RISK BASED LATTICE CUTTING FOR SEGMENTAL MINIMUM BAYES-RISK DECODING

Shankar Kumar and William Byrne

Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD 21218, USA
{skumar,byrne}@jhu.edu

ABSTRACT

Minimum Bayes-Risk (MBR) speech recognizers have been shown to give improvements over the conventional maximum a-posteriori probability (MAP) decoders through N-best list rescoring and A^* search over word lattices. Segmental MBR (SMBR) decoders simplify the implementation of MBR recognizers by segmenting the N-best lists or lattices over which the recognition is performed. We present a lattice cutting procedure that attempts to minimize the total Bayes-Risk of all word strings in the segmented lattice. We provide experimental results on the Switchboard conversational speech corpus showing that this segmentation procedure, in conjunction with SMBR decoding, gives modest but significant improvements over MAP decoders as well as MBR decoders on unsegmented lattices.

1. INTRODUCTION

A Minimum Bayes-Risk automatic speech recognizer [1, 2] attempts to find the sentence hypothesis with the least expected error under a given task specific loss function. If $l(W,W')$ is the loss function between word strings W and W', the MBR recognizer seeks the optimal hypothesis as

$$\hat{W} = \underset{W' \in \mathcal{W}}{\operatorname{argmin}} \sum_{W \in \mathcal{W}} l(W,W')P(W|A). \quad (1)$$

Here $\mathcal{W}$ is the set of all word strings allowed in the language of the recognizer. In practice, $\mathcal{W}$ is taken to be the set of most likely recognized word strings represented as an N-best list [2, 1] or a lattice [1]. $P(W|A)$ is the posterior distribution on word strings, usually obtained using an HMM acoustic model and an N-gram language model. The loss function for the word transcription task is the Levenshtein distance that measures the word error.

The search and sum in Equation 1 can be prohibitively expensive when $\mathcal{W}$ is large. However, if we reduce $\mathcal{W}$, perhaps by pruning, we risk search errors. The goal of *Segmental Minimum Bayes-Risk* decoding is to divide the single large search problem of Equation 1 into a sequence of smaller problems. We aim to do this by segmenting, or cutting, the lattice $\mathcal{W}$ into a sequence of smaller lattices. This is done as follows.

Suppose a word string $W \in \mathcal{W}$ is segmented into N substrings of zero or more words. Since each lattice path is a word string $W \in \mathcal{W}$, this segments the original lattice into N segment sets [3] $\mathcal{W}_i$, $i = 1, 2, ..., N$. We note that under Levenshtein distance, such a segmentation may restrict the potential string alignments between word sequences.

The concatenation of these segment sets yields a much larger search space. In addition to finding improved search efficiency through SMBR decoding, it is our goal to search over this larger space. To achieve this, we have to make the strong assumption that the loss function between any two word sequences $W, W' \in \mathcal{W}$ is not affected by the lattice cutting, i.e.

$$l(W,W') = \sum_{i=1}^{N} l(W_i, W'_i). \quad (2)$$

Under this assumption, we obtain the SMBR decoder as a sequence of independent decision rules

$$\hat{W}_i = \underset{W'_i \in \mathcal{W}_i}{\operatorname{argmin}} \sum_{W \in \mathcal{W}_i} l(W_i, W'_i)P_i(W_i|A), \quad (3)$$

where $\hat{W}$ is the concatenation of $\hat{W}_i$, $i = \{1, 2, ..., N\}$. Here $P_i(W_i|A)$ is the posterior probability of $W_i \in \mathcal{W}_i$, found via a lattice forward-backward procedure [4].

Previously, we have presented a method to segment word lattices into regions of low and high confidence [4]. This procedure relied on word confidence scores on the lattice links and time markers on the lattice nodes to guide segmentation.

Here, we discuss an alternate segmentation strategy that attempts to find good lattice segments by minimizing the total Bayes-Risk of all the word strings in the segmented lattice. Unlike the previous work, this method is guided by a risk criterion rather than by word confidences.

