

Segmental Minimum Bayes-Risk Decoding for Automatic Speech Recognition

Vaibhava Goel

I.B.M. T.J.Watson Research Center
Yorktown Heights, NY 10598, USA
vgoel@us.ibm.com

Shankar Kumar and William Byrne

Center for Language and Speech Processing,
Department of Electrical and Computer Engineering,
The Johns Hopkins University,
3400 N. Charles St., Baltimore, MD 21218, USA
{skumar,byrne}@jhu.edu

Abstract

Minimum Bayes-Risk (MBR) speech recognizers have been shown to yield improvements over the conventional maximum a-posteriori probability (MAP) decoders through N-best list rescoring and A^* search over word lattices. We present a Segmental Minimum Bayes-Risk decoding (SMBR) framework that simplifies the implementation of MBR recognizers through the segmentation of the N-best lists or lattices over which the recognition is to be performed. This paper presents lattice cutting procedures that underly SMBR decoding. Two of these procedures are based on a risk minimization criterion while a third one is guided by word-level confidence scores. In conjunction with SMBR decoding, these lattice segmentation procedures give consistent improvements in recognition word error rate (WER) on the Switchboard corpus. We also discuss an application of risk-based lattice cutting to multiple-system SMBR decoding and show that it is related to other system combination techniques such as ROVER. This strategy combines lattices produced from multiple ASR systems and is found to give WER improvements in a Switchboard evaluation system.

Index Terms

Minimum Bayes-Risk Decoding, Segmental Minimum Bayes-Risk Decoding, Lattice Cutting, extended-ROVER, ASR System Combination

I. INTRODUCTION

In ASR, an acoustic observation sequence $A = a_1, a_2, \dots, a_T$ is to be mapped to a word string $W = w_1, w_2, \dots, w_N$, where w_i are words belonging to a vocabulary \mathcal{V} .

We assume that a language \mathcal{W} is known; it is a subset of the set of all word strings over \mathcal{V} . This language specifies the word strings that could have produced any acoustic data seen by the ASR system. We further assume that the ASR classifier selects its hypothesis from a set \mathcal{W}_h of word strings. This set, called the *hypothesis space* of the classifier, would usually be a subset of the language. The ASR classifier can then be described as the functional mapping $\delta(A) : \mathcal{A} \rightarrow \mathcal{W}_h$.

Let $l(W, W')$ be a real valued loss function that describes the cost incurred when an utterance W belonging to language \mathcal{W} is mistranscribed as $W' \in \mathcal{W}_h$. An example loss function, the one that we focus on in this paper, is Levenshtein distance [1] that measures the minimum string edit distance (word error rate) between W and W' . This loss function is defined as the minimum number of substitutions, insertions and deletions needed to transform one word string into another.

Suppose the true distribution $P(W, A)$ of speech and language is known. It would then be possible to measure the performance of a classifier δ as

$$R(\delta(A)) = E_{P(W,A)}[l(W, \delta(A))]. \quad (1)$$

This is the expected loss when $\delta(A)$ is used as the classification rule for data generated under $P(W, A)$. Given a loss function and a distribution, the classification rule that minimizes $R(\delta(A))$ is given by [2]

$$\delta_R(A) = \operatorname{argmin}_{W' \in \mathcal{W}_h} \sum_{W \in \mathcal{W}} l(W, W') P(W|A). \quad (2)$$

We note that while the sum in Equation 2 is carried out over the entire language of the recognizer, only those word strings with non zero conditional probability $P(W|A)$ contribute to the sum. Let \mathcal{W}_e denote the subset of \mathcal{W} such that

$$\mathcal{W}_e = \{W \in \mathcal{W} | P(W|A) > 0\}. \quad (3)$$

Equation 2 can now be re-written as

$$\delta_R(A) = \operatorname{argmin}_{W' \in \mathcal{W}_h} \sum_{W \in \mathcal{W}_e} l(W, W') P(W|A). \quad (4)$$

We shall refer to the sum $\sum_{W \in \mathcal{W}_e} l(W, W') P(W|A)$ in Equation 4 as *conditional risk* and classifier given by this equation as the *Minimum Bayes-Risk* (MBR) classifier.

The set \mathcal{W}_e serves as the evidence for the MBR classifier using which it selects the hypothesis. Therefore, we shall refer to \mathcal{W}_e as the *evidence space* for the acoustic observations A . The distribution $P(W|A)$ describes the evidence space and shall be referred to as the *evidence distribution*.

Our treatment so far assumes that the true distribution over the evidence is available, however this is not the case in practice. This distribution is obtained by applying Bayes rule

$$P(W|A) = P(W)P(A|W)/P(A), \quad (5)$$

where the component distributions are approximated by models. As is commonly done, $P(W)$ is approximated as the language model and $P(A|W)$ is obtained from a hidden Markov model acoustic likelihoods.

II. SEGMENTAL MINIMUM BAYES-RISK DECODING

For most practical ASR tasks, the spaces \mathcal{W}_h and \mathcal{W}_e are large and the minimum Bayes-risk recognizer of Equation 4 faces computational problems. Previous work has focused on efficient search procedures to implement Equation 4. Here we discuss an alternate set of strategies that segment hypothesis and evidence spaces of the MBR recognizer. The segmentation transforms the original search problem into a series of search problems, which due to their smaller sizes, can be more easily solved. These strategies are collectively referred to as segmental MBR recognition [3].

For rigor, we introduce a segmentation rule $R(W)$ which divides strings in the language \mathcal{W} into N segments of zero or more words each. We denote the i^{th} segment of W as $R^i(W)$. In this way, we impose a segmentation of the space \mathcal{W} into segment sets $\mathcal{W}^1 \cdot \mathcal{W}^2 \cdot \dots \cdot \mathcal{W}^N$, where

$$\mathcal{W}^i = \{W' : W' = R^i(W), W \in \mathcal{W}\}.$$

R , when applied to \mathcal{W}_e , generates N evidence segment sets $\mathcal{W}_e^i, i = 1, 2, \dots, N$. We now define the marginal probability of any word string $W \in \mathcal{W}_e^i$

$$P_i(W|A) = \sum_{W' \in \mathcal{W}_e \cdot R^i(W')=W} P(W'|A) \quad (6)$$

The application of the segmentation rule to the hypothesis space yields hypothesis segment sets \mathcal{W}_h^i . The concatenation of the sets $\mathcal{W}_h^i, i = 1, 2, \dots, N$ yields a search space \mathcal{W}_h^* that is the cross-product of the hypothesis segment sets $\mathcal{W}_h^i, i = 1, 2, \dots, N$. Concatenating the sets may introduce new hypotheses since suffixes can be appended to prefixes in ways that were not permitted in the original space. However no hypotheses are lost through the concatenation. It is our goal to search over this larger space and, by considering more hypotheses, possibly achieve improved performance.

We now discuss the inclusion of the segmentation rule into MBR decoding. We begin by making the strong assumption that the loss function between any pair of evidence and hypothesis strings $W \in \mathcal{W}_e, W' \in \mathcal{W}_h$ distributes over the segmentation, i.e.

$$l(W, W') = \sum_{i=1}^N l(R^i(W), R^i(W')). \quad (7)$$

Under this assumption, we can now state the following proposition [4].

Proposition. An utterance level MBR recognizer given by

$$\delta(A) = \hat{W} = \operatorname{argmin}_{W' \in \mathcal{W}_h^*} \sum_{W \in \mathcal{W}_e} l(W, W') P(W|A) \quad (8)$$

can be implemented as a concatenation

$$\hat{W} = \hat{W}^1 \cdot \hat{W}^2 \dots \hat{W}^N, \quad (9)$$

where

$$\hat{W}^i = \operatorname{argmin}_{W' \in \mathcal{W}_h^i} \sum_{W \in \mathcal{W}_e^i} l(W, W') P_i(W|A), \quad i = 1, 2, \dots, N \quad (10)$$

This proposition defines the Segmental MBR (SMBR) decoder. Equation 10 follows by substituting Equation 7 into Equation 8.

A special case of segmental MBR recognition is particularly useful in practice. It arises when the strings in the hypothesis and evidence segment sets are restricted to length one or zero, i.e. individual words or the NULL word. We also assume that there is a 0/1 loss function on the segment sets

$$l(w, w') = \begin{cases} 0 & \text{if } w = w' \\ 1 & \text{otherwise.} \end{cases} \quad (11)$$

Under these conditions the segmental MBR recognizer of Equation 10 become

$$\hat{W}^i = \operatorname{argmax}_{w' \in \mathcal{W}_h^i} P(w'|A). \quad (12)$$

Equation 12 is the maximum a-posteriori decision over each hypothesis segment set. In each segment set the posterior probability of all the words are first computed based on the evidence space, and the word with the highest posterior probability is selected. We call this procedure *segmental MBR voting*. This simplification has been utilized in several recently developed N-best list and lattice based hypothesis selection procedures to improve the recognition word error rate [5], [6], [7].

This summarizes the relationship between SMBR decoding, MAP decoding and segmental MBR voting. From Equation 8, if no lattice cutting had been done, MBR decoding under the 0/1 loss function would lead to the standard MAP rule: $\hat{W} = \operatorname{argmax}_{W' \in \mathcal{W}_h} P(W'|A)$. Introducing

hypothesis space segmentation transforms the standard MAP rule to segmental MBR voting as in Equation 12.

