LATTICE-BASED SEARCH STRATEGIES FOR LARGE VOCABULARY SPEECH RECOGNITION

F. Richardson *M. Ostendorf*

Electrical, Computer and Systems Engineering
Boston University, Boston, MA 02215

J. R. Rohlicek

Bolt Beranek and Newman Inc.
Cambridge, MA 02138

ABSTRACT

The design of search algorithms is an important issue in large vocabulary speech recognition, especially as more complex models are developed for improving recognition accuracy. Recently, multi-pass search strategies have been used as a means of applying simple models early on to prune the search space for subsequent passes using more expensive knowledge sources. The pruned search space is typically represented by an N-best sentence list or a word lattice. Here, we investigate three alternatives for lattice search: N-best rescoring, a lattice dynamic programming search algorithm and a lattice local search algorithm. Both the lattice dynamic programming and lattice local search algorithms are shown to achieve comparable performance to the N-best search algorithm while running as much as 10 times faster on a 20k word lexicon; the local search algorithm has the additional advantage of accommodating sentence-level knowledge sources.

1. INTRODUCTION

N-best rescoring has been used in many speech recognition systems as a means of incorporating expensive or sentence-level knowledge sources into the system (e.g. [1, 2, 3]). For large vocabulary tasks, or tasks where the error rates are high, it is desirable to use a large N for the N-best search to insure that a good hypothesis is included in the list. However, as N is increased, N-best rescoring becomes more expensive.

A word lattice is an efficient representation of the information that is associated with an N-best list, reducing both storage costs and redundant computations. Here we use the term "lattice" to mean an acyclic, directed network with explicit time and score information, as in [4]. This usage is somewhat different from the unannotated lattices, or "word graphs", used in some progressive search strategies [1, 5]. In either case, lattices represent sentence hypotheses that have been generated by a previous recognition stage, and they have already proved to be an important structure in multi-pass search algorithms.

In this work, three lattice-based search algorithms are compared: N-best rescoring, dynamic programming (DP) and a new local search algorithm. The lattice DP algorithm is an efficient and optimal algorithm, but it only allows for the incorporation of Markov knowledge sources into the search. On the other hand, the lattice local search algorithm is a sub-optimal algorithm, but both sentence-level and Markov knowledge sources can be used. In general, the efficiency and performance of the lattice local search algorithm are dependent on the definition of the local neighborhood used for the search and the size of the lattices.

The paper proceeds by introducing the general search problem that the three different algorithms must solve, and then describes each algorithm. Next, we give experimental results on different tasks and with different knowledge sources, comparing the three algorithms in terms of speed and recognition error rate. We conclude by summarizing the results and describing conditions under which one would use each approach.

2. PARADIGM

The lattices used in this work were obtained from the BBN Byblos 5-pass lattice decoder [1], where each pass successively reduces the search space. The fourth pass of the decoder is a backward beam search using a trigram language model and cross-word triphone acoustic models. In other BU/BBN work [6, 7], the results of the fourth pass are used to generate N-best lists, which are subsequently rescored and resegmented in a fifth pass by a more detailed HMM acoustic model, providing phonetic segmentations and HMM score information as well as trigram language model scores for use in rescoring by segmental models. For the lattice rescoring work, the decoder was modified so that a lattice was produced after the fourth pass of the BBN decoder. The lattices are annotated with segmentation times and HMM word scores corresponding to the fourth-pass model.

The topology of the lattices is the same as the topology of the the trigram expanded grammar constructed by the BBN decoder. That is, the lattices are backwards and each word node in a lattice has a unique right context. Each arc therefore corresponds to a specific trigram transition. Using the notation $a]b$ to indicate word a in the right context of b. Each arc therefore corresponds to a specific trigram transition. For example, the arc $a]b \leftarrow b]c$ is associated with the trigram probability $p(w_i = a | w_{i+1} = b, w_{i+2} = c)$.

