Minimum Bayes-Risk Methods in Automatic Speech Recognition

Vaibhava Goel – IBM

William Byrne – Johns Hopkins University

Outline

- Minimum Bayes-Risk Classification Framework
  - Likelihood Ratio Based Hypothesis Testing
  - Maximum A-Posteriori Probability Classification
  - Previous Studies of Application Sensitive ASR

- Practical MBR Procedures for ASR
  - Summation over Hidden State Sequences
  - MBR Recognition with N-best Lists
  - MBR Recognition with Lattices

Outline

- Segmental MBR Procedures
  - Segmental Voting
  - ROVER
  - e-ROVER
- Experimental Results
  - Parameter Tuning within the MBR Classification Rule
  - Utterance Level MBR Word and Keyword Recognition
  - ROVER and e-ROVER for Multilingual ASR
- Summary

- Minimum Bayes-Risk Classification Framework
  - Likelihood Ratio Based Hypothesis Testing
  - Maximum A-Posteriori Probability Classification
  - Previous Studies of Application Sensitive ASR
- Practical MBR Procedures for ASR
  - Summation over Hidden State Sequences
  - MBR Recognition with N-best Lists
  - MBR Recognition with Lattices

Minimum Bayes-Risk Classification Framework

- Definition:

A : acoustic observation sequence

W : word string

$\mathbf{W}_h^A$: the hypothesis space of the observation A

$\delta(A): A \rightarrow W_h^A$: ASR classifier

$l(W, W')$: loss function, where W' is mistranscription of W

$P(W, A)$: true distribution of speech and language

Minimum Bayes-Risk Classification Framework

How to measure classifier performance?

$$\text{Using Bayes-risk} \rightarrow E_{P(W,A)}\left[l(W,\delta(A))\right] = \sum_{A}\sum_{W} l(W,\delta(A))P(W,A) \quad (2.1)$$

$$\therefore \delta(A) = \arg\min_{W' \in \mathbf{W}_h^A} \sum_{W} l(W,W')P(W \mid A) \quad (2.2) \text{ is chosen to minimize Bayes-risk}$$

$$\left(\text{but we use } \delta(A) = \arg\min_{W' \in \mathbf{W}_h^A} l(W_c, W'), W_c \text{ is the correct transcription of } A\right)$$

$$\text{Let } \mathbf{W}_e^A \text{ be the subset of } W, \text{ with nonzero } P(W \mid A) \Rightarrow \mathbf{W}_e^A = \{W \mid P(W \mid A) > 0\}$$

$$\therefore \text{Equation 2.2 can be rewritten as } \delta(A) = \arg\min_{W' \in \mathbf{W}_h^A} \sum_{W \in \mathbf{W}_e^A} l(W,W')P(W \mid A) \quad (2.4)$$

$$\text{Let } \sum_{W \in \mathbf{W}_e^A} l(W,W')P(W \mid A) = S(W') \;\; \therefore \delta(A) = \arg\min_{W' \in \mathbf{W}_h^A} S(W')$$

Minimum Bayes-Risk Classification Framework

Since the observations in $\mathbf{W}_e^A$ serve as the evidence used by MBR classifier.

$\therefore\ \mathbf{W}_e^A$ is refered as evidence space

and $P(W \mid A)$ is refered as evidence distribution

How to define loss function ? Two ways :

loss function $= l_{LRT}(X,Y) \Rightarrow$ classifier $= \delta_{\mathrm{LRT}}(A)$

method $\rightarrow$ likelihood ratio hypothesis testing

loss function $= l_{0/1}(X,Y) \Rightarrow$ classifier $= \delta_{\mathrm{MAP}}(A)$

method $\rightarrow$ maximum a - posteriori classification

Likelihood Ratio Based Hypothesis Testing

If $\mathbf{W_e} = \{H_n, H_a\}$ and $\mathbf{W_h} = \{H_n, H_a\}$ and define $l_{LRT}(X,Y) = \begin{cases} 0 & \text{if } X = H_n, Y = H_n \\ t_1 & \text{if } X = H_a, Y = H_n \\ t_2 & \text{if } X = H_n, Y = H_a \\ 0 & \text{if } X = H_a, Y = H_a \end{cases}$

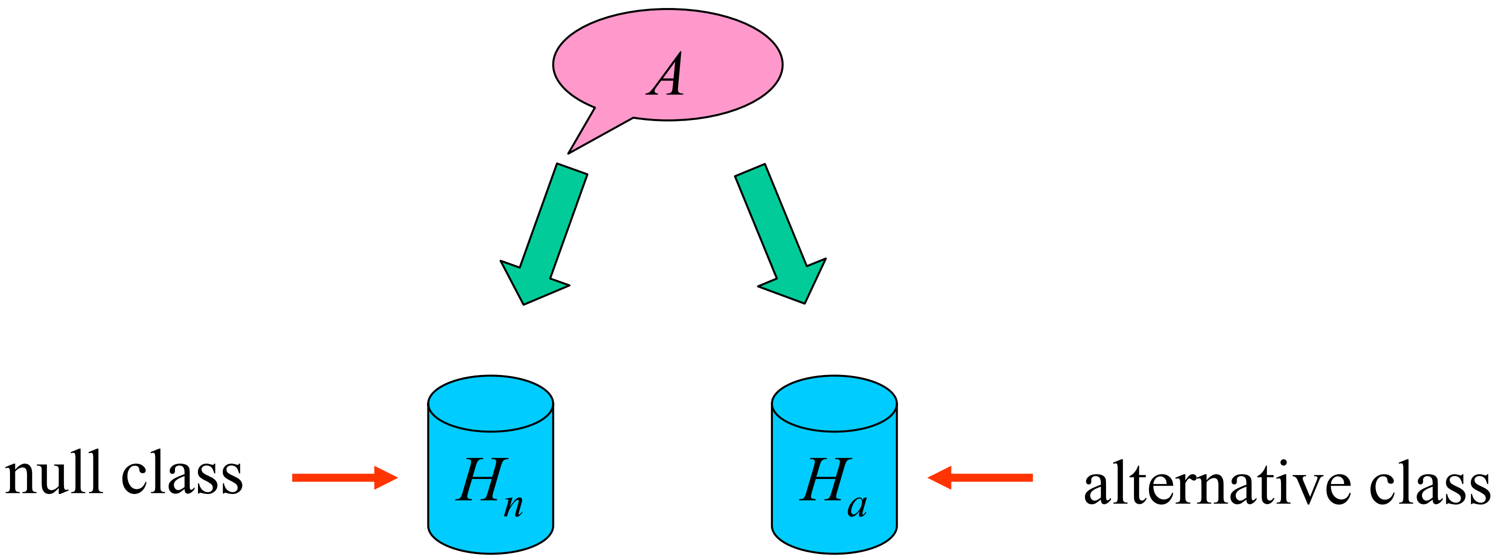
$$
\begin{aligned}
\therefore \delta(A) &= \arg\min_{W' \in \mathbf{W}_h^A} \sum_{W \in \mathbf{W}_e^A} l(W, W') P(W \mid A) \\
&= \arg\min_{W' \in \{H_n, H_a\}} \left[l(H_n, W') P(H_n \mid A) + l(H_a, W') P(H_a \mid A) \right] \\
&= \arg\min \begin{Bmatrix} \left[l(H_n, H_n) P(H_n \mid A) + l(H_a, H_n) P(H_a \mid A) \right], \\ \left[l(H_n, H_a) P(H_n \mid A) + l(H_a, H_a) P(H_a \mid A) \right] \end{Bmatrix} \\
&= \arg\min \begin{Bmatrix} \left[t_1 P(H_a \mid A) \right], \\ \left[t_2 P(H_n \mid A) \right] \end{Bmatrix} = \arg\min \begin{Bmatrix} \left[t_1 P(A \mid H_a) P(H_a) \right], \\ \left[t_2 P(A \mid H_n) P(H_n) \right] \end{Bmatrix} \\
&= \begin{cases} H_n & t_2 P(A \mid H_n) P(H_n) > t_1 P(A \mid H_a) P(H_a) \\ H_a & \text{otherwise} \end{cases} = \begin{cases} H_n & \dfrac{P(A \mid H_n)}{P(A \mid H_a)} > \dfrac{t_1 P(H_a)}{t_2 P(H_n)} = t \\ H_a & \text{otherwise} \end{cases}
\end{aligned}
$$

