FINDING CONSENSUS AMONG WORDS: LATTICE-BASED WORD ERROR MINIMIZATION

Lidia Mangu
Eric Brill
Andreas Stolcke

1Department of Computer Science, Johns Hopkins University, Baltimore, MD, U.S.A.
{lidiam,brill}@cs.jhu.edu
2Speech Technology and Research Laboratory, SRI International, Menlo Park, CA, U.S.A.
stolcke@speech.sri.com

ABSTRACT

We describe a new algorithm for finding the hypothesis in a recognition lattice that is expected to minimize the word error rate (WER). Our approach thus overcomes the mismatch between the word-based performance metric and the standard MAP scoring paradigm that is sentence-based, and that can lead to sub-optimal recognition results. To this end we first find a complete alignment of all words in the recognition lattice, identifying mutually supporting and competing word hypotheses. Finally, a new sentence hypothesis is formed by concatenating the words with maximal posterior probabilities. Experimentally, this approach leads to a significant WER reduction in a large vocabulary recognition task.

1. INTRODUCTION

Word lattices are used by most speech recognizers as a compact representation of a set of alternative hypotheses. In the standard MAP decoding approach [1] the recognizer outputs the string of words corresponding to the path with the highest posterior probability given the acoustics and a language model. However, even given optimal models, the MAP decoder does not necessarily minimize the word error rate (WER). To this end, one should maximize individual word posterior probabilities. Previous work [8] has shown how WER can be explicitly minimized in an N-best rescoring approach.

We address the problem of extracting word hypotheses with minimal expected word error from word lattices. Word lattices promise better performance than N-best lists for two basic reasons. First, they provide a larger set of hypotheses from which to choose; second, the more accurate representation of the hypothesis space gives better estimates for word posterior probabilities and, consequently, of expected word error. However, as we will see below, the lattice representation also leads to new computational problems: it is no longer feasible to compute word errors between hypotheses explicitly.

In this paper, we describe a new algorithm for carrying out a practical, approximate word error minimization on recognition lattices. Our paper is organized as follows. In Section 2 we give a mathematical formulation of the word error minimization problem and motivate the algorithm, which is described in detail in Section 3. Section 4 gives an experimental evaluation of the algorithm. Section 5 discusses related work and other possible applications of the methods developed here. Conclusions are given in Section 6.

2. APPROACH

2.1. Word Error Minimization

In the standard approach to speech recognition [1], the goal is to find the sentence hypothesis that maximizes the posterior probability $P(W|A)$ of the word sequence $W$ given the acoustic information $A$. We call this the “sentence MAP” approach. Sentence posteriors are then usually approximated as the product of a number of knowledge sources, and normalized. For example, given a language model $P(W)$ and acoustic likelihoods $P(A|W)$, we can approximate

$$P(W|A) \approx \frac{P(W)P(A|W)}{\sum_i P(W_i)P(A|W_i)} \quad (1)$$

where $k$ ranges over the set of hypotheses output by the recognizer.

Bayesian decision theory (e.g., [2]) tells us that maximizing sentence posteriors minimizes the sentence level error (the probability of having at least one error in the sentence string). However, the commonly used performance metric in speech recognition is word error, i.e., the Levenshtein (string edit) distance $WE(W, R)$ between a hypothesis $W$ and a reference string $R$. $WE(W, R)$ is defined as the number of substitutions, deletions, and insertions in $W$ relative to $R$ under an alignment of the two strings that minimizes a weighted combination of these error types.

Given word error as our objective function, we can replace the MAP approach with a new hypothesis selection approach based on minimizing the expected word error under the posterior distribution:

$$E_{P(R|A)}[WE(W, R)] = \sum_R P(R|A) WE(W, R) \quad (2)$$

2.2. The N-best Algorithm

Equation 2 provides a general recipe for computing expected word-level error from sentence-level posterior estimates. A direct algorithmic version involves two iterations: a summation over potential references $R$ and a minimization over hypotheses $W$:

$$W_e = \arg \min \sum_i P(R^k|A) WE(W^k, R^k) \quad (3)$$

We refer to the hypothesis $W_e$ thus obtained as the center hypothesis. In previous work [8], we implemented word error minimization in exactly this fashion, letting both $W$ and $R$ range over the $N$ best hypothesis output by a recognizer. In practice, this is feasible for N-best lists of as many as a few thousand hypotheses. Others have investigated the N-best approach to minimize objective functions other than standard word error, such as named-entity recognition metrics [5].

