The Lincoln Large-Vocabulary HMM CSR*

Douglas B. Paul

Lincoln Laboratory, MIT
Lexington, Ma. 02173

ABSTRACT

The work described here focuses on recognition of the Wall Street Journal (WSJ) pilot database [17], a new CSR database which supports 5K, 20K, and up to 64K- word CSR tasks. The original Lincoln Tied-Mixture HMM CSR was implemented using a time-synchronous beam-pruned search of a static network[14] and does not extend well to this task because the recognition network would be too large for currently practical workstations. Therefore, the recognizer has been converted to a stack decoder-based search strategy[1,7,16]. This decoder has been shown to function effectively on up to 64K-word recognition of continuous speech. This paper describes the acoustic modeling techniques and the implementation of the stack decoder used to obtain these results.

INTRODUCTION

The original Lincoln Tied-Mixture HMM CSR was implemented as a single-layer static network with a time-synchronous (TS) beam-pruned network-search strategy[14]. When used with a bigram language model (LM), this implementation generally requires an V^2 set of word interconnection links. This is tractable for vocabulary sizes on the order of 1K, but becomes intractable for vocabulary sizes (V) of 5K or more words. This implementation also is incompatible with or intractable for many forms of LM, such as recursive or trigram models. A stack decoder [1,7,16] with fast match[2,3,5] is used here to overcome the limitations of the original decoder structure.

Previous work focused on a 1K word task, Resource Management (RM)[18], which could be handled adequately with the TS decoder. (The same decoder was also used on the ATIS task[13].) However, only the stack decoder was usable on the WSJ task. While the theory of the stack decoder is adequately established [1,7,16], many of the implementation details are still topics for research. One topic of particular interest is fast match techniques and structures. There are also a number of pragmatic issues to be resolved for stack decoders and tree searches in general. (The stack decoder implements a tree search.)

Once a functioning stack decoder was developed, it became possible to perform recognition experiments on the WSJ database. This allowed both further debugging and development of the stack decoder and exploration of acoustic modeling techniques.

THE BASIC HMM SYSTEM

The basic system, with the exception of the decoder, is very similar to the earlier Lincoln tied-mixture (TM) systems. The system used here has two observation streams (TM-2): mel-cepstra and time differential mel-cepstra. (Due to time limitations, the second differential mel-cepstral observation stream used in the TS decoder for SI tasks was not tested.) The system uses Gaussian tied mixture [4,6] observation pdfs and treats each observation stream as if it is statistically independent of all others. Triphone models [20] are used to model phonetic coarticulation. (Cross-word triphones, which are a feature of the old TS decoder, will be implemented later.) These models are smoothed with reduced context phone models [20]. Each phone model is a three state "linear" (no skip transitions) HMM. The phone models are trained by the forward-backward algorithm using an unsupervised monophone bootstrapping procedure. The recognizer extrapolates (estimates) untrained phone models, contains an adaptive background model, allows optional intermediate silences, and can use any left-to-right stochastic LM. The LM module is interfaced via a proposed CSR-NL interface[11].

THE STACK DECODER

The stack decoder is organized according to the description in reference [16] and uses the long-span LM search control strategy. The basic paradigm used by the stack decoder is: remove the best theory from the stack, apply the fast matches (FM) to find a small number of potential successor words, evaluate the log-likelihood of these successors with the detailed matchs (DM), and insert the most promising new theories back onto the stack[2,3,5] This paradigm requires that theories of different lengths

*This work was sponsored by the Defense Advanced Research Projects Agency. The views expressed are those of the author and do not reflect the official policy or position of the U.S. Government.

be compared. Therefore, this system maintains a "least-upper-bound so-far" (lubsf) of all previously computed theory output log-likelihoods. (Acoustic log-likelihoods and the lubsf are functions of time.) The maximum of the difference between this lubsf and each theory output log-likelihood ($StSc \leq 0$) is used to determine the best theory[15,16]. Theories whose $StSc$ is less than a threshold are pruned from the stack. Reference [16] defines a method for estimating the most likely theory output time, *t_exit*. The stack entries are sorted by a major sort on *t_exit* and a minor sort on $StSc$. Thus the theories are extended primarily on a time basis.

