New Developments in Lattice-Based Search Strategies in SRI's Hub4 System

Fuliang Weng   Andreas Stolcke   Ananth Sankar

Speech Technology and Research Laboratory
SRI International
Menlo Park, California

ABSTRACT

We describe new developments in SRI's lattice-based progressive search strategy. These developments include the implementation of a new bigram lattice algorithm, lattice optimization techniques, and expansion of bigram lattices to trigram lattices. The new bigram lattice generation algorithm is based on generation of backtrace entries using a word-dependent N-best list decoding pass, followed by lattice generation from the backtrace entries. We present an algorithm to reduce the size of the bigram lattices while maintaining all valid paths. This algorithm is shown to reduce the size of the lattice by about 50%, allowing easier processing in later stages such as expansion to trigram lattices. We describe two algorithms to expand bigram lattices to trigram lattices. The first is a conventional method, while the second is a novel approach that results in compact trigram lattices that were found to be a factor of six smaller than lattices created with the conventional approach. Decoding with the new trigram lattices gave a 5% improvement in word error rate as compared to our previous search strategy which used trigram LMs to rescore N-best lists.

1. Introduction

Progressive search techniques have previously been proposed as a method of applying complex knowledge sources in decoding for automatic speech recognition (ASR) [3]. The basic idea is to use simple knowledge sources, such as non-crossword acoustic models and bigram LMs to generate a lattice of possible hypotheses. The lattices are then progressively decoded with more complex knowledge sources such as crossword acoustic models or trigram LMs.

In search strategies we have previously used, such as in [6], non-crossword acoustic models and bigram LMs were used to create word bigram lattices. Adapted acoustic models were then used to produce N-best lists from these lattices. Finally, more complex knowledge sources such as crossword acoustic models and trigram LMs were used to rescore N-best lists.

Trigram LMs are an extremely powerful knowledge source, and can hence be more effective if used to decode lattices rather than N-best lists as in our previous strategy. Our previous bigram lattice algorithm generated a subset of the full bigram LM that contained the most likely hypotheses for the sentence being decoded. However, since the subset could also contain the LM backoff node, almost any word in the lattice could be hypothesized at each word end, resulting in a very large number of possible hypotheses, slow recognition, and making expansion to trigram lattices difficult.

To correct this problem, we implemented a new bigram lattice algorithm based on a word-dependent N-best decoding pass [8] that creates backtrace entries which are then processed to create a lattice of hypotheses. This results in true lattices that do not contain the backoff node, and have a finite number of paths or hypotheses. We implemented two algorithms to expand the bigram lattices to trigram lattices. The first one is a conventional algorithm that creates a new node for every trigram context present in the bigram lattice. The second is a novel algorithm that exploits the fact that many trigram probabilities are computed using the bigram backoff, and creates compact trigram lattices that were found to be a factor of six smaller than the conventional algorithm. Since the size of the bigram lattices affects the size of the expanded trigram lattices, we also developed a bigram lattice reduction algorithm, which was found to give lattices which were a factor of two smaller.

Section 2 describes an algorithm that generates bigram lattices with good quality to serve as a base for trigram expansion. Section 3 presents a lattice compaction algorithm that tries to reduce lattice sizes while maintaining all the original hypotheses in the lattices. Section 4 gives two algorithms that expand bigram lattices to trigram lattices. Hub4-related experiments and results are given in these sections to show the effectiveness of the new algorithms. Finally, Section 6 summarizes our work.

2. New Bigram Lattice Generation Algorithm

The primary goal of designing a new bigram lattice generation algorithm is to generate simple bigram lattices with low lattice error rates for easy trigram expansion.

Our previous bigram word lattice algorithm intended to extract a subset of the full LM for an input utterance, so that the search can be narrowed in later-stage processing [3]. Unfortunately, its implementation was too simplistic in that the LM backoff node was retained in the lattice, causing almost any word in the lattice to be hypothesized at each word end, making recognition slow. It also prevented compact expansion to trigram lattices, because the connectivity of the backoff node to almost all the other nodes in a lattice results in O(n^2) trigram contexts, where n is the number of words in the lattice.

