Discriminative Graph Training for Ultra-fast Low-footprint Speech Indexing
Upendra Chaudhari, Hong-Kwang Jeff Kuo, and Brian Kingsbury

IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
{uvc, hkuo, bedk}@us.ibm.com

Abstract
We study low complexity models for audio search. The indexing and retrieval system consists of Automatic Speech Recognition (ASR), phone expansion, N-gram indexing and approximate match. In particular, the ASR system can vary tremendously in complexity ranging from a simple speaker-independent system to a fully speaker-adapted system. In this paper, we focus on a speaker-independent system with a small number of Gaussians. Such a system, with ASR followed by phone expansion, provides a good balance between speed and accuracy, allowing for the processing of large volumes of data and better retrieval performance than systems relying solely on phone recognition. Here we describe the use of discriminative training of a finite-state decoding graph for improving system accuracy while preserving speed of operation.

Index Terms: speech indexing, discriminative training, finite-state decoding graph.

1. Introduction
Contemporary audio analytics deal with an immense and ever increasing volume of data. Given constraints on available computing power, it is often helpful to adopt a multi-pass approach to audio analysis in which large volumes of data are screened using a lightweight system that identifies segments which should receive more intensive processing. The optimization of this approach is both scientifically interesting and practically important. In this work, we focus on the initial, lightweight analysis, and show how using a discriminatively trained finite-state decoding graph in the ASR component of an audio indexing system improves performance with minimal impact on computational load.

Our audio indexing system expands word transcripts into phone transcripts before building the search index. While pure phone decoding could be used, we have found that the word based approach, which includes a language model, produces better retrieval results, giving information on where word boundaries may appear. The expanded phone sequences are parsed into N-grams and indexed. For retrieval we use a constrained match that models high order confusions to account for the system dependent errors that can occur [1, 2].

The quality of the initial ASR has a significant effect on subsequent retrieval performance. While there are many ways to improve the system, most involve increasing either the size of the models or the decoding complexity, both of which result in a slower approach. In contrast, this paper shows how discriminative training, with the Minimum Classification Error (MCE) criterion, can be used to adjust the parameters of the finite-state decoding graph, without changing the size or topology of the graph, to provide better accuracy with nearly the same computational costs during audio analysis.

2. Audio Indexing
2.1. Building an index
As audio is ingested, the automatic speech recognition (ASR) system produces a word transcript, which is then converted into a phone transcript using the pronunciation dictionary. In the ASR processing, the audio is segmented into sections that are on average about 15 to 20 seconds long. In the indexing scheme, each such segment is considered to be a unique document, $D_k$, whose length is defined as the number of resulting phones. An inverted index is created with a granularity corresponding to the positions of phones. $N = 5$ is chosen as the length of the phone sequences in the index.

$$D_k \rightarrow \{h_{k,1}, h_{k,2}, h_{k,3}, \ldots\}. \quad (1)$$

For each document $D_k$, the $N$-gram at each position, $h_{k,j} = \{p_{k,j}, p_{k,j+1}, p_{k,j+2}, p_{k,j+3}, p_{k,j+4}\}$, is extracted. At the end, where full $N$-grams are not available, the sub-$N$-grams are used.

$$I = \bigcup_k D_k \quad (2)$$

is the set of all unique $N$-grams that occur. The inverted index associates to each element of $I$ the list of (document, position) pairs at which that element occurs. Note that $h \in I$ represents an observed $N$-gram and is treated as a hypothesis vector. The query is considered to be the reference, or the exact sequence of phones that is sought.

2.2. Confusion matrix
Phone confusions are represented as $P(p_i|p_j)$ which is the probability that $p_i$ is the true phone when $p_j$ is observed. Our approach also incorporates high order confusions, which are denoted by $P(p_i|p_j|p_{m})$ or $P(p_i|p_j|p_{m, n})$, which denote the probability of trigram or bigram confusions.

The distributions $P(p_j \cdots p_j) \cdots$ are derived from confusion matrices, whose parameters are estimated from held-out data which is decoded to produce phone level alignments. Phone unigram, bigram and trigram confusions are estimated.