For a given loss function, evidence space and hypothesis space, it may not be possible to find a segmentation rule such that Equation 7 is satisfied for any pair of hypothesis and evidence strings. However, given any segmentation rule, we can specify an associated *induced* loss function defined as

$$l_I(W, W') = \sum_{i=1}^N l(R^i(W), R^i(W')). \quad (13)$$

From the discussion of Proposition 1, we see that the segmental MBR recognizer is equivalent to an utterance level MBR recognizer under the loss function l_I . Therefore, the overall performance of the SMBR recognizer under a desired loss function l will depend on how well l_I approximates l .

For ASR, we are particularly interested in the Levenshtein loss function. Here, a segmentation of the hypothesis and evidence spaces will rule out some string alignments between word sequences. Therefore, under a given segmentation rule, the alignments permitted between any two word strings from \mathcal{W}_h and \mathcal{W}_e might not include the optimal alignment needed to achieve the Levenshtein distance. Therefore, the choice of a given segmentation involves a trade-off between two types of errors: search errors from MBR decoding on large segment sets and the errors in approximating the loss function due to the segmentation.

The Segmental MBR framework does not provide actual hypothesis and evidence space segmentation rules; it only specifies the constraints that these rules must obey. The construction of segment sets therefore remains a design problem to be addressed in an application specific manner. In the following sections we present procedures to construct the segment sets from recognition lattice and N-best lists under the Levenshtein loss function.

III. SMBR LATTICE SEGMENTATION

A recognition lattice is essentially a compact representation for very large N-best lists and their likelihoods. Formally, it is a directed acyclic graph, or an acyclic weighted finite state acceptor (WFSA) [8] $\mathcal{W} = (Q, \Lambda, q_s, F, \mathcal{T})$ with a finite set of states (nodes) Q , a set of transition labels Λ , an initial state $q_s \in Q$, the set of final states $F \subseteq Q$, and a finite set of transitions \mathcal{T} . The set Λ is the vocabulary of the recognizer. A transition belonging to \mathcal{T} is given by $t = (p, q, w, s)$

where $p \in Q$ is the starting state of this transition, $q \in Q$ is the ending state, $w \in \Lambda$ is a word, and s is a real number that represents a ‘cost’ of this transition. s is often computed as the sum of the negative log acoustic and language model scores on the transition. Some of the transitions in the WFST may carry the empty string $w = \epsilon$; these are termed ϵ transitions. 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 $p_1 = q_s$, $p_{i+1} = q_i$, and $q_n \in F$. The word string associated with T is w_1^n . For this word string we can obtain the joint acoustic and language model log-likelihood as $\log P(w_1^n, A) = -\sum_{k=1}^n s_k$. In this paper the finite state operations are performed using the AT&T Finite State Toolkit [9].

It is conceptually possible to enumerate all lattice paths and explicitly compute the MBR hypothesis according to Equation 4 [10]. However, for most large vocabulary ASR systems it is computationally intractable to do so. Goel et. al. [11] described an A^* search algorithm that utilizes the lattice structure to search for the MBR word string. Building on that approach, we present lattice node based segmentation procedures in which each segment maintains a compact lattice structure.

A. Lattice Segmentation using Node Sets

The ASR word lattices are directed and typically acyclic, therefore they impose a partial ordering on the lattice nodes. We say $n_1 \leq n_2$ if either $n_1 = n_2$ or there is at least one path connecting nodes n_1 and n_2 in the lattice and n_1 precedes n_2 on this path.

Let (N_s, N_e) be an ordered pair of lattice node sets such that

- P1. For all nodes $n \in Q$, there is at least one node $n' \in N_s$ such that $n \leq n'$ or $n' \leq n$.
- P2. For all nodes $n \in Q$, there is at least one node $n' \in N_e$ such that $n \leq n'$ or $n' \leq n$.
- P3. For any $n \in N_e$, there is no node $n' \in N_s$ such that $n \leq n'$.

Properties P1 and P2 essentially state that all lattice paths from lattice start to lattice end pass through at least one node of N_s and one node of N_e . Property P3 says that nodes of N_s on any lattice path precede nodes of N_e on that path. An example of N_s and N_e is depicted in the top panel of Figure 1.

Each lattice path can now be uniquely segmented into three parts by finding its first node that belongs to N_s and its first node that belongs to N_e . The portion of the path from q_s to the first node belonging to N_s is the first segment; from the first node belonging to N_s to the first node

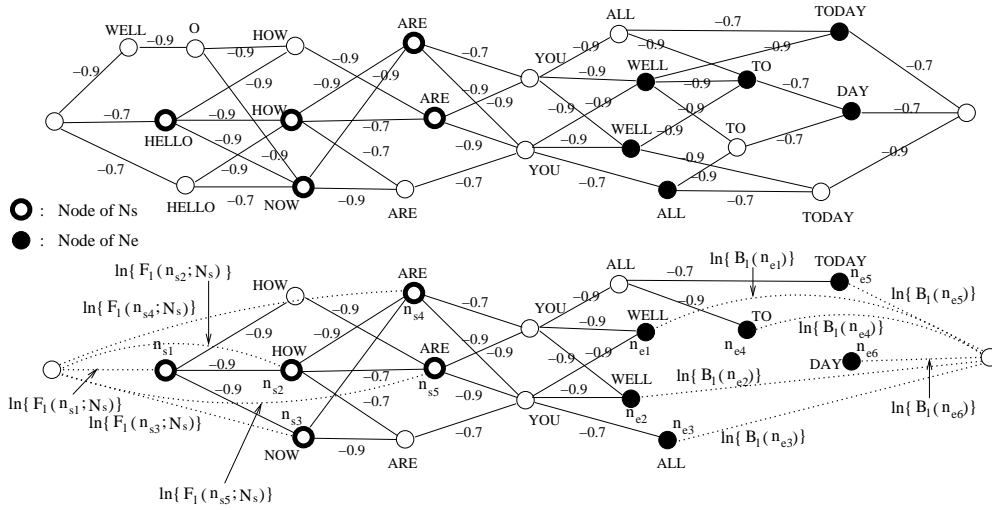


Fig. 1. Cutting a lattice based on node sets N_s and N_e (top). The lattice segment bounded by these sets is shown in the bottom panel by solid line paths.

belonging to N_e is the second segment; and from the first node belonging to N_e to a node in F is the third segment.

Segmentation of each lattice path, based on node sets $\{q_s\}$, N_s , N_e , F , defines a segmentation rules R to divide the entire lattice into three parts. In general, a rule for segmenting the lattice into $n + 1$ segments is defined by a sequence of lattice node sets $\{q_s\}$, N_1, N_2, \dots, N_n, F such that all ordered pairs (N_i, N_{i+1}) , $i = 1, \dots, n - 1$ obey P1-P3. The i^{th} lattice segment, \mathcal{W}_i , is specified by the node sets N_{i-1} and N_i . We shall say it is bounded on the left by N_{i-1} and on the right by N_i . An example lattice segment bounded by N_s and N_e is shown in the bottom panel of Figure 1. We call such node based lattice segmentation *lattice cutting* and the lattice cutting node sets as *cut sets*.

We note that lattice cutting yields segment sets \mathcal{W}_i that are more constrained than those that could be obtained by explicitly enumerating all lattice paths and segmenting them. This is due to the sharing of nodes between lattice paths. However, a useful property of lattice cutting is that each segment retains the compact lattice format. This allows for efficient implementation of MBR search on each lattice segment.

We now show that for Levenshtein loss function, fewer lattice segments necessarily result in a better approximation by the induced loss to the actual loss. Suppose we have a collection of cut sets $C = \{\mathcal{N}_i\}_{i=0}^R$ where $\mathcal{N}_0 = q_s$ and $\mathcal{N}_R = F$. This collection identifies a segmentation

rule such that the induced loss between $W, W' \in \mathcal{W}$ under the segmentation is $l_C(W, W') = \sum_{i=1}^R l(W_i, W'_i)$.

Suppose we discard a cut set \mathcal{N}_j from C to form $C' = C - \{\mathcal{N}_j\}$. This defines a new induced loss function

$$l_{C'}(W, W') = \sum_{i=1, i \neq j, i \neq j+1}^R l(W_i, W'_i) + l(W_j \cdot W_{j+1}, W'_j \cdot W'_{j+1})$$

By the definition of the Levenshtein distance (Appendix in [11])

$$l(W_j \cdot W_{j+1}, W'_j \cdot W'_{j+1}) \leq l(W_j, W'_j) + l(W_{j+1}, W'_{j+1}).$$

Hence $l_{C'}(W, W') \leq l_C(W, W')$. Therefore, if we segment the lattice along fewer cut sets, we obtain successively better approximation to the Levenshtein distance. However as the sizes of the lattice segments increase, SMBR decoding on the resulting segment sets will inevitably encounter more search errors. Our goal is therefore to choose a set C that will yield a ‘‘good’’ cutting procedure. Such a cutting procedure produces small segments that still provide a good approximation of the overall loss.

In the following two subsections we describe heuristic procedures to identify good cut sets.

B. Cut Set Selection Based on Total Risk

Our first lattice cutting procedure is motivated by the observation that under an ideal segmentation the conditional risk of each hypothesis word string is unchanged after segmentation [12]. The conditional risk after the segmentation is computed under the marginal distribution of Equation 6. Consequently the total conditional risk of all lattice hypotheses

$$R_T = \sum_{W' \in \mathcal{W}} \sum_{W \in \mathcal{W}} l(W, W') P(W|A) \quad (14)$$

would also be unchanged under this segmentation. For convenience we shall drop ‘conditional’ and refer to R_T as total risk.