Our new bigram lattice generation algorithm is based on the word-dependent N-best algorithm [8], and is similar to [4, 5]. The algorithm assumes that the starting time for a word depends on the preceding word but not on any word before that. Thus, a separate word hypothesis is propagated for each possible predecessor word. When the word ends, the score of each word ending hypothesis along with the previous word is recorded in a backtrace array. The best hypothesis is propagated along with the current word label.

To generate a lattice, we process the backtrace array to cluster to-

Lattices	Rescor-ing LM	Male F0	Male F1	Male FX	Subset Total
Old bigram	trigram	14.1	35.1	60.1	33.5
New bigram	trigram	13.9	34.5	59.1	32.9
Old bigram	fourgram	13.5	34.5	59.2	32.8
New bigram	fourgram	13.5	34.3	58.6	32.6

Table 1: Comparison of results of the previous and new bigram lattice algorithms.

gether words that have the same name and ending time in the search. Each such cluster corresponds to a node in the lattice. If the word corresponding to cluster A starts the word corresponding to cluster B in the search, then node A is connected to node B in the lattice. In the lattice generated, neither time information nor acoustic information is preserved. Either type of information is easy to recover. However, removing it makes the lattices smaller, and thus easier to expand to trigram lattices. Thus, in our approach, lattices act just as constrained language models for the subsequent recognition pass.

To generate small lattices, it is necessary to control the pruning beam-width during the search. A single pass search could be used. However, we must then use a conservatively large beam-width as only partial hypotheses scores will be compared at each frame. This results in large lattices. In a forward-backward search, we can use significantly lower pruning beam-widths in the backward search, as the entire hypothesis score can be computed at every frame using the scores from the forward search. Controlling the backward pruning threshold allows the generation of lattices that are small enough for processing, and that have low enough error rate.

We have experimented with various forward-backward pruning thresholds to optimize the search for the Hub4 task. Before the evaluation, we conducted a set of experiments that used 1800/1200 forward/backward pruning thresholds for the old lattices and 1800/700 forward/backward pruning thresholds for new bigram lattices. Pruning thresholds specify the allowed difference between the partial hypothesis score and the current best score, measured in “bytelogs,” or $\frac{1}{1024}\log_{1.0001}$ of the hypothesis probability. We used a smaller backward pruning threshold for the new algorithm because backward pruning thresholds above 700 resulted in bigram lattices that were too large for trigram expansion. Backward pruning thresholds smaller than 700 were also generated but they produced much higher lattice error rates, and we only used them as a backoff value for isolated cases where the 700 threshold produced too large a lattice. Table 1 gives the recognition results on F0, F1, and FX conditions for the male speakers of the 1996 Hub4 development set, using previous and new lattice generation algorithms to generate bigram lattices and rescoring them with last year’s Hub4/SWBD/NABN 48K vocabulary trigram and fourgram LMs described in [10]. The acoustic models used for recognition are last year’s adapted models described in [6].

From the first two rows, we can see that the new algorithm gives about a 1.8% relative improvement over the old algorithm when rescored with a trigram LM. However, this improvement did not carry over to a fourgram LM rescoring. The last two rows in the table show no significant difference between the previous bigram lattices and the new bigram lattices when rescored with a fourgram LM. However, we observed an order of magnitude speedup by using the new bigram lattices for recognition as compared to the previous bigram lattices.