THE FAST MATCH

The acoustic fast match (AFM) algorithm used here is an HMM phonetic tree generated from the vocabulary[3]. The output log-likelihood of the current theory is input to the root of the tree and the paths are evaluated using a TS beam search. If an output state's log-likelihood exceeds a threshold, the corresponding word is activated and the best score is recorded. (All references to scores in this paper refer to log-likelihoods.)

This tree search needs to be terminated to limit its computation. The beam pruning threshold used in the AFM search is computed from an estimate of the upper-bound of the AFM state log-likelihoods (AFM-bound) and, when all states are pruned, the AFM terminates. This AFM-bound is computed by a reentrant phonetic tree. (Unlike the AFM tree, the leaves of this tree connect back to the root to provide a path for a word exit to enter the next word. Thus the scores in the FM tree drop off after the word ends while the upper bound of the scores in the FM-bounding tree does not.) This reentrant tree is, in effect, an efficient implementation of a no-grammar recognizer whose only output of interest is the AFM-bound.

Once the AFM has completed, the LM fast match (LMFM) log-likelihoods are added to the AFM scores and the result is compared to another threshold. The set of words which survives the second threshold is passed to the DMs. (If an expensive LM algorithm is used, inexpensive estimates of the log-likelihoods may be used in the LMFM. Since the N-gram LMs used in this effort are very cheap to compute, the exact LM DM log-likelihood was used.)

If the FM-likelihood is guaranteed to be greater than or equal to the DM-likelihood and the FM decision threshold is the DM lubsf, the FM will be admissible. (An admissible FM is an FM which is guaranteed not to cause any search errors[3]. This statement also assumes the beam pruning is generous enough that no FM-tree search errors occur.)

Any of a number of phonetic units can be used in these trees: the goal is to minimize the total time required to compute the FMs and the DMs without increasing the error rate over that of the DMs alone. An elaborate (and expensive) FM will minimize the DM computation while a very cheap FM will result in a large amount of DM computation. Any of a large number of phonetic units can be used: triphones, left-diphones, right-diphones, monophones, upper bounding context phones, simplified network phones. (An upper bounding context phone is a diphone or monophone whose scores are an upper bound of all scores which would be produced by the triphones covered by the context phone. A simplified network phone might collapse its states into fewer states.) The two trees need not use the same phonetic units and each tree can also use a mix of phonetic units. One extreme would be triphone trees (maximally complex for a triphone based recognizer) and the other extreme would one-state monophone trees. It is also possible to use simplified observation pdfs to reduce the computation. Each of these variations must be tested to evaluate the trade-offs. The Lincoln system currently uses TM left-diphones in both trees. Since TM pdfs are relatively expensive to compute, they are cached to prevent recomputation.

Because the theories are searched in dominantly *t_exit* order, it is possible to further reduce the total AFM computation time by grouping all of theories on the stack which have *t_exit*'s within a small time zone, add their output likelihoods (for a full decode), and apply this sum as input to a single execution of the AFM tree search. (Substitute maximum for sum to perform a Viterbi decode.) This single AFM computation may be somewhat more expensive than the AFM computation for a single theory, but it reduces the number of AFM executions.

THE DETAILED MATCH

The DM is implemented as a one-word-at-a-time beam-pruned TS HMM applied to each word which survives the FMs. The input likelihood for each word decode comes from the output likelihood array in each stack entry. (This theory output log-likelihood output must time truncated in order to fit the important portion into this finite array before inserting any new theory onto the stack.) There is rarely any difficulty fitting the output of a word into this array, but it may not be possible for a continuing sound such as a zone of background (silence). This is handled by using "continuable" background models. The state of the background HMM is also stored on the stack and a long background is modeled as a succession of theories ending in background. (Of course, normal input is possible only for the first of this series of background theories. The later theories rely on the state information.) This also enables a theory to decide that a transition to background has occurred without waiting for the next word to begin.

In reference [15], a technique for eliminating theories from the stack which are "covered" by an "LM-future-equivalent" (LMF-equivalent) theory is proposed. (One theory covers another if all entries in its output log-likelihood array are greater than those of the second theory at the corresponding times.) Two theories are LMF-equivalent if the probabilities of all future word sequences are the same for both theories. Thus, for an N-gram LM, any theories which share the same N-1 final words are LMF-equivalent. Any LMF-equivalent covered theory can never beat its covering theory and therefore can be eliminated. This is analogous to a path join in a TS decoder. The mechanism also serves to eliminate the poorer of two theories which differ only in optional inter-word backgrounds. (Since optional inter-word backgrounds are not considered by the LM, they may be eliminated before determinating LMF-equivalence.) For any limited left-context-span LM, this mechanism prevents the exponential theory growth that can occur in a tree search.

