

CORRECTIVE TRAINING FOR SPEAKER ADAPTATION

Xiuyang Yu and Wayne Ward
Center for Spoken Language Understanding
University of Colorado, Boulder, Colorado

ABSTRACT

This paper reports results on an experiment to use corrective training techniques for rapid acoustic speaker adaptation in a semi-continuous speech recognition system. Decoder output is used to adjust HMM acoustic models to improve discrimination between correct words and near misses. Twenty sentences are used as an adaptation set. A speech recognizer is run on each utterance to generate a word lattice. The lattice is pruned relative to the correct path. The forward-backward algorithm is used to align each path in the lattice against the speech input and compute observation counts. For each input frame, counts in correct models are adjusted upward, and counts in incorrect models are adjusted downward. The adjusted counts are normalized to generate new observation probabilities for the models. The parameters being adjusted are the mixture weights for the semi-continuous HMMs. The technique reduced word error for a test subject by 37% relative.

INTRODUCTION

Speech recognition systems based on Hidden Markov Models typically experience significant performance degradation when a speaker is not represented well in the data that the system was trained on. Rapid speaker adaptation techniques can be very effective in improving performance for a novel speaker. These techniques use a small number of sentences (20-40), whose transcript is known, to quickly adapt HMM acoustic models to a new speaker. Such techniques are important, since in order for longer term adaptation to be able to work, the system must function well enough for the user to be productive. These techniques attempt to correct very poor models quickly to get the system to a usable point for the user. This paper reports results on an experiment to use corrective training techniques to rapidly adapt HMM acoustic models to a poorly recognized speaker.

CORRECTIVE TRAINING

Corrective training was introduced for speaker-dependent isolated word recognition by [1] and extended to speaker-independent continuous speech recognition by [2]. Hidden Markov Models are normally trained according to a maximum likelihood criterion; parameters are adjusted to maximize the probability of the training set. This process does not directly minimize the word errors, it does this indirectly by attempting to assign a high probability to the correct utterance. It makes no representation of what probability is assigned to near misses. Corrective training seeks to directly minimize the number of word errors by adjusting parameters so as to improve discrimination between correct words and near misses. The general process is:

1. Generate a set of near misses, words which are confusable with the correct words.
2. Align the correct words against the input.
3. Align a near miss against the input.
4. Modify model such that correct words are more likely and incorrect ones less likely.
5. Repeat steps 2-3 for other near misses.

Speech recognizers are used to generate the near misses. In [1] and [2], a speech recognizer was used to generate an n-best list for an isolated word or sentence. This list was used as the set of near misses. Both the correct utterance and near miss are aligned against the input. Model parameters are then adjusted to make the correct words more likely and the incorrect ones less likely.

SPHINX-II OBSERVATION ESTIMATES

In order to describe how corrective training is used to adapt our model, it is first necessary to describe the basic model. Our experimental system uses the Carnegie Mellon University Sphinx-II system for speech recognition [3],[4],[5]. Sphinx-II uses semi-continuous Hidden Markov Models [3] to model context dependent phones. Like continuous HMM systems, semi-continuous systems use a weighted sum of points on Gaussian Probability Density Functions to estimate observation probabilities. The difference is that, while continuous systems estimate a set of distributions for each HMM state in the system, semi-

continuous systems estimate a fixed set of distributions and each state combines them with state-specific weightings. Sphinx-II estimates a set of 256 Gaussian distributions for each of 4 feature streams (cep, Δcep , $\Delta\Delta\text{cep}$, power). Each HMM state in the system has a vector of weights for the distributions, which are estimated from the same training data used to estimate the means and co-variances for the distributions. In order to estimate an observation probability, the observation vector is evaluated against each of the Gaussian distributions. The 4 best scoring distributions are used to compute the estimate. The weight for each distribution is looked up in the HMM state, and the estimate of the observation probability is the weighted sum of the scores.

In order to reduce the number of parameters to be trained, the weighting vectors are tied across similar HMM states. These tied weighting vectors are referred to as senones [4]. In order to train the HMM parameters initially, the forward-backward algorithm is used to align HMM states against the vector sequences in the training data. Observation counts are accumulated for the frequency with which each senone is aligned with each distribution in the training data. These counts are used to estimate the means and co-variances of the Gaussians and also to estimate the mixture weights for the senones.

It is the senones, the weighting vectors for combining distributions that we modified with corrective training.

GENERATING NEAR MISSES

Like many state-of-the-art speech recognition systems, Sphinx-II is a two pass decoder. In the first pass, a viterbi search with a trigram language model is used to find the best path. Hypotheses explored during the Viterbi search are saved and used to create a graph of word sequence hypotheses. In the second pass, an A* search is used, perhaps with additional language models, to search the word graph for a new best path. The word graph generated by the first pass is pruned to provide our set of near misses.

First, the correct path through the graph is found using the transcript. If no correct path existed, one is generated by forced alignment. The lattice is then pruned relative to the correct path. All words better than the correct path, and all words within a threshold lower than the correct ones were kept.

CORRECTIVE TRAINING PROCEDURE

Our corrective training procedure uses the pruned word graphs as a source of near misses. The forward-backward algorithm is used to align each path in a lattice against the speech input and compute the observation counts for the models. For each frame of input, the counts for the

observation in the correct models are adjusted upward by adding a factor, and the counts in incorrect models are adjusted downward by a factor. The factor size depends on the difference between the score for the correct model and the incorrect one.

