

TASK DEPENDENT LOSS FUNCTIONS IN SPEECH RECOGNITION: A* SEARCH OVER RECOGNITION LATTICES

Vaibhava Goel, William Byrne

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

ABSTRACT

A recognition strategy that can be matched to specific system performance criteria such as word error rate or F-measure has recently been found to yield improvements over the usual maximum a-posteriori probability strategy [1] [2] [3]. In this matched-to-the-task strategy a hypothesis is chosen to minimize the expected loss or the Bayes Risk under a loss function defined by a performance measure of interest. Due to the prohibitive expense of exact implementation of this strategy, only an approximate implementation as an N-best list rescoring scheme has been used [1] [2]. Our goal is to improve the performance of such risk-based decoders by developing search strategies that can consider more hypotheses and incorporate more acoustic evidence. In this paper we present search algorithms to implement the risk-based recognition strategy over word lattices that contain acoustic and language model scores. These algorithms are extensions of the N-best list rescoring approximation and are formulated as A* algorithms. Results are reported on the Switchboard conversational telephone speech corpus. We find that lattice based rescoring yields modest but significant improvements in word error rate relative to N-best list rescoring at comparable computational cost.

1. INTRODUCTION

In many speech recognition tasks automatic speech recognition systems are evaluated on criteria other than sentence error rate. Some examples of such criteria are word error rate (WER), F-measure for Named Entity extraction tasks [4], and word-specific errors for keyword spotting tasks. It has been long known that the Bayes optimal decision rule for these applications incorporates a task dependent loss function that is closely related to the performance criterion of interest. The loss function $l(W', W)$ describes the loss incurred when an acoustic observation A produced by the word string W' is classified as belonging to W . The optimal decision rule that considers all hypotheses from the set of possible word sequences \mathcal{W} is [5]

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

We call this Bayes optimal decoding rule the *minimum-risk decoder* since it minimizes the Bayes risk

$$B(\delta(A)) = E_{P(W', A)}[l(W', \delta(A))] \quad (2)$$

which is the expected loss when $\delta(A)$ is used as the decision rule for data generated under $P(W', A)$. This distribution describes the data that will be encountered in practice.

In most applications the test set consists of many utterances and A in Equation 1 is the acoustic evidence for the whole test set, i.e. all the test set utterances considered together. Similarly, W and W' are word hypotheses for the entire test set. We now describe assumptions we make to allow the test set utterances to be processed independently. Suppose the loss function can be

broken down as

$$l(W', W) = \sum_i l(W'_i, W_i), \quad (3)$$

that is, the word strings W' and W can be divided into substrings $\{W'_i\}$ and $\{W_i\}$ such that the total loss can be computed as the summation of losses over these substrings. Linguistic considerations such as sentence and phrasal boundaries may provide one such division. If we assume also that a corresponding segmentation $\{A_i\}$ of the acoustic data can be found that makes the following conditional independence assumption plausible

$$P(W'_i|A) \approx P(W'_i|A_i) \quad (4)$$

then Equation 1 becomes

$$\delta(A_i) = \operatorname{argmin}_{W_i \in \mathcal{W}_i} \sum_{W'_i \in \mathcal{W}_i} l(W'_i, W_i) P(W'_i|A_i). \quad (5)$$

The sets \mathcal{W}_i specify the word sequences to be considered as candidates for each utterance. They are found as those word sequences with significant posterior probability given the acoustics.

We emphasize the importance of these simplifications. The first assumption reduces the spaces over which the search and the sum are performed. It may also allow separate and improved processing of subparts of a sentence. The second assumption, an approximation, reduces the computational burden of likelihood computations. In particular, computing this likelihood over an entire discourse could be prohibitively expensive. Under these two assumptions independent processing of sentences in a discourse is possible. We note that these assumptions may be difficult to ensure in practice. In particular, it may be quite difficult to find an acoustic segmentation that allows the loss function over word sequences to be decomposed as described.

Direct implementation of Equation 5 is still not possible since the true distribution $P(W_i|A_i)$ governing the observations is unknown. As a more practical matter, Equation 5 involves a **search** and a **sum** over the set of all permissible word sequences in set \mathcal{W}_i and both of these operations are impractical to implement in an unrestricted manner.