As mentioned previously, under a given segmentation, the alignments permitted between any two word strings from $\mathcal{W}$ may not include their optimal alignment associated with the Levenshtein string edit distance. Therefore the choice of a given segmentation involves a trade-off between two types of errors: search errors in the case of large lattice segments and the errors in approximating the loss function due to the segmentation.

2. RISK BASED LATTICE SEGMENTATION

Our goal is to obtain a lattice segmentation procedure that does not change the total Bayes-Risk of the word strings in the lattice. We now introduce an approximate procedure to achieve this goal. We begin by introducing the total Bayes-Risk of the word strings in the lattice.

The total Bayes-Risk of all the word strings in the lattice $\mathcal{W}$ can be written as:

$$R_T = \sum_{W' \in \mathcal{W}} \sum_{W \in \mathcal{W}} l(W,W')P(W|A) \quad (4)$$

Let $\tilde{W} = \tilde{w}_1^K$ denote the most likely (MAP) word string in the lattice: $\tilde{W} = \operatorname{argmax}_{W \in \mathcal{W}} P(W|A)$. Then we can approximate R_T by:

$$R_T \approx \sum_{W' \in \mathcal{W}} l(\tilde{W}, W')P(\tilde{W}|A). \quad (5)$$

This work was supported by the National Science Foundation under Grant No. #IIS-9810517 and Grant No. #IIS-9820687.

Suppose we can segment $W' \in \mathcal{W}$ into N substrings such that

$$l(\tilde{W}, W') = \sum_{i=1}^{N} l(\tilde{W}_i, W'_i). \quad (6)$$

Equation 5 can then be rewritten as:

$$R_T \approx P(\tilde{W}|A) \sum_{W' \in \mathcal{W}} \sum_{i=1}^{N} l(\tilde{W}_i, W'_i). \quad (7)$$

If a segmentation cannot be found to satisfy Equation 6, we would still have the right-hand side of Equation 7 as an upper bound on R_T. A segmentation that minimizes this upper bound would therefore attempt to minimize R_T. Hence a cutting procedure that satisfies Equation 6 for every $W' \in \mathcal{W}$ is optimal with respect to our risk-criterion.

In subsequent sections, we will describe a lattice cutting procedure that satisfies Equation 6. This procedure is based on operations involving Weighted Finite State Transducers (WFST) [5]. To develop this procedure, we will now describe the representation of word lattices as Weighted Finite State Acceptors (WFSA).

2.1. Word Lattices as Weighted Finite State Acceptors

A recognition word lattice $\mathcal{W}$ is a directed acyclic graph represented as a Weighted Finite State Acceptor (WFSA) [5] $\mathcal{W} = (Q, \Lambda, n_s, F, \mathcal{T})$ with a finite set of states (nodes) Q, a set of transition labels Λ, an initial state n_s, the set of final states F, and a finite set of transitions $\mathcal{T}$. The set Λ is the vocabulary of the recognizer. A transition in this WFST is given by $t = (p, q, w, s)$ where p is the starting state, q is the ending state, $w \in \Lambda$ is a word and s is a cost computed as the sum of the negative log acoustic and language model scores on the transition. A complete path through the WFSA is a sequence of transitions given by $T = \{(p_k, q_k, w_k, s_k)\}_{k=1}^{n}$ such that $p_1 = n_s$ and $q_n \in F$ and defines a word string w_1^n. Therefore we can write the joint acoustic and language model log-likelihood of a word string $W = w_1^n$ as $P(W, A) = -\sum_{k=1}^{n} s_k$.

2.2. Levenshtein Alignment through WFSTs

We now describe a finite state procedure to find the optimal alignment under the Levenshtein string edit distance between every word string $W' \in \mathcal{W}$ and the best path $\tilde{W}$. We will refer to this alignment as the Levenshtein alignment between the two strings.

We first transform the lattice $\mathcal{W}$ into an unweighted acceptor $\mathcal{W}_0$ by zeroing the scores on all lattice transitions. We also represent $\tilde{W} = \tilde{w}_1^K$ as an unweighted finite state acceptor whose transitions are given as $t = \{p, q, v\}$ where $v = \tilde{w}_k{:}k$ is a labeling that keeps track of both the words and their position in $\tilde{W}$.