2.2.1. Generating scores
The query is an arbitrary sequence of words. To perform search, each word in the query is expanded into one or more phone sequences via lookup in the ASR pronunciation lexicon and via automatic baseform generation [3]. This allows us to handle out-of-vocabulary terms and obtain alternate pronunciations for query terms. Let $Q_k$ denote a single phone sequence for a query term.

$$Q_k \rightarrow \{r_{k,1}, r_{k,2}, r_{k,3}, \ldots\}. \quad (3)$$
where $r_{k,j} = \{p_{k,j}, p_{k,j+1}, p_{k,j+2}, p_{k,j+3}, p_{k,j+4}\}$ is the phone 5-gram at position $j$ in $Q_k$. Given $r_k$ and $h_j \in I$ a match score $S(r_k, h_j)$ that incorporates high order confusions is defined [1]. Then, given a document $D$ and a query $N$-gram $r$, the best matching index element corresponding to the document is given by

$$h^*(D, r) = \arg\max_{h \in \mathcal{I} / D} S(r, h).$$

Letting $N_Q$ be the number of $N$-grams extracted from the query, the score is

$$\frac{1}{N_Q} \sum_{r \in Q} S(r, h^*(D, r)).$$

### 3. Discriminative Training

In this section, we describe how the transition weights of an integrated finite-state decoding graph are adjusted discriminatively to improve the score separation of the correct word sequence from the best competing word sequence hypothesis, using the Minimum Classification Error (MCE) criterion [4, 5]. The formulation is similar to [6]; one difference is that we use context-dependent acoustic models, so the sequences are context-dependent state sequences. Although already presented earlier [7], we include the following information for completeness.

The integrated decoding graph (call it $G$) is essentially a classical Hidden Markov Model (HMM) with two sets of parameters: the acoustic model $\Lambda$, consisting of the Gaussian densities in the HMM states, and the transition weights $\Gamma$, which specify the costs of transitions between the HMM states. The decoding graph can be constructed according to procedures described in [8, 9], which involve FST composition and optimization of the hidden Markov model, pronunciation model, and decision trees of the context-dependent HMM. The language model is a back-off $n$-gram language model, trained using the conventional maximum likelihood criterion and appropriate smoothing such as modified Kneser-Ney. The pronunciation probabilities may be arbitrarily set to uniform or may be based on estimates from aligning pronunciation variants ("lexemes") to the speech training data.

Given an acoustic observation sequence $X = x_1, x_2, \ldots, x_T$ representing the speech signal and a word sequence $W$, the conditional likelihood of $X$ is approximated as the score of the best path $S = S_0, S_1, S_2, \ldots, S_T$ through $G$ for input $X$ and output $W$. We define a discriminant function to be this score, which is a weighted combination of the sum of acoustic log likelihoods $a(X, W, S; \Lambda)$ and transition weights $b(W, S; \Gamma)$:

$$g(X, W, S; \Lambda, \Gamma) = \alpha \cdot a(X, W, S; \Lambda) + b(W, S; \Gamma),$$

where $\alpha$ is the acoustic model weight. Note that the path $S$ is a sequence of hidden states that actually specify a lexeme (specific pronunciation variant of a word) sequence as well as a leaf (context-dependent HMM state) sequence. The sum of the transition weights of the path includes the sum of the language and pronunciation model log probabilities of the associated lexeme sequence (plus other parameters such as word or silence insertion penalties, etc.).

A common strategy for a speech recognizer is to search for the word sequence $W_1$ with the largest value for this function:

$$W_1 = \arg\max_{W, S} g(X, W, S; \Lambda, \Gamma).$$

Let $W_0$ be the known correct word sequence. The misclassification function is defined to be the difference between the discriminant function and the anti-discriminant function, which is normally an $L_p$ norm weighted combination of the $N$-best competing hypotheses [10]. For simplicity, we follow [6] and just consider the decoded (single best) hypothesis $W_1$. Let the misclassification function be

$$d(X; \Lambda, \Gamma) = -g(X, W_0, S_0; \Lambda, \Gamma) + g(X, W_1, S_1; \Lambda, \Gamma).$$