Likelihood Ratio Based Hypothesis Testing

$$\therefore \delta_{LRT}(A) = \begin{cases} H_n & \text{if } \dfrac{P(A \mid H_n)}{P(A \mid H_a)} > t \\ H_a & \text{otherwise} \end{cases} \qquad (2.6)$$

The threshold t is set in an application specific manner;

it determines the balance between false rejection and false aceptance.



Maximum A-Posteriori Probability Classification

$$\text{Define } l_{0/1}(W,W') = \begin{cases} 1 & \text{if } W' \neq W \\ 0 & \text{otherwise} \end{cases}$$

$$\therefore\ \delta(A) = \arg\min_{W' \in \mathbf{W}_h^A} \sum_{W \in \mathbf{W}_e^A} l(W,W')P(W \mid A)$$

$$= \arg\min_{W' \in \mathbf{W}_h^A} \sum_{W \neq W'} P(W \mid A)$$

$$= \arg\min_{W' \in \mathbf{W}_h^A} \left(1 - P(W' \mid A)\right)$$

$$= \arg\max_{W' \in \mathbf{W}_h^A} P(W' \mid A)$$

Previous Studies of Application Sensitive ASR

- Use of ***risk minimization*** in automatic speech has not been extensive.

- Early investigations into the minimum ***Bayes-risk training criteria*** for speech recognizers were performed by ***Nadas***.

- However our focus in this chapter is in ***minimum-risk classification*** rather than ***estimation***.

Previous Studies of Application Sensitive ASR

- ***Stolcke*** et.al. proposed an approximation to a minimum Bayes risk classifier for ***generation of minimum word error rate hypothesis*** from recognition ***N-best lists***.

- Other researchers have proposed ***posterior probability*** and ***confidence based*** hypothesis selection strategies for word error rate reduction.

- Minimum Bayes-Risk Classification Framework
  - Likelihood Ratio Based Hypothesis Testing
  - Maximum A-Posteriori Probability Classification
  - Previous Studies of Application Sensitive ASR
- Practical MBR Procedures for ASR
  - Summation over Hidden State Sequences
  - MBR Recognition with N-best Lists
  - MBR Recognition with Lattices

Practical MBR Procedures for ASR

- Why difficult to implement?
  - The ***evidence and hypothesis spaces*** in Equation 2.4 tend to be ***quite large***.
  - The problem of large spaces is worsened by the fact that an ASR recognizer often has to process ***many consecutive utterances***.
  - There are efficient DP techniques for MAP recognizer, such methods ***are not yet available for an MBR recognizer*** under an arbitrary loss function.

Practical MBR Procedures for ASR

- How to implement?
  - Two implementation:
    - ***N-best list rescoring procedure***
    - ***Search over a recognition lattice***
  - Segment long acoustic data into sentence or phrase length utterances.
  - Restrict the evidence and hypothesis spaces to manageable sets of word strings.

Summation over Hidden State Sequences

- A computational issue associated with the use of HMM in the evidence distribution will be addressed.
- How to obtain the true distribution?

$$P(W \mid A) = \frac{P(W)P(A \mid W)}{P(A)} \quad (2.12)$$

Here $P(W)$ is approximated using a *language model*,
it is usually a Markov chain based N - gram model.

$P(A \mid W)$ is usually approximated using a HMM called the *acoustic model*.

Let S be the set of all the states in the acoustic HMM $P(A \mid W)$.

Let χ denote the set of all possible state sequences that could generate A.

The probability $P(A \mid W)$ is computed as

$$P(A \mid W) = \sum_{X \in \chi} P(A, X \mid W) = \sum_{X \in \chi} P(X \mid W) P(A \mid X, W) \quad (2.13)$$

Summation over Hidden State Sequences

The summation over all possible hidden state sequences is too expensive.

A computationally feasible alternative is to modify the Equation 2.4 as

$$\delta(A) = \arg \min_{W' \in \mathbf{W}_h^A} \sum_{W \in \mathbf{W}_e^A} l(W, W') P(W \mid A)$$

$$= \arg \min_{W' \in \mathbf{W}_h^A} \sum_{W \in \mathbf{W}_e^A} l(W, W') \frac{P(W) \sum_{X \in \boldsymbol{\chi}} P(X \mid W) P(A \mid X, W)}{P(A)}$$

$$= \arg \min_{W' \in \mathbf{W}_h^A} \sum_{W \in \mathbf{W}_e^A} l(W, W') \sum_{X \in \boldsymbol{\chi}} P(W, X, A)$$

$$\approx \arg \min_{(W', X') \in \mathbf{W}_h^A \times \boldsymbol{\chi}^A} \sum_{(W, X) \in \mathbf{W}_e^A \times \boldsymbol{\chi}^A} l\big((W, X), (W', X')\big) P(W, X, A)$$

where $\boldsymbol{\chi}^A$ is a sparse sampling of the most likely state sequences in $\boldsymbol{\chi}$.

Summation over Hidden State Sequences

For convenience we use

W rather than (W, X)

$\mathbf{W}_h^A$ rather than $\mathbf{W}_h^A \times \boldsymbol{\chi}^A$

$\mathbf{W}_e^A$ rather than $\mathbf{W}_e^A \times \boldsymbol{\chi}^A$

$$\therefore \text{ we have } \delta(A) = \arg\min_{W' \in \mathbf{W}_h^A} \sum_{W \in \mathbf{W}_e^A} l(W, W')P(W, A) \quad (2.15)$$

MBR Recognition with N-best Lists

The most direct approximation of Equation 2.15 is by N - best list rescoring procedures as first proposed for WER minimization.

In this approach, the evidence and hypothesis spaces are restricted to the N - best lists produced by a recognizer.

They are denoted N_e and N_h

$$\therefore \delta(A) \approx \arg\min_{W' \in N_h} \sum_{W \in N_e} l(W, W') P(W, A)$$

This approximation is particularly easy to implement for arbitrary loss functions.