We assume that the posterior probability of the most likely lattice word string dominates the total risk computation. That is

$$R_T \approx \sum_{W' \in \mathcal{W}} l(\tilde{W}, W') P(\tilde{W}|A) \quad (15)$$

where \tilde{W} denotes the most likely word string in the lattice

$$\tilde{W} = \operatorname{argmax}_{W \in \mathcal{W}} P(W|A), \quad (16)$$

and $\tilde{W} = \tilde{w}_1^K$. Our goal, then, is to find a segmentation rule so that under the ML approximation to the total risk, the following holds

$$P(\tilde{W}|A) \sum_{W' \in \mathcal{W}} l(\tilde{W}, W') = P(\tilde{W}|A) \sum_{W' \in \mathcal{W}} \sum_{i=1}^K l(\tilde{W}_i, W'_i). \quad (17)$$

Clearly if the rule segments \tilde{W} and each $W' \in \mathcal{W}$ into K substrings so that

$$l(\tilde{W}, W') = \sum_{i=1}^K l(\tilde{W}_i, W'_i). \quad (18)$$

then Equation 17 holds. In the following we describe how such a rule can be derived by first producing a *simultaneous* alignment of all word strings in \mathcal{W} against \tilde{W} and then identifying cut sets in that lattice.

1) *Lattice to Word String Alignment via Finite State Composition:* Consider the weighted lattice of Figure 2. We obtain an unweighted acceptor \mathcal{W}_0 from this lattice by zeroing the scores on all lattice transitions. We also represent the one-best word string $\tilde{W} = \tilde{w}_1^K$ as an unweighted finite state acceptor whose transitions are given as $t = \{p, q, v\}$ where $v = \tilde{w}_k.k$; this labeling keeps track of both the words and their position in \tilde{W} .

To compute the Levenshtein distance, the possible single-symbol edit operations (insertion, deletion, substitution) and their costs can be readily represented by a simple weighted transducer T [13]. T is constructed to respect the position of words in \tilde{W} (See Figure 3). 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 weighted finite state composition

$$\mathcal{A} = \mathcal{W}_0 \circ T \circ \tilde{W} \quad (19)$$

Constructed in this way, every path in \mathcal{A} specifies a word sequence $W \in \mathcal{W}$ and a sequence of string-edit operations that transform W to \tilde{W} . In its entirety, \mathcal{A} specifies all possible string-edit operations that transform all word strings in \mathcal{W} to \tilde{W} (See Figure 4).

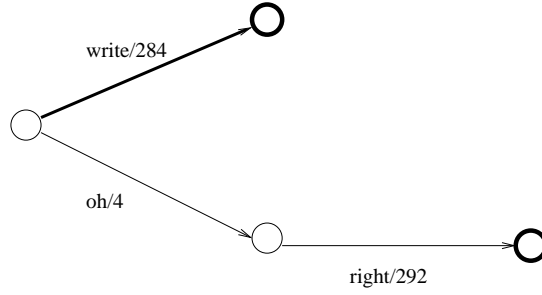


Fig. 2. A sample word lattice. The MAP hypothesis is shown in bold.

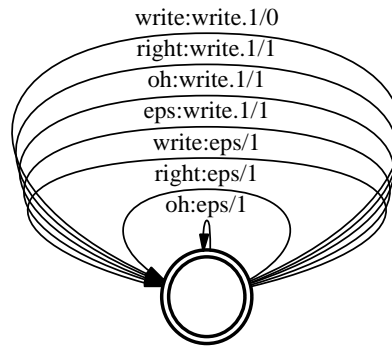


Fig. 3. The string-edit transducer T for the lattice in Figure 2. Each transition in T has the format $x : y/c$, which indicates that the input label is x , output label is y , and the cost of mapping x to y is c .

\mathcal{A} has transitions $t = (p, q, a, s)$ where a denotes an input-output symbol pair (w, v) . There are three types of transitions: (1) $w \neq \epsilon$ and $v = \tilde{w}_i.i$ which indicates a substitution of word w by word \tilde{w}_i ; (2) $w \neq \epsilon$ and $v = \epsilon$ indicates that word w is an insertion; (3) $w = \epsilon$ and $v = \tilde{w}_i.i$ shows a deletion with respect to \tilde{w}_i . The costs s on the transitions of \mathcal{A} arise from the composition in Equation 19.

2) *Compact Representation of String Alignments:* We now wish to extract from \mathcal{A} the Levenshtein alignment between every path $W \in \mathcal{W}$ and \tilde{W} . This can be done in two steps. We first perform a sequence of operations that transforms \mathcal{A} into a weighted acceptor \mathcal{A}' . \mathcal{A}' contains all the alignments links in \mathcal{A} , but represented in simplified form as an acceptor. We next use a variant of dynamic programming algorithm on the acceptor \mathcal{A}' to extract the Levenshtein alignment

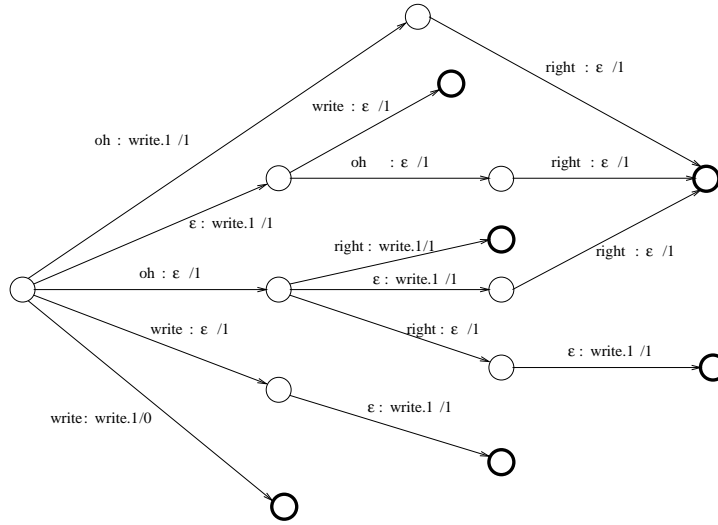


Fig. 4. The transducer \mathcal{A} for the lattice in Figure 2.

between \tilde{W} and every word string that was originally in \mathcal{W} .

The transformation of \mathcal{A} into \mathcal{A}' is as follows.

1) Project alignment information onto the input labels of \mathcal{A} , as follows :

- Sort the nodes of \mathcal{A} topologically and insert them in a queue S . Associate with each node q an integer $V(q)$. A value of $V(q) = i$ would indicate that all partial lattice paths ending at state q have been aligned with respect to \tilde{w}_1^{i-1} . Set $V(q_s) = 0$.
- While S is non-empty
 - a) $p \leftarrow \text{head}(S)$. DEQUEUE(S).
 - b) For all transitions $t = (p, q, a, s)$ leaving p , perform one of the following:
 - i) *Substitution*: If $a = (w, v)$ has $w \neq \epsilon$, $v = \tilde{w}_i.i$, set $a = (w.i, v)$ and $V(q) = i + 1$.
 - ii) *Deletion*: If $a = (w, v)$ has $w = \epsilon$ and $v = \tilde{w}_i.i$, set $V(q) = i + 1$.
 - iii) *Insertion*: If $a = (w, v)$ has $w \neq \epsilon$ and $v = \epsilon$, set $w = w_\epsilon.V(p)$ and $V(q) = V(p)$.

2) Convert the resulting transducer from Step 1 into an acceptor by projecting onto the input labels.

3) For the weighted automaton generated in Step 2, generate an equivalent weighted automaton without ϵ -transitions.

These three operations transform \mathcal{A} into a weighted acceptor \mathcal{A}' that contains the cost of *all alignments* between all lattice word strings and the best path (See Figure 5). We now relate the properties of the lattice \mathcal{W} and the finite state machines \mathcal{A} and \mathcal{A}' . By construction, corresponding to any $\bar{W} \in \mathcal{W}$ where $\bar{W} = \bar{w}_1^n$, there exist paths $T \in \mathcal{A}$ such that

1. $T = \{(p_k, q_k, a_k, s_k)\}_{k=1}^m$, $m \geq n$, $a_k = (w_k, v_k)$ where $w_1^m = \bar{w}_1^n$ if ϵ 's in w_1^m are removed and $v_1^m = \tilde{w}_1^K$ if ϵ 's in v_1^m are removed. $\sum_{k=1}^m s_k$ is total cost of the alignment specified by T .

s_k is the cost of a transition on T . Furthermore, for each $T \in \mathcal{A}$ there is a corresponding $T' \in \mathcal{A}'$ that specifies the identical alignment. That is,

2. $T' = \{(p'_k, q'_k, v'_k, s'_k)\}_{k=1}^n$ where $\text{Cost}(T) = \sum_{k=1}^m s_k = \sum_{l=1}^n s'_l = \text{Cost}(T')$ is the string edit distance between \tilde{w}_1^K and \bar{w}_1^n along the alignment specified by T and T' , and $v_1^m = \bar{w}_1^n$ if each v'_k is stripped of its $\cdot i$ and ϵ subscripts.

s'_k is the cost of a transition on T' .

