Improvements of Search Error Risk Minimization in Viterbi Beam Search for Speech Recognition

Takaaki Hori, Shinji Watanabe, Atsushi Nakamura

NTT Communication Science Laboratories, NTT Corporation, Japan

{hori,watanabe,ats}@cslab.kecl.ntt.co.jp

Abstract

This paper describes improvements in a search error risk minimization approach to fast beam search for speech recognition. In our previous work, we proposed this approach to reduce search errors by optimizing the pruning criterion. While conventional methods use heuristic criteria to prune hypotheses, our proposed method employs a pruning function that makes a more precise decision using rich features extracted from each hypothesis. The parameters of the function can be estimated to minimize a loss function based on the search error risk. In this paper, we improve this method by introducing a modified loss function, arc-averaged risk, which potentially has a higher correlation with actual error rate than the original one. We also investigate various combinations of features. Experimental results show that further search error reduction over the original method is obtained in a 100K-word vocabulary lecture speech transcription task.

Index Terms: speech recognition, beam search, pruning, search error, WFST

1. Introduction

The decoding process of speech recognition finds a sequence of words that best matches an input signal from among a large number of hypotheses. This search problem, which finds the best path in a decoding graph, essentially needs a large amount of computation especially in large vocabulary continuous speech recognition (LVCSR) tasks. To reduce the computation, many techniques have been proposed [1]. In such techniques, the most basic idea is beam search [2][3].

Current speech recognizers employ beam search to speed up the decoding process, in which unpromising partial hypotheses are pruned during decoding. However, the pruning step involves the risk of missing the best complete hypothesis by discarding a partial hypothesis that might grow into the best. Missing the best hypothesis is called *search error* and is one of the main causes of recognition errors.

A straightforward approach to reducing search errors is to use more accurate pruning techniques. For example, different criteria based on score, histogram, and word-end are introduced with different pruning thresholds [4]. In general, such thresholds need to be tuned experimentally to balance search error and decoding speed using development data.

To avoid this difficulty and reduce search errors, we proposed a novel pruning strategy based on search error risk minimization [5]. While conventional methods use heuristic criteria to prune hypotheses, our proposed method utilizes a pruning function whose parameters can be estimated automatically to minimize the search error risk in the training phase. The parameter estimation can use rich features extracted from each hypothesis, which enables a more precise decision on pruning than the conventional method.

Our approach is a sort of a search optimization framework proposed in the machine learning field [6][7]. However, it is actually difficult to apply the original learning algorithm of [6] to the LVCSR problem because it is computationally very expensive for LVCSR decoders. In our previous work, we introduced a risk-based loss function which can be calculated efficiently using recognition lattices, and showed that it could work successfully with a WFST-based speech recognizer in LVCSR tasks.

In this paper, we introduce a modified loss function and investigate effects of using various features in pruning to obtain further improvement of search efficiency. The loss function in our previous work was defined as an expected search error risk per frame, we refer to this as *frame-averaged risk*. The frame-averaged risk can be obtained by accumulating frame risks of a hypothesis over frames. However, it is not necessarily the best choice as a loss function because the risk at each frame is not additive over frames in terms of beam search. We point out this problem and propose *arc-averaged risk* in Section 3. In addition, while score and arrival arc features were only examined in our previous work, we evaluate other various features such as arrival node features, input symbol features, and their combinatorial features.

We conducted experiments with a 100K-word-vocabulary lecture speech transcription task, in which we employed a WFST-based speech recognizer including the optimized pruning function. Experimental results show that further search error reduction is achieved by using arc-averaged-risk loss function and combinatorial features. We also present some results of changing the sizes of lattices and training data used for optimization.

2. Search Error Risk Minimization

In this section, we describe the original method in our search optimization approach [5]. First, we explain the pruning function used in the decoding process, and next define the search error risk used as an objective function to optimize the pruning function. Finally we explain the optimization method to obtain appropriate parameters.

