Efficient Lattice Representation and Generation

Fuliang Weng *Andreas Stolcke* *Ananth Sankar*

Speech Technology and Research Laboratory
SRI International
Menlo Park, California
{fuliang,stolcke,sankar}@speech.sri.com

ABSTRACT

In large-vocabulary, multi-pass speech recognition systems, it is desirable to generate word lattices incorporating a large number of hypotheses while keeping the lattice sizes small. We describe two new techniques for reducing word lattice sizes without eliminating hypotheses. The first technique is an algorithm to reduce the size of non-deterministic bigram word lattices. The algorithm iteratively combines lattice nodes and transitions if local properties show that this does not change the set of allowed hypotheses. On bigram word lattices generated from Hub4 Broadcast News speech, it reduces lattice sizes by half on average. It was also found to produce smaller lattices than the standard finite state automaton determinization and minimization algorithms. The second technique is an improved algorithm for expanding lattices with trigram language models. Instead of giving *all* nodes a unique trigram context, this algorithm only creates unique contexts for trigrams that are explicitly represented in the model. Backed-off trigram probabilities are encoded without node duplication by factoring the probabilities into bigram probabilities and backoff weights. Experiments on Broadcast News show that this method reduces trigram lattice sizes by a factor of 6, and reduces expansion time by more than a factor of 10. Compared to conventionally expanded lattices, recognition with the compactly expanded lattices was also found to be 40% faster, without affecting recognition accuracy.[1]

1. INTRODUCTION

In large-vocabulary speech recognition systems, high-accuracy recognition is achieved with reasonable time and space demands through a multi-pass process [4], often using lattices as an intermediate representation. In the approach discussed here, all time and acoustic information is removed from the lattices, and a *word lattice* (or *word graph*) is generated, retaining only the language model (LM) probabilities. The lattice is then used as a constrained LM in subsequent recognition passes.

To incorporate higher-order LM probabilities, lattices typically undergo an expansion (node duplication) process [6]. It is desirable to generate lattices containing a large number of paths to minimize errors as a result of the multi-pass search. However, large lattices make expansion with higher-order N-gram LM prohibitive, and make subsequent recognition passes slow.

To address this problem, we developed two algorithms to reduce lattice sizes without changing the set of hypotheses encoded. Both algorithms are evaluated on the DARPA Hub-4 Broadcast News corpus. The first technique (described in Section 2), reduces the size of the bigram lattices generated in the first recognition pass. We also compare the new algorithm with the classic finite state automaton (FSA) determinization and minimization algorithms. The second technique (described in Section 3 and evaluated in Section 4) consists of a more compact expansion to trigram lattices. Section 5 concludes.

2. BIGRAM LATTICE REDUCTION

In SRI's latest Hub4 DECIPHER recognition system [11], bigram lattices are generated in the backward pass of a forward-backward two-pass search, based on the word-dependent N-best algorithm [8]. Backward pruning thresholds are used to control lattice sizes. The lattice generation method is similar to those described by [5] and [6].

Especially for noisy speech, the generated bigram lattices can be quite large and therefore costly to expand. A common approach in this case is to tighten the backward pruning threshold, but lattice error rates (the minimum word error by any path through the lattice) increase. Alternatively, we can try to reduce lattice sizes by removing redundant nodes and transitions, i.e., without changing the set of word hypotheses allowed by the lattice. Several algorithms with a similar goal have been developed. One approach is to view the lattices as finite state automata (FSAs) and to apply the classical FSA minimization algorithm [1]. More recently, algorithms for minimizing or reducing weighted word lattices have been developed [3, 9]. None of these approaches are directly applicable to our case because DECIPHER word lattices have words on nodes, rather than transitions, and are non-deterministic, i.e., the successor for a node is not uniquely determined by the following word. FSA determinization is thus required, a process with exponential worst-case time and space complexity (although [3] report very good performance in practice). We thus chose to develop a fast reduction algorithm that operates directly on non-deterministic lattices by eliminating local redundancies.

