Experience with a Stack Decoder-Based HMM CSR and Back-Off N-Gram Language Models[1]

Douglas B. Paul

Lincoln Laboratory, MIT
Lexington, Ma. 02173

ABSTRACT

Stochastic language models are more useful than non-stochastic models because they contribute more information than a simple acceptance or rejection of a word sequence. Back-off N-gram language models[11] are an effective class of word based stochastic language model. The first part of this paper describes our experiences using the back-off language models in our time-synchronous decoder CSR. A bigram back-off language model was chosen for the language model to be used in the informal ATIS CSR baseline evaluation test[13, 21].

The stack decoder[2, 8, 24] is a promising control structure for a speech understanding system because it can combine constraints from both the acoustic model and a long span language model (such as a natural language processor (NLP)) into a single integrated search[17]. A copy of the Lincoln time-synchronous HMM CSR has been converted to a stack decoder controlled search with stochastic language models. The second part of this paper describes our experiences with our prototype stack decoder CSR using no grammar, the word-pair grammar, and N-gram back-off language models.

N-GRAM BACK-OFF LANGUAGE MODELS

N-gram language models[2, 10] are an attractive method for estimating the probability of the sentence W by successively estimating the probability of the next word in the sentence:

$$p(W) \approx \prod_i p(w_i|w_{i-N}, \ldots, w_{i-1})$$

where N is the order of the model. They are easily computed, highly effective in reducing the perplexity of the recognition task, and the probabilities can be estimated from observed text. They also have the advantage that they are purely data-driven and therefore can be trained on databases which are too large for human inspection. The maximum likelihood (ML) estimate of the conditional probabilities is:

$$p(w_i|w_{i-N}, \ldots, w_{i-1}) = \frac{C(w_{i-N}, \ldots, w_i)}{C(w_{i-N+1}, \ldots, w_i)}$$

where C is the number of times the gram was observed in the training data. However, the number of parameters which must be estimated is V^N where V is the number of words in the vocabulary. This limits the N-gram models to trigram ($N = 3$) or lower order models due to the difficulty in estimating these probabilities from obtainable amounts of training data. There are a number of methods for smoothing the counts or probabilities to ensure non-zero probabilities for unobserved grams and to smooth the estimates of the observed grams[10]. One such method of smoothing is the N-gram back-off models[11]:

$$p(w_i|w_{i-N}, \ldots, w_{i-1}) = \begin{cases} \frac{C(w_{i-N}, \ldots, w_i)}{C(w_{i-N}, \ldots, w_{i-1})} & \text{if } C(w_{i-N}, \ldots, w_i) \geq const \\ \frac{C^*(w_{i-N}, \ldots, w_i)}{C(w_{i-N}, \ldots, w_{i-1})} & \text{if } 0 < C(w_{i-N}, \ldots, w_i) < const \\ back\_off & \text{if } C(w_{i-N}, \ldots, w_i) = 0 \end{cases}$$

where

$$back\_off = \alpha(w_{i-N}, \ldots, w_{i-1}) p(w_i|w_{i-N+1}, \ldots, w_{i-1}),$$

α is a back-off normalization weight, C^* is the Good-Turing corrected[6] estimate of C, and *const* is some constant on the order of five. (The Good-Turing correction generally yields a reduced effective count.) The method can back off recursively until the "zero-gram" is reached. The detailed mathematics of the Good-Turing correction and computing the αs may be found in reference [11] and will not be repeated here. This model is intuitively appealing because it uses the most detailed model when it can and backs off to a less detailed model when it has insufficient training data. In addition, the Good-Turing correction both improves the probability estimates for low probability grams and "frees up" some "probability space" to allow backing off to the next lower order model. In particular, some probability space is obtained from the unigrams for the zero-grams or unknown words, and therefore it can estimate the probability of an unknown word class. However, the method is not without difficulties.

[1]This work was sponsored by the Defense Advanced Research Projects Agency.

In essence, part of the gram space is covered by the N-gram pdf and the remainder is covered by the (back-off) N-1 gram pdf. (Thus the need for the back-off weight—the pdf over all of the gram space must sum to one.) Therefore if the sum of the N-gram pdf for a particular left context is one, no "space" is left for the back-off. Such a case might occur with a word pair such as "San Francisco" if it occurred more than *const* times and "San" did not occur in any other context. The heuristic applied in the Lincoln implementation was to increment the count on the context to break this deadlock.

