



# On-the-fly Lattice Rescoring for Real-time Automatic Speech Recognition

Haşim Sak<sup>1</sup>, Murat Saraçlar<sup>2</sup>, Tunga Güngör<sup>1</sup>

<sup>1</sup>Dept. of Computer Engineering, Boğaziçi University, Bebek, İstanbul, Turkey

<sup>2</sup>Dept. of Electrical & Electronics Engineering, Boğaziçi University, Bebek, İstanbul, Turkey

hasim.sak@boun.edu.tr, murat.saraclar@boun.edu.tr, gungort@boun.edu.tr

## Abstract

This paper presents a method for rescoring the speech recognition lattices on-the-fly to increase the word accuracy while preserving low latency of a real-time speech recognition system. In large vocabulary speech recognition systems, pruned and/or lower order  $n$ -gram language models are often used in the first-pass of the speech decoder due to the computational complexity. The output word lattices are rescored offline with a better language model to improve the accuracy. For real-time speech recognition systems, offline lattice rescoring increases the latency of the system and may not be appropriate. We propose a method for on-the-fly lattice rescoring and generation, and evaluate it on a broadcast speech recognition task. This first-pass lattice rescoring method can generate rescored lattices with less than 20% increased computation over standard lattice generation without increasing the latency of the system.

**Index Terms:** on-the-fly lattice rescoring, automatic speech recognition, low latency ASR

## 1. Introduction

A speech recognition lattice is a weighted directed acyclic graph where each path from the start state to a final state represents an alternative transcription hypothesis, weighted by its recognition score for a given utterance [1]. For large vocabulary speech recognition, the  $n$ -gram language models are often pruned or the order of the language model is lowered for computational reasons when building the optimized search networks of the first-pass recognition. Then the output word lattices from the first-pass are rescored offline with unpruned language models or higher order  $n$ -grams. However, the real-time speech recognition systems such as the automatic closed captioning system by Saraçlar et al. [2] that require low-latency cannot benefit from the lattice rescoring as the latency increases with the size of the output lattice [3].

In this paper, we propose an algorithmic framework for rescoring lattices on-the-fly. The lattice generation method we employ is based on the lattice generation algorithm of Ljolje et al. [1]. Although the algorithm we use for rescoring recognition hypotheses is similar to the on-the-fly hypothesis rescoring algorithm of Hori et al. [4], there are major differences. First of all, two methods concentrate on different problems. While we employ a hypotheses rescoring method for producing rescored lattices on-the-fly, Hori et al. [4] uses a similar method in an on-the-fly composition algorithm setting to achieve fast and memory-efficient decoding in extremely large vocabulary continuous speech recognition. They decompose the search network into two transducer groups and a Viterbi search is performed based on the first transducer, whereas the second transducer is used to rescore the hypotheses and the updated scores

are used in the Viterbi search. Second, we propose a more general lattice rescoring framework in terms of enabling more general rescoring models rather than setting it up as a composition of two finite-state models as in [4].

## 2. WFST-based Speech Decoding

The weighted finite-state transducers (WFSTs) are weighted directed graphs in which each arc  $a$  has a source state  $S(a)$ , a destination state  $D(a)$ , an input label  $I(a)$ , an output label  $O(a)$ , and a weight  $P(a)$ . The WFSTs provide a unified framework for representing different knowledge sources in ASR systems, e.g., hidden Markov models (HMMs), context-dependent dependency networks, pronunciation lexicons, and  $n$ -gram language models [5].

In the WFST framework, the speech recognition problem is treated as a transduction from input speech signal to a word sequence. The various knowledge sources are represented as WFSTs. A typical set of knowledge sources consists of a context-dependency network  $C$  transducing context-dependent phones to context-independent phones, a lexicon  $L$  mapping context-independent phone sequences to words, and an  $n$ -gram language model  $G$  assigning probabilities to word sequences. The composition of these models  $C \circ L \circ G$  results in an all-in-one search network that directly maps context-dependent phone (corresponding to an HMM) sequences to weighted word sequences, where weights can be combinations of pronunciation and language model probabilities. The WFST also offers finite-state operations such as *composition*, *determinization* and *minimization* to combine all these knowledge sources into an optimized all-in-one search network.