To compute the Levenshtein distance, the possible one-symbol edits (insertion, deletion, substitution) and their costs can be readily represented by a one-state weighted transducer T [6]. This T accounts for the position indices on the output string $\tilde{W}$. Furthermore, we can reduce the size of this transducer by including only transductions that map words on the transitions of $\mathcal{W}_0$ to the words in the best path $\tilde{W}$.

We can now obtain all possible alignments between $W \in \mathcal{W}_0$ and $\tilde{W}$ by the following weighted finite state composition $A = \mathcal{W}_0 \circ T \circ \tilde{W}$. In terms of AT&T Finite State Toolkit [7] commands, these operations are given as:

fsmcompose T $\tilde{W}$ | *fsmcompose* $\mathcal{W}_0$ - > A

The transducer A has transitions that can be described by $t = (p, q, a, s)$ where a denotes an input-output symbol pair (w, v). There are three types of transitions: (1) $w \neq$ eps and $v = \tilde{w}_i{:}i$ indicates a substitution of word w by word $\tilde{w}_i$; (2) $w \neq$ eps and $v =$ eps shows that word w is an insertion; (3) $w =$ eps and $v = \tilde{w}_i{:}i$ shows a deletion with respect to $\tilde{w}_i$.

We now wish to extract the Levenshtein alignment between $W \in \mathcal{W}$ and $\tilde{W}$ from the transducer A. To do this, we perform (1) A sequence of operations that transforms A into a weighted acceptor A' and (2) A forward-backward algorithm on the acceptor A'.

The sequence of operations on A are as follows.

1. *Tag Input Labels of* A
   - Sort the nodes of A topologically. Let S be a queue of lattice states. For $q \in S$, the function $V(q) = i$ implies that all partial lattice paths ending at state q have been aligned with respect to $\tilde{w}_1^{i-1}$.
   - $S \leftarrow n_s$; $V(n_s) = 0$
   - while $S \neq \phi$
     (a) $p \leftarrow$ head(S). DEQUEUE(S).
     (b) For all transitions $t = (p, q, a, s)$ leaving p
        i. *Substitution:* If $a = (w, v)$ and $v = \tilde{w}_i{:}i$, replace the input word w as $w : i$ and set $V(q) = i + 1$.
        ii. *Deletion:* If $a = (\text{eps}, v)$ and $v = \tilde{w}_i{:}i$, set $V(q) = i + 1$.
        iii. *Insertion:* If $a = (w, eps)$ and $V(p) = j$, replace the input word w by w:eps:j and set $V(q) = V(p)$.
     (c) ENQUEUE(S,q).
2. Convert the resulting transducer from Step 1 into an acceptor by projecting onto the input labels. In terms of AT&T Finite State Toolkit commands, the operation is given by *fsmproject -i*.
3. For the weighted automaton generated in Step 2, generate an equivalent weighted automaton without ϵ-transitions. This step can be carried out by the AT&T FSM toolkit's *fsmrmepsilon* operation.

These three operations transform A into a weighted acceptor A'. A' contains the cost of *all alignments* between all lattice word strings and the best path.

We now describe a type of forward-backward procedure that will select the sub-lattice $\hat{A} \subseteq A$ which specifies only the optimal alignment between each word sting in the lattice and the best path. The algorithm can be described as follows:

1. Sort the states of A' topologically.
2. *Backward Pass*: Mark each state q by the cost of the best partial path starting from q and ending on a final state of A'. This is done recursively.
3. *Forward Pass*: Maintain a queue S of lattice states.
   $S \leftarrow n_s$.
   while $S \neq \phi$
   (a) $p \leftarrow$ head(S). DEQUEUE(S).
   (b) Let T denote the set of lattice transitions $t = (p, q, v, s)$, $v = w{:}i$ or $v = w$:eps:j, leaving the state p. Let U denote the set of unique words w on T.

(c) Initialize $T' = \{\}$. For each $u \in U$,

- Compute $\hat{t} = \text{argmin}_{t \in T: w=u}\, s + b_q$. Let $\hat{t} = (p, \hat{q}, \hat{v}, \hat{s})$.
- $T' \leftarrow \hat{t}$.
- ENQUEUE(S,$\hat{q}$).