A similar difficulty occurs if one wishes to assign probabilities to known but unobserved ($C = 0$) words. The heuristic solution applied here is motivated by a comment by Katz[11] to the effect that the probability of an unobserved word is similar to the probability of a word with a count of one. Thus, any known, but unobserved words were artificially given a unigram count of one.

This, then, was how the bigram back-off model for the informal ATIS CSR baseline evaluation test[13, 21] was derived. The basic theory with the above heuristics produced a language model with a non-zero probability for an unknown word class and non-zero probabilities for known but unobserved words. To distribute this language model, a machine-independent text file format was designed for specifying back-off models which simply lists the grams, their probabilities, and, for the lower order grams, their back-off weights.

Text Modeling Performance

The N-gram back-off language models were initially tested and compared to some padded ML[2] (PML) models on several text databases as shown in Table 1. These perplexities show the effectiveness of this method when trained on limited amounts of data. Comparison between the back-off and PML models shows the back-off model never produced a higher perplexity than the PML model, but sometimes produced dramatically lower perplexities than the PML model. In one case (WSJ small, trigram), the back-off model produced more than an order-of-magnitude lower perplexity than did the PML model.

Recognition Performance

Bigram back-off language models were installed in the time-synchronous (TS) decoder[21]. The model order was limited to bigram due to the structure of the TS decoder. (The previous language models, such as the RM word-pair grammar (WPG), were simply yes-no finite state grammars.) Recognition results for the RM database are shown in Table 2. (The results in Table 2 are speaker-dependent (SD) only, but SRI has confirmed several of the relations for SI models[15].) The fair back-off model, compared to the WPG, produced a doubled error rate in spite of its much lower perplexity. To probe this effect, a cheating RM back-off model, which was guaranteed not to assign an excessively low probability to the test data, was generated. (This is demonstrated by the large decrease in the perplexity for only a small increase in the amount of training data.) Its recognition performance was very similar to that of the WPG system. BBN also observed that a classed bigram model of perplexity similar to that of the fair back-off model produced similar results to the WPG[23]. A pattern bigram model generated from the sentence patterns used to generate the RM language[22] showed a perplexity of about 20 and consistently produced lower error rates than the WPG in tests at CMU[12].

Table 2 shows the WPG and pattern bigram models to produce better recognition results and the fair back-off model to produce poorer recognition results than would be expected from the perplexity alone. This occurred for two reasons: (1) the data trained models are full-branching while the pattern trained models can disallow 94% of the transitions without the possibility of error, and (2) the probability estimates of the fair back-off model are "noisy" due to the limited training data (8K sentences, from Table 1). The RM task sentence patterns[22] made liberal use of classes (e.g. *ship-name*) in the patterns and therefore grouping the words into similar classes in the language model is a significant source of additional information and significantly smoothes the probability estimates. (The back-off models were word based and knew nothing about the word classes. One could, of course, generate a classed back-off model.) BBN observed a half bit reduction in the variance of the probability estimates in a class-based model compared to a word-based model[23]. It is relatively simple to manually generate word classes for simple tasks such as RM and ATIS but automatic methods using probabilistic classes [8, 9] are probably required for more realistic natural tasks.

Why then, did the class-based and the word-based bigram language models of the same perplexity exhibit significantly different error rates? A "noisy" model would overestimate some probabilities and underestimate some others. Since the perplexity is a weighted geometric mean and the noisiest components have the least weight, the noise level would generally have only a small effect on the perplexity. In contrast, recognition occurs on a word-by-word level and the noisy probabilities would be expected to occasionally cause recognition errors. In the context of a CSR which otherwise shows a low error rate, these "occasional" additional errors could result in a significant increase in the overall error rate. The class smoothed (less noisy) model, if the classes are appropriately chosen, would produce fewer of these additional errors.