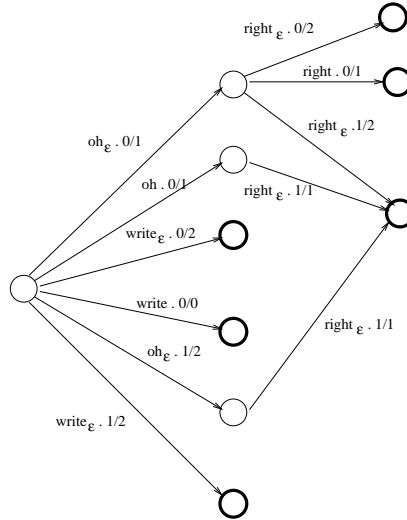


Fig. 5. The acceptor \mathcal{A}' for the lattice in Figure 2.

3) *Optimal Computation of Lattice to Word String Alignment:* We now discuss a procedure to extract the optimal alignment between paths $W \in \mathcal{W}$ and \tilde{W} . We first note that if \mathcal{A}' contained the alignment of only one word string W against \tilde{W} , we could find the desired optimal alignment through a standard dynamic programming procedure [14], [15], [16] that traverses the nodes of \mathcal{A}' in topologically sorted order and retains backpointers to the optimal partial paths to all

nodes. However, since \mathcal{A}' contains alignments of multiple strings against \tilde{W} , we need to extend the dynamic programming procedure to keep track of the identity of word strings leading into nodes. This is described in the following procedure.

- 1) Sort the nodes of \mathcal{A}' in reverse topological order (i.e. lattice final nodes first) and insert them in a queue S . For each node $q \in S$, let $b_q(y)$ denote the minimum cost of all paths that lead from node q to the lattice end node and carry the word string y . Let $a_q(y)$ be the immediate successor node of q on the path that achieves $b_q(y)$.
- 2) For each final node f of \mathcal{A}' , set $b_f(\epsilon) = 0$.
- 3) While S is non-empty
 - a) $p \leftarrow \text{head}(S)$. $\text{DEQUEUE}(S)$.
 - b) Let T denote the set of lattice transitions $t = (p, q, v, s)$ leaving state p . v is either $w.i$ or $v = w_\epsilon.i$. Let Y denote the set of unique word strings on the paths starting from state p . The word string $y = (w \cdot z)$ starts with the word w and has a suffix z .
 - c) For each $y (= w \cdot z) \in Y$,
 - i) Compute:

$$\hat{t} = \underset{t \in T: t \text{ has word label } w}{\text{argmin}} \quad s + b_q(z)$$

Denote $\hat{t} = (p, \hat{q}, \hat{v}, \hat{s})$.

$$\text{ii) } b_p(y) = \hat{s} + b_{\hat{q}}(z). \quad a_p(y) = \hat{q}.$$

Step 3 prunes all transitions leaving p that are not needed for any optimal alignment passing through p .

- 4) The procedure terminates upon reaching the start node q_s of \mathcal{A}' . The optimal alignment cost of each complete path y can be readily obtained from $b_{q_s}(y)$, and the complete alignment can be obtained by following the backtrace pointers stored in $a_p(y)$ arrays.

4) An Efficient Algorithm for Lattice to Word String Alignment: The alignment procedure described in the previous section involves the computation of the cost $b_p(y)$ for each state p in \mathcal{A}' . This cost is computed for all unique word strings y leaving state q . Therefore, it involves enumerating all the word sub-strings in the word lattice \mathcal{W} . While this is definitely impossible for most word lattices of interest, this description does clearly present the inherent complexity of the lattice to string alignment problem. In practice, we do not retain the cost $b_p(y)$ for all word sequences leaving p . For each state p , we approximate $b_p(y)$ as $b_p(y) \approx b_p^* = \min_y b_p(y) \forall y$ in

Step 3(c)ii. We now present the procedure that results from this approximation.

- 1) Sort the nodes of \mathcal{A}' in reverse topological order (i.e. lattice final nodes first) and insert them in a queue S . For each node $q \in S$, let b_q denote the minimum cost of all paths that lead from node q to the lattice end node.
- 2) For each final node f of \mathcal{A}' , set $b_f = 0$.
- 3) While S is non-empty
 - a) $p \leftarrow \text{head}(S)$. DEQUEUE(S).
 - b) Let T denote the set of lattice transitions $t = (p, q, v, s)$ leaving state p . v is either $w.i$ or $v = w_\epsilon.i$. Let U denote the set of unique words on the transitions starting from state p .
 - c) Initialize $\hat{T} = \{\}$.
 - d) For each $w \in U$,
 - i) Compute:

$$\hat{t} = \underset{t \in T: t \text{ has word label } w}{\text{argmin}} \quad s + b_q$$

Denote $\hat{t} = (p, \hat{q}, \hat{v}, \hat{s})$.

- ii) $\hat{T} \leftarrow \hat{t}$.
- e) Compute:

$$b_p = \min_{i \in \hat{T}} \hat{s} + b_{\hat{q}}$$

- f) Prune transitions $t \in T$ and $t \notin \hat{T}$.
- 4) The procedure terminates upon reaching the start node q_s of \mathcal{A}' .

As a result of the simplification, the information maintained by the $a_p(w)$ and the $b_p(w)$ arrays can be stored with the lattice structure of \mathcal{A}' . This is therefore a pruning procedure of \mathcal{A}' and we call the resulting acceptor $\hat{\mathcal{A}}$. For the example \mathcal{A}' of Figure 5, $\hat{\mathcal{A}}$ is shown in Figure 6.

The transitions of $\hat{\mathcal{A}}$ have 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) $v = w_\epsilon.i$ 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.

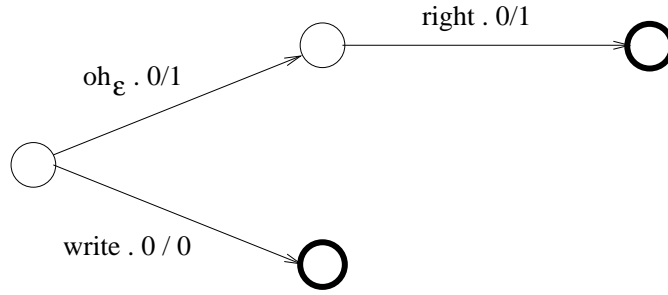


Fig. 6. The acceptor $\hat{\mathcal{A}}$ for the lattice in Figure 5.

We note that the acceptors \mathcal{W} and $\hat{\mathcal{A}}$ have identical word sequences, therefore, we can get the acoustic and language model scores for $\hat{\mathcal{A}}$ by composing it with \mathcal{W} .

5) *Risk Based Lattice Cutting*: Referring back to Section III-A, we introduce lattice segmentation as the process of identifying lattice cut sets to satisfy property P1-P3. The process of generating $\hat{\mathcal{A}}$ identifies a correspondence between each word in \tilde{W} and paths in $\hat{\mathcal{A}}$. Each word \tilde{w}_i in \tilde{W} is aligned with a collection of arcs in $\hat{\mathcal{A}}$. These arcs either fall on distinct paths (e.g. hello.0 in Figure 7) or form connected subpaths (e.g. well_ε.0 · o.0 in Figure 7). For each word \tilde{w}_i , we define the lattice cut node set \mathcal{N}_i as the terminal nodes of all the subpaths that align with \tilde{w}_i . This defines K cut sets if there are K words in \tilde{W} . We also define \mathcal{N}_0 as $\{q_s\}$.

In this way we use the alignment information provided in $\hat{\mathcal{A}}$ to define the lattice cut sets that segment the lattice into K sublattices. We call this procedure *Risk-Based Lattice Cutting* (RLC). This procedure ensures that every lattice path passes through exactly one node from each lattice node cut set. A determinized version of the lattice from Figure 1 and its acceptor $\hat{\mathcal{A}}$ are shown in the top and bottom panels of Figure 7. The bottom panel also displays the cuts obtained along the node sets.

The segmentation procedure, modulo the errors introduced by the approximate procedure used to generate $\hat{\mathcal{A}}$, is optimal with respect to the MAP word hypothesis. Every path $W' \in \mathcal{A}$ has a corresponding path $\{p_k, q_k, w_k, s_k\}_{k=1}^N$ in $\hat{\mathcal{A}}$ such that $l(\tilde{W}, W') = \sum_{i=1}^K l(\tilde{W}_i, W'_i) = \sum_{k=1}^K s_k$. In this way, the costs in $\hat{\mathcal{A}}$ agree with the loss function desired in Risk-based lattice cutting (RLC).