(d) Prune all the transitions $t \in T; t \notin T'$

The role of Step 3c is to generate a set of transitions T' where each transition $t \in T'$ corresponds to a unique word leaving the node p. Among all the transitions with the same word w and leaving node p, the partial path starting with $t \in T'$ has the least cost. When the above algorithm terminates, we will have a pruned acceptor $\hat{A} \subseteq A'$ that contains the Levenshtein alignment of every $W' \in \mathcal{W}$ with respect to $\tilde{W}$.

The transitions of $\hat{A}$ are of the form $t = (p, q, v, s)$. Either (a) $v = w{:}i$ that indicates the word w has aligned with $\tilde{w}_i$ (substitution) or (b) $v = w$:eps:i indicates that word w occurs as an insertion before $\tilde{w}_i$. We can insert ϵ-transitions whenever the partial path ending on state q has aligned with $\tilde{w}_1^i$ and the partial path ending on q', a successor node of q has aligned with $\tilde{w}_1^{i+2}$. This will allow for deletions.

2.3. Risk Based Lattice Cutting

The previous section describes a procedure to obtain the Levenshtein alignment between every word string $W' \in \mathcal{W}$ and the MAP word string $\tilde{W} = \tilde{w}_1^K$. Using this alignment, we now cut $\mathcal{W}$ into K segments so that the substrings in the i^{th} segment set $\mathcal{W}_i$ align with the word $\tilde{w}_i$. This segmentation therefore satisfies Equation 6. This risk-based lattice cutting (RLC) procedure also yields $K-1$ node sets N_1,N_2,...,N_{K-1}, that cut $\mathcal{W}$ into K segments. A heavily pruned lattice $\mathcal{W}$ and its acceptor $\hat{A}$ are shown in the top and bottom panels of Figure 1. The bottom panel also displays the cuts obtained along the $K-1$ node sets. The segmentation procedure described above is optimal with respect to the MAP word hypothesis. However, the loss between arbitrary word strings in $\mathcal{W}$ will be increased by sub-optimal alignment due to the segmentation; therefore Equation 2 need not be satisfied.

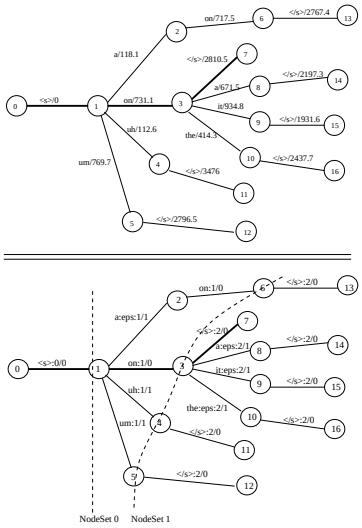

Fig. 1. (top) A word lattice $\mathcal{W}$ and (bottom) its acceptor $\hat{A}$ showing the Levenshtein alignment between $W \in \mathcal{W}$ and $\tilde{W}$ (shown as the path in bold).

Let $I = \{I_1, I_2, ..., I_r\} \subseteq \{1, 2, ..., K-1\}$ be a set of indices such that we segment the lattice $\mathcal{W}$ along the node sets $N_i; i \in I$. In addition, we also define indices I_0 and I_{r+1} corresponding to the initial state n_s and the set of final states F. Therefore, choosing I specifies a segmentation of $\mathcal{W}$. The loss between $W, W' \in \mathcal{W}$ under the segmentation I is given by

$$L_I(W, W') = \sum_{i=1}^{r+1} L(W_{I_{i-1}}^{I_i}, W'^{I_i}_{I_{i-1}}). \tag{8}$$

Let $I' = \{I_1, I_2, ..., I_{j-1}, I_{j+1}, ..., I_r\}$ be obtained from I by excluding the index I_j. We note that

$$\begin{aligned} & L_I(W, W') - L_{I'}(W, W') \\ &= L(W_{I_{j-1}}^{I_j}, W'^{I_j}_{I_{j-1}}) + L(W_{I_j}^{I_{j+1}}, W'^{I_{j+1}}_{I_j}) \\ &- L(W_{I_{j-1}}^{I_{j+1}}, W'^{I_{j+1}}_{I_{j-1}}) \geq 0. \end{aligned} \tag{9}$$