The use of classes or other language model smoothing techniques is another example of a trade-off that occurs throughout the design of speech recognizers—that of matching the complexity of the model to the amount of available training data. A prespecified-class classed model requires less training data because it has fewer parameters, but because it combines the words into groups, can never model as much detail as the word-based models. As the amount of training data is increased, there will be a point beyond which the word-based models will achieve a better perfor-

[2]The PML models are just ML models where each N-gram count is initialized with a small "count." The appropriate bins are then summed to produce the N-1 gram counts and the equation for ML probabilities is used to compute the probability. The method is simple but can yield very poor probability estimates for low probability words.

mance on natural tasks than the classed models. (Of course, the word-based modeling can at best equal the performance of a correctly-classed model if the language is truly classed. In general, one would expect only artificial languages to be truly classed.)

THE STACK DECODER

A copy of the Lincoln time synchronous (TS) decoder HMM CSR[18, 20, 21] has been converted to a stack decoder[2, 8, 24] using the optimal A* search[19]. The current prototype supports most of the features of the current TS recognizer[20, 21]. It uses multiple observation streams, has adaptive background model estimation, optional interword silences, and context dependent phone modeling. The current implementation can be used with or without a language model. The language models are integrated using the CSR-NL interface[17]. Language model modules have been built for word-pair and bigram/trigram back-off language models. Unlike the TS decoder implementation which is limited to outputting the single best sentence theory, the stack decoder implementation can output a top-N sentence theory list.

The current prototype does not yet use tied mixtures[4, 7] (TM). For simplicity, it uses Top-1 observation pruning mode, which is equivalent to a discrete observation recognizer using observation pdfs generated by a TM trainer. (This was done to delay dealing with the issue of caching the mixture sums. The changes required to convert to an inefficient TM system are trivial.) It also does not yet use cross-word phone modeling pending solution of several implementation issues.

The system includes a tree-structured fast match[1, 3] to reduce the computation required by the detailed acoustic match. This fast match uses a beam-pruned TS search of a phonetic tree built from HMM phone models to locate the the possible one word extensions to the current theory (partial sentence). To reduce computation, only the observation pdfs are used—the transition probabilities are ignored. The current fast match reduces the number of possible next words to about 15% of the vocabulary for the SD (RM) task using triphone models.

The system has been tested in no-grammar (NG) mode using SD triphone models on the RM task. In some respects, the stack decoder does a better job than does the TS decoder. It sometimes locates a higher probability path through the recognition network than does the TS decoder with a reasonable pruning threshold. (Unfortunately, these paths usually contain a recognition error.) The fixed pruning threshold of the TS decoder terminates these paths, while the stack decoder continues to extend them. If the pruning threshold of the TS system is increased to a value that would ordinarily be considered excessive, the TS system will also find these paths. The stack decoder automatically finds these paths because it has, in effect, adaptive pruning which does not require any fixed thresholds.

The system has also been tested using WPG and N-gram back-off language models through the CSR-NL interface. The potential search errors due to an interaction between the acoustic and language models described and verified by simulation in [19] have been observed on real speech data. In fact, this search error can be caused by the word insertion penalty alone. (This appears to be rather infrequent—it was forced experimentally by using a relatively large insertion penalty.) Initial checks for the search error, which used the WPG without probabilities, indicated that the interaction was a minor problem. (The WPG without probabilities gave better recognition results than the WPG with $\frac{1}{branching\_factor}$ probabilities in the TS recognizer.)

In contrast to NG and the WPG, the interaction becomes a major problem when one of the stochastic (bigram or trigram back-off) language models is used. A simple, but inefficient solution is to increase the tie-breaking factor in the equation for the stack ordering criterion[19]:

$$StSc_i = \max_t L_i(t) - \hat{\text{lub}}L(t) - \epsilon t$$

where $L_i(t)$ is the likelihood of theory i and $\hat{\text{lub}}L(t)$ is the least-upper-bound-so-far on the theory likelihoods and ϵ is the tie-breaking factor which favors shorter over longer theories. (For the NG case, ϵ need only be a very small number greater than zero.) This is a poor solution because a large value of ϵ will also greatly increase the computation. When a sufficiently large value of ϵ is used, the stack decoder functions as expected with either the trigram or bigram back-off language models. Another possible solution is to run the decoder in top-N mode and select the best sentence after some number of sentences has been output. This approach also significantly increases the computation.