The following N-best list rescoring approximation has been proposed earlier [1]

$$\delta(A_i) = \operatorname{argmin}_{W_i \in \mathcal{W}_i^{n,b1}} \sum_{W'_i \in \mathcal{W}_i^{n,b2}} l(W'_i, W_i) P(W'_i|A_i) \quad (6)$$

where $\mathcal{W}_i^{n,b1}$ and $\mathcal{W}_i^{n,b2}$ are relatively small lists of word sequences with high posterior probabilities given the i^{th} utterance. We call $\mathcal{W}_i^{n,b1}$ the *hypothesis space* and $\mathcal{W}_i^{n,b2}$ the *evidence space*. The goal of the decoding procedure is to pick a minimum risk hypothesis from $\mathcal{W}_i^{n,b1}$ using evidence in $\mathcal{W}_i^{n,b2}$. Equation 6 can be implemented efficiently for a specified loss function if an estimate of the word sequence posterior probabilities can be obtained.

The N-best list rescoring approximation follows from an explicit enumeration of the word sequences in the search and hypothesis spaces. To avoid search errors and to consider more acoustic evidence, it is of interest to increase the size of these two spaces from N-best lists to dense word lattices produced by an ASR system.

We present a single stack A-star search and obtain an underestimate and an over-estimate of the cost needed for the search. For loss functions that do not depend on time segmentation of hypotheses, a prefix-tree based simplification of the single stack algorithm is then derived. For yet a further subset of loss functions, including the usual Levenshtein distance based loss for WER reduction tasks, we describe additional efficiencies in computation and storage.

2. A* LATTICE SEARCH FOR MINIMUM RISK DECODING

For our purposes a recognition lattice is a compact representation of a very large N-best list. As discussed above, we will consider a decoding procedure in which a recognition lattice for each utterance is processed separately; thus the utterance index i is no longer needed. Each lattice is an acyclic directed graph $(\mathcal{N}, \mathcal{E})$ with one start node and one exit node; \mathcal{N} is the set of nodes and \mathcal{E} is the set of edges. Each node in the lattice is labeled by a word and a time, (w, t) , representing the hypothesis that the utterance contains that word ending at that time. A path through the lattice is a sequence n_0^k of connected nodes. A prefix is a sequence n_0^k if n_0 is the start node of the lattice; n_k need not be the end node of the lattice. If n_k is not the end node, the prefix is called a *partial prefix*; otherwise it is called a *complete path*. A path n_0^k specifies a word sequence $w_{n_1^k}^{n_k}$ and a segmentation of the acoustic observation data $a_{n_0}^{n_k}$ beginning at time t_{n_0} and ending t_{n_k} . We assume that there is a one to one correspondence between prefixes and word sequences, i.e. no two prefix node sequences have identical words and word boundaries. A partial prefix is then a partial hypothesis and a complete path is a complete hypothesis.

The lattice represents a sparse subset of all possible word sequences with all possible word ending times. We denote this collection of complete hypotheses as \mathcal{W}_{lat} and use W to indicate a particular word sequence with word ending times. When specified as $W \in \mathcal{W}_{lat}$, W must be a complete hypothesis through the lattice. Otherwise W may be partial or complete.

Using Bayes Rule, we define our lattice decoding task by approximating Equation 1 as

$$\operatorname{argmin}_{W \in \mathcal{W}_{lat}} \sum_{W' \in \mathcal{W}_{lat}} l(W', W) P(W', A). \quad (7)$$

This search over the lattice can be effectively implemented as an A* algorithm. The goal here is to find the minimum cost complete hypothesis, where the cost of any complete hypothesis W is measured as

$$S(W) = \sum_{W' \in \mathcal{W}_{lat}} l(W', W) P(W', A). \quad (8)$$

The algorithm proceeds by extending partial hypotheses forward in time. Each hypothesis, partial or complete, has an associated cost with two properties:

- P1.** The cost of a complete hypothesis is the true cost (Equation 8) of that hypothesis.
- P2.** The cost of a partial hypothesis is a lower bound on the minimum cost that can be obtained by extending the hypothesis through the lattice to completion.