It is of interest to increase the size of these two spaces to the recognition lattice, e.e., to consider more candidates in the search and the sum.

MBR Recognition with Lattices

- Multistack prefix tree $A*$ search algorithm uses recognition lattices as the hypothesis and evidence.

Lattice Definitons :

A compact representation for a large set of word strings and their time boundaries.

$(N, \varepsilon, n_s, n_e, \rho)$ an acyclic directed graph

N : the set of nodes

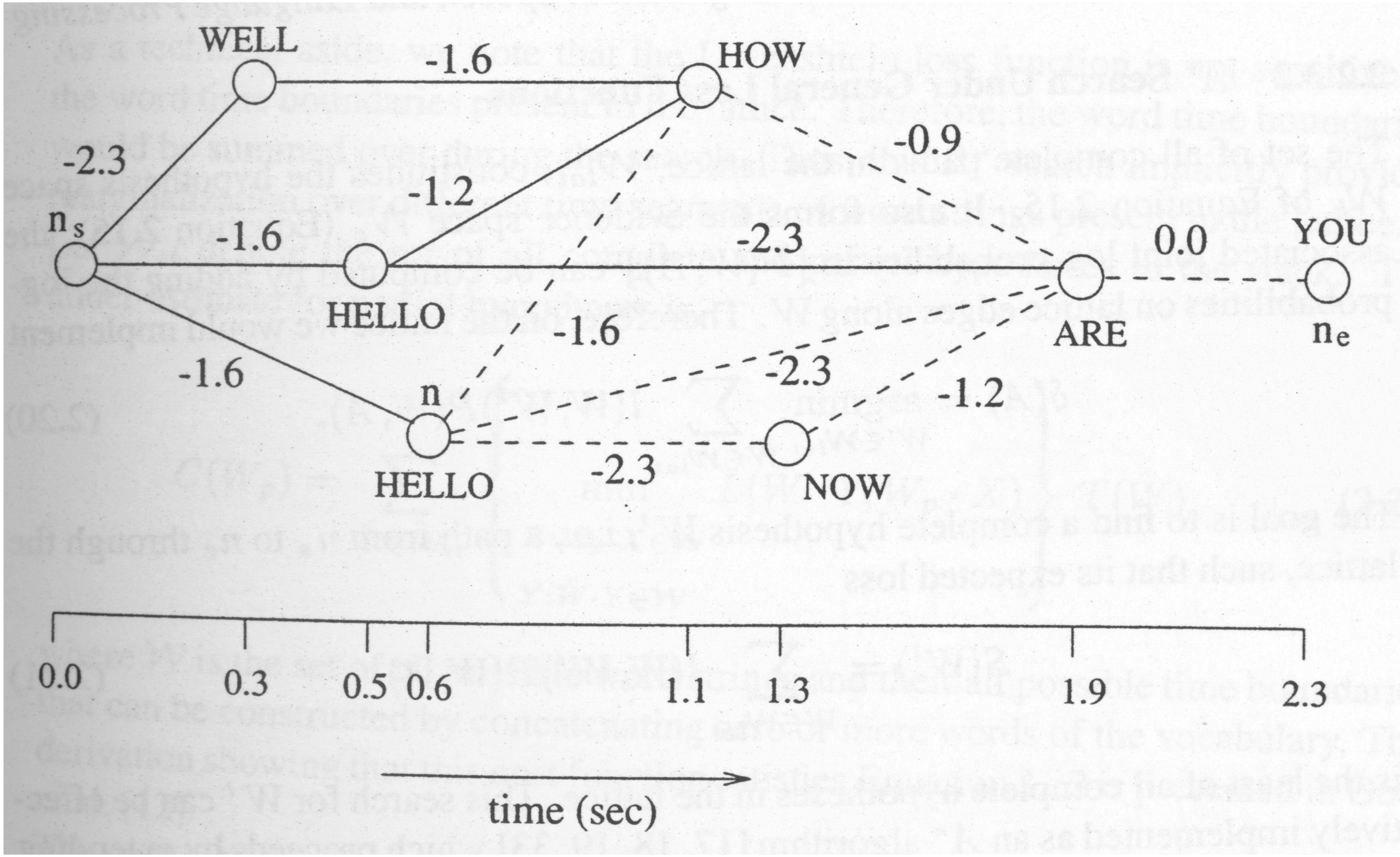
ε : the set of edges

n_s : the unique lattice start node

n_e : the unique lattice end node

$\rho : N \times \varepsilon \to N$ specifies lattice connectivity.

MBR Recognition with Lattices



MBR Recognition with Lattices

$Path$ or $Complete\ \ path$: start node n_s to end node n_e

$Path\ Segment\ (W_x)$: (internal) node n_x to (internal) node n_y

(W_e) : (internal) node n_x to end node n_e

$Partial\ \ path\ (W_p)$: n_s to (internal) node n_x

Acoustic segments corresponding to these three path will be $A(W_x)$, $A(W_e)$ and $A(W_p)$ respectively.

The sum of log - probability on the edges will be $P(W_x, A(W_x))$, $P(W_e, A(W_e))$ and $P(W_p, A(W_p))$ respectively.

MBR Recognition with Lattices

The partial path log-probability of $W_p = L_f(W_p) = \ln\{P(W_p, A(W_p)\}$. The lattice backward

log-probability of W_p is $L_b(W_p) = \ln\left\{\sum_{W_e: W_p \cdot W_e \in W_{lat}} P(W_e, A(W_e) \mid W_p)\right\}$ (2.18)

where W_{lat} denotes the st of all complete paths in the lattice.