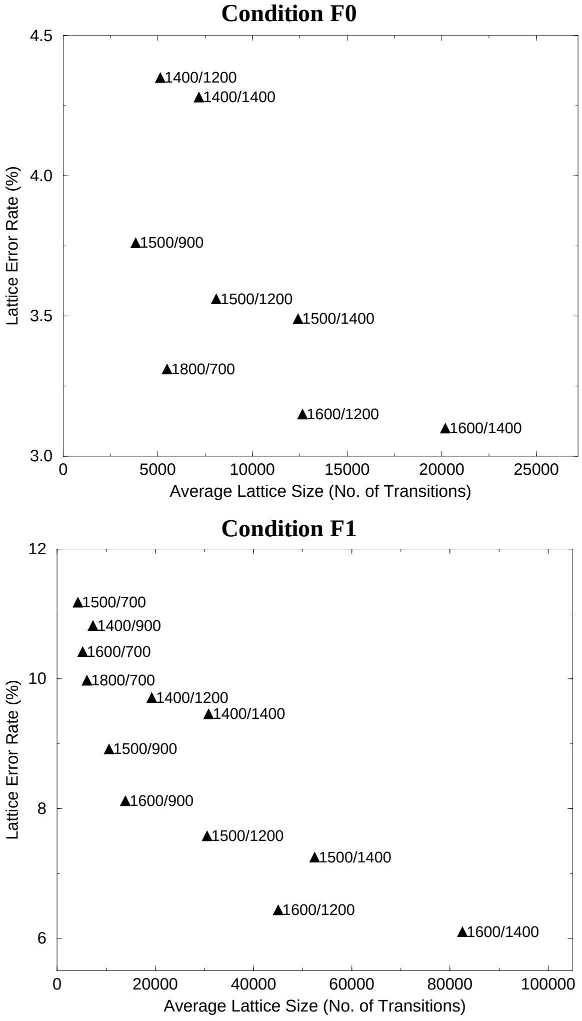


Figure 1: Average lattice sizes (number of transitions) and lattice error rates as a function of forward-backward beam width. Each datapoint corresponds to a pair of forward/backward pruning thresholds. The first graph shows result for condition F0, the second one for condition F1 (note the scales differ).

After the evaluation, further work was done on finding optimal pruning thresholds for generating bigram lattices. Figure 1 shows the dependence of average lattice size and lattice error rate on the choice of pruning parameters, for conditions F0 and F1 in the 1996 Hub4 development data. Based on these experiments, we selected 1600/1200 as the forward/backward pruning thresholds to get another set of bigram lattices using the new bigram lattice generation algorithm. When a resulting bigram lattice was too big for trigram expansion, we subsequently used 1500/1200 and 1500/900 as back-off values, similar to what we did during the evaluation. As shown in Figure 1, the new pruning thresholds reduced bigram lattice error rates over the old setting at 1800/700, by about 5% for F0, and by about 35% for F1. The attendant increase in lattice size was partially compensated by the lattice reduction algorithm decribed next.

3. Lattice Reduction

For very noisy speech, the bigram lattices generated in the forward-backward recognition pass are quite large and therefore difficult to expand. The usual way to obtain smaller bigram lattices for this type of data is to tighten the backward pruning threshold. As a result, the lattice word error rates increase. Here, we explore an alternative way to make the resulting lattices smaller. The idea is to combine identical (sub)paths in the lattices so that the redundant nodes and transitions are removed. Similar research has been reported in the computer science literature [1], where standard algorithms for minimizing deterministic finite state automata are given. More recently, algorithms for minimization of weighted transducers were also developed [2, 9]. These approaches differ from ours in several aspects. First, we are dealing with word lattices, a dual representation of finite state automata, where nodes are states and transitions are labeled with words. Second, and more important, our lattices are nondeterministic and there is no requirement that the resulting lattices be deterministic. Third, we need a fast algorithm to process our lattices. With these differences in mind, we designed a simple reduction algorithm that greatly reduce redundancy in our lattices.

The key observation underlying our algorithm is that if two nodes in the lattice have the same word label and the same set of successor (or predecessor) nodes, they can be merged without changing the language of the lattice, where the language of a lattice is defined as the set of all the word strings starting at the initial node and ending at the final node. Depending on whether we are merging nodes according to their predecessor node set or their successor node set, we can have either forward or backward reduction algorithms. Multiple iterations can also be performed. For simplicity, we only describe the backward reduction algorithm; the forward one is symmetrical.

Backward Reduction Algorithm: Let $S_{out}(n)$ and $S_{in}(n)$ be the set of successor nodes and the set of predecessor nodes of node n, respectively. Let $word(n)$ denote the word name of a lattice node n.

```
• For each lattice node n in reverse topological order (starting with the final node):
    – for each pair of predecessor nodes $(i, j)$ of node n:
        ∗ if $word(i) = word(j)$ and $S_{out}(i) = S_{out}(j)$, then merge nodes $i$ and $j$
```

Experiments were conducted with one iteration of the backward reduction algorithm, on the F0 and F1 portions of the 1996 Hub4 development set. Using 1600/1200 as the forward-backward pruning thresholds, the algorithm reduced lattice sizes by about 50%, as shown in Table 2. However, using an unadapted version of this year's acoustic model [7], we observed no statistically significant difference in recognition accuracy between original and reduced lattices (Table 3).