The words passed to the DM by the FM are generally acoustically similar and thus frequently share many of the triphones. Therefore the same observation pdfs are likely to be needed more than once. As in the FM, the TM likelihoods are cached to minimize the cost of reuse.

This stack decoder does not yet include cross-word phonetic models. It will be possible to add them to the system, but they will certainly increase the complexity of the acoustic DM and perhaps also of the AFM (depending on the type of phonetic unit used in the AFM). Since the system still has some known difficulties/bugs, the implementation of the cross-word phonetic models will be delayed until these problems are under control. Since the 5K word WSJ vocabulary already contains over 6K word-internal triphones and cross-word triphone models will greatly increase this number, practical machine size limits dictate that clustered triphones [9,10] or lower context phonetic units, such as semiphones [14], be used to reduce the memory required to implement cross-word phonetic models.

RECOGNITION RESULTS

The initial work developing and implementing the above described stack decoder was performed using the Resource Management (RM) database[18]. The WSJ-pilot database training and development-test data has only been fully available for about 5 weeks (as of this writing) and therefore the number of experiments that have been performed on it is limited. Where possible, results will be reported on the WSJ-pilot database, but some results will be quoted from work performed on the RM databases. All results must be considered preliminary, particularly since, as noted above, only non-cross-word triphones are being used and the recognizer has known but as yet unfixed algorithmic/implementation bugs.

One result that became obvious very quickly after transitioning to the WSJ data was that algorithmic decisions made on the RM data could be very inappropriate for the WSJ task (and presumably any similar large vocabulary task). For instance, work on the RM task suggested that a triphone FM tree with a monophone FM-bounding tree was a good choice for the AFM. This worked very well for RM but rather slowly for WSJ. The triphone FM dominated the computation and was so slow that it slowed down the entire system. The diphone trees mentioned above were significantly faster for WSJ and still worked very well for RM. Similarly, the run-times are much longer and the recognition error rates are much higher for WSJ experiments indicating that it is a significantly harder task than RM. The stack decoder is also more than an order-of-magnitude faster than the TS decoder on an RM with a (full-branching) bigram LM task.

A series of no-LM tests using RM training and test data was performed to demonstrate the large vocabulary capability of the stack decoder. Since a dictionary was not available at the time this test was performed, a "tri-letter" dictionary was used (ie. each three letter sequence is used in the same fashion as one would use a triphone). The recognizer used RM words augmented with WSJ words to achieve the desired vocabulary. Over a vocabulary size range of 1K to 64K words, the system ran effectively with computation time proportional to the square root of the vocabulary size. The stack decoder used in this test contained a triphone-based FM and thus this result is mostly indicative of the FM computational requirements. This decoder was also demonstrated on the 64K-word task using a perplexity 79 bigram LM.

The stack decoder was tested on a variety of the conditions provided by the WSJ-pilot database (Table 1). Due to the limited time available and the immature state of the decoder, only a subset of the available conditions could be tested. Since we were primarily interested in the performance of the decoder, only closed vocabulary tests were performed. (In a closed vocabulary test, all words in the test set are in the recognizer's vocabulary.) The language models are N-gram back-off LMs[8,12]. The bigram models are "baseline" models and the dictionary is a function word dependent triphone dictionary derived from the "baseline" dictionary supplied by Dragon. (The baseline components are standardized components supplied with the database[17].)

Inspection of the actual output of the system reveals a non-trivial number of malfunctions. (The total effect of these problems on the results is probably less than 10% of the numbers in Table 1.) In some cases, the likelihood of the output sentence is less than the likelihood of the correct sentence. This could be caused by a pruning error (either FM or DM) or a bug in one (or more) of the routines. Another problem which shows up is an incorrect likelihood for the output theory, probably due to occasional errors in locating the most likely output time for a theory (*t_exit*).

Inspection of these results (Table 1) suggests several observations. Comparison of lines 2 and 3 show a significant improvement (8.0% v. 10.1% word error) when 2400 rather than 600 SD training sentences are used. Thus, the "knee" in the function of performance vs. amount of training data is not reached by 600 SD training sentences. Comparison of the LSD trained systems shows the error rate to increase less than linearly with the perplexity: V=5K, p=44: 6.0%; V=5K, p=80: 8.0%; V=5K, p=118: 10.5%; V=20K, p=158: 13.6%; and V=20K, p=236: 18.0%.

