CORRECTIVE AND REINFORCEMENT LEARNING
FOR SPEAKER-INDEPENDENT CONTINUOUS SPEECH RECOGNITION

Kai-Fu Lee, Sanjoy Mahajan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

This paper addresses the issue of learning hidden Markov model (HMM) parameters for speaker-independent continuous speech recognition. Bahl et al. [1] introduced the corrective training algorithm for speaker-dependent isolated word recognition. Their algorithm attempted to improve the recognition accuracy on the training data. In this work, we extend this algorithm to speaker-independent continuous speech recognition. We use cross-validation to increase the effective training size. We also introduce a near-miss sentence hypothesization algorithm for continuous speech training. The combination of these two approaches resulted in over 20% error reductions both with and without grammar.

1. Introduction

At present, hidden Markov models (HMMs) constitute the predominant approach to automatic speech recognition. Most HMM-based systems use the Baum–Welch (or forward-backward) algorithm [2, 3, 4], which adjusts the parameters to approximate the maximum-likelihood estimates (MLE) of the HMM parameters. Maximum likelihood estimators have many desirable properties; however, maximum likelihood estimation has one serious flaw: it assumes that the underlying models are correct, which is not the case [5].

Bahl et al. [1] introduced the corrective training algorithm for HMMs as an alternative to the forward-backward algorithm. Whereas the forward-backward algorithm attempts to increase the probability that the models generated the training data, corrective training attempts to maximize the recognition rate on the training data. This algorithm has two components: (1) error-correction learning improves correct words and suppresses misrecognized words, and (2) reinforcement learning improves correct words and suppresses near-misses. Applied to the IBM speaker-dependent, isolated-word office correspondence task, this algorithm reduced the error rate by 16%.

In this study, we extend the corrective and reinforcement learning algorithm to speaker-independent, continuous speech recognition. Speaker independence presents some problems, because corrections appropriate for one speaker may be inappropriate for another. However, with a speaker-independent task, it is possible to collect and use a large training set. We also propose the use of cross-validation to further increase the effective training data size.

Extension to continuous speech is more problematic. With isolated-word input, both error-correcting and reinforcement training are relatively straightforward, since all errors are simple substitutions. Bahl, et al. [1] determined both mis-recognized words and near-misses by matching the utterance against the entire vocabulary. However, with continuous speech, the errors include insertions, deletions, and phrase-substitutions. These problems make reinforcement learning difficult. We propose an algorithm that hypothesizes near-miss sentences for any given sentence. First, a dynamic programming algorithm produces an ordered list of likely phrase substitutions. Then, this list is used to hypothesize the near-miss sentences used in reinforcement learning.

We applied this modified corrective training procedure to the 997-word DARPA continuous resource management task, using the speaker-independent database. We report error rate reductions of more than 20% over the standard MLE-trained SPHINX System [6].

In this paper, we first give a brief overview of hidden Markov models and maximum likelihood training in Section 2. We present our modified corrective training algorithm in Section 3, and our results in Section 4. Section 5 finishes with a brief conclusion.

2. HMM and Maximum Likelihood Training

A hidden Markov model (HMM) can be characterized by:

- \( \{s\} \): A set of states including an initial state \( S_i \) and a final state \( S_f \).
- \( \{a_{ij}\} \): A set of transitions where \( a_{ij} \) is the probability of taking a transition from state \( i \) to state \( j \).
- \( \{b_j(k)\} \): The output probability matrix: the probability of emitting symbol \( k \) when taking a transition from state \( i \) to state \( j \).

The forward-backward algorithm [4] is used to estimate \( a \) and \( b \) iteratively. For each iteration, the estimates from the previous iteration are used to compute a set of counts, where a count, \( c_{ij}(k) \), represents the frequency that the symbol \( k \) is observed, and that the transition from \( i \) to \( j \) is taken. Baum [2] showed that \( a \) and \( b \) can be re-estimated from these counts to increase the likelihood of generating the training data, unless a local maximum has been reached. Although the forward-backward algorithm guarantees only a local maximum, it efficiently produces an approximation to the maximum-likelihood estimates (MLE) of the HMM parameters.

In spite of the advantages of maximum-likelihood estimation, it suffers a serious problem — it assumes that the underlying models are correct [5]. However, HMMs are poor models of real speech, due mainly to the Markov indepen-
3. Corrective and Reinforcement Training

In view of the dependence of MLE on the problematic assumptions of HMMs, Bahl, et al. [1] proposed the IBM corrective training algorithm for speaker-dependent isolated-word recognition. This algorithm attempts to tune the models to minimize recognition errors. This criterion has a definite practical appeal, since error rate, not sentence likelihood, is the bottom line for speech recognition.

