Submitted to ICASSP-95

USING A STOCHASTIC CONTEXT-FREE GRAMMAR AS A LANGUAGE MODEL FOR SPEECH RECOGNITION

Daniel Jurafsky, Chuck Wooters, Jonathan Segal, Andreas Stolcke, Eric Fosler, Gary Tajchman, and Nelson Morgan*

International Computer Science Institute
1947 Center Street, Suite 600
Berkeley, CA 94704, USA
& University of California at Berkeley
{jurafsky,wooters,tajchman,jsegal,stolcke,fosler,morgan}@icsi.berkeley.edu

ABSTRACT

This paper describes a number of experiments in adding new grammatical knowledge to the Berkeley Restaurant Project (BeRP), our medium-vocabulary (1300 word), speaker-independent, spontaneous continuous-speech understanding system (Jurafsky *et al.* 1994). We describe an algorithm for using a probabilistic Earley parser and a stochastic context-free grammar (SCFG) to generate word transition probabilities at each frame for a Viterbi decoder. We show that using an SCFG as a language model improves word error rate from 34.6% (bigram) to 29.6% (SCFG), and semantic sentence recognition error from from 39.0% (bigram) to 34.1% (SCFG). In addition, we get a further reduction to 28.8% word error by mixing the bigram and SCFG LMs. We also report on our preliminary results from using discourse-context information in the LM.

1. TIGHT COUPLING

A number of researchers have proposed ways to use natural-language-backend information in the speech recognition process. Moore *et al.* (1989) used a unification-based CFG to generate word transitions for a Viterbi recognizer. Goodine *et al.* (1991) describe a system which uses the CFG-based TINA parser to predict next words for the SUMMIT speech recognizer, Kita & Ward (1991) used a CFG to filter bigram follow-sets for the Sphinx recognizer. Hauenstein & Weber (1994) also used a unification-based CFG to filter bigram follow-sets. In all these cases, the CFG was used to generate or filter the word-transition list, but not to assign probabilities. Goddeau (1992) extended these results by using a probabilistic LR parser to actually produce word-transition probabilities.

Our tight coupling model extends these models to general SCFGs by augmenting a probabilistic version of the Earley algorithm (Stolcke 1993) to compute word transition probabilities from an SCFG.

The system we have augmented, the BeRP system, is a speech understanding system which answers questions about restaurants in the city of Berkeley, California, inspired by earlier consultants like VOYAGER (Zue *et al.* 1991). BeRP consists of a RASTA-PLP feature extractor, a multilayer perceptron (MLP) phonetic probability estimator, a Viterbi decoder, an HMM lexicon, a natural language interpreter which incorporates a stochastic context-free grammar, and a database of restaurants.

*Currently at Dept. of Defense

The SCFG used in BeRP consists of 1389 hand-written context-free rules. The non-terminals in the rules are very specific to the corpus, and hence to the restaurant domain. The rule probabilities are learned from the 4786-sentence BeRP corpus with the EM algorithm. Figure 1 shows a sample of the grammar rules.

0.38 s → ISENTENCES
0.62 ISENTENCES → IWANTTO VP
0.71 VP → EATVERB EATOBJ
0.17 VP → SPENDVERB MONEY
0.11 VP → TRAVELVERB DISTANCE

Figure 1: Sample Grammar Rules

2. USING THE SCFG

We have experimented with a number of ways to use the information provided by the SCFG:

1. Use the SCFG to smooth the bigram grammar, by taking our original corpus, adding a pseudo-corpus generated from the SCFG, and building the bigram with Monte-Carlo sampling on this joint corpus.
2. Use the SCFG to smooth the bigram grammar by generating the *characteristic* bigram for the SCFG in closed form.
3. Use the SCFG directly to provide word transition probabilities on each frame.
4. Use a mixture of the SCFG and bigram probabilities directly to provide word transition probabilities on each frame.