RAPID SPEAKER ENROLLMENT

There are four basic methods of producing acoustic models for speech recognition: static SI training, static SD training, rapid speaker enrollment, and recognition-time adaptation. The two static methods train the models using prerecorded data and do not change the models thereafter. Rapid speaker enrollment records a small amount of data from a speaker and uses the data to adapt an existing set of models. Recognition-time adaptation adapts the models to the speaker during the recognition process and may be supervised or unsupervised depending on whether or not the speaker corrects the recognition output. We have added a rapid enrollment mode to our TM trainer.

The rapid enrollment algorithm used is: read an existing set of TM models into the trainer and adapt (train) only the Gaussians based upon the new data[19]. To date, only a few pilot experiments using one test speaker have been performed, shown in Table 2. (The recognition experiments were performed using an obsolete version of the recognizer with a higher error rate than the one used to produce the database results, so the two tables should not be compared.) These results suggest that the adaptation algorithm is operational, but are too statistically weak to draw any firm conclusions. They suggest that another speaker's SD models may give poor initial performance, but are improved significantly by the rapid enrollment process. Both SI models perform better initially, but are only improved a small amount by the enrollment. All three sets of rapid-enrolled models gave similar performance. And, as usual, SD models, given enough training data, yield the best performance.

DISCUSSION AND CONCLUSIONS

The results of these investigations suggest that the stack decoder will be a viable competitor to time synchronous approaches. (This should come as no surprise since IBM has had operational stack decoders for years[1].) These results also show that a number of additional strategies, such as covered LMF-equivalent theory elimination are necessary to achieve useful speeds. (In one test where a bug prevented the covered LMF-equivalent theory elimination, a 4000 element stack overflowed after 10 CPU hours. After the bug was fixed, the sentence decoded in 10 minutes with a maximum stack size of less than 100. This much improvement while dramatic, was rare—the system with the bug successfully decoded many other sentences.) Very few sentences require a stack size exceeding a few hundred theories. Several other techniques, such as efficient fast matches and sharing each fast match across a group of theories—which can limit the number of acoustic fast matches to less than one per input observation were found to be important. Tied mixture pdfs are expensive to compute and the caching of the pdfs is also vital to achieving adequate speeds. Even with the caching, the pdf computation can be the single most expensive operation.

The tests on rapid speaker enrollment reported here are little more than pilot tests for debugging purposes and no strong conclusions can be drawn. The results, however, show promise and will require more rigorous testing.

So far, we have not addressed such issues as recognition-time speaker adaptation and language-model adaptation (ie. handling out-of-vocabulary words at recognition time). The current tests show error propagation not to be a serious problem, so the initial reaction to an out-of-vocabulary word—a recognition error—should not cause problems elsewhere in the input. Nor have we had a chance to test on the spontaneous data recorded as part of the WSJ-pilot database.

The recognition results achieved on the WSJ-pilot database are encouraging. Even without cross-word phonetic models (cross-word phonetic models halved our error rates for RM using the TS decoder[14]), the error rates are high enough to show the WSJ task to be very challenging, but not so high that one is intimidated by the task. We hope to improve our future performance by fixing some of the bugs, by improving the quality of our modeling techniques, and by making the system more able to adapt to its user and environment.