The basic algorithm for corrective training is given below. We only provide a sketch here. A detailed description will be given later.

1. Train HMM parameters using MLE.
2. For each correct training utterance*:
   1. Generate a set of confusable utterance.
   2. Increase the likelihood of the correct utterance.
   3. Decrease the likelihood of the confusable utterance.
3. Iterate step 2 over entire training set.

The two central problems of this algorithm are: (1) how to generate a set of confusable utterances (sentences), and (2) how to increase and decrease likelihoods of the sentences. In the next two sections, we will address these two problems.

3.1. Confusable Sentence Generation

A confusable sentence is either a sentence that receives a higher score than the correct sentence, or one that has a lower score that is within a threshold of the correct sentence. We call the former a misrecognition, and the latter a near-miss.

For an isolated-word task, misrecognitions and near-misses can be simply generated by matching the input word against all words in the vocabulary. This was, in fact, the algorithm used by IBM [1]. In order to deal with our speaker-independent continuous task, we modified this algorithm.

The next two sections will described how we generated misrecognitions and near-misses.

3.1.1. Misrecognition Generation

It is natural to generate misrecognitions on a spoken sentence by running the recognition algorithm on the sentence. Our only departure from the IBM algorithm was that instead of using the same data to train the initial HMM probabilities and to perform corrective training, we use cross-validation. First, the training data is divided into two partitions, and HMMs are trained on each partition. Then, HMMs trained from one partition are used to recognize the sentences from the other. Not only will we obtain many more errors this way, but these errors will also be more realistic. We then use these errors to correct the models trained on the entire set.

3.1.2. Near-Miss Generation

Bahl, et al. [1] found that reinforcement (near-miss) learning aided the convergence of corrective training considerably. For isolated-word tasks, near-miss training is conceptually simple, since the only errors are simple word-for-word substitutions (such as for → fur). To generate near-misses, Bahl, et al. used a list of near-miss words produced by a fast-match algorithm [7].

Such an approach is unsuitable for continuous speech, where we need to produce near-miss sentences given a correct sentence. This information is unavailable from a continuous speech recognizer due to pruning. One possibility is to use anti-supervision [8]. However, since our algorithms are already very time consuming, we chose an efficient algorithm that decomposes this problem as follows:

1. Produce a long list of near-miss phrase substitutions, where each phrase may have zero to several words.
2. Use this list to hypothesize near-miss sentences by substituting one or more of the near-miss phrases with their respective replacements.

The first step is to generate a list of likely system errors. We model these errors using near-miss phrase substitutions. A near-miss phrase substitution is a pair of phrases, where each phrase may have zero to three words. We generate these confusable phrases as follows. First, we use cross-validation recognition to obtain realistic misrecognized sentences for each training sentence. Then, to find plausible phrase errors, each misrecognized sentence is matched against the corresponding correct one by a dynamic programming (DP) algorithm. We could then define near-miss phrases as phrase alignments that have reasonable costs. Details of this algorithm can be found in [9, 10].

We processed 4150 pairs of correct and recognized sentences using the above algorithm, obtaining a list of 13000 phrase substitutions. This list of near-miss phrase substitutions are then used by the sentence hypothesizer to heuristically hypothesizes likely near-miss sentences. For each correct sentence, we randomly replaced phrases in the sentence with the corresponding confusable phrase. The likelihood of a random substitution is proportional to the goodness of the match between the confusable phrases. We used an average of 6 near-miss sentences per original sentence.

3.2. Changing Likelihood of Sentences

Given a set of correct and confusable sentences, how do we modify the HMM parameters to increase the likelihood of the former and decrease the likelihood of the latter? Our algorithm for this purpose is identical to that of IBM's. Basi-

---

*An utterance is a word in the IBM algorithm, and a sentence in ours.
only the parameters corresponding to that word will be corrected; everything else will be cancelled out. The exact algorithm for incrementing and decrementing counts will be described in the next section.

3.3. Final Algorithm Description

The full corrective training and reinforcement algorithm is shown below:

1. Partition the training sentences into two equal sections for cross-validation and train a set of models on each half.
2. Recognize each half with the models trained on the other half (cross-validation).
3. Perform the DP algorithm on misrecognized sentences to obtain a list of near-miss phrase substitutions.
4. Hypothesize a list of near-miss sentences from the list of near-miss phrase substitutions.
5. Generate the list of confusable sentences \( \omega_n \) by combining the misrecognitions from Step 2 and the near-misses from Step 4.
6. Run the forward-backward algorithm on each training sentence \( s \), using the model for the correct sentence \( w \) to obtain the counts \( c_{ij}^{\text{new}} \). Do the same with each \( \omega_n \) to obtain the counts \( c_{ij}^{\text{old}} \). Then, replace the each original count \( c_{ij} \) with \( c_{ij} + \gamma (c_{ij}^{\text{old}} - c_{ij}^{\text{new}}) \). \( \gamma \) is an adjustment factor:
   - For misrecognitions, it is set to \( \beta \).
   - For near misses, it decreases linearly from \( \beta \) to 0 as the difference between \( \log P(u|w) \) and \( \log P(u|\omega_n) \) decreases from 0 to \( -\delta \).
7. Smooth the resulting models with the MLE parameters, using a weighted average.**
8. Go to step 3, until a sufficient number of iterations have been run.