Hypotheses are kept in a stack which is sorted by cost, with the smallest cost hypothesis at the top. At every iteration the hypothesis at the top of the stack is extended. The algorithm terminates when the hypothesis occupying the top slot in the

stack is complete. This search yields a sequence of lattice nodes from which the minimum cost sentence hypothesis and its word boundary times can be derived.

In order to carry out the search we introduce the *backward log-likelihood* (Figure 1) of a lattice node n , the *forward log-likelihood* (Figure 1) of a hypothesis W , and the *total log-likelihood* of W .

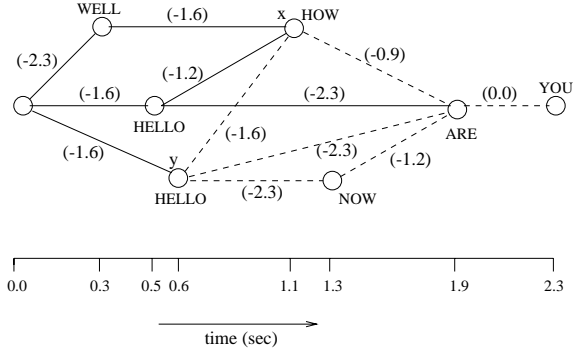


Figure 1. The *forward log-likelihood* of partial hypothesis WELL HOW (ending at node x) is the log of the likelihood of that path. The *backward log-likelihood* of node y is the log of the sum of likelihoods of all lattice paths, depicted by broken lines, from y to the lattice end node.

The *backward log-likelihood* $L_b(n)$ is given by

$$\ln\left\{\sum_{x_1^p:w_0^n \cdot w_{x_1^p}^{x_p} \in \mathcal{W}_{lat}} P(a_{x_0}^{x_p} | w_{x_1^p}^{x_p}, n) P(w_{x_1^p}^{x_p} | n)\right\} = \ln\left\{\sum_{x_1^p:w_0^n \cdot w_{x_1^p}^{x_p} \in \mathcal{W}_{lat}} \prod_{i=1}^p P(a_{x_{i-1}^p}^{x_i^p} | w_{x_i^p}) P(w_{x_i^p} | x_{i-1}^p)\right\} \quad (9)$$

where w_0^n denotes the partial hypothesis starting at the lattice start node¹ and ending at node n . The x_i are indices of nodes that follow n with the constraint that x_p is the lattice end node. It is straightforward to compute the *backward log-likelihood* recursively by traversing the lattice in reverse. A similar computation yields the *forward log-likelihood* of a hypothesis defined as

$$\begin{aligned} L_f(W = w_{n_1}^{n_k}) &= \ln\{P(a_{n_0}^{n_k} | w_{n_1}^{n_k}) P(w_{n_1}^{n_k})\} \\ &= \ln\left\{\prod_{i=1}^k P(a_{n_{i-1}}^{n_i} | w_{n_i}) P(w_{n_i} | n_{i-1})\right\} \end{aligned} \quad (10)$$

where n_0 is the start node of the lattice. The *total log-likelihood* of this hypothesis is given by

$$T(W = w_{n_1}^{n_k}) = L_f(W) + L_b(n_k). \quad (11)$$

As described above, a valid hypothesis cost $S(W)$ should obey properties **P1** and **P2**. That is, for a partial hypothesis W_p

$$S(W_p) \leq \min_{X:W_p \cdot X \in \mathcal{W}_{lat}} \sum_{W' \in \mathcal{W}_{lat}} l(W', W_p \cdot X) P(W', A) \quad (12)$$

and for a complete hypothesis W_c

$$S(W_c) = \sum_{W' \in \mathcal{W}_{lat}} l(W', W_c) P(W', A). \quad (13)$$

A possible cost function for partial hypotheses is

$$S(W_p) = \sum_{\tilde{W} \in \mathcal{W}_{st}} \min_{\substack{X:W_p \cdot X \in \mathcal{W} \\ Y:\tilde{W} \cdot Y \in \mathcal{W}}} l(\tilde{W} \cdot Y, W_p \cdot X) e^{T(\tilde{W})} \quad (14)$$

¹ w_0 is the word label associated with the start node of the lattice.

where \mathcal{W}_{st} is the set of partial and complete hypotheses present in the stack and $T(\tilde{W})$ is the *total log-likelihood* (Equation 11) of \tilde{W} . It is straightforward to show that this cost function satisfies Equation 12 and **P1**.