4. Algorithms for Expansion to Trigram Lattices

One of the main purposes of this work is to use trigram LMs at an early stage. The central component is the algorithm that expands a bigram lattice to a trigram lattice, i.e., a lattice whose transitions contain the trigram probabilities. Thus, more context information is encoded in the lattice and used in the subsequent re-recognition pass. We note that for simplicity and concreteness, our discussion here is focussed on trigram lattices, but that the algorithms described generalize to N-gram models of higher order.

	F0	F1	Subset Total
Before Reduction	12641	45033	30083
After Reduction	6777	23892	15993

Table 2: Bigram lattice sizes before and after reduction.

To place trigram probabilities on the lattice transitions we must create a unique two-word context for each transition. For example, in Figure 2, a node labeled *c* and its transitions (c, d) and (c, e) are duplicated to guarantee the uniqueness of the trigram contexts for placing $p(d|bc)$ and $p(e|bc)$ on the transitions (c, d) and (c, e), respectively. When a central node with label c has two predecessor nodes labeled with the same word a, only one additional node and its corresponding outgoing transitions need to be duplicated, as shown in Figure 3.

The conventional trigram expansion algorithm, presented below, works by duplicating nodes and transitions in the manner indicated throughout the lattice.

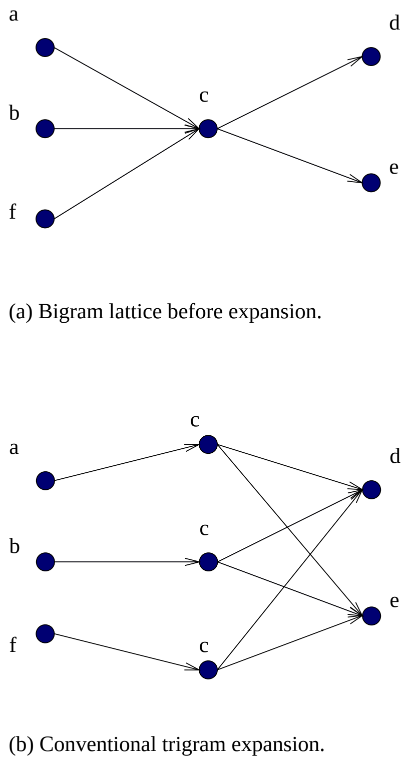
Conventional lattice expansion algorithm:

```
• For each node $n$ of the lattice, in topological order:
    – For each predecessor node $i$ of $n$:
        ∗ for each successor node $k$ of $n$:
            · if a node $j$ with $word(n)$ was already created for trigram context $(word(i), word(n))$ and $word(k)$, connect node $i$ to node $j$.
            · otherwise, create node $j$ and label it with $word(n)$, connect node $i$ to node $j$ and node $j$ to node $k$, put $p(word(k)|word(i)word(n))$ on transition $(j, k)$
    – remove node $n$ and all its incoming and outgoing transitions
```

While this algorithm correctly implements trigram probabilities in the lattice, it does so at a considerable increase in lattice size. On our Hub4 development set, the number of lattice transitions increased about 10-fold using the conventional approach. We therefore developed an alternative expansion algorithm that takes advantage of the backoff structure of the N-gram model. For most trigram language

	F0	F1	Subset Total
Before Reduction	18.81	37.25	29.06
After Reduction	18.79	37.25	29.05

Table 3: 1-best word error rates using bigram lattice before and after reduction.



(a) Bigram lattice before expansion.

(b) Conventional trigram expansion.