When this misclassification function is strictly positive, a sentence recognition error has been made. To formulate an error function appropriate for gradient descent optimization, a smooth, differentiable function ranging from 0 to 1 such as the sigmoid function is chosen to be the class loss function for a specific utterance $X_i$:

$$l_i(X_i) = l(d(X_i)) = \frac{1}{1 + \exp(-\gamma d(X_i) + \theta)}.$$

where $\gamma$ and $\theta$ are constants which control the shape and the shift of the sigmoid function, respectively. Our objective is to minimize the loss function over all utterances in the training corpus:

$$l(X) = \sum \sum l_i(X_i).$$

The transition-weight parameters can be adjusted iteratively (with step size $\epsilon$) to minimize the objective function using the following update equation:

$$\Gamma_{t+1} = \Gamma_t - \epsilon \nabla l_i(X_i).$$

The gradient of the loss function is

$$\nabla l_i(X_i; \Lambda_1, \Gamma_1) = \sum \sum \frac{\partial l_i}{\partial d_i} \frac{\partial d_i}{\partial d(X_i; \Lambda, \Gamma)},$$

where the first term is the slope associated with the sigmoid class-loss function and is given by:

$$\frac{\partial l_i}{\partial d_i} = \gamma l(d_i) (1 - l(d_i)).$$

If we regard $\Gamma$ as a vector of transition weights $s_j$, to compute $\frac{\partial l(X_i; \Lambda, \Gamma)}{\partial s_j}$, we can take the partial derivatives with respect to each $s_j$. Using the definition of $d$ in Equation 7 and after working out the mathematics, we get:

$$\frac{\partial d(X_i; \Lambda, \Gamma)}{\partial s_j} = -I(W_0, s_j) + I(W_1, s_j),$$

where $I(W_0, s_j)$ denotes the number of times the transition $S_j$ is taken in the best aligned path of $X_i$ to the word sequence $W$. For each utterance in the training data, the algorithm is counting the transitions for the correct string and the decoded hypothesis. Transitions for the correct string increase the corresponding transition weights, while those for the decoded string decrease the weights. The amount of increase or decrease is proportional to the step size $\epsilon$, the value of the slope of the sigmoid function and the difference in the number of times the transition appears. The slope of the sigmoid function is close to 0 for very large positive $d$, so little adjustment is made for a sentence for which the total score of the correct string is much worse than the score of the competing string. This decreases the effect of outliers, for example of utterances whose transcripts are erroneous. Notice that the only dependence on the acoustic
scores (more specifically, the difference in the total path scores) in the equations is in the slope $\frac{\partial l_i}{\partial d_i}$, which determines how much influence a particular training sample has in updating the parameters.

With gradient descent optimization, there is a choice of batch mode, which collects the statistics over all the training data before making an update to the model, or online mode, where the model is updated after processing each training sample and typically the sample order is randomized. Although online mode may result in faster convergence, batch mode has the advantage of allowing for parallelism in collecting statistics of the training data. In this paper, we use batch mode training.

4. Experimental Setup

The automatic speech recognition system used for the audio indexing experiments is a speaker-independent recognizer using PLP-derived features and a quinphone acoustic model with approximately 1200 context dependent states and 30000 Gaussians. The acoustic model is trained using maximum likelihood on 430 hours of audio from the 1996 English Broadcast News Speech corpus (LDC97S44), the 1997 English Broadcast News Speech corpus (LDC98S71), and the TDT4 Multilingual Broadcast News Speech corpus (LDC2005S11). To obtain training transcripts for the TDT4 audio, we used a lightly supervised training method in which we decoded the audio with a biased language model and used the resulting transcriptions as references [11].

The language model used for decoding is a unigram model with 84087 words trained on a collection of 335M words from the following data sources: 1996 CSR Hub4 Language Model data (LDC98T31), EARS BN03 closed captions, GALE Phase 2 Distillation GNG Evaluation Supplemental Multilingual data (LDC2007E02), Hub4 acoustic model training transcripts (LDC97T22 and LDC98T28), TDT4 closed captions (LDC2005T16), TDT4 newswire (LDC2005T16), GALE Broadcast Conversations (LDC2005E82, LDC2006E33, LDC2006E84, LDC2006E91, LDC2007E05, and LDC2007E45), and GALE Broadcast News (same catalog numbers as GALE Broadcast Conversation). The pronunciation lexicon has 1.08 pronunciation variants per word, on average.

The experiments are carried out on the English Hub4 broadcast news Eval97 and Eval98 data (referred to as “h4e” in the plots). A total of 6 hours of data is segmented into 1484 segments. Retrieval corresponds to finding the segments (or documents) that contain a given query. A total of 6116 single word queries were defined. The search results are reported as average precision (p) and recall (r) per query (in %) in the figures.