The decoding for the best path in the resulting network is a single-pass Viterbi search. In this work, we implemented a WFST-based Viterbi speech decoder to experiment with the on-the-fly lattice rescoring using the OpenFst library [6].

### 2.1. One-Best Decoding

The decoding for the best path of arcs  $\mathbf{a} = a_1 \dots a_n$  from the initial state to a final state in a search network  $T$  given acoustic feature vectors  $\vec{x}$  for an utterance can be formulated as in [1]:

$$\max_{\mathbf{a}} P(\vec{x}[0, \tau], \mathbf{a}) = \max_{a, t_1, \dots, t_{n-1}} \prod_{i=1}^n P(\vec{x}[t_{i-1}, t_i] | I(a_i)) P(a_i)$$

The probability  $P(\vec{x}[t_{i-1}, t_i] | I(a_i))$  represents the likelihood assigned by the acoustic model for the context-dependent phone  $I(a_i)$  when applied to the feature vectors  $\vec{x}$  for the time interval  $t_{i-1}, t_i$ . The probability  $P(a_i)$  is the language model probability for the arc  $a_i$  in  $T$ .

We can formulate this equation in terms of the best scoring path to each state of  $T$  at a given time instant for the Viterbi algorithm. Let  $B(s)$  be the set of all paths  $a_1 \dots a_k$  in  $T$  from the initial state to state  $s$  and define that best scoring path as:

$$\alpha(s, t) = \max_{a \in B(s), t_1, \dots, t_{k-1}} \prod_{i=1}^k P(\vec{x}[t_{i-1}, t_i] | I(a_i))P(a_i)$$

Then:

$$\begin{aligned} \max_{\mathbf{a}} P(\vec{x}[0, t], \mathbf{a}) &= \\ \max_s \alpha(s, t) &= \\ \max_{D(a)=s} P(a) \left[ \max_{t' < t} P(\vec{x}[t', t] | I(a))\alpha(S(a), t') \right] \quad (1) \end{aligned}$$

In Equation (1), the Viterbi decoding of the best scoring path at a time instant is expressed as two nested (max) loops: the outer loop considers each possible active arc  $a$  ending in state  $s$ , and the inner loop finds the optimal start time  $t'$  for  $a$  by combining the acoustic likelihood of  $a$  between  $t'$  and  $t$  with the best path score from the start to state  $S(a)$  at time  $t'$ .

## 2.2. Lattice Generation

The lattice generation method for on-the-fly lattice rescoring algorithm that we propose is based on the the lattice generation algorithm of Ljolje et al. [1], which is repeated here briefly. Their lattice generation algorithm involves no extra computation over the normal Viterbi algorithm other than the negligible time needed to add states and arcs into the transducer lattice as it is being constructed. Each state of a phone-to-word transducer lattice  $L$  corresponds to a pair  $(t, s)$  of a time frame in the recognition and a state from the recognition transducer  $T$ . The initial state is the pair of utterance start time 0 and the start state of  $T$ . The final states are the pairs consisting the utterance end time  $\tau$  and a final state from  $T$ . If, during the Viterbi recursion of Equation (1), we have identified the optimal start time  $t'$  for arc  $a$  ending in state  $D(a)$  at time  $t$ , then a corresponding arc is added to  $L$  from state  $(t', S(a))$  to state  $(t, D(a))$ . If necessary, state  $(t, D(a))$  was first created; state  $(t', S(a))$  must already have been in  $L$  by induction. The new arc has input label  $I(a)$ , output label  $O(a)$ , and weight  $P(\vec{x}[t_{i-1}, t_i] | a)P(a)$ . All this information is readily available from the Viterbi recursion. We store the index for the pair  $(t, D(a))$  in the decoder active state data structure for  $D(a)$  at the current time, and we store the index for the pair  $(t', S(a))$  in the decoder active arc data structure for  $a$ . The generated phone-to-word transducer lattice can also be converted to a word lattice and pruned relative to the best scoring path through the entire lattice as explained in [1].