It appears likely that the interaction problem is due to combining two fundamentally different types of score (log-probabilities) together. The HMM observation and transition probabilities (i.e. the acoustic scores) are accumulated as a function of time and the language model probabilities are accumulated as a function of the number of states traversed. The mixture of these dissimilar scores appears to be damaging the estimation of the least upper bound used to perform the A* search[16] in the stack decoder.

On the average, the current prototype stack decoder runs significantly faster than does the TS decoder on a SD no-grammar task. This is probably due to the adaptive-pruning-threshold like behavior of the stack decoder which allows it to pursue the minimum number of theories required to decode the sentences—a small number on the "easy" sentences and a larger number on the "harder" sentences, whereas the TS decoder must always use a worst-case pruning threshold. With the word-pair grammar, the two systems run at similar speeds. The strategies proposed above for combating the interaction problem with the stochastic language models slow the stack decoder sufficiently to cause it to run to significantly slower than the TS decoder. The strategies also increase the number of theories which must be held on the stack.

CONCLUSION

Stochastic language models will be an important component in future speech recognition and understanding systems. The N-gram language models are a class of model which is easily trained from observed data and provides

significant constraints to the recognition process and were therefore chosen for use in the informal ATIS CSR baseline evaluation test. The required number of parameters, however, is too large to be trained from practical amounts of data. Backing off to lower order models to estimate the probability of unobserved N-grams is an effective method for dealing with finite training data. The fact that these models are purely data driven is both an advantage and a disadvantage—they are free from often erroneous human bias, but also cannot incorporate human knowledge. One method of incorporating human knowledge in limited tasks is to smooth the probability estimates by grouping the words into human-defined classes and estimating the language model on the classes.

The stack decoder is an attractive control strategy for a speech understanding system because it can combine information from the acoustic matching and any of a variety of language models/natural language systems into a single integrated search. The current prototype is not mature enough to use in a practical recognition/understanding system, but is showing promise. The no-grammar recognition works fairly well—but no-grammar recognition is not the goal. The goal of the effective integration of the language model and the acoustic modeling has not yet been achieved due to the interaction between the two knowledge sources preventing estimation of the proper least upper bound of the theory likelihoods. Once this problem is overcome, the stack decoder should become a practical structure for speech recognition.

REFERENCES

1. A. Averbuch, L. Bahl, R. Bakis, P. Brown, A. Cole, G. Daggett, S. Das, K. Davies, S. De Gennaro, P. de Souza, E. Epstein, D. Fraleigh, F. Jelinek, S. Katz, B. Lewis, R. Mercer, A. Nadas, D. Nahamoo, M. Picheny, G. Shichman, and P. Spinelli, "An IBM-PC Based Large-Vocabulary Isolated-Utterance Speech Recognizer," Proc. ICASSP 86, Tokyo, April 1986.

2. L. R. Bahl, F. Jelinek, and R. L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-5, March 1983.

3. L. R. Bahl, P. S. Gopalakrishnam, D. Kanevsky, D. Nahamoo, "Matrix Fast Match: A Fast Method for Identifying a Short List of Candidate Words for Decoding," Proc. ICASSP 89, Glasgow, May 1989.

4. J.R. Bellegarda and D.H. Nahamoo, "Tied Mixture Continuous Parameter Models for Large Vocabulary Isolated Speech Recognition," Proc. ICASSP 89, Glasgow, May 1989.

5. A. Derr and R. Schwartz, "A Simple Statistical Class Grammar for Measuring Speech Recognition Performance," Proceedings October, 1989 DARPA Speech and Natural Language Workshop, Morgan Kaufmann Publishers, October, 1989.

6. I. J. Good, "The Population Frequencies of Species and the Estimation of Population Parameters," Biometrika, vol. 40, no.3 and 4, 1953.

7. X. D. Huang and M.A. Jack, "Semi-continuous Hidden Markov Models for Speech Recognition," Computer Speech and Language, Vol. 3, 1989.

8. F. Jelinek, "A Fast Sequential Decoding Algorithm Using a Stack," IBM J. Res. Develop., vol. 13, November 1969.

9. F. Jelinek, R. Mercer, and S. Roucos, "Classifying Words for Improved Statistical Language Models," Proc. ICASSP 90, Albuquerque, NM, April 1990.