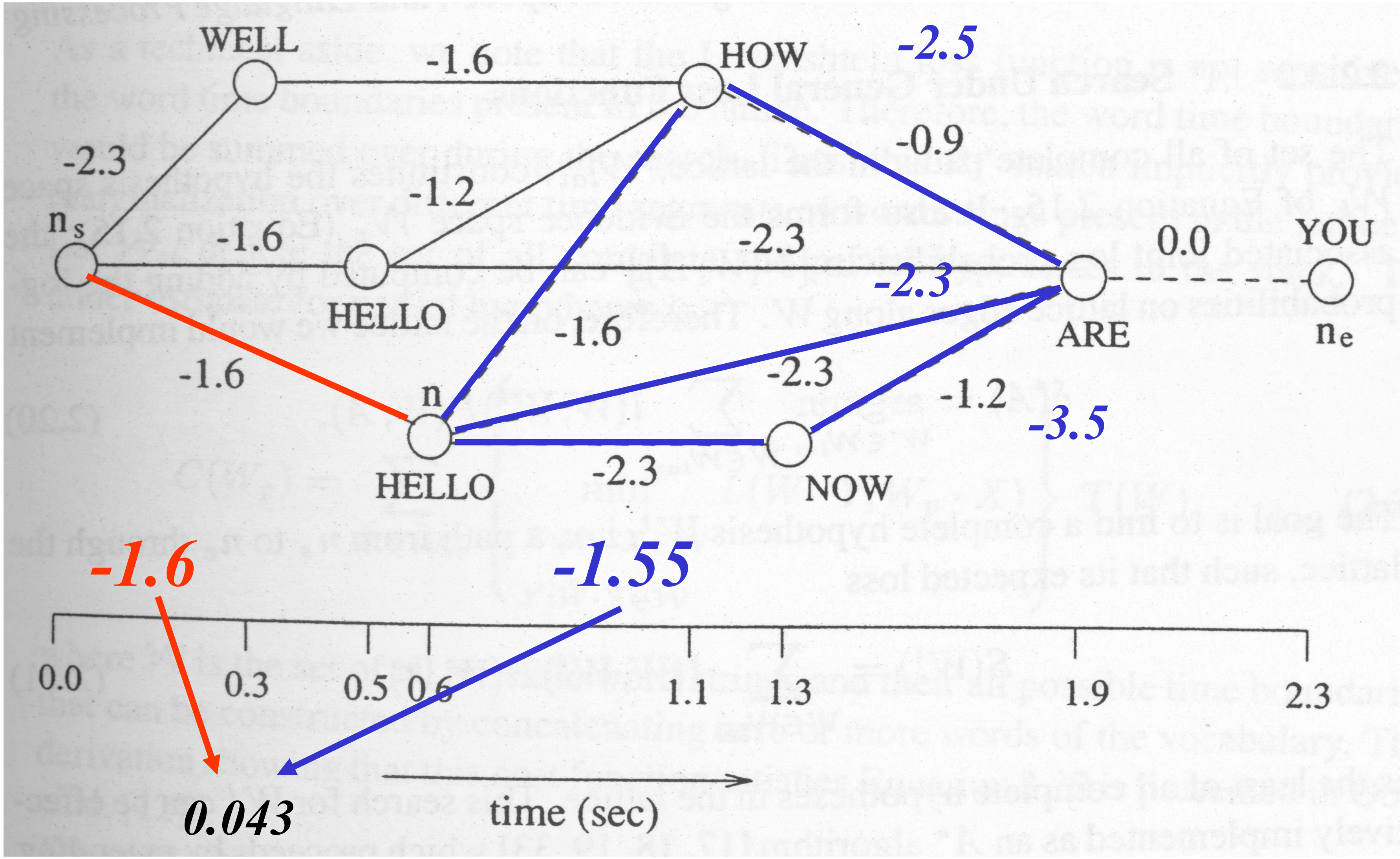
The lattice total probability of $W_p = T(W_p) = \exp\{L_f(W_p) + L_b(W_p)\}$

$$= \exp\left\{\ln[P(W_p, A(W_p))] + \ln\left[\sum_{W_e: W_p \cdot W_e \in W_{lat}} P(W_e, A(W_e) \mid W_p)\right]\right\}$$

$$= \exp\left\{\ln\left[\sum_{W_e: W_p \cdot W_e \in W_{lat}} P(W_e, A(W_e) \mid W_p) P(W_p, A(W_p))\right]\right\}$$

$$= \exp\left\{\ln\left[\sum_{W_e: W_p \cdot W_e \in W_{lat}} P(W_p \cdot W_e, A)\right]\right\} = \sum_{W_e: W_p \cdot W_e \in W_{lat}} P(W_p \cdot W_e, A)$$

MBR Recognition with Lattices



MBR Recognition with Lattices

The jonit log - probability $\ln\{P(W, A)\}$ can be computed by adding the log - probabilities on lattice edges along W.

Therefore, on the lattice we would implement

$$\delta(A) = \arg\min_{W' \in \mathbf{W}_{lat}} \sum_{W \in \mathbf{W}_{lat}} l(W,W')P(W, A) \quad (2.20)$$

The goal is to find a complete hypothesis W', i.e., a path from n_s to n_e through the lattice, such that its expected loss

$S(W') = \sum_{W \in W_{lat}} l(W,W')P(W, A)$ is the least of all complete hypothesis.

This search for W' can be effectively implemented as an A^* algorithm.

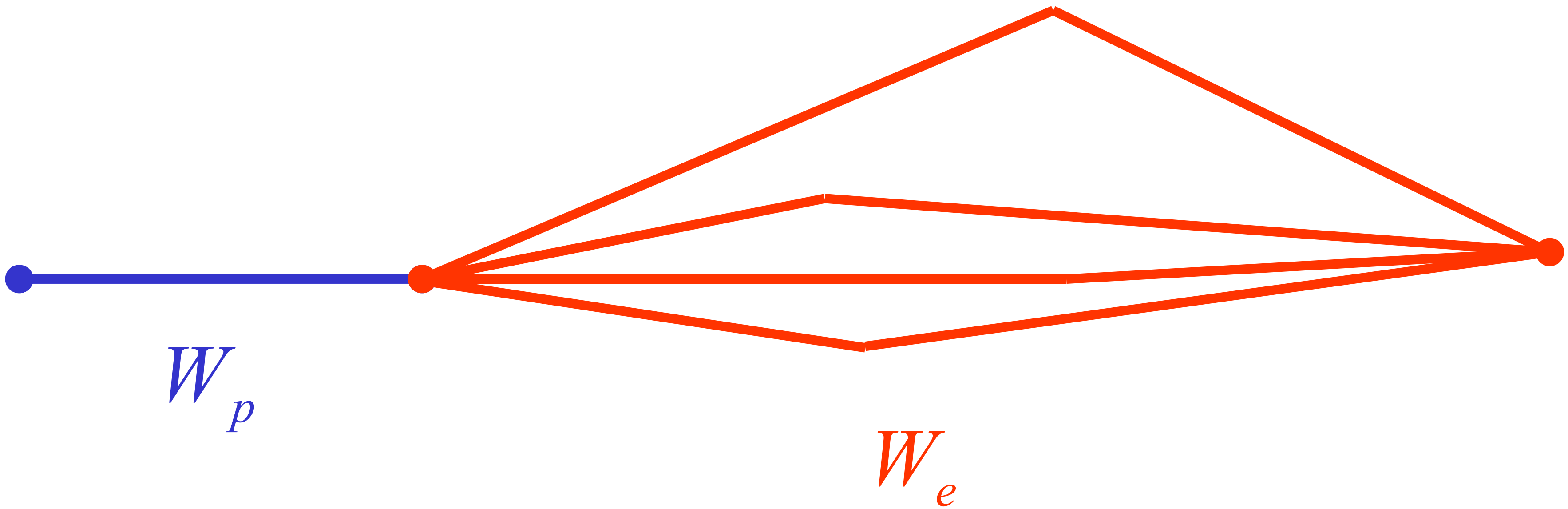
MBR Recognition with Lattices

- Two cost functions are required for the search.
  - The first cost function is associated with each hypothesis W_p, whether partial or complete.
    - Its value is a lower bound on the expected loss that can be obtained by extending the hypothesis through the lattice to completion.

$$C(W_p) \le \min_{W_e : W_p \cdot W_e \in W_{lat}} \sum_{W \in W_{lat}} l(W, W_p \cdot W_e) P(W, A) \quad (2.22)$$

$$C(W_p) \le \min_{W_e} S(W_p \cdot W_e)$$

MBR Recognition with Lattices



MBR Recognition with Lattices

- The second cost function in only associated with complete hypotheses.
  - It is an over-estimate of the expected loss of a complete hypothesis W'

$$\overline{C}(W') \geq \sum_{W \in W_{lat}} l(W, W') P(W, A) = S(W') \quad (2.23)$$

- Hypotheses are kept in a priority queue sorted by cost C, with the smallest cost hypothesis at the top.
- At every iteration the hypothesis at the top of the stack is extended.