2.3. Lattice-based Word Error Minimization

In moving to lattice-based hypothesis selection, we are faced with a computational problem. The number of hypotheses contained in a lattice is several orders of magnitudes larger than in N-best lists, making a straightforward computation of the center hypothesis as in (3) infeasible. A natural approach to this problem is to exploit the structure of the lattice for efficient computation of the center hypothesis. Unfortunately, there seems to be no efficient (e.g., dynamic programming) algorithm of this kind. The main difficulty is that the objective function is based on pairwise string distance, a non-local measure. A single word difference anywhere in a lattice path can have global consequences on...
Word $e$ be the set of words, and $p(F) = \sum_{e \in F} p(e)$ be the total posterior probability of a link subset $F \subset E$.

Formally, an alignment consists of an equivalence relation over the word hypotheses (edges) in the lattice, together with a total ordering of the equivalence classes, such that the ordering is consistent with that of the original lattice. Each equivalence class corresponds to one "position" in the alignment, and the members of a class are those word hypotheses that are "aligned to each other," i.e., represent alternatives.

The lattice defines a partial order $\preceq$ on the links. For $e, f \in E$, $e \preceq f$ if

- $e = f$
- $F_{\text{node}}(e) = F_{\text{node}}(f)$
- $\exists e' \in E$ such that $e \preceq e'$ and $e' \preceq f$.

Informally $e \preceq f$ means that $e$ "comes before" $f$ in the lattice.

Now let $E \subset 2^E$ be a set of equivalence classes on $E$, and let $\preceq$ be a partial order on $E$. We say that $\preceq$ is consistent with the lattice order if $e_1 \preceq e_2 \implies [e_1] \preceq [e_2]$, for all $e_1, e_2 \in [e_1], [e_2], [e_3] \in E$. Consistency means that the equivalence relation preserves the temporal order of word hypotheses in the lattice.

Given a lattice, then, we are looking for an ordered link equivalence that is consistent with the lattice order and is also a total (linear) order, i.e., for any two $e_1, e_2 \in E$, $[e_1] \preceq [e_2]$ or $[e_2] \preceq [e_1]$. Many such equivalences exist; for example, one can always sort the links topologically and assign each link its own class. However, such an alignment would be very poor: it would vastly overestimate the word error between hypotheses.

We initialize the link equivalence such that each initial class consists of all the links with the same starting and ending time and the same word label. Starting with this initial partition, the algorithm successively merges equivalence classes until a totally ordered equivalence is obtained.

Correctness and termination of the algorithm are based on the following observation. Given a consistent equivalence relation with two classes $E_1$ and $E_2$ that are not ordered ($E_1 \not\preceq E_2$ and $E_2 \not\preceq E_1$), we can always merge $E_1$ and $E_2$ to obtain a new equivalence that is still consistent and has strictly fewer unordered classes. We are thus guaranteed to create a totally ordered, consistent equivalence relation after a finite number of steps.

Our clustering algorithm has two stages. We first merge only clusters corresponding to same word instances (intra-word clustering), and then start grouping together heterogeneous clusters (inter-word clustering), based on the phonetic similarity of the word components. At the end of the first stage we are able to compute word posterior probabilities, but it is only after the second stage that we are able to compute competing word hypotheses in specific regions of the speech signal.

### 3.1 Intra-word Clustering

The purpose of this step is to group together all the links corresponding to same word instance. Candidates for merging at this step are all the clusters that are not in relation and correspond to the same word. The metric used for intra-word clustering is the following similarity measure between two sets of links:

$$\text{SIM}(E_1, E_2) = \max_{e_1 \in E_1} \overline{\text{overlap}}(e_1, e_2) \cdot p(e_1) \cdot p(e_2)$$

where $\overline{\text{overlap}}(e_1, e_2)$ is defined as the time overlap between the two links normalized by the sum of their lengths. The temporal overlap is weighted by the link posteriors so as to make the measure less sensitive to unlikely word hypotheses. At each step we compute the similarity between all possible pairs of cluster candidates, and merge those that are most similar. At the end of this iterative process we obtain a link equivalence relation that has overlapping instances of the same word clustered together.
3.2. Inter-word Clustering
At this step we start grouping together clusters corresponding to
different words. Candidates for merging are any two classes that
are not in relation. The algorithm stops when no more candidates
are available, i.e., a total order has been achieved.

The metric used for inter-word clustering is the following simil-
arity measure based on a phonetic similarity between words:

\[
SIM(F_1, F_2) = \frac{\text{avg}}{w_1 \in \text{Words}(F_1) \ni \text{avg}} \{ \text{sim}(w_1, w_2) \}
\]

\[
w_1 \in \text{Words}(F_1) \ni \text{avg} \{ \text{sim}(w_1, w_2) \}
\]

where \( \text{sim}(\cdot, \cdot) \) is the phonetic similarity between two words,
computed using the most likely phonetic base form. In our im-
plementation we defined phonetic similarity to be 1 minus the edit
distance of the two phone strings, normalized by the sum of their
lengths. Other, more sophisticated definitions are conceivable,
\text{eg.}, by taking phone similarities into account.