6) *Periodic Risk Based Lattice Cutting*: The alignment obtained in Section III-B.1 is ensured to be optimal only relative to the MAP path. It is not guaranteed that $l(W, W') = l_I(W, W')$ for $W \neq \tilde{W}$. Following the discussion in Section III-A, we note that if we segment the lattice along

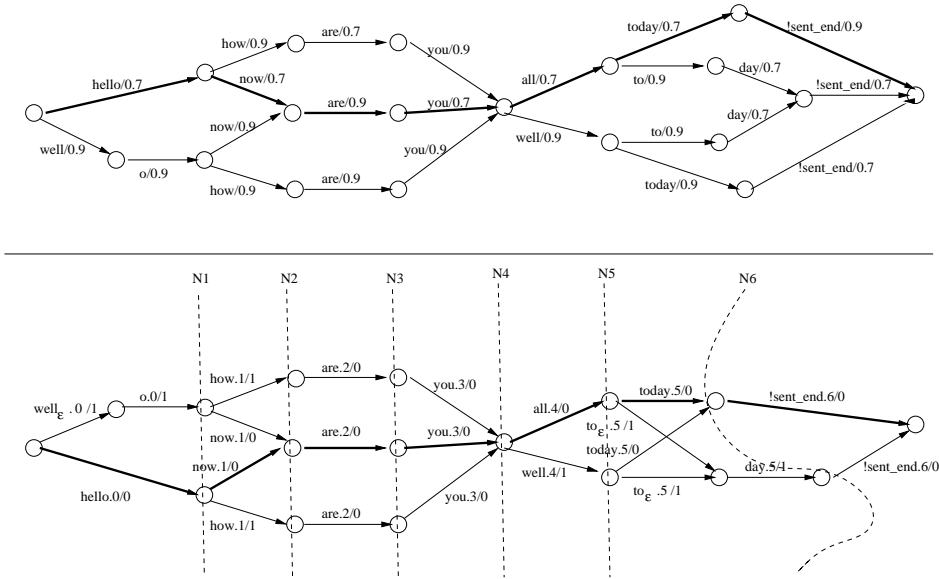


Fig. 7. (top) A word lattice \mathcal{W} and (bottom) its acceptor $\hat{\mathcal{A}}$ showing the Levenshtein alignment between $W \in \mathcal{W}$ and \tilde{W} (shown as the path in bold). The bottom panel shows the segmentation along the 6 nodesets obtained by the risk-based lattice cutting procedure.

fewer cut sets, we obtain better approximations to the Levenshtein loss function. However, this leads to larger lattice segments and therefore greater search errors in MBR decoding.

One solution that attempts to balance the trade-off between search errors and errors in approximating the loss function is to segment the lattice by choosing node sets N_i at equal intervals or periods. A period of k specifies the cut sets N_1, N_{k+1}, N_{2k+1} and so on. Therefore the set $C = \{q_s, N_1, N_{k+1}, \dots, N_{nk+1}, F\}$ where n is the largest integer so that periodic cuts can be found. We call this procedure *periodic risk-based lattice cutting* (PLC). If the loss function approximation obtained by cutting \mathcal{W} into K segment sets 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. Figure 8 shows the sub-lattices obtained by periodic risk-based lattice cutting on the lattice from Figure 7.

C. Cut Set Selection Based on Word Confidence

Our next procedure to identify good lattice cutting node sets uses word level confidence scores [17]. In this procedure word boundary times are used to derive alignment between sen-

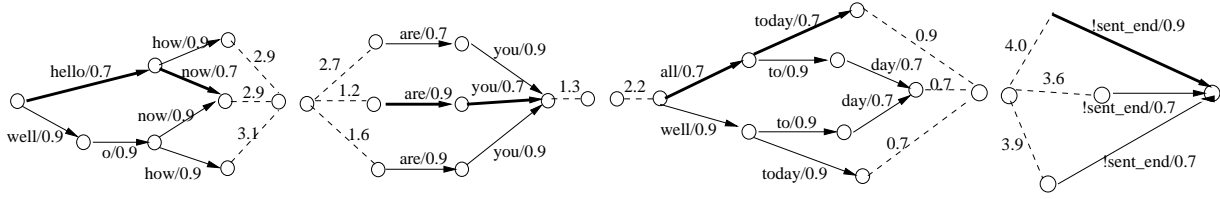


Fig. 8. Lattice segments obtained by periodic risk-based lattice cutting on the lattice from Figure 7 (Period = 2).

tence hypotheses that avoids computing the alignment corresponding to the exact Levenshtein distance [18]. As before, we begin by identifying the MAP lattice path. We compute the confidence score of the i^{th} link lattice link l on that path as follows:

- 1) Compute the lattice forward-backward probability of l [17].
- 2) Identify other lattice links that have a time overlap of at least 50% with l . Among these links, keep only those that have the same word label as l .
- 3) Compute the lattice forward-backward probabilities of all these links. Add their probabilities to that of l to obtain the *confidence score* for l .

Next, high confidence links on the MAP path are identified by comparing each link’s confidence score to a global threshold. Consecutive high confidence links identify high confidence lattice regions, and lattice cut node sets are derived as follows.

- 1) For each stretch of consecutive high confidence links along the MAP path, identify the leftmost and the rightmost links, denoted l and r , respectively.
- 2) Find all those lattice links that have a time overlap of more than 50% with l . The start nodes of these links form the left boundary node set of a lattice cut.
- 3) Find all those lattice links that have a time overlap of more than 50% with r . The end nodes of these links form the right boundary node set of the lattice cut.
- 4) Add nodes to the left and right boundary node sets to ensure that properties P1 and P2 are met.

We note that in contrast to risk-based lattice cutting, this procedure allows a lattice path to pass through more than one node of a given node set. The top panel of Figure 9 depicts confidence based cutting of the lattice of Figure 1.

We now introduce the notion of “pinching” in a lattice segment. If the largest value of the marginal probability (Equation 6) in a lattice segment is above a threshold, we collapse the

lattice segment to the MAP path \tilde{W}_i belonging to the segment. The bottom panel of Figure 1 shows the pinched version of the middle cut; the high confidence region can be represented by a single word sequence.

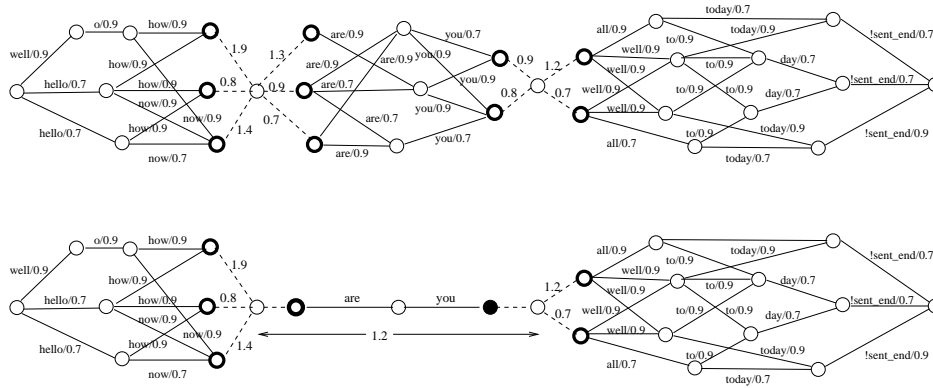


Fig. 9. Segmenting the lattice from Figure 1 into regions of low and high confidence based on word confidence scores.

D. SMBR Decoding of a Lattice Segment

To generate SMBR hypothesis from a lattice segment (Equation 10) we require $P_i(W|A)$ for each word string in that segment. Since our lattice cutting node set selection procedures described above may yield multiple start nodes which may be successive nodes on some path through \mathcal{W}_i , care must be taken in computing $P_i(W|A)$. It is found by summing over all paths through \mathcal{W} along which W is the longest subsequence in \mathcal{W}_i , i.e. as the sum over all these paths whose longest subpath in \mathcal{W}_i is W . In the following we describe a modified lattice forward-backward procedure that respects this restriction and yields the desired marginal probability.

Let W be a complete path in the lattice and let W_p be a prefix of W . We use $L_f(W_p)$ to denote the joint log-likelihood of observing W_p and the acoustic segment that corresponds to W_p . $L_f(W_p)$ can be obtained by summing the log acoustic and language model scores present on the lattice links that correspond to W_p . Similarly, for a suffix W_s of W , we use $L_b(W_s)$ to denote the joint log-likelihood of observing W_s together with its corresponding acoustic segment, conditioned on the starting node of W_s . $P(A)$ can be computed as $e^{L_f(n_e)}$.

Let $E^h(W')$ denote the first node of an arbitrary lattice path segment W' . Let $E^f(W')$ denote the last node of W' , and let $E(W')$ be the set of all lattice nodes through which W' passes, including $E^h(W')$ and $E^f(W')$. Let W_i be a path in a lattice segment \mathcal{W}_i bounded by node sets

N_s and N_e . Let $n_1 = E^h(W_i)$ and $n_2 = E^f(W_i)$. We first define a *lattice forward probability* of n_1 , $F_l(n_1)$, which is the sum of partial path probabilities of all partial lattice paths ending at n_1 . That is,

$$F_l(n_1) = \sum_{W_p: E^f(W_p)=n_1} e^{L_f(W_p)}. \quad (20)$$

However, paths that pass through any node of N_s before they reach n_1 would contribute a segment longer than W_i to this cut. We exclude their probability by defining a *restricted forward probability* of n_1 , restricted by the node set N_s , as

$$F_l(n_1; N_s) = \sum_{\substack{W_p : E^f(W_p) = n_1 \\ E(W_p) \cap N_s = \{n_1\}}} e^{L_f(W_p)}. \quad (21)$$

We also define *lattice backward probability* of the final node of W_i , using the backward log-likelihood $L_b(W_s)$, as

$$B_l(n_2) = \sum_{W_s: E^h(W_s)=n_2} e^{L_b(W_s)}. \quad (22)$$

We have stated earlier (Section III-A) that each lattice path can be segmented into three parts by finding its first node belonging to N_s and its first node belonging to N_e . As a result there is no path in the sub-lattice that contains two nodes in N_e . We therefore do not need to define a restricted backward probability analogous to the restricted forward probability.