2.1. Using the SCFG to Smooth the Bigram

In the first two methods, we use the SCFG to smooth the bigram grammar, and then use this improved bigram grammar in the recognizer. The first method extends an idea of Zue *et al.* (1991), who used an advanced language model to generate random sentences from which to train a word-pair model. We extended this idea to generation of bigrams by Monte-Carlo sampling, by using our SCFG-based parser in generation mode to generate a pseudo-corpus of 200,000 sentences, adding in our regular BeRP corpus, and then using our standard bigram-building tools on the combined corpus.

In the second method, we have shown (Stolcke & Segal 1994) that it is possible to generate a bigram from a stochastic context-free grammar *directly*, by computing its *characteristic n-gram* in closed form. The method computes the expected bigram counts for strings generated by each of the nonterminals in the grammar by solving a system of linear equations derived from the grammar rule probabilities. We have implemented this algorithm recently, and will use it instead of the Monte Carlo method to generate our future bigrams.

2.2. Using the SCFG directly as the LM

In the third method, we use the SCFG directly as the LM for the recognizer. We begin by abstracting away from probabilities. Consider the problem of using a CFG to produce a follow-set, given a prefix string. For example, if the recognizer passes the string *I want British* to the parser, it will produce the follow words "food", "restaurants", "places", "cuisine", etc. The parser parses the prefix string, and then looks at every non-terminal symbol that the Earley parser is predicting next. For each such non-terminal, we look up its *left-corner* list – the list of terminal symbols which the non-terminal can generate on the left fringe of some parse tree. This list can be computed in advance.

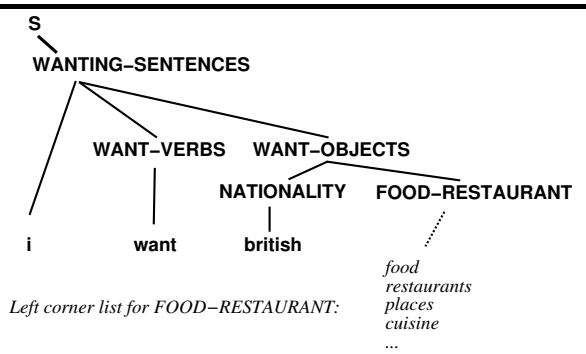

Figure 2: Prefixes and Left-Corner lists

The recognizer needs more than just follow sets, however. In this case, it needs the various probabilities P('food' | 'I want British'), P('restaurants' | 'I want British') etc.; i.e., for each word w_i in the follow set, we need to compute

$$P(w_i|w_1 w_2 \ldots w_{i-1}) \tag{1}$$

To compute these probabilities, we first augment the left corner list to produce the probability that a given non-terminal expands to a terminal. For a given pair of symbols $< X, w >$, where X is a non-terminal and w is a terminal, the left-corner probability is the probability that X generates some string which begins with w. Jelinek & Lafferty (1991) give an algorithm for computing this left-corner probability for every pair of non-terminals and terminals in the grammar with a single matrix-inversion.

If all sentences were unambiguous, this would be sufficient to produce the correct transition probabilities. However, sentences are ambiguous. Because of this, there will be multiple parses for each prefix, and hence we will need to combine the left-corner probabilities for non-terminals from different parses. We can do this by weighting the follow-set for each parse, or derivation, by the probability of the derivation.

$$P(w_i|w_1 \ldots w_{i-1}) = \sum_{d \in \textit{derivations}} P(d)\,P(w_i|w_1 \ldots w_{i-1}, d) \tag{2}$$

Thus the parser must be able to compute *prefix probabilities* for derivations of input strings. For a given parse, the prefix probability is just the product of the probabilities of all the rules used in the parse. In order to compute this probability efficiently, we augment our probabilistic chart parser by annotating each edge of the chart with quantities: a prefix probability and an inside probability. Each edge-creation action computes the inside probability and prefix probability for the new edge from the old edges and the grammar rule probabilities. Readers with interest in the details of this probabilistic Earley computation are referred to Stolcke (1993), which extends the simpler prefix algorithm used in BeRP to deal with left-recursive grammars and unit productions.