3.3. Pruning
Typical word lattices contain links with very low posterior prob-
ability. Such links are negligible in computing the total posterior
probabilities of word hypotheses, but they can have a detrimen-
tal effect on the alignment. This occurs because the alignment
preserves consistency with the lattice order, no matter how low how
the probability of the links imposing the order is. For example,
in Figure 2 we see BE and ME, which are phonetically similar
and overlap in time, and should therefore be mutually exclusive.
However, even a single path with BE preceding ME, no mat-
ter how low in probability, will prevent BE and ME from be-
ning aligned. To eliminate such cases we introduce a preliminary
pruning step that removes all links whose posteriors are below an
ability among all links at a given position. This is equivalent
to finding the path through the confusion graph with the highest
combined link weight.

3.4. Confusion Networks
The total posterior probability of an alignment class can be
strictly less than 1. That happens when there are paths in the
original lattice that do not contain a word at that position; the
missing probability mass corresponds precisely to the probabil-
ity of a deletion (or null word). We explicitly represent deletions
by a link \( e_- \) with the corresponding empty word \( \text{Word}(e_-) = \_ \).

For example, in the lattice in Figure 1(a) there are some hy-
potheses having \( "T" \) as the first word, while others have no cor-
responding word in that position. The final alignment thus con-
tains two competing hypotheses in the first position: the word
\( "T" \) (with posterior equal to the sum of all hypotheses starting
with that word), and the null word (with posterior equal to the
sum of all other hypotheses).

As illustrated in Figure 1(b), the alignment is itself equivalent
lent to a lattice, which we refer to as a confusion network. The
confusion network has one node for each equivalence class of
original lattice nodes (plus one initial/final node), and adjacent
nodes are linked by one edge per word hypothesis (including the
null word).

3.5. The Consensus Hypothesis
Once we have a complete alignment it is straightforward to ex-
tract the hypothesis with the lowest expected word error. Let \( C_i, i = 1, \ldots, I \) be the final link equivalence classes making up
the alignment. We need to choose a hypothesis \( W = w_1, \ldots, w_L \) such that \( w_i = "-" \) or \( w_i = \text{Word}(e_i) \) for some \( e_i \in C_i \). It is
easy to see that the expected word error of \( W \) is the sum total of
word errors for each position in the alignment. Specifically, the
expected word error at position \( i \) is

\[
1 - \sum_{e \in C_i} \text{Word}(e) = w_i \quad \text{P}(e) \quad \text{if} \quad w_i \neq "-" \quad \text{and} \quad w_i \quad \text{if} \quad w_i = "-"
\]

In other words, the best hypothesis is obtained by picking the
links in the confusion graph that have the highest posterior prob-
ability among all links at a given position. This is equivalent
to finding the path through the confusion graph with the highest
combined link weight.

3.6. Score Scaling
Posterior probability estimates are based on a combination of
recognizer acoustic and language model scores (Equation 1). Since
the posteriors are combined additively, rather than maxim-
imized, it is important to scale the scores correctly. Contrary to
standard practice in MAP decoding, it is better to reduce the dy-
amic range of the acoustic scores than to increase that of the
language model [8]. In our experiments we used

\[
\log P(W|A) = \log P(W) + \log P(Q|W) + \frac{1}{\lambda} \log P(A|W) + C
\]

where \( \lambda \) is the language model weight, \( P(Q|W) \) is the aggregate
pronunciation probability, and \( C \) is a normalization constant.\footnote{A word insertion penalty did not prove beneficial, but if used it
should also be scaled by \( \frac{1}{\lambda} \).
}

4. RESULTS AND ANALYSIS
4.1. Comparison to MAP Scoring
We carried out experiments on the Switchboard conversational
telephone speech corpus [4] to test the performance of lattice-
based word error minimization. The first column in Table 1
(Set I) shows the results on a set of 2427 utterances from 14
conversations that formed the development test set at the 1997
Johns Hopkins University LVCSR Workshop (WS97 dev-test).
The consensus hypothesis results in an absolute WER reduction
of 1.4% over the baseline, the standard MAP approach. To con-
firm the consistency of the improvement we ran similar experi-
ments for two more sets of lattices. Set II consists of lattices for
the same set of utterances, but obtained with different acoustic
models. Set III is based on a different set of utterances and was
generated using the same acoustic models as Set I, with a base-
line WER that is more than 4% higher than that of Set I. On all
three test sets we obtain similar and significant WER reductions
over the baseline. The algorithm to construct the hypothesis align-
ment and extract the best hypothesis is fast: on a 400 MHz Pentium-II pro-
cessor it ran in about 0.5x \text{real time on average for the Switch-
board data.}
<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Word Error Rate (%)</th>
<th>WER reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>38.5</td>
<td>-1.4</td>
</tr>
<tr>
<td>Consensus</td>
<td>37.1</td>
<td>-1.4</td>
</tr>
<tr>
<td>(\Delta )WER</td>
<td>-1.4</td>
<td>-1.1</td>
</tr>
</tbody>
</table>