REFERENCES

1. 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.
2. L. Bahl, P. S. Gopalakrishnam, D. Kanevsky, D. Nahamoo, "Matrix Fast Match: A Fast Method for Identifying a Short List of Candidate Words for Decoding," ICASSP 89, Glasgow, May 1989.
3. L. Bahl, S. V. De Gennaro, P. S. Gopalakrishnam, R. L. Mercer, "A Fast Approximate Acoustic Match for Large Vocabulary Speech Recognition," submitted to ASSP.
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. L. S. Gillick and R. Roth, "A Rapid Match Algorithm for Continuous Speech Recognition," Proceedings June 1990 Speech and Natural Language Workshop, Morgan Kaufmann Publishers, June, 1990.
6. X. D. Huang and M.A. Jack, "Semi-continuous Hidden Markov Models for Speech Recognition," Computer Speech and Language, Vol. 3, 1989.
7. F. Jelinek, "A Fast Sequential Decoding Algorithm Using a Stack," IBM J. Res. Develop., vol. 13, November 1969.
8. 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.
9. K. F. Lee, *Automatic Speech Recognition: The Development of the SPHINX System,* Kluwer Academic Publishers, Norwell, MA, 1989.
10. D.B. Paul and E. A. Martin, "Speaker Stress-Resistant Continuous Speech Recognition," Proc. ICASSP 88, New York, NY, April 1988.
11. D. B. Paul, "A CSR-NL Interface Specification," Proceedings October, 1989 DARPA Speech and Natural Language Workshop, Morgan Kaufmann Publishers, October, 1989.
12. D. B. Paul, "Experience with a Stack Decoder-Based HMM CSR and Back-Off N-Gram Language Models," Proc. DARPA Speech and Natural Language Workshop, Morgan Kaufmann Publishers, Feb. 1991.
13. D. B. Paul, "New Results with the Lincoln Tied-Mixture HMM CSR System," Proceedings Fourth DARPA Speech and Natural Language Workshop, Morgan Kaufmann Publishers, February, 1991.
14. D. B. Paul, "The Lincoln Tied-Mixture HMM Continuous Speech Recognizer," ICASSP 91, Toronto, May 1991.
15. D. B. Paul, "Algorithms for an Optimal A* Search and Linearizing the Search in the Stack Decoder," ICASSP 91, Toronto, May 1991.
16. D. B. Paul, "An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition with a Stochastic Language Model," this proceedings.
17. D. B. Paul and J. M. Baker, "The Design for the Wall Street Journal-based CSR Corpus," this proceedings.
18. P. Price, W. Fisher, J. Bernstein, and D. Pallett, "The DARPA 1000-Word Resource Management Database for Continuous Speech Recognition," ICASSP 88, New York, April 1988.
19. D. Rtischev, "Speaker Adaptation in a Large-Vocabulary Speech Recognition System," Masters Thesis, MIT, 1989.
20. R Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul, "Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech," Proc. ICASSP 85, Tampa, FL, April 1985.

	Tr. sent	Punct	Vocab	LM	Perp	Wd err (std dev)
1. LSD	2400	VP	5K	TG	44	6.0% (.5%)
2. LSD	2400	VP	5K	BG	80	8.0% (.6%)
3. SD (LSD)*	600	VP	5K	BG	80	10.1% (.6%)
4. LSD	2400	NVP	5K	BG	118	10.5% (.7%)
5. SD	600	VP	5K	BG	80	12.6% (.4%)
6. SI-84	7200	VP	5K	TG	44	15.0% (.8%)
7. SI-84	7200	VP	5K	BG	80	19.3% (.8%)
8. SI-12	7200	VP	5K	BG	80	21.7% (.8%)
9. LSD	2400	VP	20K	BG	158	13.6% (.9%)
10. LSD	2400	NVP	20K	BG	236	18.0% (.8%)

Table 1: WSJ Development Test Results: * LSD speaker subset of line 5; LSD=longitudinal SD (3 spkr subset of SD); SD: 12 speakers; SI-84: train on 84 speakers, test on 10 SI-test speakers; SI-12: trained on all 12 SD speakers, test on 10 SI-test speakers; VP=verbalized punctuation; NVP=non-verbalized punctuation; TG=trigram; BG=bigram; std dev=binomial standard deviation; The dictionary is a function-word dependent triphone dictionary. All bigram language models are the "baseline" models and all tests use a closed recognition vocabulary.

Training condition	Adapted	Wd err rate	Comments
LSD-2400 (test speaker)	no	11%	Standard LSD
LSD-2400 (test speaker)	yes	13%	(control)
LSD-2400 (non-test speaker)	no	30%	
LSD-2400 (non-test speaker)	yes	21%	
SI-84	no	22%	Standard SI-84
SI-84	yes	18%	
SI-12	no	19%	Standard SI-12
SI-12	yes	18%	

Table 2: Rapid Enrollment Test Results: These are pilot results tested on one non-database speaker using an obsolete version of the stack decoder and a non-standard (biased) set of 20 short WSJ test sentences containing 168 words, so comparisons should not be made with the development test results. The standard deviation of the error rates is about 3%. Enrollment was performed using the standard WSJ 40 adaptation sentences recorded by the test speaker. Test conditions: 1 speaker, VP, 5K, BG (p=80).