## 3. Lattice Rescoring

### 3.1. Lattice Rescoring with Composition

The word lattices output from a speech recognizer generally contain both the acoustic model and the language model scores for the hypotheses as explained in the previous section. For rescoring the lattices offline, the scores from the language model of the first-pass for a transcription hypothesis is subtracted from that hypothesis' lattice score and the resulting weighted finite-state automaton is intersected with the rescoring language model automata. However, with a simple modification to the lattice generation algorithm of [1] given in section 2.2 - by assigning only the acoustic model score  $P(\vec{x}[t_{i-1}, t_i] | a)$  to a new lattice arc and omitting the language model score  $P(a)$

- we can get rid of the score subtraction step. Then we can just intersect the output lattice with the better language model for rescoring. For short utterances, this method of rescoring is very effective and has a very small latency. However, for real-time speech recognition systems that require decoding long utterances as in Saraçlar et al. [2], this method is not feasible since the memory for generating and storing the lattice increases rapidly and the composition with large lattices leads to significant latency. This method also requires that the rescoring model could be represented as weighted finite-state automata, which may not be the case, for instance, for the discriminative language models with complex feature sets.

In the ASR experiments, we implemented this method as a baseline to compare with the on-the-fly rescoring method.

### 3.2. On-the-fly Lattice Rescoring

For on-the-fly lattice rescoring, we need an algorithm for lattice generation and rescoring recognition hypotheses. The lattice generation algorithm is based on the algorithm given in section 2.2. The recognition hypothesis rescoring methodology is conceptually similar to the on-the-fly hypothesis rescoring algorithm of Hori et al. [4], however the motivation and implementation of hypothesis rescoring is different.

Each state of a phone-to-word rescored transducer lattice  $R$  corresponds to a tuple  $(t, s, h)$  of a time frame in the recognition, a state from the recognition transducer  $T$ , and an  $n$ -gram history for a path arriving to state  $s$  at time instant  $t$ . Since there may be multiple paths arriving at a state at the same time during decoding with possibly different  $n$ -gram histories, we need to keep track of  $n$ -gram word histories for active arcs and states. The initial state is the tuple of utterance start time 0, the start state of  $T$ , and the sentence start symbol  $\langle s \rangle$ . The final states are the tuples consisting the utterance end time  $\tau$ , a final state from  $T$ , and the sentence end symbol  $\langle /s \rangle$ . If, during the Viterbi recursion of Equation (1), we have identified the optimal start time  $t'$  for arc  $a$  ending in state  $D(a)$  with an  $n$ -gram history  $h$  at time  $t$ , then a corresponding arc is added to  $R$  from state  $(t', S(a), h)$  to state  $(t, D(a), h')$  with updated history  $h'$ . If the output label  $O(a)$  of that arc  $a$  is  $\epsilon$ ,  $h'$  is the same with  $h$ . If not, then the new history  $h'$  is formed by dropping the oldest word and appending the output label  $O(a)$  to the history. If necessary, state  $(t, D(a), h')$  was first created; state  $(t', S(a), h)$  must already have been in  $L$  by induction. The new arc has input label  $I(a)$ , output label  $O(a)$ , and weight  $P(\vec{x}[t_{i-1}, t_i] | a)P(O(a)|h)$ .  $P(O(a)|h)$  is the language model score assigned by the rescoring language model to the current  $n$ -gram. If the output symbol is  $\epsilon$ , it is taken as 1 to use the acoustic model score. All this information is readily available from the Viterbi recursion. We store the index for the tuple  $(t, D(a), h')$  in the decoder active state data structure for  $D(a)$  at the current time, and we store the index for the pair  $(t', S(a), h)$  in the decoder active arc data structure for  $a$ .