Table 1. Comparing the consensus hypotheses to the baseline

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Word Error Rate (%)</th>
<th>WER reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-best (Center)</td>
<td>37.9</td>
<td>-1.1</td>
</tr>
<tr>
<td>Lattice (Consensus)</td>
<td>37.4</td>
<td>-1.1</td>
</tr>
</tbody>
</table>

Table 2. Comparison of N-best (center) and lattice-based (consensus) word error minimization

4.2. Lattices versus N-best lists: Result Analysis

We compared the lattice-based consensus hypothesis to the N-best based center hypothesis (Equation 3). The maximum number of hypotheses per utterance was 300. Table 2 shows both results on the WSJ97 dev-test data (Set I).

We also conducted two diagnostic experiments to further pinpoint where the improvement over the N-best approach was coming from. First, we computed word posteriors from the lattice using the consensus method, but then limited the choice of sentence hypothesis to those in the N-best list. The result was a 0.3% higher WER than the plain consensus hypothesis (last line in Table 2). We conclude that 0.5% of the overall 0.8% reduction comes from improved posterior estimates, while the rest can be attributed to the larger candidate set for hypothesis selection.

The second diagnostic experiment was designed to quantify the difference between the word error resulting from multiple alignment (\(WFE\)) and the standard word error based on pairwise alignment (\(WFE\)). We sampled a large number of pairs of hypotheses from the posterior distribution represented by our lattices, and compared \(WFE\) and \(WFE\). The total number of substitutions, deletions, and insertions under the two alignments differed by only 0.15 on average. This shows that the suboptimal nature of the alignment is negligible in practice, and more than justified by the computational advantages it affords.

5. RELATED AND FUTURE WORK

In essence, our approach replaces sentence-level posterior probabilities with word-level posteriors as the objective function for speech recognition, corresponding to the word-based error metric commonly used. As a result, our method is related to several algorithms based on posterior word probabilities. Conversely, our method is essentially an estimator of posterior word probabilities, and as such could benefit a number of other tasks. Here, we point out some of these related tasks.

As shown in [10], word spotting can be accomplished by estimating word posteriors from the N-best output of a large vocabulary recognizer. Based on our results, we would expect improved word spotting results when using the lattice-based posteriors obtained as described here.

A closely related problem is the estimation of word confidence measures for large vocabulary recognizers. The N-best based posterior is one of the most informative features for confidence estimation [9]; consequently, we can expect improved results with lattice-based posteriors. Conversely, work on confidence measures suggests that other recognizer features can be combined with acoustic and language model scores to yield improved posterior estimates, and therefore fewer word errors.

Finally, we note that our algorithm is similar to the ROVER algorithm for combining 1-best outputs from multiple recognizers [3], in that it combines multiple, weighted hypotheses into a single alignment for voting at the word level. We note that ROVER might give even better results if it used not just the 1-best output from various recognizers, but instead used the full confusion networks and associated posteriors from each recognizer.

6. CONCLUSION

We have developed a new method for finding the sentence in a recognition lattice that minimizes expected word error, unlike the standard MAP approach that minimizes sentence error. The core of the method is a clustering procedure that identifies mutually supporting and competing word hypotheses in a lattice, constructing a total order over all word hypotheses. Together with word posterior probabilities computed from recognizer scores, this allows an efficient extraction of the hypothesis with minimum expected number of errors. Experiments on the Switchboard corpus show that this approach results in significant WER reductions, both over the standard MAP approach and compared to a previous word error minimization technique based on N-best lists.

ACKNOWLEDGMENTS

We thank Mitch Weintraub for valuable discussions and suggestions on the word hypothesis clustering problem. Also thanks to Frederick Jelinek, Sanjeev Khudanpur, David Yarowsky, Radu Florian, Ciprian Chelba and Jun Wu for useful feedback during STIMULATE meetings and Dimitra Vergyri for providing the lattices used in these experiments. The work reported here was supported in part by NSF and DARPA under NSF grant IRI-9618874 (STIMULATE). The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the funding agencies.

REFERENCES