4. Results and Discussion

In order to compare results with corrective training, and to establish an initial parameter set, we must first generate a set of MLE HMMs. So we first used the forward-backward algorithm to train the SPHINX system. SPHINX uses discrete hidden Markov models with three codebooks quantized from:

- (1) 12 LPC cepstrum coefficients, (2) 12 differenced LPC cepstrum coefficients, and (3) power and differenced power.

More details about SPHINX can be found in [11, 12, 13, 14].

SPHINX was applied to the 997-word TI Resource Manage-

** During training, we found that corrective training would make some probabilities too small for test data. As a result, a few sentences could not be recognized. To remedy this problem, we use a large \( \beta \) and then smooth the trained parameters with the MLE parameters.

Table 1: Error rates for corrective and reinforcement training without grammar.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Errors on Training Set</th>
<th>Errors on Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (MLE)</td>
<td>6122 (18.8%)</td>
<td>375 (29.4%)</td>
</tr>
<tr>
<td>1</td>
<td>4008 (10.9%)</td>
<td>316 (24.7%)</td>
</tr>
<tr>
<td>2</td>
<td>2218 (6.8%)</td>
<td>302 (23.6%)</td>
</tr>
<tr>
<td>3</td>
<td>1896 (5.2%)</td>
<td>299 (23.4%)</td>
</tr>
</tbody>
</table>

Table 2: Error rates for corrective and reinforcement training with a word-pair grammar.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Errors on Training Set</th>
<th>Errors on Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (MLE)</td>
<td>1799 (4.9%)</td>
<td>81 (6.3%)</td>
</tr>
<tr>
<td>1</td>
<td>809 (2.2%)</td>
<td>68 (5.3%)</td>
</tr>
<tr>
<td>2</td>
<td>689 (1.9%)</td>
<td>64 (5.0%)</td>
</tr>
<tr>
<td>3</td>
<td>650 (1.8%)</td>
<td>62 (4.9%)</td>
</tr>
</tbody>
</table>

IBM reported error rate reductions of 16% and 88% on test data and training data respectively [1]. With more training, we report 20% and 72% without grammar; 23% and 63% with grammar. Thus, we have not only demonstrated the extensibility of the corrective training algorithm to speaker-independent continuous speech recognition, but also narrowed the gap between training and testing results through the use of more training and cross validation.

A more recent experiment that combined between-word
coarticulation modeling and corrective training has led to error rates of 18.1% and 3.8% for no grammar and the word-pair grammar, respectively. More detailed results can be found in [16, 13].

5. Conclusion

Hidden Markov models with maximum-likelihood estimation constitute the predominant approach to automatic speech recognition today. However, maximum likelihood produces inferior results when the underlying models are incorrect, which HMM's obviously are as models of real speech. For this reason, the IBM corrective training algorithm produced good results when applied to the IBM isolated-word, speaker-dependent, office-correspondence task. In this paper, we extended this algorithm to continuous, speaker-independent speech recognition.

In order to increase the effective training size, we used cross-validation. In order to extend the algorithm to continuous speech, we introduced an algorithm that hypothesized near-miss sentences. This algorithm has two components: (1) a near-miss phrase substitution generator that used a dynamic programming algorithm to produce a long list of possible phrase substitutions, and (2) a non-deterministic near-miss sentence hypothesizer that used the phrase substitution list to hypothesize possible near-miss sentences from a correct sentence. These enhancements, aided by the use of a large training database, led to error rate reductions of 20.3% without grammar, and 23.4% with a word-pair grammar.

These results clearly demonstrated that the corrective training algorithm is applicable to speaker-independent, continuous speech recognition. The applicability of this algorithm to continuous speech is demonstrated through the use of novel algorithms for near-miss sentence hypothesization. The applicability of this algorithm to speaker-independent recognition is also important, because much more training data can be collected in speaker-independent mode. In the future, we hope to collect and use larger multi-speaker databases, investigate more efficient training algorithms, as well as examine other techniques for competitive learning.

Acknowledgments

The authors wish to thank the CMU Speech Group for their support and contributions. This research was sponsored by DARPA Contract N00039-85-C-0163.

References