Therefore $L(W, W) \leq L_{I'}(W, W') \leq L_I(W, W')$, where the inequalities are a direct consequence of the definition of Levenshtein distance. Thus if we segment the lattice along fewer cuts, we obtain successively better approximations to the Levenshtein distance. However as the size of the lattice segments increases, SMBR decoding on the resulting segment sets will involve greater search errors. Our goal is therefore to choose a set I that will yield a "good" cutting procedure.

2.4. Periodic Risk-Based Lattice Cutting

We now provide one solution that balances the trade-off between search errors and errors in loss approximation. Here, we segment the lattice by choosing node sets N_i at equal intervals or periods. A period of k would imply that we choose the cuts N_1, N_{k+1}, N_{2k+1} and so on. Therefore the set $I = \{1, k+1, ..., nk+1\}$ where n is the largest integer such that $nk + 1 \leq K - 1$. We call this procedure *periodic risk-based lattice cutting* (PLC). If the loss function approximation obtained by cutting $\mathcal{W}$ into K segment sets (as in Section 2.3) is good, the cutting period k tends to be smaller and vice-versa. The choice of the cutting period is found experimentally to reduce the word error rate on a development set. We note that the RLC procedure is identical to the PLC procedure with period 1.

2.5. Construction of sublattices for SMBR Decoding

The preceding sections 2.1 to 2.4 discuss how to segment lattices under the loss function. This generates a sequence of sublattices $\mathcal{W}_i$ taken from the original lattice $\mathcal{W}$. To carry out SMBR, we assign scores to the links of the sublattices so that the sublattice assigns $P_i(W_i|A)$ to the word strings $W_i \in \mathcal{W}_i$, where the $P_i(W_i|A)$ are consistent with the scores in the original lattice $\mathcal{W}$. Each sublattice is then rescored via Equation 3.

3. PERFORMANCE OF SMBR DECODERS

We now report our experiments on the Switchboard LVCSR task. The lattice segmentation procedures were tested on the Switchboard2 portion of the 1998 Hub5 evaluation set (swbd2-98) and the Switchboard1 portion of the 2000 Hub5 evaluation set (swbd1-01). A description of the acoustic and language models used is given in the JHU LVCSR Hub5 system description [8]. The AT&T Large Vocabulary Decoder was used to generate an initial set of one-best hypotheses using HTK [9] cross-word triphone acoustic models,

trained on VTN-warped data, with a pruned version of SRI 33K trigram word language model. The one-best hypotheses were used to train MLLR transforms with two regression classes, for Speaker Adaptive Training (SAT) versions of the acoustic models. The decoder was then used to generate an initial set of lattices for the test set using MLLR and using pruned versions of SRI 33K word trigram language model. The initial lattices were rescored using the unpruned SRI 33K trigram language model and then again using SAT acoustic models with MLLR.

Given a lattice segmentation, several MBR procedures are available to carry out the calculation in Equation 3. An A^* procedure [1] attempts an exact, if heavily pruned, implementation. Alternatively, an N-best list can be generated from each $\mathcal{W}_i$, and an N-best approximation can be used in the search and sum of Equation 3 [1, 2]. A third procedure called Extended ROVER (e-ROVER) [10, 4] retains the sum over an N-best list but employs a greatly enlarged search space. Experiments using all three techniques were performed. In the latter two techniques, N-best lists of size 250 were used.

Figure 2 presents the performance of A^* SMBR decoding with periodic lattice cutting at cutting periods of 1 through 14. As can be seen, the optimal cutting period is 6 on this set (swbd2-98). We note that N-best rescoring and e-ROVER also achieved their optimal performance at period 6 on this test set. On swbd1-00 set, we found that all the procedures achieved their optimal performance at period 4. This suggests that cutting procedures should be tuned to the task to which they are applied.