For complete hypotheses the cost given by Equation 13 is clearly not easy to obtain. Rather than computing this exact cost we find an *over-estimate*. If the over-estimated cost of the complete hypothesis at the top of the stack is less than the under-estimated cost of the next hypothesis in the stack, the algorithm stops. This use of an over-estimate in the A^* search may mean that the optimum candidate is not identified as soon it would be if the exact cost was available. However efficient procedures to compute over-estimates are available for some loss functions and this is necessary for the A^* procedure to be tractable.

For the Levenshtein distance loss function $L(\cdot, \cdot)$ between sentences, an over estimate for the complete hypothesis W_c can be computed as follows

- Let $C(\tilde{W})$ be the length of the longest suffix of $\tilde{W} \in \mathcal{W}_{st}$ in the lattice.
- Append each hypothesis \tilde{W} in the stack by $C(\tilde{W})$ instances of *dummy* (out of vocabulary) markers D .
- Compute the over-estimate

$$\bar{S}(W_c) = \sum_{\tilde{W} \in \mathcal{W}_{st}} L(\tilde{W} \cdot D_1^{C(\tilde{W})}, W_c) e^{T(\tilde{W})} \quad (15)$$

The following algorithms can be implemented to find the desired hypothesis in the recognition lattice. These algorithms differ in the assumptions made about the loss function.

I. Single stack search

Using the cost function of Equation 14 for the under-estimate computations and that of Equation 15 for the over-estimate computation, the following algorithm can be used to obtain the desired hypothesis:

1. Mark the lattice nodes by the *backward log-likelihood* (9). Also, at each node, keep the length of the longest path from that node to the end of the lattice.
2. Maintain a stack of lattice nodes. These correspond to partial hypotheses ending at these nodes. Each stack entry contains $L_f(W)$ (Equation 10), $T(W)$ (Equation 11), and $S(W)$ in addition to the hypothesis W itself. The stack ordering is defined first by increasing values of $S(W)$, and second by decreasing values of $T(W)$ in cases of identical $S(W)$.
3. Initialize the search by inserting the start node of the lattice, i.e. the NULL partial hypothesis, into the stack.
4. Extend each chosen node, i.e. each partial hypothesis to be extended, by all lattice arcs that leave that node. Compute $S(W)$ for each of the newly created W .
5. Insert these newly created partial hypotheses at their appropriate places in the stack. Pruning may be applied during the insertion.
6. Check the stack top for complete hypotheses. For each complete hypothesis at the top of the stack compute the over-estimate $\bar{S}(W_c)$.
7. If the over-estimate $\bar{S}(W_c)$ is smaller than the cost $S(W)$ of hypotheses in the stack, W_c is the desired candidate. Otherwise choose the stack top for extension and go to step 4.

II. Prefix tree search

In our treatment so far each hypothesis implicitly has an associated time segmentation so that hypotheses with same word contents but different time segmentation are different. We now

describe a procedure that merges identical word hypotheses with different time segmentations. Let U be the operator that strips the time segmentations from hypotheses. U applied to a set of hypotheses with time segmentations gives another set of hypotheses by first removing the time segmentations from the hypotheses and then clearing all the duplications. For the loss functions that do not depend on the time segmentation of hypotheses, the cost function of Equation 14 can be rearranged, using the operator U , as