2.1. Pruning function

Let H_t be a set of hypotheses in the queue at time frame t in a time-synchronous Viterbi beam search. Let $\Phi(h, H_t)$ be a feature vector extracted from hypothesis h in H_t. We define a pruning function $d(h; H_t, \Lambda)$ as a linear discriminant model of the following form:

$$d(h; H_t, \Lambda) = \Lambda \cdot \Phi(h, H_t), \tag{1}$$

where Λ is a weight vector for feature vector $\Phi(h, H_t)$. We discard hypothesis h if $d(h; H_t, \Lambda) > 0$, or keep it if $d(h; H_t, \Lambda) \leq 0$. Only the hypotheses kept in the queue are expanded in the next time frame.

We must determine what features we extract for $\Phi(h, H_t)$ based on our prior knowledge of the search problem. In [5], we

used the following:

- Score-based beam feature:

$$\phi_S(h, H_t) = \left\{ \max_{g \in H_t} \alpha(g) - \alpha(h) \right\} - B_S, \quad (2)$$

  where $\alpha(h)$ is a Viterbi score of hypothesis h and B_S is a threshold; i.e., this indicates a result of conventional beam pruning.

- Arrival arc feature:

$$\phi_a(h, H_t) = \begin{cases} 1 & if\ \mathrm{a}[h] = a \\ 0 & if\ \mathrm{a}[h] \neq a, \end{cases} \quad (3)$$

  where a is an arc in the decoding graph such as a WFST. If we have K arcs in the graph, K feature components $\phi_{a_1}(h, H_t), \ldots, \phi_{a_K}(h, H_t)$ exist. $\mathrm{a}[h]$ indicates the latest arc reached by h in the graph.

Suppose a weight vector for the above features is $\Lambda = \{\lambda_S, \lambda_N, \lambda_{a_1}, \ldots, \lambda_{a_K}\}$. Then the pruning function is calculated as:

$$d(h; H_t, \Lambda) = \lambda_S \phi_S(h, H_t) + \lambda_{\mathrm{a}[h]}. \quad (4)$$

For the terms related to arrival arc features, only $\lambda_{\mathrm{a}[h]}$ remains because $\phi_{\mathrm{a}[h]}(h, H_t)$ is 1 and all the others are 0. Thus the pruning criterion changes depending not only on its score but also on its arrival arc.

2.2. Search error risk

The pruning function potentially discriminates hypotheses more precisely by extracting high-dimensional feature vectors. However, manually finding an appropriate set of parameters is not easy when the number of dimensions increases. Therefore, we define a loss function and find the parameters to numerically minimize the loss function in the training step. In this paper, we use an empirical loss based on the expectation of search error risk.

First we define the risk that hypothesis h is pruned by $d(h; H_t, \Lambda)$ as:

$$r(d(h; H_t, \Lambda)) = r(\Lambda \cdot \Phi(h, H_t)), \quad (5)$$

where $r(x)$ is a risk function that takes a value larger than 0 if $x \leq 0$ or gets close to 0 if $x < 0$. For example, sigmoid and hinge functions have this property.

Given utterance $O = o_1, \ldots, o_T$, which is an acoustic feature vector sequence of length T, we assume that the search error risk in decoding O is the following expected value of Eq. (5) averaged by the number of frames:

$$L(O; \Lambda) = \frac{\sum_{g \in H_T} P(g|O) \sum_{t=1}^{T} r(\Lambda \cdot \Phi(h_t(g), H_t))}{T}, \quad (6)$$

where $P(g|O)$ is the posterior probability of complete hypothesis g given O, and $h_t(g)$ is the partial hypothesis of g up to time t from the initial node. Since $P(g|O)$ is the relative score given by the decoder, loss $L(O; \Lambda)$ indicates an expected risk of missing high-scored complete hypotheses during the beam search. We refer to this expected risk of Eq. (5) as frame-averaged risk.