We have described how the parser is able to compute follow-set probabilities for each string that is passed to it by the recognizer. We turn now to the tight-coupling interface. For each frame, the decoder must compute word strings to pass to the parser. A bigram-based recognizer would simply look up the bigram transition probability for each word that can end at the current frame. Since an SCFG-based recognizer will use the entire prefix to compute the transition probabilities, the recognizer must perform a *backtrace* to determine the prefix associated with the word. The optimal algorithm would search through the Viterbi array to find the N-best word strings or the equivalent word lattice, and either pass each string to the recognizer or parse the lattice directly. In practice, we use a simple (but we believe poor) approximation to the N-best algorithm, in which at each 10 ms frame, the decoder finds the 10 words most likely to end, and for each performs a single backtrace to find 10 strings. Each of these is passed to the parser, which computes a probability vector over the follow-set words. The recognizer then uses each of these probabilities as the transition probability from the word ending at frame f to each of the words which the follow-set vector gives a non-zero probability of starting at frame $f + 1$. If a word is included in the follow-set of more than one backtrace, we pick the maximum probability (Viterbi) backtrace.

If the parser fails at any point in parsing a backtrace, it backs off to the bigram grammar to compute word-transition probabilities for the remainder of the sentence. As Figure 3 shows, this backoff is quite rare, only happening for a very small number of the sentences (mostly the very long sentences). Thus even sentences whose correct transcription falls outside the CFG are usually forced into the nearest CFG-grammatical string.

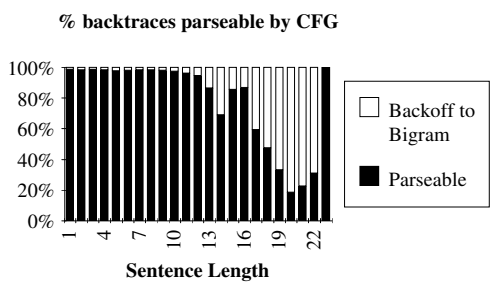

Figure 3: Percentage of backtraces covered by the SCFG

One of the most challenging design aspects of this algorithm

was achieving reasonable time-performance, since the decoder requires word-transition probabilities after every 10 ms frame, requiring on average 2400 calls to the parser per sentence. Despite this large number, our prototype tightly-coupled recognizer runs just 36% slower than our non-tightly-coupled recognizer using bigram probabilities. To achieve this speed, we optimized the algorithm extensively by using efficient indexing in the grammar and the chart, making use of shared substring information for the prefix computation, and adding a cache between the recognizer and the parser to avoid reparsing repeated backtraces.

2.3. Mixing SCFG and SCFG-bigram

The final way to use SCFG information relies on the intuition that the SCFG and the SCFG-smoothed-bigram offer complimentary sources of knowledge about grammar. Where the SCFG is best at modeling long-distance dependencies and hierarchical structure, the SCFG-bigram is best at local and lexical dependencies. Our idea is to mix the two models on a frame-by-frame basis. We have experimented with two versions of this mixing. In one, we weight the models equally:

$$P(w_i|\textit{prefix}) = 0.5P(w_i|\textit{prefix,SCFG}) + 0.5P(w_i|\textit{prefix,Bigram}) \quad (3)$$

In the second, we weight each model by how likely it is given the prefix (which we compute using Bayes' rule); this reflects the intuition that we should rely more on the model which demonstrates a better fit with previous input:

$$P(w_i|\textit{prefix}) = P(\textit{SCFG}|\textit{prefix})P(w_i|\textit{prefix,SCFG}) + P(\textit{Bigram}|\textit{prefix})P(w_i|\textit{prefix,Bigram}) \quad (4)$$

3. RESULTS AND CONCLUSIONS

Our tight coupling systems were tested on a test set of 364 sentences, drawn from the same corpus as the 4786-sentence training sentences (see Jurafsky *et al.* (1994) for details on the corpus collection). Table 1 presents our word error results.