#### 3.2.1. Implementation Details

Figure 1 shows a partial decoding process of the Viterbi search progressing in time. The upper part of the figure shows the states activated at a time instant  $t_i$ . Each active state stores a pointer to a list of rescoring tokens as shown above each state. In the rescoring token list, we store the forward acoustic score of an active state at the creation time of the list. This score is used to update the rescoring token scores with the accumulated acoustic score. During the Viterbi decoding, we store a list of maximum  $N$  rescoring tokens in each active state, where

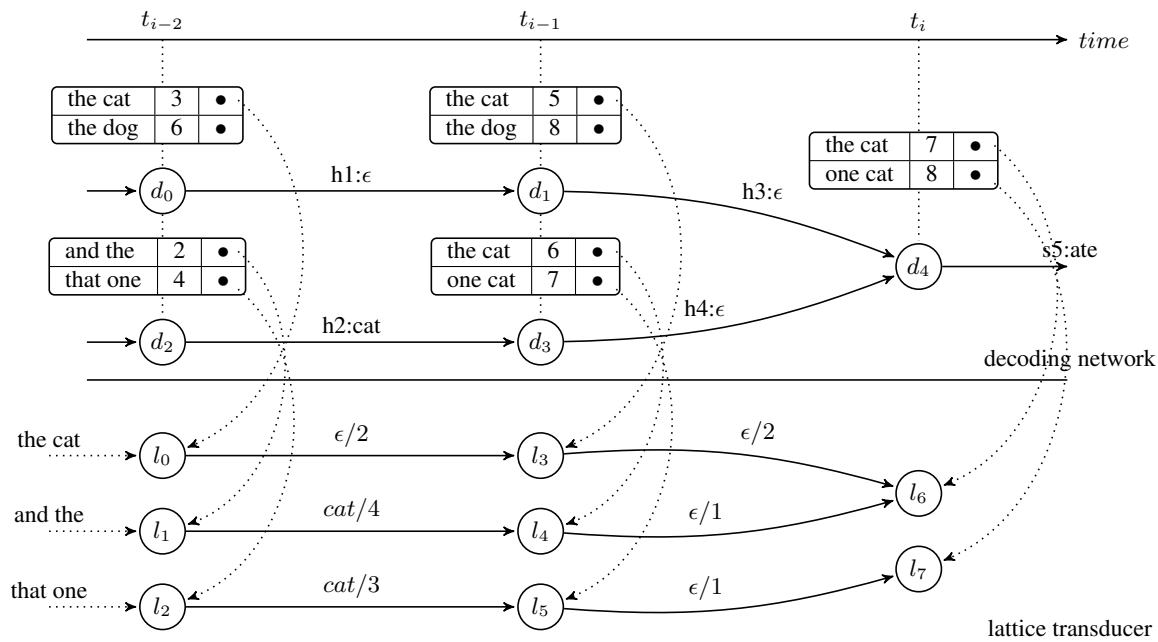


Figure 1: Hypotheses and associated lattice rescoring information during decoding.

each token represents a word trace (path) from the rescoring language model. A rescoring token contains the current  $n$ -gram history, the score for the current rescoring state, and a pointer to a lattice state that is being generated on-the-fly. When two hypotheses meet at the same active state, the rescoring token lists are merged and the maximum  $N$  tokens with the  $N$ -best scores are kept.

During decoding, each active arc has an associated HMM with a number of active HMM states. In Figure 1, the active HMM states are not shown for clarity, but the decoding process is very similar in the internal HMM states. When two hypotheses meet at the same HMM state in an active arc, only the rescoring token list having the token with the best score is retained. The pointers to the rescoring token lists are shared pointers with reference counting. Therefore in the internal Viterbi decoding of HMM states within an active arc, we can just propagate the pointers without copying the lists.