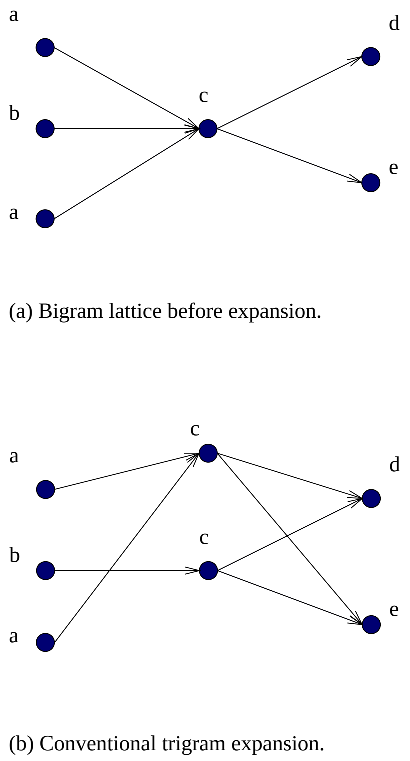
Figure 2: Conventional expansion of a bigram lattice to a trigram lattice when all incoming nodes have different labels.

models, the number of trigrams is much smaller than the number of all possible trigrams. If we can share the bigram backoff weights for trigram contexts, then we need to duplicate only enough nodes to uniquely represent the explicit trigram probabilities in the lattice.

The idea underlying the algorithm is to factor backed-off trigram probabilities $p(w_{i+2}|w_i w_{i+1})$ into the backoff weight $bo(w_i w_{i+1})$ and the bigram probability $p(w_{i+2}|w_{i+1})$, and to multiply the backoff weight onto the weight of the (w_i, w_{i+1}) transition, while keeping only the bigram estimate on the (w_{i+1}, w_{i+2}). Thus, no node duplication is required. Since backoff weights and probabilities combine multiplicatively, the total score along a path from w_i through w_{i+1} to w_{i+2}) will include the correct trigram probability $p(w_{i+2}|w_i w_{i+1})$.

Figure 4 illustrates the compact expansion idea given that there is only one explicit trigram probability $p(d|ac)$. Notice that only one node labeled c and its incoming transition from the node labeled a and outgoing transition to the node labeled d are created. $p(d|ac)$ is placed on the transition from the newly created node to the node labeled d. The weight on transition from the node labeled a to the newly created node was copied from the weight on the transition from the node labeled a to the original node labeled c. After the explicit trigrams are processed, the outgoing transitions from the original node labeled c are weighted with their corresponding bigram probabilities $p(d|c)$ and $p(e|c)$. Furthermore, bigram backoff weights $bo(a, c)$, $bo(b, c)$, and $bo(f, c)$ are multiplied onto the corresponding incoming transitions of the original node labeled c.

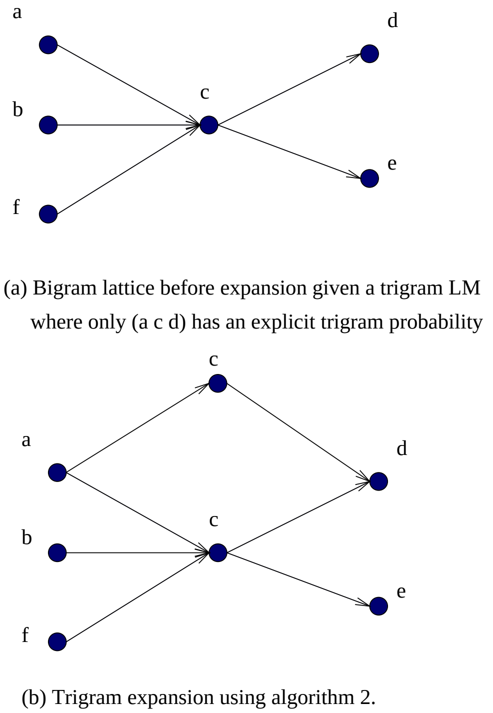
A potential problem for the compact algorithm is that even for ex-



(a) Bigram lattice before expansion.

(b) Conventional trigram expansion.

Figure 3: Conventional expansion of a bigram lattice to a trigram lattice when some incoming nodes have a same label.



(a) Bigram lattice before expansion given a trigram LM where only (a c d) has an explicit trigram probability

(b) Trigram expansion using algorithm 2.