For all of the rescoring algorithms investigated in this work the following knowledge sources are used: the BU Stochastic Segment Model (SSM) [6], a relative frequency duration model, either a trigram language model or a sentence-level mixture language model, and the number of words,

This work was supported by ONR, grant number ONR-N00014-92-J-1778

phones and inter-word silences. In all cases, the SSM segment and frame scores are cached in order to reduce redundant computation and the phonetic segmentations on the lattices are used to constrain the search with the SSM. The scores from the different knowledge sources are linearly combined with weights estimated from the N-best hypotheses of a development test set, using a grid-based optimizer [8]. Therefore, N-best rescoring must be performed in order to estimate weights for all algorithms.

3. LATTICE-BASED SEARCH ALGORITHMS

For all of the lattice-based search algorithms, the first step in decoding requires expanding the annotated word lattice to include multiple word pronunciations and triphone contexts with optional inter-word silences. The process of adding multiple pronunciations introduces triphone nodes without time information, so segmentation times for these nodes are estimated using right-context mean phone durations. The times are used to constrain the search and reduce the cost of the segmental acoustic model. Given the expanded lattice, three different search algorithms are investigated here: the lattice DP, N-best and local search algorithms, as described below.

3.1. Lattice DP Search

The lattice DP algorithm is an optimal algorithm which guarantees that the highest scoring hypothesis will be found. The algorithm is also efficient in that each word node in the lattice is only evaluated once, although different word nodes may contain the same word (associated with a different trigram context) and thus there is still some redundant computation.

To specify the DP search, let $\{n_s, n_t, \mathcal{N}, \mathcal{A}\}$ define a lattice, where n_s and n_t are the first and last nodes of the lattice, $\mathcal{N}$ is the set of all nodes, and $\mathcal{A}$ is the set of all triphone arcs $\{\gamma(n_i, n_j) : n_i, n_j \in \mathcal{N}\}$. $R(n_i)$ is the set of constrained times for node n_i, and $\mathbf{CS}(\gamma(n_i, n_j), t, \tau)$ is the combined score for the arc $\gamma(n_i, n_j)$ and the triphone segment times t and τ, which includes a weighted combination of: the log likelihoods of the segment acoustic model, the duration model, and the language model (if the arc is at a word boundary), and phone, word and silence insertion penalties depending on arc location. $\mathcal{N}$ is topologically sorted so that all of the scores reaching a node at a given time have been evaluated before its successor nodes are reached. The lattice DP algorithm is then

1. Initialize: $J^*(0, n_s) = 0$
2. For each $n_i \in \{\mathcal{N} - n_s\}$ calculate:
   For each $t \in R(n_i)$, sequentially

$$J^*(t, n_i) = \max_{\gamma(n_i,n_j)\in\mathcal{A}, \tau\in R(n_j)} \{J^*(\tau, n_j) + \mathbf{CS}(\gamma(n_i, n_j), t, \tau)\} \qquad (1)$$

The argument that maximizes the score for each (t, n_i) pair is stored in a traceback structure which contains the highest scoring path at the end. For an utterance of length T, $J^*(T, n_t)$ gives the starting point of the traceback. As in other recognition work, a beam search could be used to reduce the cost of the lattice DP, but it was not explored in the experiments reported here.

3.2. Lattice N-Best Rescoring

N-best rescoring is performed on the lattices using the lattice DP algorithm and constraining the path through the lattice to correspond to one N-best hypothesis at a time. The modification to Equation 1 is trivial: triphone arcs corresponding to words that are not in the N-best hypothesis are not scored.

Lattice N-best rescoring differs from rescoring an N-best list in that the fifth segmentation pass of the decoder is not needed and the data structures for representing the search space are more efficient. In addition to providing a performance baseline, the lattice N-best algorithm is useful for providing the N-best hypothesis scores that we use in weight estimation for all of the algorithms.