Using the restricted forward probability of n_1 and lattice backward probability of n_2 , the marginal probability of W_i can be computed as

$$P_i(W_i|A) = \frac{1}{P(A)} F_l(n_1; N_s) P(W_i, A(W_i)|n_1) B_l(n_2), \quad (23)$$

where $A(W_i)$ denotes the acoustic segment corresponding to W_i .

We note that if the node set N_s is such that no lattice path passes through two nodes of N_s , the restricted forward probability of n_1 , $F_l(n_1; N_s)$, will be identical to its lattice forward probability $F_l(n_1)$. In this case, the marginal probability of W_i will be obtained by summing over all lattice paths that pass through W_i . This is the well known lattice *forward-backward probability* of W_i [19].

Having obtained $P_i(W|A)$, the SMBR hypothesis can be computed using the A^* search procedure described by Goel et. al. [11]. Alternatively, an N-best list can be generated from each segment and N-best rescoring procedure of Stolcke et. al. [10] can be used.

IV. SMBR N-BEST LIST SEGMENTATION

An N-best list is an enumeration of N most likely word strings given an acoustic observation; it can be generated from a lattice as word strings with N highest log likelihood values. An N-best list can itself be considered as a special “linear” lattice where each node, except the start and end nodes, has exactly one incoming and one outgoing transition. An example N-best list derived from the lattice of Figure 7 is displayed as a linear lattice in Figure 10.

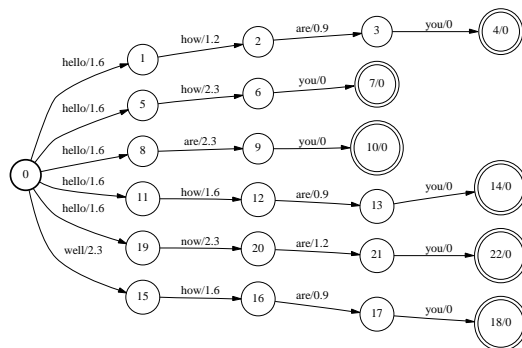


Fig. 10. An N-best List from the lattice in Figure 7.

For SMBR rescoring of an N-best list we can apply the lattice cutting methods described in Sections III-C and III-B. Alternatively, we could use the ROVER [5] procedure which was originally proposed by Fiscus [5] to combine MAP hypotheses from multiple ASR systems, and later extended to N-best lists [20]. ROVER is similar to total risk based cutting of N-best lists with the most significant difference being that the total risk based lattice cutting allows for multiple consecutive words in each segment set (Figure 7); in contrast, ROVER yields at most one consecutive word in a segment set.

An alternative N-best list SMBR rescoring procedure that generalizes both ROVER and total risk based lattice cutting is a procedure termed e-ROVER, as described here. We first define a process of *joining* two consecutive segment sets. In joining two segment sets we replace those two sets by one *expanded* set that contains all the paths from the original pair of sets. This is illustrated in Figure 11.

The e-ROVER procedure for constructing SMBR evidence and hypothesis spaces can be described as follows [3].

- 1) Segment N-best lists, as in ROVER, so that each segment contains at most one consecutive word [5].

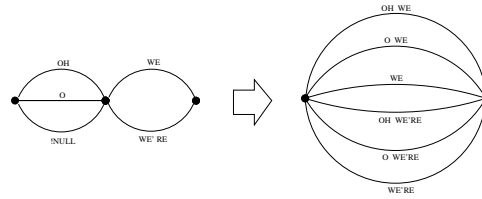


Fig. 11. Joining two segment sets.

- 2) Determine the posterior probability of words in segment sets, according to Equations 6 and 12.
- 3) “Pinch” segment sets in which the largest value of the posterior probability is above a *pinching threshold*. Join all adjacent unpinched segment sets.

The procedure of pinching and expanding the segment sets is shown in Figure 12. Hypotheses in e-ROVER are formed sequentially according to Equations 9 and 10.

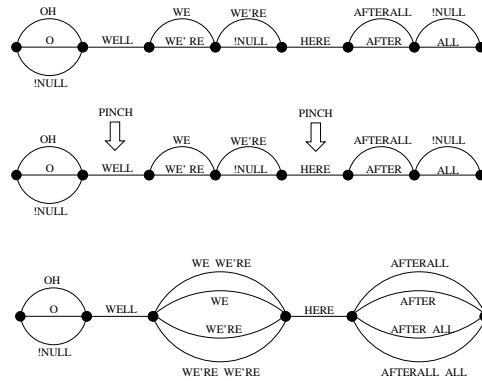


Fig. 12. e-ROVER WTN construction.

That e-ROVER generalizes total risk based lattice cutting is evident by observing that the segment sets in e-ROVER contain hypotheses which were not present in the original N-best lists. Furthermore, we note that the hypothesis and evidence spaces in e-ROVER are identical to those in ROVER. However, the loss function in e-ROVER provides a better approximation to the word error rate due to the improved segmentation. Since they are both instantiations of Equation 9, e-ROVER directly extends ROVER and would be reasonably expected to yield a lower word error rate.

V. APPLICATIONS TO ASR SYSTEM COMBINATION

In addition to its role in simplifying MBR decoding, the segmental MBR decoding framework has applications to ASR system combination. These techniques involve combining either word

lattices or N-best lists produced by several ASR systems.

Let $\mathcal{W}^k, k = 1, 2, \dots, J$ be recognition lattices or N-best lists from J ASR systems. Let $P^k(W|A)$ be the evidence distribution of the k^{th} system over \mathcal{W}^k . A common evidence space can be obtained by taking a union or intersection of these K lattices or N-best lists. The evidence distribution over this space can be derived by taking the arithmetic mean

$$P(W|A) = \frac{1}{J} \sum_k P^k(W|A),$$

or a geometric mean

$$P(W|A) = \left[\prod_k P^k(W|A) \right]^{\frac{1}{J}}$$

of the J evidence distributions.

In the case of N-best lists, the SMBR recognition can be carried out over the N-best list generated above. The SMBR decoding procedures that can be applied on this space include ROVER [5] and e-ROVER as described in section IV.

Combination of lattices from multiple systems is not as straightforward. One possible scheme is described in the following.

- 1) Select the hypothesis with the overall highest posterior probability among the MAP hypotheses from the J systems. This is obtained as

$$\hat{k} = \operatorname{argmax}_{k=1,2,\dots,J} P^k(\hat{W}^k|A) \quad (24)$$

$$\hat{W} = \hat{W}^{\hat{k}}. \quad (25)$$

- 2) Segment each lattice with respect to \hat{W} using the periodic risk-based lattice-cutting procedure (Section III-B) into N sections. This gives us $N \times J$ sub-lattices given by $\mathcal{W}_l^k, k = 1, 2, \dots, J, l = 1, 2, \dots, N$. We note that \hat{W} need not be present in all of the J lattices since the procedure described in Section III-B can be used to align the lattice to any word string.
- 3) For each section $l = 1, 2, \dots, N$, we now create new segment-sets by combining the J corresponding sub-lattices $\mathcal{W}_l^k, k = 1, 2, \dots, J$. We have considered two combination schemes:
 - a) We perform a weighted finite-state intersection [8] of the corresponding sub-lattices.

This is equivalent to multiplying the posterior probability of hypotheses in the indi-

vidual sub-lattices.

$$\text{For } l = 1, 2, \dots, N \quad (26)$$

$$\mathcal{W}_l = \cap_{k=1}^J \mathcal{W}_l^k \quad (27)$$

$$P_l(W|A) = \left[\prod_{k=1}^J P_l^k(W|A) \right]^{\frac{1}{J}} \quad \forall W \in \mathcal{W}_l. \quad (28)$$

- b) We perform a weighted finite state union of the corresponding sub-lattices followed by a weighted finite state determinization under the $(+, \times)$ semiring [8]. This is equivalent to adding the posterior probability of hypotheses in the individual sub-lattices.

$$\text{For } l = 1, 2, \dots, N \quad (29)$$

$$\mathcal{W}_l = \cup_{k=1}^K \mathcal{W}_l^k \quad (30)$$

$$P_l(W|A) = \frac{1}{J} \sum_{k=1}^J P_l^k(W|A) \quad \forall W \in \mathcal{W}_l. \quad (31)$$

- 4) Finally, we perform SMBR decoding (Equations 10 and 9 in Section II) on the sub-lattices \mathcal{W}_l obtained by the above combination schemes.

A schematic of multi-system SMBR decoding using three sets of lattices is shown in Figure 13.

VI. SMBR DECODING EXPERIMENTS

All our SMBR decoding experiments were carried out on large vocabulary ASR tasks. We first present results of the risk and confidence based lattice cutting procedures described in Section III. We then present experiments with multiple system combination using the N-best list based e-ROVER procedure described in Section IV and the lattice based system combination scheme presented in Section V.