Figure 4: Compact expansion of a bigram lattice to a trigram lattice.

plicit trigram probabilities, the lattice retains a path using the backoff transitions, which might have a higher weight than the correct trigram transition and therefore be preferred during search. For example, in Figure 4(b), there are two paths labeled (a, c, d), and during search the incorrect lower path will be chosen if $p(d|ac) < p(d|c) * bo(ac)$. Our solution to this problem is to preprocess the trigram LM to eliminate all trigram probabilities that are lower than the corresponding (improper) backoff estimate, and to renormalize the LM. Experiments show that in practice, this only eliminates a small fraction of trigrams. As shown below, there is no significant difference between lattices containing improper transitions and those created from the pruned LM.

Compact lattice expansion algorithm: Let $weight(i, j)$ be the aggregate probability on transition (i, j).

For each node n in the lattice in topological order:

```
• for each predecessor node $i$ of $n$:
    – for each successor node $k$ of $n$:
        ∗ if there is an explicit trigram probability for $(word(i), word(n), word(k))$,
            · if a node $j$ with $word(n)$ was already created for trigram context $(word(i), word(n))$ and $word(k)$, connect node $i$ to node $j$
            · otherwise, create node $j$, label it with $word(n)$, connect node $i$ to node $j$ and node $j$ to node $k$, and set $weight(j, k) = p(word(k)|word(i)word(n))$
        ∗ otherwise, mark transitions $(i, n)$ and $(n, k)$
    – if transition $(i, n)$ is not marked, remove $(i, n)$; otherwise, set $weight(i, n) = weight(i, n) * bo(word(i), word(n))$
• for each end successor node $k$ of $n$:
    – if transition $(n, k)$ is not marked, remove $(n, k)$;
    – otherwise, set $weight(n, k) = p(word(k)|word(n))$
• if no incoming transitions are marked, remove node $n$ and all its incoming and outgoing transitions.
```

5. Lattice Expansion Experiments

Experiments were conducted before and after the 1997 Hub4 evaluation. Because of time constraints, we could only test the conventional expansion algorithm before the evaluation. Trigram lattices were obtained from bigram lattices using 1800/700 as the forward-backward pruning thresholds and the backoff procedure described in Section 2. Recognition on male F0, F1, and FX conditions of the 1996 Hub4 development data with last year's adapted acoustic models [6] was performed on these trigram lattices through conventional expansion.

Contrastive results are given in Table 4. The first row shows the results when using the old lattice generation algorithm to generate bigram lattices, generating N-best lists from these lattices, and rescoring the N-best hypotheses with a trigram LM. The second row gives results using the new lattice generation algorithm instead to generate bigram lattices. The last row shows the results of 1-best recognition on conventionally trigram lattices, expanded from bigram lattices obtained with the new lattice generation algorithm. We see a 1.8% improvement with the new lattice generation algorithm and better than a 3% improvement with the expanded trigram lattices.

	Male F0	Male F1	Male FX	Subset Total
Old bigram lat. N-best rescoring	14.1	35.1	60.1	33.5
New bigram lat. N-best rescoring	13.9	34.5	59.1	32.9
New bigram lat. Tri lat. expansion	13.5	33.1	57.9	31.9

Table 4: Improvement with new lattice algorithms.

After the 1997 Hub4 evaluations, we conducted further experiments with trigram lattice expansion using the reduced bigram lattices described in Section 3. As we noted before, these lattices have smaller lattice error rate than the bigram lattices we used for the evaluations. We expanded the reduced bigram lattices using both the conventional and compact trigram lattice expansion algorithms. It was found that the compact expansion algorithm was ten times faster than the conventional algorithm. Further, Table 5 shows that the size of the trigram lattices from the compact expansion algorithm is only a about a sixth of those from the conventional expansion algorithm.

Recognition experiments were carried out using the conventional and compact trigram lattices using this year's unadapted acoustic models [7]. Table 6 shows that there is no difference in performance between the conventional and compact trigram lattices. However, as we noted before, the compact lattices can be generated ten times faster than the conventional lattices, and are also six times smaller. As shown in the last row of Table 6, there is no significant difference for compact trigram expansion between using the original trigram LM versus one pruned to eliminate trigrams that lead to improperly weighted transitions.