MBR Recognition with Lattices

- When to terminate?
  - When there is a complete hypothesis at the top, its second cost $\overline{C}$ is computed. If this over-estimate cost $\overline{C}$ is smaller than the under-estimate cost C of the next stack hypothesis.
  - There is no partial hypothesis left in the stack.
- Why we use over-estimate?
  - Since A* usually employ an exact expected loss for complete hypotheses; however, this is prohibitively expensive to find in our case.

Single Stack Search Levenshtein Loss Function

- We now present usable const functions for the Levenshtein distance $L(W, W')$
- These costs are not unique, and the efficiency of the search depends on the quality of both the under-estimate and the over-estimate.
- The Levenshtein loss function is not sensitive to the word time boundaries.
- Therefore, the word time boundaries would be summed over during the search.

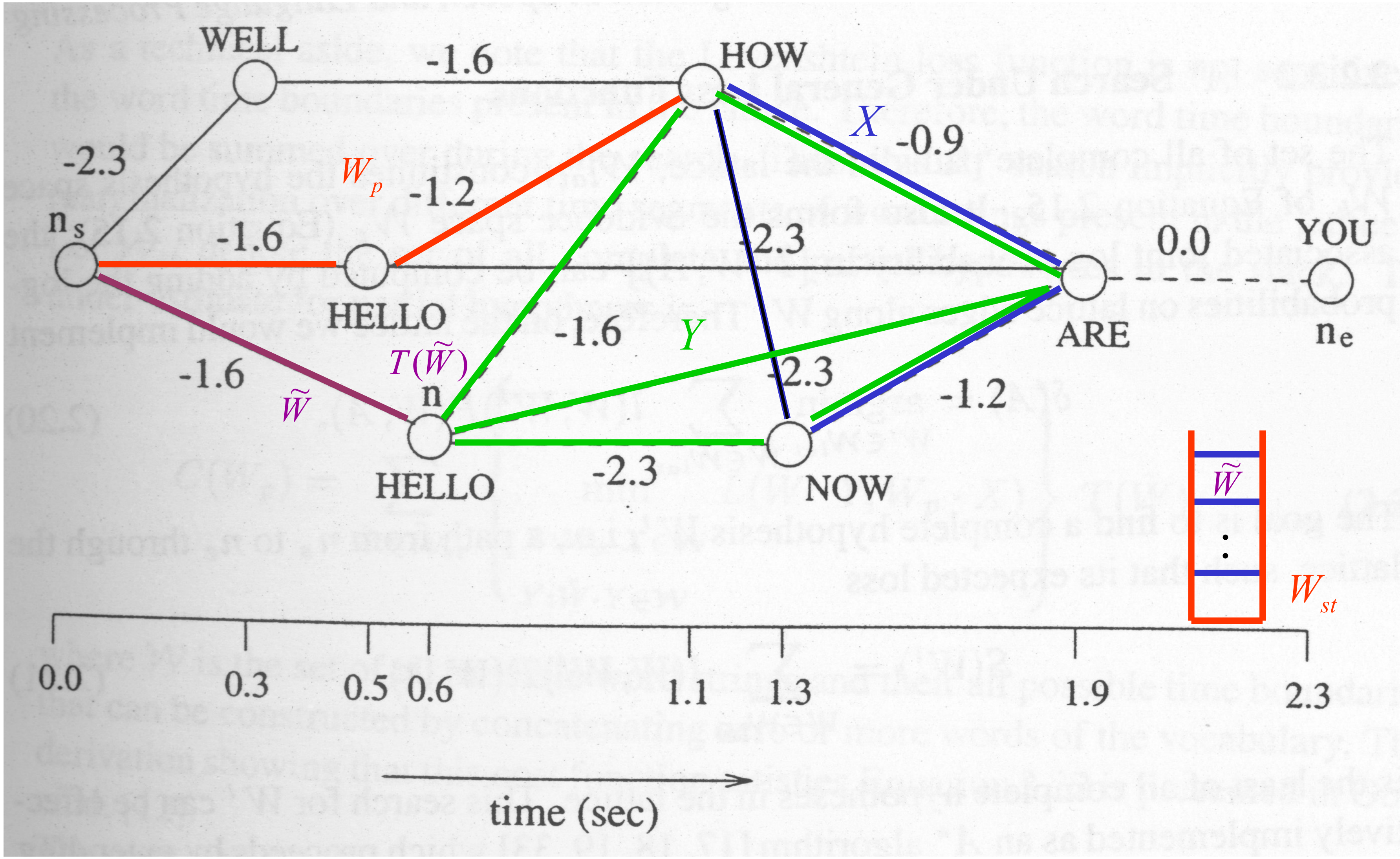
Single Stack Search Levenshtein Loss Function

Let W_{st} denote the set of all complete and partial hypotheses in the stack. The under - estimate for partial hypotheses is

$$C(W_p) = \sum_{\tilde{W} \in W_{st}} \left\{ \min_{\substack{X : W_p \cdot X \in W \\ Y : \tilde{W} \cdot Y \in W}} L(\tilde{W} \cdot Y, W_p \cdot X) \right\} T(\tilde{W}) \quad (2.24)$$

where W is the set of all possible word strings and their all possible time boundaries that can be constructed by concatenating zero or more words of the vocabulary.

Single Stack Search Levenshtein Loss Function



Single Stack Search Levenshtein Loss Function

The derivation showing that this cost function satisfies Equation 2.22

The over - estimate for a complete hypothesis W' can be computed as follows :

- For a hypothesis $\tilde{W}$ in stack W_{st}, let $N(\tilde{W})$ be the length of the longest path from its end node to the lattice end node n_e.
- Append each hypothesis $\tilde{W}$ in the stack by $N(\tilde{W})$ instances of out of vocabulary markers D. These markers do not match any word in the vocabulary.
- Compute the over - estimate

$$\overline{C}(W') = \sum_{\tilde{W} \in W_{st}} L(\tilde{W} \cdot D_1^{N(\tilde{W})}, W') T(\tilde{W}) \quad (2.25)$$

Single Stack Search Levenshtein Loss Function

With the under - estimate (2.24) and the over - estimate (2.25), the following single stack search algorithm can be used to find the desired hypothesis in the recognition lattice.

1. Mark the lattice nodes by the lattice backward log - probability(2.18). At each node keep the length of the longest path to the end of the lattice.
2. Maintain a stack W_{st} of partial and complete hypotheses. Each partial stack entry contains a hypothesis W_p, $L_f(W_p)$(2.17), $T(W_p)$(2.19), and $C(W_p)$(2.24). Each complete stack entry contains a hypothesis W', $T(W')$, $C(W')$, and $\overline{C}(W')$(2.25). The stack ordering is defined first by increasing values of $C(\cdot)$, and second by decreasing values of $T(\cdot)$ in cases of identical $C(\cdot)$.
3. Initialize the search by inserting the start node of the lattice, i.e., the NULL hypothesis, into the stack.