3.3. Lattice Local Search

The lattice local search algorithm is an iterative strategy that successively evaluates small changes from the current top hypothesis. The algorithm was motivated by the split-and-merge phone recognition search algorithm proposed in [9] and consists of the following steps:

1. Initialize:
   CurrHyp is set to the 1-best hypothesis from the BBN decoder. Find the best new segmentation and associated score of this hypothesis.
2. Iterate:
   Evaluate all hypotheses in the local neighborhood of **CurrHyp**. **NewHyp** is set to the hypothesis that has the largest increase in score for the sentence hypothesis.
3. If **NewHyp** is the same as **CurrHyp** then Stop, otherwise go to 2.

The local neighborhood of a current path **CurHyp** in the lattice consists of all paths that form simple loops with **CurHyp**, as illustrated in Figure 1. We refer to such a deviation from the current path as a "local path". These "local paths" are sequences of words in the lattice that are not part of **CurHyp**, but are connected to it. For this work, six types of local paths, depicted in Figure 1, were used for the local neighborhood: "insertions," "deletions," "substitutions," "splits," "merges," and "double substitutions." The neighborhood was motivated by the fact that the vast majority of errors in the WSJ task are simple one or two-word sequences. We also investigated smaller neighborhoods, but the recognition accuracy degraded.

During the lattice local search, the acoustic model rescoring involves resegmentation of the region of the utterance corresponding to the local path. The language model score change (a log likelihood ratio) may be computed at the sentence level or only over the local path, depending on the model. In the current implementation, which resegments only the local path, a second resegmentation step is included after choosing the new path since changes in word labels can affect the segmentations of words beyond the local neighborhood. The local path scores are cached to reduce redundant calculation.

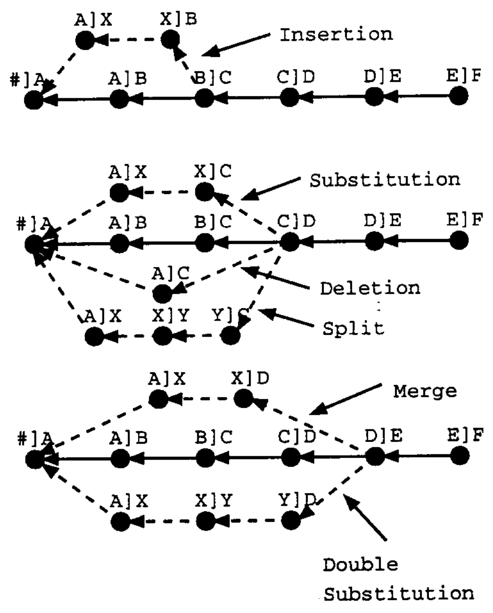

Figure 1: Six local paths that comprise the local neighborhood for the lattice local search.

4. EXPERIMENTS

The lattice-based algorithms were evaluated on the ARPA 1993 Wall Street Journal (WSJ) corpus [10], which consists of read WSJ articles collected with a clean microphone. Two separate tests were performed: the 1993 Hub 1 (20,000 words) and Hub 2 (5,000 words) tests. In both cases, the same acoustic model is used which is trained on all of the acoustic training data (WSJ0 and WSJ1). Experiments were also performed on the Switchboard corpus [11], but the error rates were too high (>50%) to draw meaningful conclusions about performance trade-offs (although the conclusions about speed trade-offs scaled for this Switchboard task).

4.1. Lattice Statistics

To illustrate the importance of lattice (vs. N-best) representations and the effect of task difficulty, Table 1 gives lattice statistics for three tasks: the 5k vocabulary WSJ-H2 and open vocabulary WSJ-H1 tasks (both high quality read speech) and the roughly 5k vocabulary Switchboard task (spontaneous, conversational speech). The statistics include the N (from N-best) that was used to generate the lattice, the sentence inclusion rate (percentage of time the correct hypothesis can be parsed by the lattice), the word coverage (average word accuracy, obtained by finding the lowest error parse of the correct sentence in the lattice), and the accuracy of the 1-best HMM hypothesis. As the tasks get more difficult, indicated by a decrease in 1-best accuracy, the sentence inclusion and word coverage rates fall off dramatically despite the increase in N.