[1] The work reported here was funded by DARPA under contract N66001-94-C-6048

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

Table 1: Bigram lattice sizes before and after reduction.

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 set of word hypotheses encoded by the lattice. Depending on whether nodes are merged according to their predecessor node set or their successor node set, we have a 'forward' or a 'backward' reduction pass, respectively. To get the most out of this approach, forward and backward passes should be iterated until no more redundancies are found. For brevity, we describe only the backward reduction algorithm; the forward version is symmetrical.

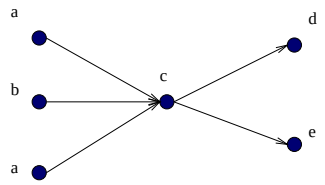
Backward reduction algorithm: Let $S_{out}(n)$ be the set of successor nodes of node n. Let $word(n)$ denote the word name of lattice node n.

- 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

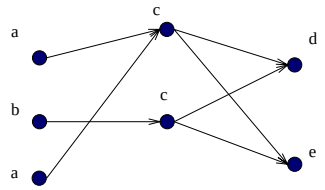
The runtime for this algorithm is proportional to the number of nodes times some constant that depends on the maximum fan-in and fan-out of the lattice nodes. A more aggressive reduction algorithm can be obtained if, instead of $S_{out}(i) = S_{out}(j)$, only a certain percentage of overlap between the two outgoing node sets is required for node merging. This will produce smaller lattices but might add new hypotheses to the lattice. We have not yet evaluated the tradeoffs associated with this variant.

Experimentally, we found that almost all of the eventual size reduction occurred in a single pass of the backward algorithm. This can be explained by the way the recognizer operates. Multiple hypotheses of the same word tend to be generated, starting at different neighboring frames, but ending at the same time. Furthermore, lattices tend to be more bushy at the beginning of an utterance; toward the end of the utterance pruning has eliminated a proportionally larger number of hypotheses. Both effects lead to node merging based on successors being more effective.

We evaluated the effectiveness of the reduction algorithm on lattices generated from the 1996 DARPA Hub4 development test set. Only the F0 (high-bandwidth, planned speech) and F1 (spontaneous speech) conditions of that set were included, with F1 generally giving considerably larger lattices. A single backward reduction pass reduced lattice sizes by about 50%, as shown in Table 1. Recognition from the reduced lattices gave a very small (not statistically significant) reduction in word error, which might be a result of fewer search errors.



(a) Bigram lattice before expansion.



(b) Conventional trigram expansion.

Figure 1: Conventional expansion of a bigram lattice to a trigram lattice when some incoming nodes have the same label.

We also compared our local reduction algorithm to the FSA determinization/minimization approach. We first converted our node-based word lattices into the dual FSA representation, a process which maps each node to exactly one FSA transition. We then performed FSA determinization and minimization using the AT&T FSM Toolkit [2]. Since bigram probabilities can always be retrofitted into a word lattice without changing its structure, we first set all transition probabilities to 1, effectively turning the weighted FSA operations into their classical, non-weighted counterparts. For comparison purposes, we then transformed the minimized FSA back into a node-based word lattice.[2]

We found that the local reduction algorithm produced slightly smaller lattices than the FSA-based determinization/minimization. The average number of transitions after FSA-based processing was about 5% larger for Hub4 F0 lattices, and 12% larger for F1 lattices. Recognition accuracies with both kinds of lattices were identical, as expected. With regard to recognition speed, the determinization/minimization approach could be advantageous because determinism narrows the search space at word transitions. However, this has to be balanced with other overhead in the recognizer that is proportional to the lattice size (such as time for input). On our test set, and using the DECIPHER recognizer, both non-deterministic reduced and determinized/minimized lattices gave virtually identical recognition times.