10. F. Jelinek, "Self-Organized Language Modeling for Speech Recognition," in *Readings in Speech Recognition*, A. Weibel and K. F. Lee, ed., Morgan Kaufmann Publishers, 1990.

11. S. M. Katz, "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer," ASSP-35, pp 400-401, March 1987.

12. K. F. Lee, *Automatic Speech Recognition: The Development of the SPHINX System*, Kluwer Academic Publishers, Norwell, MA, 1989.

13. F. Kubala, S. Austin, C. Barry, J. Makhoul, P. Placeway, R. Schwartz, "BYBLOS Speech Recognition Benchmark Results," Proc. DARPA Speech and Natural Language Workshop, Morgan Kaufmann Publishers, Feb. 1991.

14. M. Liberman, "Text on Tap: the ACL/DCI," Proceedings October, 1989 DARPA Speech and Natural Language Workshop, Morgan Kaufmann Publishers, October, 1989.

15. H. Murveit, personal communication.

16. N. J. Nilsson, "Problem-Solving Methods of Artificial Intelligence," McGraw-Hill, New York, 1971.

17. D. B. Paul, "A CSR-NL Interface Specification," Proceedings October, 1989 DARPA Speech and Natural Language Workshop, Morgan Kaufmann Publishers, October, 1989.

18. D. B. Paul, "Speech Recognition using Hidden Markov Models," Lincoln Laboratory Journal, Vol. 3, no. 1, Spring 1990.

19. D. B. Paul, "Algorithms for an Optimal A* Search and Linearizing the Search in the Stack Decoder," Proc. DARPA Speech and Natural Language Workshop, Morgan Kaufmann Publishers, June 1990.

20. D. B. Paul, "The Lincoln Tied-Mixture HMM Continuous Speech Recognizer," Proc. DARPA Speech and Natural Language Workshop, Morgan Kaufmann Publishers, June 1990.

21. D. B. Paul, "New Results with the Lincoln Tied-Mixture HMM CSR System," Proc. DARPA Speech and Natural Language Workshop, Morgan Kaufmann Publishers, Feb. 1991.

22. P. Price, W. fischer, J. Bernstein, and D. Pallett, "The DARPA 1000-Word Resource Management Database for Continuous Speech Recognition," Proc. ICASSP 88, New York, April 1988.

23. R. Schwartz, personal communication.

24. D. G. Sturtevant, "A Stack Decoder for Continuous Speech Recognition," Proc. DARPA Speech and Natural Language Workshop, Morgan Kaufmann Publishers, Oct. 1989.

Table 1: Perplexities of N-gram Back-off (BO) and PML Language Models

Database	Perplexity 1-gram BO	1-gram PML*	2-gram BO	2-gram PML*	3-gram BO	3-gram PML*	Training Words	Training Vocab.	Unknown Test Words
ATIS, June 90	114	122	24	28	18	35	8600	552	1.3%
ATIS, baseline	125	-	18	-	13	-	45K	1065	.2%
RM, fair	258	258	26	27	14	25	71K	991	0%
RM, cheat	254	-	16	-	6	-	89K	991	0%
WSJ, small	715	1608	365	1512	274	3926	130K	13K	6.9%
WSJ, large	1215	1334	287	492	172	1541	4.8M	64K	.8%

* pad chosen to minimize test set perplexity
June 90 = June 90 ATIS training data
baseline = "baseline" ATIS training data, see below
RM, fair = trained on RM1 + RM2 training data
RM, cheat = trained on "fair" + test data
WSJ = Wall Street Journal sampler[14]

Table 2: RM SD Bigram Language Model Recognition Results

Grammar	type	Trained from	Modeling Unit	Test Set Perplexity	Word Err
WPG	(cheat)	patterns[22]	word	60	1.7%
back-off	fair	data	word	26	3.0%
back-off	cheat	data	word	16	1.5%
class bigram[5]	fair	data	class	24	~WPG[23]
pattern bigram	(cheat)	patterns[22]	word	20[12]	~.7*WPG[12]

Note: A "fair" language model is not trained on the test data, a "cheating" model is trained on the test data. Therefore, the pattern trained models must be classed as cheating because they were trained on the entire RM language—which includes the test data.