] for the HUB-1 corpus is 3.4%. For the pruned 50-best lattices, the oracle WER is 7.8%. We see that by pruning the lattices using the trigram model, we already introduce additional error. Because of the memory usage and time required for parsing word lattices, we were unable to test our model on the original "acoustic" HUB-1 lattices, and are thus limited by the oracle WER of the 50-best lattices, and the bias introduced by pruning using a trigram model. Where available, we also present comparative scores of the sentence error rate (SER) — the percentage of sentences in the test set for which there was at least one recognition error. Note that due to the small (213 samples) size of the HUB-1 corpus, the differences seen in SER may not be significant.

We see an improvement in WER for our parsing model alone ($\alpha = \beta = 0$) trained on 1 million words of the Penn Treebank compared to a trigram model trained on the same data — the "Treebank Trigram" noted in Table 3. This indicates that the larger context considered by our model allows for performance improvements over the trigram model alone. Further improvement is seen with the combination of acoustic, parsing, and trigram scores ($\alpha = 1/16, \beta = 1$). However, the combination of the parsing model (trained on 1M words) with the lattice trigram (trained on 40M words) resulted in a higher WER than the lattice trigram alone. This indicates that our 1M word training set is not sufficient to permit effective combination with the lattice trigram. When the training of the head-driven parsing model was extended to the BLLIP 1987 corpus (20M words), the combination of models ($\alpha = 1/16, \beta = 1$) achieved additional improvement in WER over the lattice trigram alone.

The current best-performing models, in terms of WER, for the HUB-1 corpus, are the models of Roark (2001), Charniak (2001) (applied to *n*-best lists by Hall and Johnson (2003)), and the SLM of Chelba and Jelinek (2000) (applied to *n*-best lists by Xu et al. (2002)). However, *n*-best list parsing, as seen in our evaluation, requires repeated analysis of common subsequences, a less efficient process than directly parsing the word lattice.

The reported results of (Roark, 2001) and (Chelba, 2000) are for parsing models interpolated with the lattice trigram probabilities. Hall and John-

[7] The WER of the hypothesis which best matches the true utterance, *i.e.*, the lowest WER possible given the hypotheses set.

Training Size	Lattice/List	OP	WER				Number of Edges (per word)
			S	D	I	T	
1M	Lattice	N	10.4	3.3	1.5	15.2	1788
1M	List	N	10.4	3.2	1.4	15.0	10211
1M	Lattice	Y	10.3	3.2	1.4	14.9	2855
1M	List	Y	10.2	3.2	1.4	14.8	16821
20M	Lattice	N	9.0	3.1	1.0	13.1	1735
20M	List	N	9.0	3.1	1.0	13.1	9999
20M	Lattice	Y	9.0	3.1	1.0	13.1	2801
20M	List	Y	9.0	3.3	0.9	13.3	16030

Table 2: Results for parsing HUB-1 n-best word lattices and lists: OP = overparsing, S = substutitions (%), D = deletions (%), I = insertions (%), T = total WER (%). Variable beam function: $\hat{b} = b/log((w+2)/2)$. Training corpora: 1M = Penn Treebank sections 02-21; 20M = BLLIP section 1987.

Model	n-best List/Lattice	Training Size	WER (%)	SER (%)
Oracle (50-best lattice)	Lattice		7.8	
Charniak (2001)	List	40M	11.9	
Xu (2002)	List	20M	12.3	
Roark (2001) (with EM)	List	2M	12.7	
Hall (2003)	Lattice	30M	13.0	
Chelba (2000)	Lattice	20M	13.0	
Current ($\alpha = 1/16, \beta = 1$)	**List**	**20M**	**13.1**	**71.0**
Current ($\alpha = 1/16, \beta = 1$)	**Lattice**	**20M**	**13.1**	**70.4**
Roark (2001) (no EM)	List	1M	13.4	
Lattice Trigram	Lattice	40M	13.7	69.0
Current ($\alpha = 1/16, \beta = 1$)	**List**	**1M**	**14.8**	**74.3**
Current ($\alpha = 1/16, \beta = 1$)	**Lattice**	**1M**	**14.9**	**74.0**
Current ($\alpha = \beta = 0$)	**Lattice**	**1M**	**16.0**	**75.5**
Treebank Trigram	Lattice	1M	16.5	79.8
No language model	Lattice		16.8	84.0