To obtain the empirical loss, it is impractical to compute Eq. (6) since we have to enumerate all possible complete hypotheses for each utterance. So we use the following approximation based on lattice representation:

$$L(O; \Lambda) \approx \frac{\sum_{\tilde{a} \in A(O)} P(\tilde{a}|O) r(\Lambda \cdot \bar{\Phi}(\tilde{a})) l_{\tilde{a}}}{\sum_{\tilde{a} \in A(O)} P(\tilde{a}|O) l_{\tilde{a}}} \quad (7)$$

where $A(O)$ is a set of arcs in the lattice generated by the decoder for O, and $P(\tilde{a}|O)$ is a posterior probability of lattice arc $\tilde{a}$ that can be derived from the forward-backward probabilities of the lattice. $\Gamma(\tilde{a})$ indicates an arc in the decoding graph, which $\tilde{a}$ corresponds to. $l_{\tilde{a}}$ is the duration time of lattice arc $\tilde{a}$. $\bar{\Phi}(\tilde{a})$ is a mean vector of the feature vectors in the lattice arc and can be computed as:

$$\bar{\Phi}(\tilde{a}) = \frac{\sum_{t=b_{\tilde{a}}}^{e_{\tilde{a}}-1} \sum_{h \in H_t : \mathrm{a}[h] = \Gamma(\tilde{a})} \Phi(h, H_t)}{l_{\tilde{a}}}. \quad (8)$$

The mean vector can be stored for each lattice arc when generating the lattice by the decoder. $b_{\tilde{a}}$ and $e_{\tilde{a}}$ are the starting and ending times of $\tilde{a}$, and therefore $l_{\tilde{a}} = e_{\tilde{a}} - b_{\tilde{a}}$.

2.3. Optimization

Now we show how to optimize the parameters using training corpus $\mathbf{O} = O_1, \ldots, O_M$ consisting of M utterances. Optimal parameter setting $\hat{\Lambda}$ is computed by minimizing the empirical loss and regularization penalty

$$\hat{\Lambda} = \arg\min_{\Lambda} \{R(\Lambda) + L(\mathbf{O}; \Lambda)\}, \quad (9)$$

where empirical loss $L(\mathbf{O}; \Lambda)$ is computed as

$$L(\mathbf{O}; \Lambda) = \frac{\sum_{m=1}^{M} L(O_m; \Lambda) T_m}{\sum_{m=1}^{M} T_m}. \quad (10)$$

We employ the gradient descent method to obtain $\hat{\Lambda}$, in which the parameters are iteratively updated in the following manner:

$$\Lambda' = \Lambda - \eta \frac{\partial \{R(\Lambda) + L(\mathbf{O}; \Lambda)\}}{\partial \Lambda}, \quad (11)$$

where η represents the learning rate. Once a set of lattices is generated for the training data, our proposed method can efficiently update Λ without looking at the data.

We introduced a regularization penalty $R(\Lambda)$ in Eq. (9). If we try to minimize only $L(\mathbf{O}; \Lambda)$, the decoding speed will be slower because $L(\mathbf{O}; \Lambda)$ is not convex for Λ and therefore does not converge. We constrain Λ by the regularization penalty that makes the objective function convex and ensures its convergence.

The penalty consists of two components. One is L2 norm $R_1(\Lambda) = ||\Lambda||^2$ commonly used in this kind of optimization, which prevents the absolute value of each coefficient in Λ from becoming too large. The other is a square of the expected value of the pruning function, $R_2(\Lambda)$, which ensures that the expected value is close to zero by conserving the degree of pruning and keeps the decoding speed constant. Details are described in [5].

Accordingly we use the following penalty:

$$R(\Lambda) = \zeta R_1(\Lambda) + \gamma R_2(\Lambda), \quad (12)$$

where ζ and γ are positive constants.

3. Arc-averaged-risk loss function

As mentioned in Introduction, the original loss function, i.e. the frame-averaged risk of Eq. (6), is not necessarily the best choice for reducing word errors occurring by search errors. Suppose a hypothesis takes risk r in consecutive 10 frames and another one takes a risk equal to $r \times 2$ at one frame. The frame-averaged risk of the former hypothesis is five times larger than that of the later one, where the risk in other frames is assumed to be zero. However, when we consider the principle of beam search, the potential of missing the former hypothesis could be lower

than that of the later because the best complete hypothesis can be lost by only one search error. This means that the risk at each frame is not additive over frames, unlike the previously proposed loss function in Eq. (6). Therefore, we need to focus on the magnitude of the risk rather than duration.