3. TRIGRAM LATTICE EXPANSION

The second approach to obtain smaller expanded N-gram lattices is to optimize the expansion step itself. Again, the purpose of N-gram lattice expansion is to allow higher-order N-gram probabilities

[2]The reverse conversion constructs a node for each unique pair of FSA node and incoming transition symbol. This produces best results if the FSA is deterministic. Conversions back and forth between the two representations are designed to be exact inverses.

to be assigned to the word transitions so as to increase accuracy in subsequent recognition passes. The discussion here is based on trigram lattices for simplicity, but the ideas generalize to higher-order N-grams.

To place trigram probabilities on the lattice transitions, we must create a unique two-word context for each transition. For example, in Figure 1, a node labeled c and its transitions (c,d) and (c,e) are duplicated to guarantee the uniqueness of the trigram contexts before placing probabilities $p(\cdot|ac)$ and $p(\cdot|bc)$ on the transitions. 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 must be duplicated. The conventional trigram expansion algorithm [5, 6] performs this node duplication exhaustively, as follows.

Conventional lattice expansion algorithm:

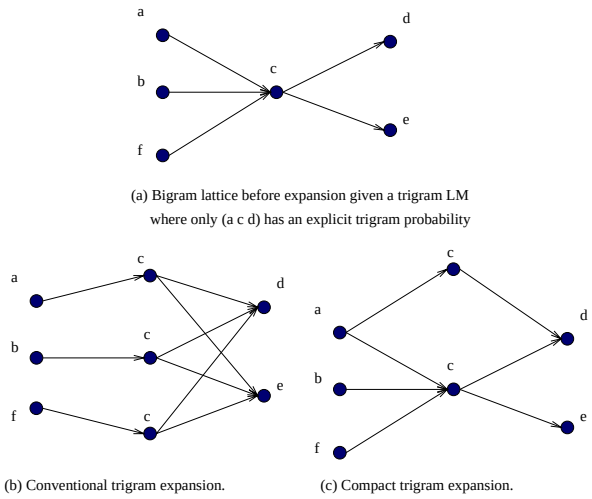
- For each node n of the lattice, in topological order:
  - For each predecessor node i of n:
    - for each successor node k of n:
      - if a node j with $word(n)$ was already created for trigram context $(word(i), word(n))$ and $word(k)$, connect node i to node j.
      - otherwise, create node j 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.

The conventional expansion algorithm ensures unique trigram histories by copying a node labeled w_i if it appears in *at least one* trigram $w_{i-1}w_iw_{i+1}$. However, one copy for each predecessor w_{i-1} in the lattice is created, even if those predecessors do not have a trigram in the LM. By contrast, the new expansion algorithm only creates one copy of w_i for each *explicit trigram* $w_{i-1}w_iw_{i+1}$ in the LM.

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

Figure 2 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 a node and outgoing transition to the d node are created. $p(d|ac)$ is placed on



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

(b) Conventional trigram expansion.

(c) Compact trigram expansion.

Figure 2: Compact expansion of a bigram lattice to a trigram lattice.

the transition from the new c node to the d node. The weight of the (a,c) transition is copied as well. After the explicit trigrams are processed, the outgoing transitions from the original c node are weighted with their corresponding bigram probabilities $p(d|c)$ and $p(e|c)$. Furthermore, bigram backoff weights $bo(ac)$, $bo(bc)$, and $bo(fc)$ are multiplied onto the corresponding incoming transitions of the original c node.

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 it
  - 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 it
  - 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.

Algorithm	F0	F1	Total
Conventional	123107	488738	319985
Compact	29113	76396	54573
Conventional/minimized	59559	207535	139238
Compact/minimized	59957	216387	144188

Table 2: Trigram lattice sizes in terms of average number of transitions.