$$\begin{aligned} S(W_p) &= \sum_{\tilde{W} \in \mathcal{W}_{st}} \min_{\substack{X:W_p \cdot X \in \mathcal{W} \\ Y:\tilde{W} \cdot Y \in \mathcal{W}}} l(\tilde{W} \cdot Y, W_p \cdot X) e^{T(\tilde{W})} \quad (16) \\ &= \sum_{\substack{\tilde{W} \in U(\mathcal{W}_{st}), \\ \tilde{W} : U(\tilde{W}) = \tilde{W}}} \min_{\substack{X:W_p \cdot X \in \mathcal{W} \\ Y:\tilde{W} \cdot Y \in \mathcal{W}}} l(\tilde{W} \cdot Y, W_p \cdot X) e^{T(\tilde{W})} = \\ &\sum_{\tilde{W} \in U(\mathcal{W}_{st})} \min_{\substack{X:W_p \cdot X \in \mathcal{W} \\ Y:\tilde{W} \cdot Y \in \mathcal{W}}} l(\tilde{W} \cdot Y, W_p \cdot X) \sum_{\tilde{W}:U(\tilde{W})=\tilde{W}} e^{T(\tilde{W})} \end{aligned}$$

Although the cost $S(W_p)$ is computed for each partial hypothesis W_p , it is equal for hypotheses with identical word contents but different time boundaries. This observation, along with the rearrangement presented above, immediately suggests a prefix tree based implementation of the desired search. The algorithm can be described as follows

1. Mark the lattice nodes by the *backward log-likelihood*. At each node keep the length of the longest path starting from that node.
2. Maintain a stack of prefix tree nodes. Each prefix tree node identifies the lattice nodes that correspond to that prefix. The stack is ordered by $S(W_p)$ (Equation 16).
3. Initialize the prefix tree root node with the lattice start node. Initialize the search by putting the prefix tree root node in the stack.
4. Extend the chosen prefix tree node by all lattice arcs that follow that prefix in the lattice. Compute $S(W_p)$ for each newly created prefix.
5. Insert the newly created nodes at their appropriate place in the stack.
6. Check the stack top for complete prefixes. For the complete prefix W_c , compute the over-estimate $\bar{S}(W_c)$ according to Equation 15.
7. If the over-estimate is smaller than the cost $S(W)$ of all other prefixes, W_c is the desired hypothesis. Otherwise choose the stack top for extension and go to step 4.

For Levenshtein distance based loss functions the prefix tree facilitates the storage of alignments between pairs of hypotheses needed for computation of

$$\min_{\substack{X:W_p \cdot X \in \mathcal{W} \\ Y:\tilde{W} \cdot Y \in \mathcal{W}}} l(\tilde{W} \cdot Y, W_p \cdot X) \quad (17)$$

in the Equation 16 above. A *square grid* can be constructed, each node of which is labeled with a prefix tree node pair. Each node of the square tree thus represent two partial hypotheses and can be used to maintain their Levenshtein distance, as well as other information needed in the search. Due to the recursive nature of the distance, the nodes of the square grid can be added progressively.

Multi-stack organization

The quality of the under-estimate of Equation 16 depends on the length (in words) of the partial hypotheses. In order to better compare the under-estimates of partial hypotheses, the single

stack is split into multiple stacks, so that each stack contains hypotheses of equal length. The hypothesis with the least score among the top hypotheses of these stacks, i.e. the same one that would have been chosen in the single stack, is extended. This re-organization allows better pruning of hypotheses.

3. EXPERIMENTS AND RESULTS

The prefix tree search algorithm was evaluated on the Switchboard LVCSR task [6]. The test set consisted of 2427 utterances from 19 conversations (38 sides) that formed the dev-test at the 1997 Johns Hopkins University LVCSR Workshop; full details of the test set definition, language models used, and other details are given in [7]. A set of bigram lattices was generated that used acoustic models incorporating interjection and word boundary information. These lattices were then rescored with a trigram language model and HMMs adapted for each speaker using VTLN. The resulting lattices had an average of 446.6 nodes and 1629.7 links for each word in the reference transcriptions. The MAP candidate from these lattices had WER of 38.5% and served as our baseline for comparisons.

In order to compare the speed of the prefix tree search to that of the N-best list rescoring procedure, N-best lists of 1000 entries were arranged as a lattice and search was carried out on these lattices. This allowed comparing the prefix tree search to the N-best search when both were restricted to use identical hypothesis and evidence spaces. The speed of the prefix tree search was found to be almost identical to that of the N-best list rescoring procedure.