Each rescoring token in a rescoring token list stores a unique  $n$ -gram history to keep track of the state information in the rescoring language model. In Figure 1, we use a 3-gram rescoring language model, therefore we only need to store bigram word histories. This way of implementation provides a more general framework for rescoring, since we can also use rescoring models that cannot be represented as finite-state transducers. In the experiments, we used a 3-gram language model and used the SRILM toolkit [7] to train language models and assign probabilities to  $n$ -grams on-the-fly. However, it is also possible to use other language models such as a discriminative language model to assign scores. Note that we can also use discriminative models with acoustic, duration and language model based features such as [8] in the first pass by simple modifications to the proposed lattice rescoring algorithm.

As indicated above, we generate the word lattices as in Ljolje et al. [1], however one can also use the method by Saon et al. [9]. The lower part of Figure 1 shows the rescored lattice being generated on-the-fly. Each rescoring token has a pointer to a

lattice state as shown by dashed arcs from decoding network to the lattice network. When an active arc in the decoding network is expanded to a new state, that state is activated and the rescoring token list propagated from the active arc is updated with the new lattice pointers while adding new lattice arcs and states.

## 4. Experiments

We evaluated the performance of the rescoring algorithms on a Turkish broadcast news transcription task. The acoustic model uses hidden Markov models (HMMs) trained on 188 hours of broadcast news speech data [10]. In the acoustic model, there are 10843 triphone HMM states and 11 Gaussians per state with the exception of the 23 Gaussians for the silence HMM. The test set contains 3.1 hours of speech data that has been pre-segmented into short utterances (2,410 utterances and 23,038 words). We used the geometric duration modeling in the decoder.

The speech decoder is our implementation of a WFST-based decoder using the OpenFst library for finite-state operations and model representations [6]. We implemented the lattice generation algorithm of Ljolje et al. [1] to produce the lattices. The proposed algorithm for on-the-fly lattice rescoring has also been implemented in the decoder.

The text corpora that we used for building  $n$ -gram language models are composed of about 200 million-words BOUN NewsCorpus collected from news portals in Turkish and 1.3 million-words text corpus (BN Corpus) obtained from the transcriptions of the Turkish Broadcast News speech database [10]. The language model of the decoding network is a 3-gram language model with a vocabulary size of 200K words. This model is estimated by linearly interpolating two language models trained over the BOUN NewsCorpus and the BN corpus to reduce the effect of out-of-domain data. The language model trained on the BOUN NewsCorpus is pruned due to high memory requirements while building the optimized search net-

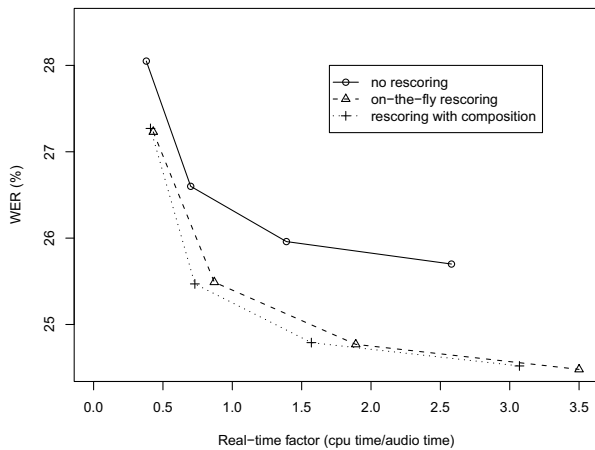


Figure 2: Word error rate versus real-time factor obtained by changing the pruning beam width

work using the SRILM toolkit [7]. The language model used for lattice rescoreing experiments use the unpruned language models with the same  $n$ -gram order of 3.

We give speech recognition results for three systems. All the systems are single-pass systems. In the first system, the decoder generates a lattice using the pruned search network without any rescoreing. The second system uses the pruned search network to generate a lattice containing only acoustic model scores and the resulting lattice is composed with the unpruned language model to rescore it as soon as the recognition for each utterance ends. The third system uses our proposed on-the-fly lattice rescoreing method.