Since it is difficult to associate the search error risk with word error rate explicitly, we assume that when a search error occurs within an arc in a decoding graph, the arc is removed from the path found as the recognition result, and this increases at most one word error [1]. Based on this assumption, we define the arc-averaged-risk loss function, i.e. the expected risk per arc, which is calculated as an expected value of maximum pruning risk within each arc on the path of the hypothesis:

$$L(O;\Lambda) = \frac{\sum_{g\in H_T} P(g|O) \sum_{\tilde{a}\in A_g} \max_{t:b_{\tilde{a}}\le t<e_{\tilde{a}}} r(\Lambda\cdot\Phi(h_t(g),H_t))}{\sum_{g\in H_T} P(g|O)|A_g|}, \tag{13}$$

where A_g is a set of arcs on the path of hypothesis g, and $|A_g|$ represents the length of the path based on the arcs.

Consequently, the empirical loss $L(\mathbf{O};\Lambda)$ is computed as:

$$L(\mathbf{O};\Lambda) = \frac{\sum_{m=1}^{M} L(O_m;\Lambda)|A(O_m)|}{\sum_{m=1}^{M}|A(O_m)|}, \tag{14}$$

where

$$|A(O_m)| = \sum_{g\in H_{Tm}} P(g|O_m)|A_g|,$$

and H_{Tm} is the hypothesis queue at frame T_m of utterance O_m.

The expected loss based on lattice representation results in

$$L(O;\Lambda) \approx \frac{\sum_{\tilde{a}\in A(O)} P(\tilde{a}|O) r(\Lambda\cdot\hat{\Phi}(\tilde{a}))}{\sum_{\tilde{a}\in A(O)} P(\tilde{a}|O)}. \tag{15}$$

where

$$\hat{\Phi}(\tilde{a}) = \max_{t:b_{\tilde{a}}\le t<e_{\tilde{a}}} \sum_{h\in H_t:\mathrm{a}[h]=\Gamma(\tilde{a})} \Phi(h,H_t). \tag{16}$$

We assume here

$$r(\Lambda\cdot\hat{\Phi}(\tilde{a})) = \max_{t:b_{\tilde{a}}\le t<e_{\tilde{a}}} \sum_{h\in H_t:\mathrm{a}[h]=\Gamma(\tilde{a})} r(\Lambda\cdot\Phi(h,H_t)). \tag{17}$$

Note that we need to choose features that satisfy this assumption. The score-based beam feature and binary features used in this work satisfy it if $\lambda_S > 0$.

4. Additional features

The number of dimensions of arrival arc feature is equal to the number of arcs in the decoding graph, which can be a very large number in LVCSR, and therefore difficult to estimate all the parameters reliably with limited data. We incorporate the following elements as lower dimensional features in the pruning function.

- Arrival node feature

$$\phi_n(h,H_t) = \begin{cases} 1 & if\ \mathrm{n}[h] = n \\ 0 & if\ \mathrm{n}[h] \neq n \end{cases} \tag{18}$$

where n indicates a node in the decoding graph, and $\mathrm{n}[h]$ is the node that h has arrived in. This feature represents a location of the hypothesis as well as the arrival arc feature.

- Input symbol feature

$$\phi_i(h,H_t) = \begin{cases} 1 & if\ \mathrm{i}[\mathrm{a}[h]] = i \\ 0 & if\ \mathrm{i}[\mathrm{a}[h]] \neq i \end{cases} \tag{19}$$

where i indicates an input symbol in the decoding graph, and $\mathrm{i}[a]$ is the input symbol associated with arc a. In a WFST for speech recognition, an input symbol corresponds to an emission probability distribution of HMM or a sequence of distributions. Thus, the feature is potentially effective for search errors caused by acoustic models.