A potential problem for this approach is that even for explicit 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 2b, there are two paths labeled (a,c,d), and during search the incorrect lower path will be chosen if $p(d|ac) < p(d|c) * bo(ac)$. The proper solution is to preprocess the trigram LM to prune all trigram probabilities that are lower than the corresponding (improper) backoff estimate, and to renormalize the LM. Experiments on Hub4 data showed that in practice, this eliminates only a small fraction of trigrams, not significantly changing the power of the LM or the final recognition results. We found that leaving the improper paths in the lattice also did not have a significant effect on recognition accuracy, compared to using the pruned LM.

4. LATTICE EXPANSION EXPERIMENTS

Experiments were conducted with both the conventional and the compact trigram expansion algorithms. The trigram LM used for expansion was SRI's 1996 Hub4 48,000-word trigram LM described in [10]. Using bigram lattices from the Hub4 F0 and F1 development test sets as the input lattices to the two algorithms, we found that the compact expansion algorithm was 10 times faster than the conventional algorithm. Furthermore, as shown in Table 2, the size of the compact trigram lattices is only about one sixth that of the conventional ones. We also applied the weighted determinization/minimization algorithms described in [3] and implemented by [2] to both conventionally and compactly expanded trigram lattices. As shown in the last two rows of Table 2, determinization/minimization reduced the size of conventional lattices by 56%. However, determinization and minimization more than doubled the size of compact trigram lattices. This is likely a result of the backoff structure, which introduces non-determinism into the lattice (see Figure 2c).

Recognition experiments were carried out using the (non-deterministic) conventional and compact trigram lattices with SRI's 1997 Hub4 unadapted acoustic models [7]. Word recognition error rates showed no difference in performance between the conventional and the compact trigram lattices. However, recognition speed with the compact lattices was 40% faster than with the conventional lattices.

Given the same time and memory limitations, the more compact lattice expansion step allowed us to relax the pruning during initial lattice generation, resulting in a decreased lattice error rates. Previously, lattices had been limited to 3.31% word error for the F0 condition and 9.98% for the F1 condition. Using less pruning and the more efficient expansion, lattice errors were reduced to 3.15% and 7.38%, respectively. At least with our present recognition system, however, we did not observe lower final 1-best error rates.

5. CONCLUSION

We have described two algorithms to keep word lattices small without sacrificing lattice or word recognition error rates. A bigram lattice reduction algorithm merges lattice nodes that can be shown to be locally redundant, halving the size of lattices obtained from our recognizer. Experimentally, the algorithm seems to be superior to an alternative approach based on FSA determinization and minimization. Furthermore, we developed a new trigram lattice expansion algorithm that reduces trigram lattice sizes by a factor of 6 and expansion time by a factor of 10. Recognition with the resulting lattices is 40% faster as compared to conventional trigram lattices. Due to reduced resource demands, we were able to significantly lower the lattice error rate using the new algorithm.

6. REFERENCES

1. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, 1979.
2. M. Mohri, F. Pereira, and M. Riley. FSM Library—general-purpose finite-state machine software tools. http://www.research.att.com/sw/tools/fsm/.
3. M. Mohri and M. Riley. Weighted determinization and minimization for large vocabulary speech recognition. In G. Kokkinakis, N. Fakotakis, and E. Dermatas, editors, *Proc. EUROSPEECH*, vol. 1, pp. 131–134, Rhodes, Greece, 1997.
4. 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.
5. H. Ney and X. Aubert. A word graph algorithm for large vocabulary, continuous speech recognition. In *Proc. ICSLP*, pp. 1355–1358, Yokohama, 1994.
6. J. Odell. *The Use of Context in Large Vocabulary Speech Recognition*. Ph.D. thesis, Cambridge University Engineering Department, Cambridge, U.K., 1995.
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*, pp. 91–96, 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.
11. F. Weng, A. Stolcke, and A. Sankar. New developments in lattice-based search strategies in SRI's Hub4 system. In *Proceedings DARPA Broadcast News Transcription and Understanding Workshop*, pp. 138–143, Lansdowne, VA, 1998.