Figure 2 shows the word error rate versus run-time factor obtained by varying prune beam widths from 9 to 12 for three systems. As expected rescoreing with unpruned language models improves the accuracy. Since the utterances in our test set are short, the baseline rescoreing method has not a significant effect on run-time for small beam-widths. The on-the-fly rescoreing method achieves about the same accuracy with the baseline rescoreing method. The proposed lattice rescoreing method can generate rescored lattices with less than 20% increased computation over standard lattice generation. However, the accuracy improvement for a real-time system (i.e. the real-time factor 1 in the graph) is very significant for the on-the-fly rescoreing method without increasing the latency of the baseline system with no rescoreing.

## 5. Discussion and Conclusions

We presented a general algorithmic framework for on-the-fly lattice rescoreing. Applications such as real-time closed captioning of news broadcasts require low-latency. In such applications, offline lattice rescoreing may not be used due to added latency of the rescoreing, which increases with the size of the output lattice. The proposed on-the-fly lattice rescoreing brings accuracy improvement of lattice rescoreing without increasing the latency of the system.

As a general framework, the language model for rescoreing can be any model that can assign scores to  $n$ -grams. For instance, it is possible to use a discriminative language model [11] to rescore the lattices in this framework. This is important since the discriminative models with complex feature sets cannot be

represented as finite-state automata. The lattices can even be rescored with multiple models. The implementation can also be easily modified to enable on-the-fly rescoreing with more complex models such as [8].

The resulting speech decoder can also be easily tweaked to do parameter estimation for discriminative language modeling using the simple perceptron algorithm. This has the advantage that the parameter estimation is done over dynamically generated lattices rather than static lattices.

## 6. Acknowledgements

This work was supported by the Boğaziçi University Research Fund under the grant numbers 06A102 and 08M103, the Scientific and Technological Research Council of Turkey (TÜBİTAK) under the grant numbers 107E261 and 105E102. Haşim Sak is supported by the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610 and TÜBİTAK BİDEB 2211. Murat Saraçlar is supported by the TUBA-GEBIP award.

## 7. References

- [1] A. Ljolje, F. Pereira, and M. Riley, "Efficient general lattice generation and rescoreing," in *Eurospeech*, 1999, pp. 1251–1254.
- [2] M. Saraçlar, M. Riley, E. Bocchieri, and V. Goffin, "Towards automatic closed captioning: Low latency real time broadcast news transcription," in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 2002.
- [3] H.-K. J. Kuo, B. Kingsbury, and G. Zweig, "Discriminative training of decoding graphs for large vocabulary continuous speech recognition," in *ICASSP*, 2007.
- [4] T. Hori, C. Hori, Y. Minami, and A. Nakamura, "Efficient WFST-based one-pass decoding with on-the-fly hypothesis rescoreing in extremely large vocabulary continuous speech recognition," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 15, no. 4, pp. 1352–1365, 2007. [Online]. Available: <http://dx.doi.org/10.1109/TASL.2006.889790>
- [5] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.
- [6] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *CIAA 2007*, ser. LNCS, vol. 4783. Springer, 2007, pp. 11–23, <http://www.openfst.org>.
- [7] A. Stolcke, "SRILM – an extensible language modeling toolkit," in *Proceedings of ICSLP*, vol. 2, 2002, pp. 901–904, <http://www.speech.sri.com/projects/srilm/>.
- [8] M. Lehr and I. Shafran, "Discriminatively estimated joint acoustic, duration and language model for speech recognition," in *ICASSP*, 2010.
- [9] G. Saon, D. Povey, and G. Zweig, "Anatomy of an extremely fast LVCSR decoder," in *Interspeech*, 2005, pp. 549–552.
- [10] E. Arısoy, D. Can, S. Parlak, H. Sak, and M. Saraçlar, "Turkish broadcast news transcription and retrieval," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 17, no. 5, pp. 874–883, 2009.
- [11] B. Roark, M. Saraçlar, and M. Collins, "Discriminative  $n$ -gram language modeling," *Computer Speech and Language*, vol. 21, no. 2, pp. 373–392, April 2007.