Table 1: *Lattice statistics for the development test sets of different tasks.*

Task	WSJ-H2	WSJ-H1	SWBD
N	100	500	2000
Sentence inclusion	79%	53%	10%
Word coverage	98%	93%	64%
1-best accuracy	91%	81%	43%

4.2. Search Trade-Offs

In the experiments below, we present speed and accuracy figures for the different search algorithms. The accuracy figures are standard word error rates, including substitution, insertion and deletion errors. The speed figures are the number of times real time required for recognizing an utterance on a Sparc 20/50.

In initial experiments, we verified that the lattice N-best rescoring algorithm had the same speed and accuracy costs as the original N-best rescoring algorithm. In fact the performance was slightly better, probably because the time windows used to constrain the possible segmentations with the acoustic model could potentially be wider when taken from the lattice rather than an HMM resegmentation. Interestingly, the speed of N-best rescoring for the H1 task (using a 20k dictionary) is almost twice that of the 5k H2 task when using the same size N to generate lattices. We hypothesize that the larger vocabulary task generates more active triphones for the N-best hypotheses which results in a decrease in the amount of computation that is saved by score caching.

The next series of experiments compared the performance of the different algorithms for Markov knowledge sources, for the H2 (N=100) and H1 (N=500) WSJ tasks. On both tasks, we see a slight improvement in performance for the DP algorithm and a slight degradation in performance for the sub-optimal local search algorithm relative to N-best rescoring. However, both the DP and local search algorithms are significantly faster than N-best rescoring, with the speed difference increasing as a function of the size of the lattices. This gain is despite the use of score caching across hypotheses in the N-best rescoring search. The local search algorithm, for these lattices, is not much different in speed than the DP search, despite the fact that it converges in only a few steps (roughly 2 iterations for H2 and 2.5 iterations for H1). The local search would be faster if not for redundant computation, some of which can be reduced by word caching, increasing the resegmentation region associated with the local neighborhood (to avoid subsequent resegmentation), and using a bigram rather than a trigram expanded lattice.

For Markov knowledge sources, there appears to be no advantage to using the local search algorithm. However, there are many interesting knowledge sources that are not Markov, particularly in language modeling where one might want to capture long-distance dependence. The speed and performance of the lattice local search algorithm with a sentence-level mixture language model [12] are presented in

Table 2: *WSJ H2 and H1 speed and error rates for both development and evaluation test sets comparing the lattice N-best, DP and local search algorithms when only Markov knowledge sources are used.*

Test	N-Best		DP		Local Search	
H2:dev	13.5	7.4%	4.6	7.3%	4.6	7.5%
H2:eval	13.9	6.2%	4.1	6.2%	3.7	6.5%
H1:dev	102.6	15.6%	8.0	15.4%	8.6	15.9%
H1:eval	94.6	14.3%	9.1	13.9%	9.8	14.5%

Table 3: *WSJ H1 development and evaluation test results for lattice local search algorithm with the sentence-level mixture language model.*

	Lattice N-best		Local Search	
Test	Speed	Error Rate	Speed	Error Rate
H1:dev	103	15.4%	9.0	15.7%
H1:eval	95	13.7%	9.6	13.5%

Table 3. Note that the local search algorithm is still much more efficient than N-best rescoring and achieves slightly better error rates on the WSJ H1 evaluation test than DP with only Markov knowledge sources. Further gain could probably be attained at a small additional cost by using additional or more complex sentence-level models.