Table 3: Comparison of WER for parsing HUB-1 words lattices with best results of other works. SER = sentence error rate. WER = word error rate. "Speech-like" transformations were applied to all training corpora. Xu (2002) is an implementation of the model of Chelba (2000) for n-best list parsing. Hall (2003) is a lattice-parser related to Charniak (2001).

son (2003) does not use the lattice trigram scores directly. However, as in other works, the lattice trigram is used to prune the acoustic lattice to the 50 best paths. The difference in WER between our parser and those of Charniak (2001) and Roark (2001) applied to word lists may be due in part to the lower PARSEVAL scores of our system. Xu et al. (2002) report inverse correlation between labelled precision/recall and WER. We achieve 73.2/76.5% LP/LR on section 23 of the Penn Treebank, compared to 82.9/82.4% LP/LR of Roark (2001) and 90.1/90.1% LP/LR of Charniak (2000). Another contributing factor to the accuracy of Charniak (2001) is the size of the training set — 20M words larger than that used in this work. The low WER of Roark (2001), a top-down probabilistic parsing model, was achieved by training the model on 1 million words of the Penn Treebank, then performing a single pass of Expectation Maximization (EM) on a further 1.2 million words.

6 Conclusions

In this work we present an adaptation of the parsing model of Collins (1999) for application to ASR. The system was evaluated over two sets of data: strings and word lattices. As PARSEVAL measures are not applicable to word lattices, we measured the parsing accuracy using string input. The resulting scores were lower than that original implementation of the model. Despite this, the model was successful as a language model for speech recognition, as measured by WER and ability to extract high-level information. Here, the system performs better than a simple n-gram model trained on the same data, while simultaneously providing syntactic information in the form of parse trees. WER scores are comparable to related works in this area.

The large size of the parameter set of this parsing model necessarily restricts the size of training data that may be used. In addition, the resource requirements currently present a challenge for scaling up from the relatively sparse word lattices of the NIST HUB-1 corpus (created in a lab setting by professional readers) to lattices created with spontaneous speech in non-ideal conditions. An investigation into the relevant importance of each parameter for the speech recognition task may allow a reduction in the size of the parameter space, with minimal loss of recognition accuracy. A speedup may be achieved, and additional training data could be used. Tuning of parameters using EM has lead to improved WER for other models. We encourage investigation of this technique for lexicalized head-driven lattice parsing.

Acknowledgements

This research was funded in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Advice on training and test data was provided by Keith Hall of Brown University.

References

L. R. Bahl, F. Jelinek, and R. L. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:179–190.

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of Fourth DARPA Speech and Natural Language Workshop*, pages 306–311.

J.-C. Chappelier and M. Rajman. 1998. A practical bottom-up algorithm for on-line parsing with stochastic context-free grammars. Technical Report 98-284, Swiss Federal Institute of Technology, July.

Eugene Charniak, Sharon Goldwater, and Mark Johnson. 1998. Edge-Based Best-First Chart Parsing. In *6th Annual Workshop for Very Large Corpora*, pages 127–133.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 1999. *BLLIP 1987-89 WSJ Corpus Release 1*. Linguistic Data Consortium.

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*, pages 132–129, New Brunswick, U.S.A.

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

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14:283–332.

Ciprian Chelba. 2000. *Exploiting Syntactic Structure for Natural Language Modeling*. Ph.D. thesis, Johns Hopkins University.

Christopher Collins. 2004. *Head-Driven Probabilistic Parsing for Word Lattices*. M.Sc. thesis, University of Toronto.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Joshua Goodman. 1997. Global thresholding and multiple-pass parsing. In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing*.

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

Frederick Jelinek. 1997. *Information Extraction From Speech And Text*. MIT Press.

Lidia Mangu, Eric Brill, and Andreas Stolcke. 2000. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Computer Speech and Language*, 14(4):373–400.

Hwee Tou Ng and John Zelle. 1997. Corpus-based approaches to semantic interpretation in natural language processing. *AI Magazine*, 18:45–54.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing*, May.

Mosur K. Ravishankar. 1997. Some results on search complexity vs accuracy. In *DARPA Speech Recognition Workshop*, pages 104–107, February.

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

Brian Roark. 2002. Markov parsing: Lattice rescoring with a statistical parser. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 287–294.

Ann Taylor, Mitchell Marcus, and Beatrice Santorini, 2003. *The Penn TreeBank: An Overview*, chapter 1. Kluwer, Dordrecht, The Netherlands.

Hans Weber, Jörg Spilker, and Günther Görz. 1997. Parsing n best trees from a word lattice. *Kunstliche Intelligenz*, pages 279–288.

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