Single Stack Search Levenshtein Loss Function

4. If there are incomplete hypotheses in the stack, extend the top incomplete hypothesis by all lattice arcs that leave its end node. Compute $C(W_p)$ for each of the newly created partial hypothesis W_p. Compute $C(W')$, and $\overline{C}(W')$ for each newly created complete hypothesis W'. Otherwise, if there are no incomplete stack hypotheses, select the hypothesis with least $\overline{C}(W')$. This is the desired candidate.
5. Update the cost estimates(2.24 and 2.25) of all other partial and complete stack hypotheses after adding theses newly created hypotheses to the evidence space. Insert the newly created hypotheses at their appropriate places (sorted first by $C(\cdot)$ and second by $T(\cdot)$ in case of ties) in the stack. Pruning may be applied during the insertion.
6. If there is a complete hypothesis at the top of the stack and its over - estimate is smaller than the under - estimate of second stack hypothesis (partial or complete), it is the desired candidate and the search ends. Otherwise ot to step 4.

Prefix Tree Search Under Levenshtein Loss Function

Since the Levenshteion distance does not depend on the time segmentation of hypotheses

$\therefore$ time information $\rightarrow$ removed.

Let U be the operator that strips the time segmentations from hypotheses.

Given a partial hypothesis W_p from the stack W_{st}, let $\Phi_p = U(W_p)$ be its word contents.

Let $T(\Phi_p) = \sum_{\tilde{W} \in W_{st} : U(\tilde{W}) = \Phi_p} T(\tilde{W})$ be the induced total probability of Φ_p over the current stack.

Prefix Tree Search Under Levenshtein Loss Function

The cost function can be rearranged using the operator U as

$$C(W_p) = \sum_{\tilde{W} \in W_{st}} \min_{\substack{X : W_p \cdot X \in W \\ Y : \tilde{W} \cdot Y \in W}} L(\tilde{W} \cdot Y, W_p \cdot X) T(\tilde{W})$$

$$= \sum_{\Phi \in U(W_{st})} \sum_{\tilde{W} \in U(\tilde{W}) = \Phi} \min_{\substack{a : \Phi_p \cdot a \in U(W) \\ b : \Phi \cdot b \in U(W)}} L(\Phi \cdot b, \Phi_p \cdot a) T(\tilde{W})$$

$$= \sum_{\Phi \in U(W_{st})} \min_{\substack{a : \Phi_p \cdot a \in U(W) \\ b : \Phi \cdot b \in U(W)}} \sum_{\tilde{W} \in U(\tilde{W}) = \Phi} L(\Phi \cdot b, \Phi_p \cdot a) T(\tilde{W})$$

$$= \sum_{\Phi \in U(W_{st})} \min_{\substack{a : \Phi_p \cdot a \in U(W) \\ b : \Phi \cdot b \in U(W)}} L(\Phi \cdot b, \Phi_p \cdot a) T(\Phi)$$

$$= C(\Phi_p)$$

Prefix Tree Search Under Levenshtein Loss Function

Therefore the cost of a partial hypothesis W_p depends only on its word contents $\Phi_{p.}$

This suggests that we can introduce a *prefix tree* as a compact representation of the word sequences associated with all partial hypotheses in the stack.

A node in the prefix tree identifies a set of hypotheses and their end nodes in the lattice.

Prefix Tree Search Under Levenshtein Loss Function

It is the same as the single stack search except that

1. The stack contains prefix tree nodes and is ordered first by $C(\Phi_p)$ and then by $T(\Phi_p)$ in case of ties.
2. The lattic paths corresponding to the prefix tree node at the top of the stack are extended by one word. These extensions yield a new set of prefix tree nodes to be inserted in the stack.

The over - estimate is still computed according to Equation 2.25

A significant advantage of using prefix trees for the Levenshtein distance is that they facilitate storage and computation of

$$\min_{\substack{a:\Phi_p \cdot a \in U(W) \\ b:\Phi \cdot b \in U(W)}} L(\Phi \cdot b, \Phi_p \cdot a) \rightarrow \textit{partial hypothesis comparison cost}$$

Pruning and Multistack Organization of the Prefix Tree Search

- Equation 2.24 and 2.25 didn’t take pruning into account.
- When entries are pruned from the stack, Equation 2.24 is still a valid under-estimate but Equation 2.25 is no longer a valid over-estimate.
- It is however a valid over-estimate for the sub-lattice of the original lattice that could be constructed by completion of the partial hypotheses in the pruned stack.

Pruning and Multistack Organization of the Prefix Tree Search

- The single stack search and the prefix tree search have the disadvantage that the costs of partial hypotheses of different lengths are compared.
- This is acceptable under the search formulation, but is not a good comparison for use in pruning since it favors short hypotheses.→ sub-optimal.
- How to solve it?
  - Using multistack implementation that maintains a separate stack for each hypothesis length.
  - It has been found to have better pruning characteristics in practice.

- Segmental MBR Procedures
  - Segmental Voting
  - ROVER
  - e-ROVER
- Experimental Results
  - Parameter Tuning within the MBR Classification Rule
  - Utterance Level MBR Word and Keyword Recognition
  - ROVER and e-ROVER for Multilingual ASR
- Summary

Segmental MBR Procedures

- Segmental MBR (SMBR).
- Utterance level recognition → sequence of simpler MBR recognition.
- The lattices or N-best lists are segmented into sets of words.
- Advantages:
  - The segmentation can be performed to identify high confidence regions within the evidence space.
  - Within the regions we can produce reliable word hypotheses.
  - But SMBR focuses on the low confidence regions.

Segmental MBR Procedures

Definition	evidence	hypothesis
space	W_e $\downarrow R_e$ $W_e^1, \cdots, W_e^N$	W_h $\downarrow R_h$ $W_h^1, \cdots, W_h^N$
word sequence	W $\downarrow R_e$ $R_e^1(W), \cdots, R_e^N(W)$	W' $\downarrow R_h$ $R_h^1(W'), \cdots, R_h^N(W')$
reconstruct		$W'^1, \cdots, W'^N$ $\downarrow J_h$ W'

Segmental MBR Procedures

- Assume that the utterance level loss can be found from the losses over the segment sets as

$$l(W,W') = \sum_{i=1}^{N} l^i(R_e^i(W), R_h^i(W))$$

where l^i is a loss function defined on the i^{th} segment set.