Table 1 shows how SMBR rescoring performance varies under different lattice segmentation procedures. Rescoring of the entire lattice corresponds to no segmentation at all, while RLC is the most extensive segmentation. PLC was performed with a cutting period of 6 (on both the test sets). Under all 3 rescoring procedures, PLC gave the best performance, even improving over MBR rescoring of the original lattice. While it does provide improvements over the MAP baseline, RLC is less effective than even the MBR decoding on the original lattice. This shows the necessity of choosing a proper cutting period in these approaches.

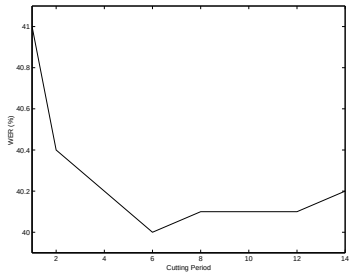

Fig. 2. Performance of *Periodic Lattice Cutting* with different cutting periods for A^* SMBR decoding on the swbd2-98 test set

4. DISCUSSION

We have introduced a risk based lattice cutting procedure for SMBR decoding of word lattices, and have described a particular implementation called the periodic risk-based lattice cutting. It attempts to segment the lattice while balancing the modeling errors that arise in approximating the loss function through segmentation against the MBR search errors. The procedure improves upon our lattice cutting work that was based on word confidence scores [4].

Our experiments have shown that MBR decoding procedures such as A^* search, N-best rescoring and e-ROVER are improved when applied to the lattice segments obtained by periodic cutting. Although the absolute WER improvements in WER are small, they are obtained as the final steps in a Hub-5 LVCSR evaluation system [8]. As an extension of the ROVER technique [3], lattice cutting can also be used in confidence assignment and in system combination. Hence these lattice cutting procedures have applications beyond the reduction of WER via MBR decoding.

Decoding Strategy		WER(%)	
		swbd2-98	swbd1-00
MAP (baseline)		41.1	26.0
N-best rescoring	Entire lattice	40.4	25.6
	RLC	41.0	25.9
	PLC	40.1	25.4
A^* search	Entire lattice	40.4	25.5
	RLC	41.0	25.9
	PLC	40.0	25.4
e-ROVER	Entire lattice	40.5	25.7
	RLC	40.5	25.7
	PLC	39.9	25.3

Table 1. WER Results from SMBR decoding on lattice segments

Acknowledgements We would like to thank Petr Podvesky of Charles University, Prague for useful discussions. We would also like to thank Michael Riley for use of the AT&T Large Vocabulary decoder and FSM Toolkit and Andreas Stolcke for the use of the SRI language model.

5. REFERENCES

[1] V. Goel and W. Byrne, "Minimum Bayes-Risk automatic speech recognition," *Computer Speech and Language*, vol. 14(2), pp. 115–135, 2000.

[2] A. Stolcke, Y. Konig, and M. Weintraub, "Explicit Word Error Minimization in N-Best List Rescoring," in *Eurospeech 1997*, Rhodes, Greece, 1997, vol. 1, pp. 163–165.

[3] J. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)," in *ASRU 1997*, 1997, pp. 347–354.

[4] V. Goel, S. Kumar, and W. Byrne, "Confidence based lattice segmentation and minimum Bayes-Risk decoding," in *Eurospeech 2001*, Aalborg, Denmark, 2001, vol. 4, pp. 2569–2572.

[5] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16(1), pp. 69–88, 2002.

[6] M. Mohri, F. Pereira, and M. Riley, "The design principles of a weighted finite-state transducer library," *Theoretical Computer Science*, vol. 231, no. 1, pp. 17–32, 2000.

[7] M. Mohri, F. Pereira, and M. Riley, *ATT General-purpose finite-state machine software tools*, 2001, http://www.research.att.com/sw/tools/fsm/.

[8] W. Byrne, "The JHU March 2001 Hub-5 Conversational Speech Transcription System," in *Proceedings of the NIST LVCSR Workshop*. NIST, 2001.

[9] S. Young et. al., *The HTK Book, Version 3.0*, July 2000.

[10] V. Goel, S. Kumar, and W. Byrne, "Segmental Minimum Bayes-Risk ASR Voting Strategies," in *ICSLP 2000*, Beijing, China, 2000, vol. 3, pp. 139–142.