We can also use a combination of those features. If we use all the features described in this paper, the pruning function can be computed as:

$$d(h;H_t,\Lambda) = \lambda_S\phi_S(h,H_t) + \lambda_{\mathrm{a}[h]} + \lambda_{\mathrm{n}[h]} + \lambda_{\mathrm{i}[\mathrm{a}[h]]}. \tag{20}$$

Since this computation consists of only a few multiplications and additions, there is still no problem in decoding even if we use all the features.

5. Experiments

We conducted experiments on a lecture speech transcription task from the Corpus of Spontaneous Japanese (CSJ) [8]. Speech recognition system SOLON developed at NTT Communication Science Labs was used for the experiments, in which a one-pass WFST-based decoder employs a pair of WFSTs to be composed during decoding by a fast on-the-fly composition technique [9].

We constructed 100K-word lexicon and trigram language models. The language model was trained using transcripts covering 2672 lectures in CSJ. The Kneser-Ney back-off method was applied with a frequency cutoff of 1 for the bigrams and trigrams.

Tied-state triphone HMMs with 5,000 states and 32 Gaussians per state were trained with Minimum Classification Error criterion [10] using 230 hours of lecture speech data. The data were digitized with 16-kHz sampling and 16-bit quantization and transformed to 39-dimensional acoustic features consisting of 12 MFCCs and a log-power, and their delta and delta-delta components. Cepstrum Mean Subtraction was applied to each utterance.

To estimate parameters (Λ) that minimize the search error risk defined in Eq. (6) or (13), we used 20 lectures (approximately 4 hours) which were different from the 230-hour data used for acoustic model training. A lattice was also generated for each utterance in advance. ζ and γ in Eq. (12) were experimentally determined.

The weight for score-based beam feature λ_S was fixed to 1. In addition, other binary features were defined only on the first one of the two WFSTs composed during decoding, i.e., the HMM-state-to-Unigram WFST ($HCLG_1$). The number of dimensions of the feature space resulted in 1 for the score-based beam, 193.525 for arrival arcs, 56,671 for arrival nodes, and 28,154 for input symbols. To obtain $\hat{\Lambda}$, we used an RPROP technique for updating Λ in the gradient descent method [10].

First we evaluated the arc-averaged-risk loss function proposed in this paper. Figure 1 shows the relationship between word accuracy (WACC) and decoding time (real time factor: RTF) when we changed the threshold for score-based beam B_S, where Λ was optimized for each B_S. Our approach based on

[1] Although one search error may actually induce more word errors, we simplify that as one word error.

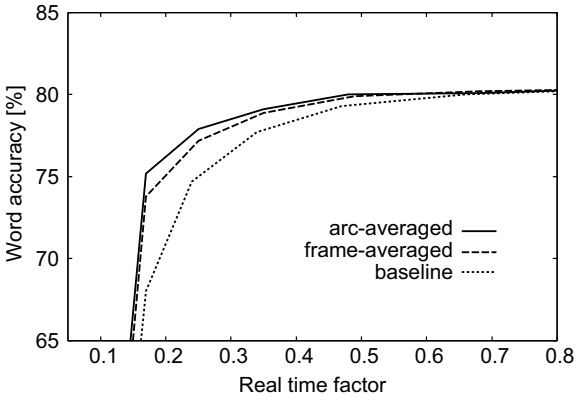

Figure 1: RTF vs. WACC in decoding with the baseline, frame-averaged, and arc-averaged risks.

Table 1: Features and word accuracy [%]

Feature				B_S			
ϕ_S	ϕ_a	ϕ_n	ϕ_i	110	120	130	140
○	-	-	-	68.0	74.7	77.7	79.3
○	○	-	-	75.2	77.9	79.1	**80.0**
○	-	○	-	75.5	77.9	79.3	79.8
○	-	-	○	73.0	77.0	78.7	79.8
○	○	○	-	76.1	**78.4**	**79.5**	**80.0**
○	○	-	○	75.3	77.9	79.2	79.9
○	-	○	○	75.2	77.8	79.2	79.8
○	○	○	○	**76.3**	**78.4**	**79.5**	79.9

the search optimization significantly outperformed the baseline which only used score-based pruning as regards word accuracy when it performed fast decoding. The arc-averaged-risk loss function further reduced the search errors compared to the original loss. Thus, we show the effectiveness of the arc-averaged-risk loss, and use it in the following experiments.