Proposition. An utterance level MBR recognizer of can be implemented as a concatenation of N MBR recognizers $\delta(A) = J_h(\delta^i(A)|_{i=1}^{N})$ (2.29)

where $\delta^i(A) = \arg\min_{W'^i \in W_h^i} \sum_{W^i \in W_e^i} l^i(W^i, W'^i) P^i(W^i \mid A)$ (2.30)

and $P^i(W^i \mid A)$ is the marginal probability over the

i^{th} evidence set $P^i(W^i) = \sum_{W \in W_e : R_e^i(W) = W^i} P(W \mid A)$ (2.31)

Segmental MBR Procedures

- Therefore, under the assumption of Equation 2.28, utterance level MBR recognition becomes a sequence of smaller MBR recognition problems.
- In practice it may be difficult to segment the evidence and hypothesis spaces.
- Utterance level induced loss function is defined as

$$l_I(W,W') = \sum_{i=1}^{N} l^i(R_e^i(W), R_h^i(W)) \quad (2.32)$$

- The overall performance under the desired loss function l should depend on how well l_I approximates l

Segmental Voting

- Special case of segmental MBR recognition
- Suppose each evidence and hypothesis segment set contains at most one word.
- There is a 0/1 loss function on segment sets.
- Under these conditions the segmental MBR recognizer of Equation 2.30 becomes

$$\delta(A^i) = \arg\max_{W'^i \in W_h^i} P^i(W'^i \mid A) \quad (2.33)$$

- The utterance level induced loss for segmental voting

$$l_{\text{seg-vote}}(W, W') = \sum_{i=1}^{N} l_{0/1}(R_e^i(W), R_h^i(W')) \quad (2.34)$$

Segmental Voting

- We will now describe two versions of segmental MBR recognition used in state-of-the-art ASR systems.

- Both these procedures attempt to reduce the word error rate (WER) and thus are based on the Levenshtein loss function.

ROVER

- Recognizer Output Voting for Error Reduction (ROVER) is an N-best list segmental voting procedure.
- It combines the hypotheses from multiple independent recognizers under the Levenshtein loss.

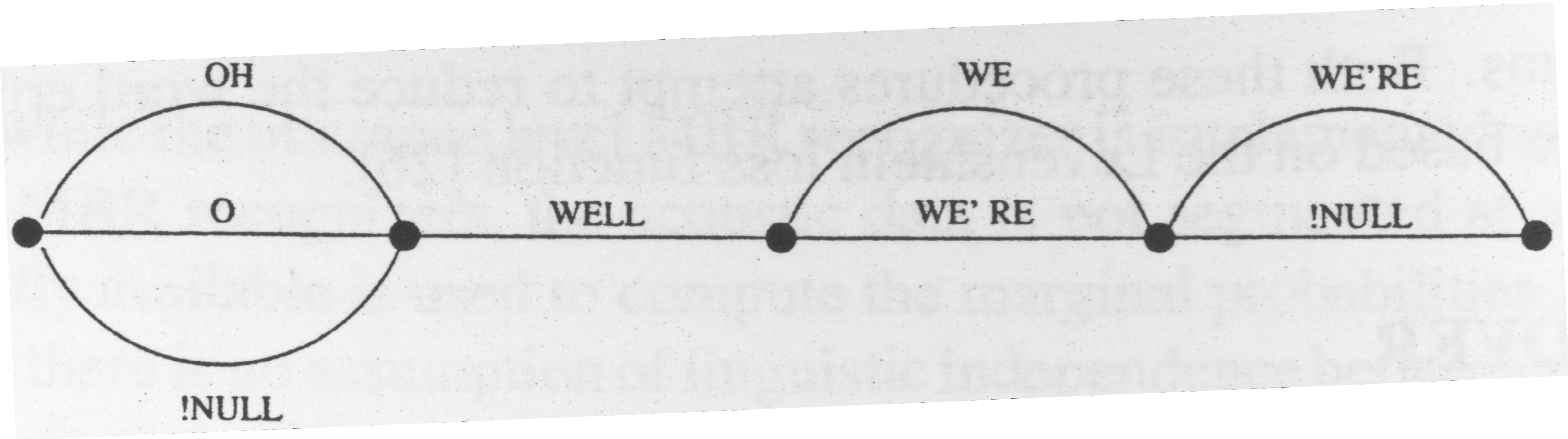
Let $N_m, m = 1, \cdots, K$ be N - best lists produced by K recognition systems in response to acoustics A, and let P_m be the posterior distribution associated with N_m

$$P(W \mid A) = \sum_{m=1}^{K} \alpha_m P_m(W \mid A) \quad , W \in N_e \quad , \sum_{m=1}^{K} \alpha_m = 1$$

The set N_e and $P(W \mid A)$ are the evidence space and the evidence distribution used by ROVER

ROVER

- The word strings of N_e are arranged in a word transition network (WTN) that represents an approximate *simultaneous alignment* of these hypotheses.



Segmental Voting

- The utterance level induced loss in ROVER is derived

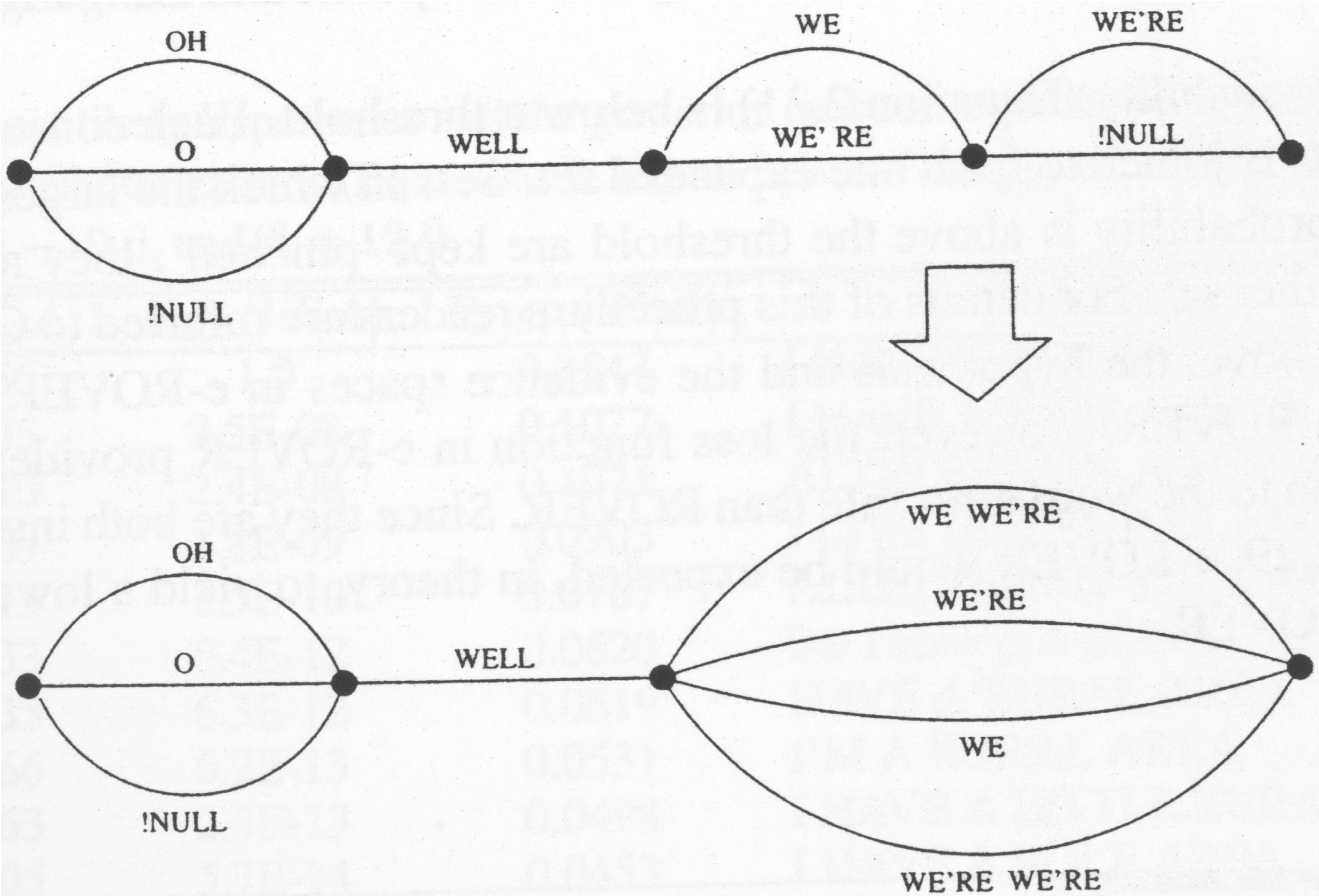
$$l_{\text{ROVER}}(W, W') = \sum_{i=1}^{N} l_{0/1}(R_e^i(W), R_h^i(W')) \quad (2.36)$$