- (1) For each utterance, run the forward-backward algorithm on the correct word sequence (transcript) to align it against the input. This defines the “correct” labels for each frame of input.
- (2) Process each word graph to find near miss phrases. These are found by comparing word sequences in the graph to the correct path. A sequence of one or more incorrect words is a near miss.
- (3) For each near miss:
 - a) Use the forward-backward algorithm to align the near miss phrase against the corresponding frames of input and accumulate the observation counts $i_{s,d}$, the number of times each distribution d is associated with each senone s for incorrect alignments. An alignment is not considered incorrect just because it is part of an incorrect path. If the senone for the frame has the same base phone as a senone assigned to the frame in the correct alignment, the senone is not considered incorrect and the counts are not adjusted.
 - b) Use the forward-backward algorithm to align the correct word sequence against the corresponding frames of input and accumulate the observation counts $c_{s,d}$, the number of times each distribution d is associated with each senone s for correct alignments.
 - c) Use these counts to compute adjustments to the weight vectors:

$$\text{adj}_{s,d} = \gamma (c_{s,d} - i_{s,d})$$

$$\gamma = 1 - [\log P(\text{correct}) - \log P(\text{nearmiss})] / \beta$$

The γ parameter controls the step size of the adjustment. $\log P(\text{nearmiss})$ is the average log probability over the frames of the near miss. It is the log of the probability produced by the forward algorithm for the nearmiss alignment divided by the number of frames. $\log P(\text{correct})$ is the average log probability over the corresponding frames of the correct alignment. It is the log probability produced by the forward algorithm for the corresponding correct alignment divided by the number of frames. β is the pruning threshold and is used to scale the difference in average log score. For this experiment a value of 5.0 was used for β . Correct alignments cause positive adjustments and incorrect alignments produce negative adjustments.

(4) Normalize the adjustments:

$$\text{adj}_{s,d} = \text{adj}_{s,d} / \sum \text{abs}(\text{adj}_{s,i}) \quad i=1, \dots, 256$$

(5) Interpolate adjustments with original weights :

For each senone s

For each distribution d

$$W_{s,d} = W_{s,d} + \tau \text{adj}_{s,d}$$

τ mixing parameter set from development data

(6) Normalize the model mixture weights.

For each senone s, normalize the weight for each distribution by the sum across distributions:

$$W_{s,d} = W_{s,d} / \sum W_{s,i} \quad i=1, \dots, 256$$

This process is applied iteratively with models no longer being adjusted as they fall below a threshold from the correct one. This process tends to make the correct paths “rise to the top” of the graphs.

EVALUATION

The procedure was tested in a 20,000 word medical transcription system. The Carnegie Mellon University Sphinx-II system was the speech recognition system used. The system was a prototype being field tested at a clinic for internal medicine. The speakers were physicians dictating patient records. One of the physicians in the clinic had an exceptionally high word error rate when using the system. Two hundred twenty nine utterances were recorded of this physician using the system. These were divided into an adaptation set of 20 utterances (191 words), and a test set of 209 utterances (1951 words). The test speaker had not been seen in the data the system was trained on. New models were created by running the corrective training procedure on the 20 utterances in the adaptation set. The recognizer was then run on the 209 utterance test set with both the old models and the new models, and their word error rates measured. The error rates for both models were also measured on the adaptation set. Results are shown in table 1.

	Adapt set	Test set
Old models	19.4	19.1
Adapted models	2.1	12.0
% reduction	89.2	37.2

Table 1

Reduction in Error Rate with Corrective Training

The error rate on the test set for the original models was 19.1%. This was reduced to 12.0% by corrective training, which is a reduction of 37.2% in error. The original models have a similar error rate on both the test set and the adaptation set, which would be expected if the adaptation set is representative of the test set. The adapted models had a 2.1% error rate on the adaptation set. With additional iterations on the data, the error on the training set could be

made lower, but would run the risk of over-training to the data and not transferring to test sets. A development test set is used to determine the number of iterations.

Table 2 shows how the error rate varies as a function of the mixing parameter t. This parameter is an interpolation weight that controls how much the models are modified by the adjustments.

τ	Adapt set	Test set
.1	2.6	13.3
.2	2.1	12.0
.3	2.6	12.3
.4	2.6	12.6

Table 2

Error rate as a function of mixing parameter τ

CONCLUSION

Corrective training techniques do help adapt acoustic models to a speaker, given a small amount of transcribed acoustic data from the speaker. We have not yet adapted on larger adaptation sets and therefore do not yet know the optimal set size for adaptation. We have also not yet gathered data from additional speakers for evaluation. We will be evaluating the technique on additional speakers in the near future.

While adapting acoustic models to a speaker certainly has benefits, it also has limitations. While error on the adaptation set should be very low, it probably should not be zero. One reason for this is that dictionary pronunciations will not exactly match the phone strings in fluent speech. If the transcripts generate a sub-optimal “correct” phone string to align for adaptation, the wrong models will be tuned. The model will work in the current utterance, but will not transfer in the correct context. We are working to jointly adapt pronunciation models and acoustic models with the types of techniques described.

REFERENCES

1. Bahl, L.R., Brown, P.F., De Souza, P.V., Mercer, R.L.: "A New Algorithm for the Estimation of Hidden Markov Model Parameters". IEEE International Conference on Acoustics, Speech, and Signal Processing. April, 1988.
2. Lee, K-F. and Mahajan, S., "Corrective and Reinforcement Learning for Speaker-Independent Continuous Speech Recognition", CMU Technical Publication CMU-CS-89-100, 1989.

3. Huang, X.D. and Jack, M.A., "Semi-continuous hidden Markov models for speech recognition," *Computer Speech and Language* 3 pp. 239-251, 1988.
4. Hwang, M., Shared-Distribution Hidden Markov Models for Speech Recognition , *IEEE Trans. on SAP*, pp. 414-420, October,1993.
5. Ravishankar, M.K., "Efficient Algorithms for Speech Recognition," Unpublished PhD Dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA May 1996.