Finally, we compared recognition performance using the conventional trigram lattices generated using the bigram lattices used for the 1997 evaluations, and the post-evaluation bigram lattices. Recall the difference between these lattices is the forward-backward pruning thresholds used, with the result that the post evaluation lattices had a lower lattice error rate. Again we used this year's unadapted acoustic models [7]. Yet, as shown in Table 7, there was no significant change in performance with these differently pruned lattices.

6. Summary

We have reported recent improvements of lattice-based search techniques in our Hub4 system. Results show that a new bigram lattice

Expansion Algorithm	F0	F1	sub-total Total
Conventional	123107	488738	319985
Compact	29113	76396	54573

Table 5: Trigram lattice sizes in terms of average number of transitions.

Expansion Algorithm	F0	F1	Subset Total
Conventional	14.77	32.02	24.36
Compact	14.68	32.32	24.49
Compact(pruned LM)	14.68	32.31	24.48

Table 6: 1-best recognition word error rates of trigram lattices expanded with different algorithms.

generation algorithm, together with the expansion of trigram lattices, results in about 5% improvement over our previous approach of trigram rescoring N-best lists from the previous bigram lattices. Using trigram lattices alone results in about a 3% improvement over trigram rescoring of N-best from bigram lattices. A significant speedup of recognition with the new lattices was also observed, enabling us to develop algorithms more efficiently.

In addition to these experiments, we have been working on various pruning techniques to achieve smaller and higher-quality lattices. We optimized the global pruning by controlling forward-backward search beam-widths, and we developed a lattice reduction algorithm to reduce lattice sizes about 50% while maintaining all the original paths. Furthermore, we developed a new trigram lattice expansion algorithm, which speeds up expansion by a factor of ten as compared with the conventional algorithm, and also results in a factor of six reduction in trigram lattice size.

Ackowledgment

This work was sponsored by DARPA through the Naval Command and Control Ocean Surveillance Center under contract N66001-94-C-6048.

References

1. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Mass., 1979.

2. M. Mohri and M. Riley. Weighted determinization and minimization for large vocabulary speech recognition. In *Proc. EUROSPEECH*, vol. 1, pp. 131–134, Rhodes, Greece, 1997.

3. H. Murveit, J. Butzberger, V. Digalakis, and M. Weintraub. Large-vocabulary dictation using SRI's DECIPHER speech recognition system: Progressive search techniques. In *Proc. ICASSP*, vol. II, pp. 319–322, Minneapolis, 1993.

4. H. Ney and X. Aubert. A word graph algorithm for large vocabulary, continuous speech recognition. In *Proc. ICSLP*, pp. 1355–1358, Yokohama, 1994.

5. J. Odell. *The Use of Context in Large Vocabulary Speech Recognition*. Ph.D. thesis, Cambridge University Engineering Department, Cambridge, U.K., 1995.

6. A. Sankar, L. Heck, and A. Stolcke. Acoustic modeling for the SRI Hub4 partitioned evaluation continuous speech recognition system. In *Proceedings DARPA Speech Recognition Workshop*, pp. 127–132, Chantilly, VA, 1997.

7. A. Sankar, F. Weng, Z. Rivlin, A. Stolcke, and R. R. Gadde. The development of SRI's 1997 Broadcast News transcription system. In *Proceedings DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, 1998.

8. R. Schwartz and S. Austin. A comparison of several approximate algorithms for finding multiple (N-BEST) sentence hypotheses. In *Proc. ICASSP*, vol. 1, pp. 701–704, Toronto, 1991.

9. N. Ström. *Automatic Continuous Speech Recognition with Rapid Speaker Adaptation for Human/Machine Interaction*. Ph.D. thesis, KTH, Stockholm, 1997.

10. F. Weng, A. Stolcke, and A. Sankar. Hub4 language modeling using domain interpolation and data clustering. In *Proceedings DARPA Speech Recognition Workshop*, pp. 147–151, Chantilly, VA, 1997.

Bigram Lattice	F0	F1	Subset Total
1997 Evaluation	14.74	31.96	24.32
Post-evaluation	14.77	32.02	24.36

Table 7: 1-best recognition word error rates of trigram lattices expanded from different bigram lattices.