Two forms of pruning were used during the prefix tree search. The likelihood of the most likely completion of each partial hypothesis was compared with the likelihood of the most likely complete hypothesis in the lattice and the partial hypothesis was rejected if the likelihood ratio exceeded a threshold. Partial hypotheses were also pruned by their cost under-estimates whenever the square grid became unmanageably large. Under these pruning conditions the prefix tree search took approximately twice as long as the N-best list rescoring procedure with a 25 element hypothesis set and a 1000 element evidence set.

	WER
Baseline (MAP)	38.5
N-best rescoring	37.9
Prefix tree search	37.5

Table 1. Word error rate result of the prefix tree search and its comparison to the N-best list rescoring procedure. For N-best list rescoring a hypothesis space of 25 elements and an evidence set of 1000 elements was used.

4. DISCUSSION AND CONCLUSIONS

We have formulated an A^* search algorithm for implementation of the minimum Bayes-risk decision rule on a recognition lattice for the task of word error rate reduction. Two procedures, a single stack based search and a prefix tree and square grid based multi-stack search, are presented. The prefix tree implementation, while comparable to the N-best list rescoring procedure in speed, yields slightly better word error rate performance on our test set and explores many more candidates during search.

The efficiency of our A^* algorithms critically depends on the quality of the under-estimates (Equation 14) and of the over-estimates (Equation 15) used in the search. Better estimates might also prevent search space explosion by only exploring the better candidates. We are continuing our efforts to improve these estimates.

The construction of the square grid relies on the recursive nature of the Levenshtein distance. It may be difficult to apply the prefix-tree-square-grid based search to other loss functions that cannot be found recursively.

The single stack search requires finding an under-estimate and an over-estimate of the true cost of the best completion of a par-

tial hypothesis. Furthermore these estimates must be found using only the other partial hypotheses in the stack. The single stack search may be used for other loss functions only if these conditions can be met.

The assumption of Equation 3 allows independent processing of subparts of sentences if the total loss over sentences can be well approximated by the sum of losses over these subparts. The work of Mangu et.al. [8] towards reduction of word error rate can be interpreted in the light of this assumption where each subpart can contain at most one word.

The A^* search can be extended to the recognition trellis. It may therefore be possible to implement the search for the minimum error hypothesis using the lattice as a word graph only. This would further generalize the search by not restricting it to the time segmentations, acoustic scores, and language model scores provided in the lattice.

5. ACKNOWLEDGMENT

We are grateful to Entropic Cambridge Research Laboratory, Cambridge, UK, for providing the software for lattice generation and rescoring.

REFERENCES

- [1] A. Stolcke, Y. Konig, and M. Weintraub. Explicit word error minimization in N-best list rescoring. *Eurospeech-97*, pp. 163–165, Rhodes, Greece, 1997.
- [2] V. Goel, W. Byrne, and S. Khudanpur. LVCSR rescoring with modified loss functions: A decision theoretic perspective. *ICASSP-98*, pp. 425–428, Seattle, WA, 1998.
- [3] V. Goel, W. Byrne. Task dependent loss functions in speech recognition: Application to Named Entity extraction. *ESCA ETRW Workshop on Accessing Information from Spoken Audio*, pp. 49–53, Cambridge, UK, 1999.
- [4] C. J. Van Rijsbergen. *Information Retrieval*. 2nd edition, London, Butterworths, 1979.
- [5] P. J. Bickel and K. A. Doksum. *Mathematical Statistics: Basic Ideas and Selected topics*. Holden-Day Inc., Oakland, CA, 1977.
- [6] J. J. Godfrey, E. C. Holliman, and J. McDaniel. Switchboard: Telephone speech corpus for research and development. *Proc. ICASSP*, pp. 517–520, San Francisco, CA, 1992.
- [7] *Proceedings of the 1997 Large Vocabulary Continuous Speech Recognition Workshop*, Johns Hopkins University, Baltimore, MD, 1997.
- [8] L. Mangu, E. Brill, and A. Stolcke. Finding Consensus Among Words: Lattice-Based Word Error Minimization. To appear in *Eurospeech-99*, Budapest, Hungary, 1999.