5. Results

The speed and accuracy of the decoding are controlled using two forms of pruning. The first is the standard likelihood-based beam pruning that is used in many Viterbi decoders. The second is a form of Gaussian shortlisting in which the Gaussians in the acoustic model are clustered into 1024 clusters, each of which is represented by a single Gaussian. When the decoder gets a new observation vector, it computes the likelihood of the observation under all 1024 cluster models and then ranks the clusters by likelihood. Observation likelihoods are then computed only for those mixture components belonging to the top maxL1 clusters; for components outside this set a default, low likelihood is used. To illustrate the trade-offs in speed vs. accuracy that can be achieved by varying the two pruning parameters, we sweep through different values for the parameters and measure decoding accuracy, reported as word error rate (WER), and decoding speed, reported as times faster than real time (xRT). For example, a system that operates at 50xRT will require one minute of time (measured as elapsed time) to process 50 minutes of speech.

Figure 1 presents data from 100 ASR trials run on a grid of values for maxL1 and the pruning beam, showing both individual measurements and the convex hull of the measurements for both the baseline system and one with a discriminatively trained decoding graph. The key finding is that the discriminatively trained system dominates the baseline system: for a given speed, the discriminatively trained system always has a better word error rate.

For example, at about 30xRT, the best WER for the baseline is about 47.5% while the best WER for the discriminatively trained system is about 45%, an absolute improvement of 2.5%. Considering a fixed level of performance, say a WER of 47.5%, Figure 1 shows that using the discriminatively trained graph improves the speed from 30xRT to about 40xRT. Thus, while maintaining the same level of accuracy, the speed-up allows one to process 33% more speech data in a fixed amount of time.

It is interesting to note that, for a given setting of the maxL1 and pruning beam parameters, the discriminatively trained system is usually slightly slower than the baseline system. This appears to happen because more competing paths are retained by the beam pruning with the discriminatively trained decoding graph. To better understand this effect, we performed a comparison of the two graphs. For each state in the graphs (recall that they have the same topology) having multiple exit arcs, we measured the difference (gap) between the weight on the best-scoring arc and the weight on the second best-scoring arc. We then looked at how this gap was changed by the discriminative training. On average, we find that, for states having more than one exit arc, discriminative training decreases the gap between the best exit arc and the next best exit arc by 0.00059, where the arc scores are log (base e) probabilities. Our graphs had 68488 states with multiple exit arcs. Of those, the discriminative training decreased the gap for 27335 states, increased the gap for
25826 states, and left the gap unchanged for 15327 states. Thus, although the MCE criterion increases the total score separation of the correct word sequence from the best competing word sequence hypothesis, at the level of the individual state, it (on average) decreases the separation between competing paths. The crucial distinctions to remember here are that the MCE training includes acoustic scores, while our analysis does not, and that the MCE training considers scores of entire paths, while our analysis considers only arcs leaving individual states. On the whole, however, for a given set of pruning parameters, the discriminatively trained graph provides a significant accuracy improvement with minor loss in decoding speed.

Performance on an audio search task for a subset of the points is represented in Figure 2. Again, the discriminative system is compared against the baseline and for the same decoding points is represented in Figure 2. Again, the discriminative system is compared against the baseline and for the same decoding speed, the recall is higher for the discriminative system.

Figure 2: Recall vs. decoding speed for baseline and discriminative systems.

Precision for the same set of points is shown in Figure 3. It is important to mention here that both the recall and the precision are improving with the discriminatively trained graph, and we are not trading one off for the other. In scoring the results of our search, we select the top scoring documents (audio segments) for each query. Recall and precision are both improved with the discriminatively trained graph because the correct documents receive a comparatively higher score.

6. Conclusions

We have shown that it is possible to improve the accuracy of audio search without compromising the speed at which data is ingested and analyzed. The baseline system used a small speaker-independent system to first generate word based transcripts from input data, which were subsequently expanded into phone sequences and indexed to allow both in and out of vocabulary retrieval. Speed and accuracy are both critical, and this paper presents a method to improve the accuracy of retrieval by using discriminative training to optimize the parameters of the decoding graph for ASR. For a given decoding speed for the analysis, the updated system shows significant improvement in search accuracy.

7. Acknowledgments

This work was partially supported by the Defense Advanced Research Projects Agency under contract No. HR0011-06-2-0001. We thank Hagen Soltau for the IBM ASR package Attila and Stan Chen for the IBM FST tools.

8. References