- This loss is similar to the Levenshtein distance between strings *W* and *W'* when their alignment is specified by the WTN.

e-ROVER

- Extended-ROVER (e-ROVER)
- The utterance level loss function of e-ROVER is given as follows.
  - Start with initial WTN
  - Merge two consecutive set.
  - Let the loss function on the expanded set be the Levenshtein distance.
  - The loss function on correspondence sets that did not expand remains the 0/1 loss.

e-ROVER


OH
O
!NULL
WELL
WE
WE' RE
WE'RE
!NULL
OH
O
!NULL
WELL
WE WE'RE
WE'RE
WE
WE'RE WE'RE

e-ROVER

- The utterance level induced loss in e-ROVER is

$$l_{\text{e-ROVER}}(W,W') = \sum_{i=1, i\neq m, i\neq m+1}^{N} l_{0/1}(R_e^i(W), R_h^i(W')) + L(W^m, W'^m) \quad (2.37)$$

Here, W^m and W'^m are word subsequences from the joined segment sets.

- It follows from the definition of Levenshtein distance that

$$L(W,W') \leq l_{\text{e-ROVER}}(W,W') \leq l_{\text{ROVER}}(W,W')$$

e-ROVER

- There are two consequences of joining correspondence sets:
  - After the joining operation, the loss function on the expanded set is no longer the 0/1 loss but is instead the Levenshtein distance.
  - The size of the expanded set grows exponentially with the number of joining operations, making Equation 2.30 progressively difficult to implement.
- Therefore, it is important to determine the sets to be joined carefully so as to yield maximum gain in Levenshtein distance approximation with minimum combinations of the correspondence sets.

- Segmental MBR Procedures
  - Segmental Voting
  - ROVER
  - e-ROVER
- Experimental Results
  - Parameter Tuning within the MBR Classification Rule
  - Utterance Level MBR Word and Keyword Recognition
  - ROVER and e-ROVER for Multilingual ASR
- Summary

Parameter Tuning within the MBR Classification Rule

- The joint distribution $P(W, A)$ to be used in the MBR recognizers is derived by combining probabilities from acoustic and language models.
- It is customary in ASR to use two tuning parameters in the computation of joint probability

$$P_{\alpha,\beta}(W, A) = e^{\alpha|W|} P(A \mid W) P(W)^{\beta} \quad (2.38)$$

where $|W|$ is the number of words in word string W

$\alpha(\textit{word insertion penalty}) \rightarrow$ negative constant,

causes a decrease of probability with increasing $|W|$

$\beta(\textit{language model scale factor}) \rightarrow$ scales the language

model probability relative to the acoustic model probability

Parameter Tuning within the MBR Classification Rule

We have found it useful to introduce an additional *likelihood scale factor* γ

$$P_{\alpha,\beta,\gamma}(W, A) = \left[e^{\alpha|W|} P(A \mid W) P(W)^{\beta}\right]^{\frac{1}{\gamma}} \quad (2.39)$$

TABLE 2.1

Example ten most likely hypotheses and the posterior probability of these hypotheses under two different parameterizations (Equations 2.38 and 2.39) of the posterior distribution. $\gamma = 15.0$, $\alpha = -10.0$, and $\beta = 12.0$.

$\ln P_{\alpha,\beta}(W, A)$	$P_{\alpha,\beta}(W\|A)$	$P_{\alpha,\beta,\gamma}(W\|A)$	Sentence
-22402.56	1.0	0.3547	I HAVE A RURAL AREA
-22420.05	2.5E-08	0.1077	I HAVE A REAL RURAL AREA
-22420.10	2.4E-08	0.1074	ALTHOUGH IN A RURAL AREA
-22422.69	1.8E-09	0.0903	I LIVE IN A RURAL AREA
-22425.15	1.5E-10	0.0767	ALTHOUGH IT WILL AREA
-22428.33	6.4E-12	0.0620	SO I HAVE A RURAL AREA
-22428.35	6.3E-12	0.0619	HAVE A RURAL AREA
-22430.66	6.2E-13	0.0531	I'M A RURAL AREA
-22431.63	2.4E-13	0.0498	I HAVE A LITTLE RURAL AREA
-22433.05	5.7E-14	0.0453	I HAVE A ROLE AREA

Optimization of Likelihood Parameters

Let $\delta_{\alpha,\beta,\gamma}$ be the minimum - risk recognizer incorporating the parameterized distribution $P_{\alpha,\beta,\gamma}(W, A)$ of Equation 2.39

We optimize α, β, and γ to minimize the empirical risk of $\delta_{\alpha,\beta,\gamma}$

$$\sum_{(W,A)} l(W, \delta_{\alpha,\beta,\gamma}(A)) \qquad (2.40)$$

over a database $\mathrm{T} = \{(W, A)\}$ of labeled utterances.

Since the utterance labels are known, this is *supervised optimization*.

Unsupervised optimization : minimize the empirical risk using the most likely evidence string in place of the truth.

Utterance Level MBR Word and Keyword Recognition

TABLE 2.2
Evaluation of parameter tuning and recognition procedures for minimization of WER and KER.

Baseline (MAP): WER = 38.5, KER = 43.2

	Recognition & Tuning Criterion	Parameter Tuning Strategy	Likelihood Scale Factor (γ)	Recognition Strategy: N-best WER	N-best KER	A^* WER	A^* KER
A	WER	LM Scale	12.0	37.9	42.9	37.7	42.5
	WER	Supervised	15.2	37.9	43.0	37.5	42.4
	WER	Unsupervised	15.2	37.9	43.0	37.5	42.4
B	KER	LM Scale	12.0	38.7	42.5	N/A	41.9
	KER	Supervised	18.0	38.7	42.4	N/A	41.6
	KER	Unsupervised	15.5	38.8	42.0	N/A	41.4

ROVER and e-ROVER for Multilingual ASR