	Word Error
Bigram	34.6
SCFG-Smoothed Bigram	29.6
SCFG	29.6
SCFG/SCFG-Bigram Weighted Mixture	29.5
SCFG/SCFG-Bigram Equal Mixture	28.8

Table 1: BeRP Tight Coupling Performance

Note that the SCFG gave a 5.0% improvement in word error over the bigram, significant at the .005 level. The SCFG and the SCFG-smoothed bigram performed equally, and the mixture models were slightly but not significantly better than either SCFG model. One conclusion we can reach is that compiling the SCFG into a bigram preserved most of the useful information. Additionally, the equal-mixture model seemed to do the best, although the difference with the other mixture and SCFG models was not significant – we plan to rerun these experiments on a larger test set. We suspect that the relative success of the equal-mixture over the weighted-mixture model was due to a useful side effect of equal-mixtures which penalizes the bigram model by normalizing it to 0.5 just in those backtraces where the SCFG returns a zero probability. Use of the SCFG also improved the semantic sentence error from 39.0% (bigram) to 34.1% (all the systems incorporating the SCFG), although this difference was not statistically significant with only 364 sentences.

In our most recent language model experiment, we train pragmatic-context-specific bigrams. Because BeRP is a mixed-initiative system, users often respond to questions asked by the system. For each question the system can ask, we build a subcorpus of responses from our training set, and train a bigram (smoothed with responses to other questions). Then during recognition, we switch between these bigrams depending on the system's latest question. In very preliminary experiments with this discourse-based tight coupling, we show a non-significant 2% reduction in word error, but we are extremely optimistic that these results will improve.

Further details of the BeRP system are presented in Wooters (1993) and Jurafsky *et al.* (1994).

Acknowledgments

This work was partially funded by ICSI and an SRI subcontract from ARPA contract MDA904-90-C-5253. Partial funding for the BeRP system development also came from ESPRIT project 6487 (The Wernicke project).

4. REFERENCES

GODDEAU, DAVID. 1992. Using probabilistic shift-reduce parsing in speech recognition systems. In *ICSLP-92*, I.321—324, Banff, Canada.

GOODINE, DAVID, STEPHANIE SENEFF, LYNETTE HIRSCHMAN, & MICHAEL PHILLIPS. 1991. Full integration of speech and language understanding in the MIT spoken language system. In *Proceedings of Eurospeech 91*, 24—26, Genova, Italy.

HAUENSTEIN, ANDREAS, & HANS H. WEBER. 1994. An investigation of tightly coupled time synchronous speech language interfaces using a unification grammar. In *Proceedings of AAAI-94 Workshop on Integration of Natural Language and Speech Processing*, 42–49.

JELINEK, FREDERICK, & JOHN D. LAFFERTY. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics* 17.315–323.

JURAFSKY, DANIEL, CHUCK WOOTERS, GARY TAJCHMAN, JONATHAN SEGAL, ANDREAS STOLCKE, ERIC FOSLER, & NELSON MORGAN. 1994. The Berkeley restaurant project. In *ICSLP-94*, Yokohama, Japan. to appear.

KITA, KENJI, & WAYNE H. WARD. 1991. Incorporating LR parsing into SPHINX. In *IEEE ICASSP-91*, I.269–272.

MOORE, ROBERT, FERNANDO PEREIRA, & HY MURVEIT. 1989. Integrating speech and natural-language processing. In *Proceedings DARPA Speech and Natural Language Workshop*, 243—247.

STOLCKE, ANDREAS. 1993. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. Technical Report TR-93-065, ICSI, Berkeley, CA. To appear in *Computational Linguistics*.

——, & JONATHAN SEGAL. 1994. Precise n-gram probabilities from stochastic context-free grammars. In *Proceedings of the 32nd ACL*, 74–79, Las Cruces, NM.

WOOTERS, CHARLES C., 1993. *Lexical Modeling in a Speaker Independent Speech Understanding System*. Berkeley, CA: University of California dissertation. available as ICSI TR-92-062.

ZUE, VICTOR, JAMES GLASS, DAVID GOODINE, HONG LEUNG, MICHAEL PHILLIPS, JOSEPH POLIFRONI, & STEPHANIE SENEFF. 1991. Integration of speech recognition and natural language processing in the MIT VOYAGER system. In *IEEE ICASSP-91*, I.713–716.