A. SMBR Recognition with Lattices

Our lattice cutting procedures were tested on the Switchboard-2 portion of the 1998 Hub5 evaluation set (SWB2) and Switchboard-1 portion of the 2000 Hub5 evaluation set (SWB1). For both these test sets an initial set of one-best hypotheses were generated using the AT&T

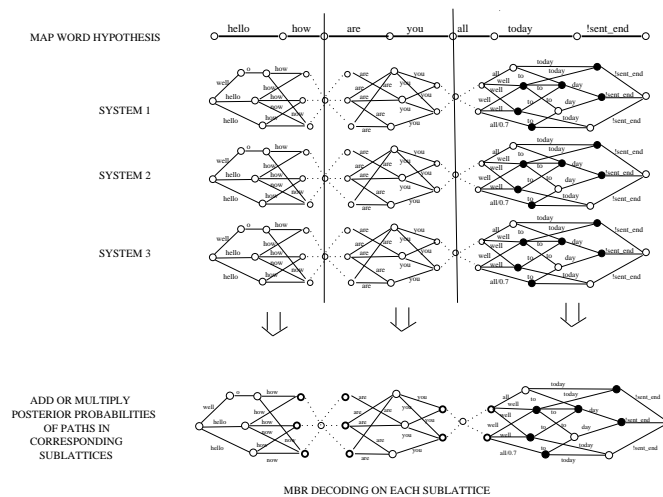


Fig. 13. Multiple-system SMBR Decoding via Lattice Combination.

large vocabulary decoder [8]. HTK [21] cross-word triphone acoustic models, trained on VTN-warped data, with a pruned version of SRI 33K trigram word language model [20] were used. The one-best hypotheses were then used to train MLLR transforms, with two regression classes, for speaker adaptive training (SAT) version of the acoustic models. These models were used to generate an initial set of lattices under the language model mentioned above. These lattices were rescored using the unpruned version of SRI 33K trigram language model and then again using SAT acoustic models with unsupervised MLLR on the test set. Details of the system are given in JHU 2001 LVCSR Hub5 system description [22].

Lattices were segmented using the three procedures described in this article: risk based lattice cutting (Section III-B.5), periodic risk based lattice cutting (Section III-B.6), and confidence based lattice cutting (Section III-C). Once a lattice segmentation was obtained, the following procedures were investigated to compute the SMBR hypothesis. An A^* search over each segment [11] attempts an exact, if heavily pruned, implementation of the MBR decoder. Alternatively, an N-best list was generated from each segment and then rescored using the min-risk procedure [10], [11]. As a third approach, the e-ROVER procedure of Section IV was applied. In the latter two techniques, N-best lists of size 250 were used. For confidence based lattice cutting, a confidence threshold of 0.9 was used in all cases.

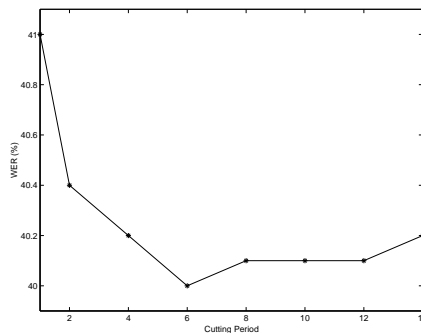


Fig. 14. Performance of *Periodic Lattice Cutting* with different cutting periods for A^* SMBR decoding on the SWB2 held out set.

For periodic risk based lattice cutting, the optimal segmentation period was determined on two held out sets, one corresponding to each test set. Cutting periods of 1 through 14 were tried and for each segmentation the SMBR hypothesis was generated using one of A^* , N-best list rescoring, or e-ROVER procedures. Figure 14 presents the word error rate of the A^* SMBR decoding on the held out set corresponding to the SWB2 test set. As can be seen, the optimal cutting period is 6 on this set. N-best rescoring and e-ROVER also achieved their optimal performance at period 6 on this data set. On the held out set corresponding to the SWB1 test set, the optimal cutting period was found to be 4 under all three hypothesis generation procedures. This suggests that optimal lattice cutting period is relatively insensitive to the hypothesis generation method but should be tuned to the task to which periodic lattice cutting is applied.

Table I presents a comparison of different lattice segmentation and hypothesis generation procedures. PLC was performed with a cutting period of 6 (on both test sets, even though 4 was found to be optimal for SWB1).

We note that all SMBR procedures yielded a gain over the MAP baseline for both test sets. In addition, PLC and confidence based lattice segmentation consistently further improved the word error rate over the no segmentation case, which advocates the use of SMBR procedures over MBR decoding without segmentation. Among the various hypothesis generation procedures, PLC with period 6 was found to have the best performance. In all cases, e-ROVER performance was the best, although with period 6, e-ROVER is ahead only by a small margin.

Decoder		WER(%)	
		SWB2	SWB1
MAP (baseline)		41.1	26.0
SMBR Decoding			
Segmentation Strategy	MBR Decoding Strategy		
No Segmentation	A^* search	40.4	25.5
	e-ROVER	40.5	25.7
	N-best rescoring	40.4	25.6
RLC (Period 1)	A^* search	41.0	25.9
	e-ROVER	40.5	25.7
	N-best rescoring	41.0	25.9
PLC (Period 6)	A^* search	40.0	25.4
	e-ROVER	39.9	25.3
	N-best rescoring	40.1	25.4
Confidence based	e-ROVER	40.0	25.2
	N-best rescoring	40.2	25.4

TABLE I

SMBR LATTICE RESCORING PERFORMANCE.

B. System Combination Results

1) *N-best List Combination*: The experiments involving combination of N-best lists from multiple systems were performed on a multi-lingual language independent acoustic modeling task [23]. This task consisted of combining recognition outputs in Czech language from three systems : a triphone system trained on one hour of Czech voice of America (Cz-VOA) database (Sys1); a triphone system trained on 72 hrs. of English and then adapted to one hour of Czech (Sys2); and Sys1 output rescored with Sys2 models. The test set consisted of 748 held out utterances from the Cz-VOA broadcast [24]. 250 hypotheses were taken from each system along with their distributions restricted to these 250-best lists. The baselines (MAP hypotheses) in these systems had error rates of 29.42%, 35.24%, and 29.22%, respectively.

We created a single N-best list of up to 750 hypotheses by merging (via union operations

as described in Section V) the 250-best lists from the 3 systems. On this merged list, N-best ROVER yields an absolute improvement of 3.28% over the 29.22% baseline. Its comparison with e-ROVER is shown in Figure 15. The top panel in this figure shows the fraction of the pinched sets as a function of the pinching threshold. A threshold of 0.0 pinches all the sets, equivalent to N-best ROVER, while any threshold above 1.0 results in no pinching at all. We note that a segment set is pinched when largest value of posterior probability in the set is greater than or equal to the pinching threshold. Since regions of high confidence have segments which have a posterior probability of 1.0, these segments are pinched even at a threshold of 1.0. As the pinching threshold increases (i.e. for fewer pinched sets) the number of hypotheses in the expanded sets grows so large that MBR rescoring becomes infeasible without heavy pruning. For pinching thresholds greater than 1.0 we are effectively performing MBR rescoring with the large hypothesis space obtained by full expansion of all segment sets (Figure 11). By contrast, the original MBR decoding has only the unexpanded N-best list as its hypothesis space. This points out the need to achieve a proper balance between size of the hypothesis space versus the computational complexity of the MBR search.

The bottom panel in Figure 15 shows the effect of pinching threshold on the word error performance of e-ROVER. We note that performance under all thresholds is better than the performance of N-best ROVER. The threshold of 1.0 yields the best performance of 0.56% absolute improvement over N-best ROVER and hence a total of 3.84% absolute over the baseline error rate of 29.22%. We see a degradation in performance for thresholds larger than 1.0, owing to the pruning of the expanded sets.

2) *Lattice Combination*: Our experiments with combining lattices from multiple systems and their SMBR decoding were carried out on the development set of the LVCSR RT-02 evaluation. A description of the acoustic and language models used is given in the JHU LVCSR RT-02 system description [25]. In this system, MMI acoustic models were used to generate an initial set of lattices under the SRI 33K trigram language model [20]. These lattices were then rescored with DLLT acoustic models and DSAT acoustic models [26] to yield two other sets of lattices. These three sets of lattices were then used for system combination as described in Section V.

The performance of the lattice combination experiments is reported in Table II. In these experiments, we use a cutting period of 6 for the periodic risk-based lattice cutting. We tested these

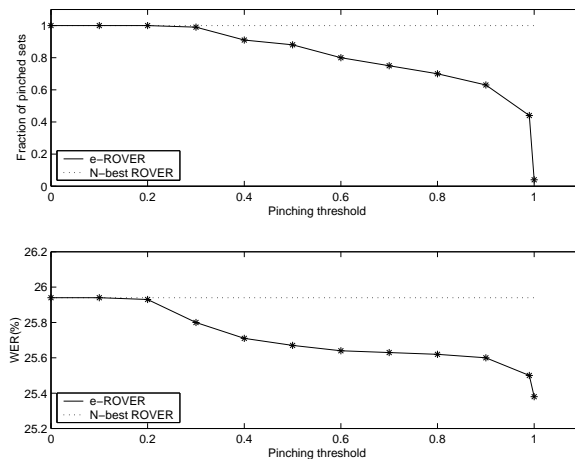


Fig. 15. Fraction of pinched segment sets and word error rate performance of e-ROVER as a function of pinching threshold. The rightmost point in the top panel shows that fraction of pinched sets is 0 when pinching threshold exceeds 1.0. The WER in this condition is given in the rightmost point of the lower panel.

procedures on the Switchboard1 portion of the 2000 Hub5 evaluation set (SWB1), Switchboard2 portion of the 1998 Hub5 evaluation set (SWB2) and the Switchboard-Cellular development set released in 2000 (SWB2C). The Table II is organized as follows. We first report the performance of the MAP hypothesis from each system. We next give results by a simple system combination technique (Lattice-Intersect) that intersects the three lattices and obtains the MAP hypothesis from the resulting lattice [8]. We also report results by the ROVER system combination scheme [5] on the MAP hypotheses from the three systems. We then finally present the results by the two multi-system SMBR decoding schemes (Union-SMBR and Intersect-SMBR) presented in Section V. The e-ROVER procedure was used to compute the MBR hypothesis in both these schemes.