Second we compared several features for the pruning function. In Table 1, ϕ_S, ϕ_a, ϕ_n, and ϕ_i represent score-based beam, arrival arc, arrival node, and input symbol features, respectively. The circle "○" indicates the feature used in the pruning function. The results show that each feature has a certain effect on search error reduction, but that of Φ_i is smaller than those of Φ_a and Φ_n, and furthermore the combination of all the features yields a better accuracy than others.

Finally we conducted additional experiments on changing the sizes of lattices and training data used for optimization to investigate its potential of further error reduction. The results are shown in Table 2, where only score-based beam and arrival arc features were used except the condition of the last row in the table. "Raw lattice" means that the lattices generated from the decoder were used for optimization without post lattice pruning. "Pruned lattice" corresponds to the case we pruned lattice arcs based on a posterior probability of each arc. In the "1-best", we used the best path in each lattice. With these results, it is shown that using large lattices improves the accuracy of the pruning function in our framework. We also increased the training data from 20 lectures to 60 lectures (10.6 hours) in which the additional 40 lectures were a part of the 230-hour data set, and then used combinatorial features consisted of arrival arc, arrival node, and input symbol. With all the above attempts, we finally achieved approximately 60% search error reduction from the baseline in fast decoding conditions.

Table 2: Impact of lattice size and training data size on word accuracy [%]

	Lattice density	B_S 110	120	130	140
Baseline (no opt.)	-	68.0	74.7	77.7	79.3
Raw lattice	125	**75.2**	**77.9**	**79.1**	**80.0**
Pruned lattice	23	74.4	77.3	78.8	79.8
1-best	1	74.0	77.2	78.9	79.8
+Large training set w/ raw lattice	112	76.2	78.3	79.3	79.9
+Combinatorial feature	112	**76.9**	**78.6**	**79.5**	**80.0**

6. Conclusion

We have reported some improvements in a search error risk minimization approach to fast beam search for speech recognition. We introduced a modified loss function, arc-averaged risk, which potentially had a higher correlation with actual word error rate than the original one, and also investigated various combinations of features and the impact of lattice size and the amount of training data used for optimization. In a 100K-word vocabulary lecture speech transcription task, we achieved further search error reduction over the original method.

7. References

[1] X. L. Aubert, "An overview of decoding techniques for large vocabulary continuous speech recognition," *Computer Speech and Language*, vol. 16, pp. 89–114, 2002.

[2] B. Lowerre, "The HARPY speech recognition system," *PhD theses, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, USA*, 1976.

[3] H. Ney, R. Haeb-Umbach, BH. Tran, and M. Oerder, "Improvements in beam search for 10000-word continuous speech recognition," in *Proc. ICASSP*, 1992, vol. I, pp. 9–12.

[4] S. Ortmanns, A. Eiden, and H. Ney, "Improved lexical tree search for large vocabulary speech recognition," in *Proc. ICASSP*, 1998, vol. II, pp. 817–820.

[5] T. Hori, S. Watanabe, and A. Nakamura, "Search error risk minimization in viterbi beam search for speech recognition," in *Proc. ICASSP*, 2010, pp. 4934–4937.

[6] H. Daume III and D. Marcu, "Learning as search optimization: Approximate large margin methods for structured prediction," in *Proc. ICML*, 2005, pp. 169–176.

[7] Y. Xu and A. Fern, "On learning linear ranking functions for beam search," in *Proc. ICML*, 2007, pp. 1047–1054.

[8] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous speech corpus of japanese," in *Proc. LREC2000*, 2000, pp. 947–952.

[9] T. Hori, C. Hori, Y. Minami, and A. Nakamura, "Efficient WFST-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1352–1365, 2007.

[10] E. McDermott, T. J. Hazen, J. L. Roux, A. Nakamura, and S. Katagiri, "Discriminative training for large vocabulary speech recognition using minimum classification error," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 1, pp. 203–223, 2007.