5. CONCLUSIONS

Presented here are two algorithms which have been shown to be more efficient than N-best rescoring while attaining similar or even better results. The lattice DP algorithm has been shown, in general, to perform better than N-best rescoring but has the drawback of only allowing Markov knowledge sources to be incorporated in the search. The lattice local search is as efficient as the lattice DP algorithm while achieving slightly lower error rates, and could potentially be more efficient on a larger lattice as preliminary experiments on the Switchboard task indicated. In addition, the local search has the advantage of allowing sentence-level knowledge sources such as a sentence-level mixture language model into the search. Therefore, with sentence-level knowledge sources, the lattice local search would generally be the best choice, unless the initial starting point is very bad, in which case one might want to do a preliminary DP rescoring or use N-best rescoring. In all cases, we still use the lattice N-best search to score hypotheses for use in score combination weight estimation.

6. ACKNOWLEDGMENTS

The authors gratefully acknowledge the help of researchers at BBN, particularly Rich Schwartz, for providing access to the BBN decoder and for help with our code modifications.

7. REFERENCES

[1] L. Nguyen, R. Schwartz, Y. Zhao and G. Zavaliagkos, "Is N-Best Dead?," *Proc. of the ARPA Human Language Technology Workshop,* March 1994.

[2] M. Rayner, D. Carter, V. Digalakis, P. Price, "Combining Knowledge Sources to Reorder N-Best Speech Hypothesis Lists," *Proc. of the ARPA Human Language Technology Workshop,* March 1994.

[3] M. Ostendorf, A. Kannan, S. Austin, O. Kimball, R. Schwartz and J. R. Rohlicek, "Integration of Diverse Recognition Methodologies Through Reevaluation of N-Best Sentence Hypotheses," *Proc. of the DARPA Workshop on Speech and Natural Language,* Feb. 1991, pp. 83-87.

[4] F. Alleva, X. Huang and M. Hwang, "An Improved Search Algorithm Using Incremental Knowledge for Continuous Speech Recognition," *IEEE Int. Conf. Acoust., Speech, Signal Processing,* Apr. 1993, pp. 307-310.

[5] H. Murveit, J. Butzberger, V. Digalakis and M. Weintraub, "Large-Vocabulary Dictation Using SRI's DECIPHER^{TM} Speech Recognition System: Progressive Search Technique," *IEEE Int. Conf. Acoust., Speech, Signal Processing,* Apr. 1993, pp. 319-322.

[6] M. Ostendorf, F. Richardson, S. Tibrewal, R. Iyer, O. Kimball, J. R. Rohlicek, "Stochastic Segment Modeling for CSR: The BU WSJ Benchmark System", *ARPA Spoken Language Systems Technology Workshop,* March 1994.

[7] G. Zavaliagkos, Y. Zhao, R. Schwartz and J. Makhoul, "A Hybrid Segmental Neural Net/Hidden Markov Model System for Continuous Speech Recognition", *IEEE Trans. on Speech and Audio Processing,* Vol. 2, No. 1, Part 2, pp. 151–160, Jan 1994.

[8] A. Kannan, M. Ostendorf and J. Rohlicek, "Weight Estimation for N-Best Rescoring," *Proc. of the DARPA Workshop on Speech and Natural Language,* Feb. 1992, pp. 455-456.

[9] V. Digalakis, M. Ostendorf and J. R. Rohlicek, "Fast Search Algorithms for Phone Classification and Recognition Using Segment-Based Models," *IEEE Transactions on Signal Processing,* December 1992, pp. 2885-2896.

[10] F. Kubala, *et al.*, "The Hub and Spoke Paradigm for CSR Evaluation," *Proc. of the ARPA Human Language Technology Workshop,* March 1994.

[11] J. Godfrey, E. Holliman, J. McDaniel, "Switchboard: Telephone Speech Corpus for Research and Development", *IEEE Int. Conf. Acoust., Speech, Signal Processing,* March 1992, pp. 517-520.

[12] R. Iyer, M. Ostendorf and J. R. Rohlicek, "Language Modeling with Sentence-Level Mixtures," *Proceedings of the ARPA Workshop on Human Language Technology,* March 1994.