We observe that the multiple system SMBR decoding via either the union or intersection scheme is better than 1) intersecting lattices and obtaining the MAP hypothesis or 2) performing a ROVER on the MAP hypotheses from the three systems. Furthermore, we note that adding posteriors of the paths in the sub-lattices (Union-SMBR) turns out to be better than multiplying them (Intersect-SMBR).

VII. CONCLUSIONS

We have presented the Segmental Minimum Bayes-Risk Decoding framework for Automatic Speech recognition. This framework allows us to decompose an utterance level Minimum

Decoding Strategy	WER(%)		
	SWB1	SWB2	SWB2C
MAP			
Sys 1 (MMIE)	24.5	39.2	39.6
Sys 2 (DLLT)	24.0	38.7	38.8
Sys 3 (DSAT)	24.5	39.3	39.5
Lattice-Intersect	24.0	38.4	38.7
1-best ROVER	23.8	38.1	38.2
SMBR Decoding			
Intersect - SMBR	23.5	37.8	38.0
Union- SMBR	23.3	37.8	37.8

TABLE II

EXPERIMENTS: MULTIPLE-SYSTEM SMBR DECODING VIA LATTICE COMBINATION.

Bayes-Risk Recognizer into a sequence of smaller sub-utterance recognizers. Therefore, a large search problem is decomposed into a sequence of simpler, independent search problems. Though the utterance level MBR decoder is implemented as a sequence of MBR decoders on hypothesis and evidence space segments, the acoustic data is not segmented at all. The marginal probability of a word string within a segment set is computed based on acoustic and language model scores that span the entire utterance ; these might have a much greater span than any string in the segment set. In addition, there is no assumption of linguistic independence between word strings belonging to adjacent segments. This is not the case when the entire conversation level decoder is simplified to decoders at the utterance level; by contrast, in that case we do segment acoustic data and assume acoustic and linguistic independence between utterances.

We have described several procedures for segmenting word lattices into sub-lattices for SMBR decoding. The confidence based lattice cutting relies on word-level confidences and time-marks in a lattice to perform segmentation; while total and periodic risk based lattice cutting strategies attempt to find segments that preserve the total Bayes-risk of all word strings in the lattice. These procedures identify node sets that can be used to segment the lattice. However, we have shown

that the selection of cut sets must be made considering both SMBR search errors and errors due to poor approximation of the loss function. We introduced periodic risk based lattice cutting as a cut set selection procedure that finds a balance between these two types of modeling errors. Lattice cutting, in conjunction with SMBR decoding gives consistent improvements as the final stage of an LVCSR evaluation system. In addition, the risk based cutting procedure has been shown to form the basis for novel discriminative training procedures [27]. We note that the two cutting procedures give similar WER performance although the risk based cutting procedure is more suited to system combination since it does not rely on word boundary times which can easily vary across multiple systems.

We have discussed how popular ASR system combination techniques such as ROVER and N-best ROVER are instances of SMBR recognition. In the SMBR framework, we presented the extended-ROVER technique that improves upon N-best ROVER by better approximating the WER and improving performance on a language independent acoustic modeling task.

Finally, we showed how the risk-based lattice segmentation can be applied to multiple system SMBR decoding on lattices produced by several ASR systems. We presented two schemes to merge posteriors of word strings in sub-lattices and then performing SMBR decoding. The system combination scheme performs better than the output produced by a MAP decoder on each of the individual lattices or on a intersection of the lattices. These techniques are effective in combining results from different systems, as is particularly apparent from the multilingual system combination experiments.

SMBR recognition has been shown to be a useful framework for automatic speech recognition. It transforms the overall MBR recognition problem into a sequence of smaller, independent decisions that are easier to solve than the original problem. As a modeling technique, it allows us to focus in on individual recognition errors during the search process. We have also shown how SMBR can be used to describe and enhance ASR system combination procedures. SMBR is a powerful framework for the development and description of novel ASR decoding strategies.

ACKNOWLEDGEMENTS

We would like to thank Petr Podvesky of Charles University, Prague for useful discussions about lattice cutting. 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. This work was supported by the National Science Foundation under Grant No. #IIS-9810517 and Grant No. #IIS-9820687.

REFERENCES

- [1] V. I. Levenshtein, “Binary codes capable of correcting spurious insertions and deletions of ones,” *Problems of Information transmission*, vol. 1, no. 1, pp. 8–17, 1965.
- [2] P. J. Bickel and K. A. Doksum, *Mathematical Statistics: Basic Ideas and Selected topics*, Holden-Day Inc., Oakland, CA, USA, 1977.
- [3] 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.
- [4] V. Goel and W. Byrne, “Recognizer output voting and DMC in minimum Bayes-risk framework,” in *Research Notes No. 40, Center for Language and Speech Processing*, 2000.
- [5] 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.
- [6] L. Mangu, E. Brill, and A. Stolcke, “Finding consensus in speech recognition: word error minimization and other applications of confusion networks,” *Computer Speech and Language*, vol. 14(4), pp. 373–400, 2000.
- [7] G. Evermann and P. Woodland, “Posterior probability decoding, confidence estimation and system combination,” in *Proceedings of the NIST Speech Transcription Workshop*, College Park, MD, USA, 2000.
- [8] 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.
- [9] M. Mohri, F. Pereira, and M. Riley, *AT&T General-purpose finite-state machine software tools*, 2001, <http://www.research.att.com/sw/tools/fsm/>.
- [10] 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.
- [11] V. Goel and W. Byrne, “Minimum Bayes-risk automatic speech recognition,” *Computer Speech and Language*, vol. 14(2), pp. 115–135, 2000.
- [12] S. Kumar and W. Byrne, “Risk based lattice cutting for segmental minimum Bayes-risk decoding,” in *ICSLP 2002*, Denver, CO, USA, 2002, pp. 373–376.
- [13] 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.
- [14] R. E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, USA, 1957.
- [15] D. Sankhoff and J. B. Kruskal, *Time Warps, String Edits and Macromolecules: The Theory and Practice of String Comparison*, Addison-Wesley Publishing Company, Inc., Reading, MA, USA, 1983.
- [16] M. Mohri, “Edit-distance of weighted automata,” in *Seventh International Conference on Implementation and Application of Automata*, Jean-Marc Champarnaud and Denis Maurel, Eds., 2002.
- [17] 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.
- [18] F. Wessel, R. Schlueter, and H. Ney, “Explicit word error minimization using word hypothesis posterior probabilities,” in *Proceedings of ICASSP-01*, Salt Lake City, UT, USA, 2001, pp. 33–36.
- [19] F. Wessel, K. Macherey, and R. Schlueter, “Using word probabilities as confidence measures,” in *Proceedings of ICASSP-98*, Seattle, WA, USA, 1998, pp. 225–228.

- [20] A. Stolcke, H. Bratt, J. Butzberger, H. Franco, V. R. Rao Gadde, M. Plauche, C. Richey, E. Shriberg, K. Sonmez, F. Weng, and J. Zheng, “The SRI March 2000 Hub-5 conversational speech transcription system,” in *Proceedings of the NIST Speech Transcription Workshop*, College Park, MD, USA, 2000.
- [21] S. Young et. al., *The HTK Book, Version 3.0*, July 2000.
- [22] W. Byrne, A. Gunawardana, S. Kumar, and V. Venkataramani, “The JHU March 2001 Hub-5 conversational speech transcription system,” in *Proceedings of the NIST LVCSR Workshop*. NIST, 2001, Available at <http://www.clsp.jhu.edu/research/rteval/>.
- [23] W. Byrne, P. Beyerlein, J. Huerta, S. Khudanpur, B. Marthi, J. Morgan, N. Peterek, J. Picone, D. Vergyri, and W. Wang, “Towards language independent acoustic modeling,” in *IEEE Conference on Acoustics, Speech and Signal Processing*, Istanbul, Turkey, 2000, pp. 1029–1032.
- [24] Linguistic Data Consortium, *Voice of America Broadcast News Czech Transcript Corpus*, 2000, <http://www ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2000T53>.
- [25] W. Byrne, V. Doumpiotis, S. Kumar, S. Tsakalidis, and V. Venkataramani, “The JHU 2002 large vocabulary speech recognition system,” in *Proceedings of the NIST RT-02 Workshop*. NIST, 2002, Available at <http://www.clsp.jhu.edu/research/rteval/>.
- [26] S. Tsakalidis, V. Doumpiotis, and W. Byrne, “Discriminative linear transforms for feature normalization and speaker adaptation in HMM estimation,” in *ICSLP 2002*, Denver, Colorado, USA, 2002, pp. 2585–2588.
- [27] V. Doumpiotis, S. Tsakalidis, and W. Byrne, “Lattice segmentation and minimum Bayes-risk discriminative training,” in *Eurospeech 2003*, Geneva, Switzerland, 2003.