Class-dependent Score Combination for Speaker Recognition

Luciana Ferrer[1] *Kemal Sönmez*[2] *Sachin Kajarekar*[2]

[1] Department of Electrical Engineering
Stanford University, Stanford, CA, USA
[2] Speech Technology and Research Laboratory
SRI International, Menlo Park, CA, USA
lferrer@stanford.edu {kemal,sachin}@speech.sri.com

Abstract

Many recent performance improvements in speaker recognition using higher-level features, as demonstrated in the NIST Speaker Recognition Evaluation (SRE) task, rely on combinations of multiple systems modeling a large variety of features. The diversity of the large set of features starting from short-term acoustic spectrum features all the way to habitual word usage from a large set of speakers in a multitude of settings (acoustic environment, speaking style, quantities of enrollment/test data) results in a challenging model combination task. In this work, we are presenting a class-based score combination technique that relies on clustering of both the target models and the test utterances in a vector space defined by a set of speaker-specific transformation parameters estimated during transcription of the talker's speech by automatic speech recognition (ASR). We show that significant performance gains are obtained by using the first few principal components of a model transform for clustering the speaker verification trials into classes for (target speaker, test utterance) pairs, and then training a separate combiner for each class. We report results on the NIST SRE 2004 and FISHER datasets.

1. Introduction

Recent strides in speaker recognition using higher-level features, as demonstrated in the NIST Speaker Recognition Evaluation (SRE) task, have come to rely on combinations of multiple speaker models of a variety of classes of features [1]. The short-term acoustic spectrum features, the traditional basis of a GMM adapted speaker verification system, are combined at the score level with a large number of systems using features that cover various aspects of prosody and habitual word usage. The heterogeneity of the rich set of features used to model the large set of speakers and generate scores for speaker verification in a variety of settings (acoustic environment, speaking style, quantities of enrollment/test data) results in a challenging model combination task. The central goal of system combination is estimating a statistical mapping from the set of scores produced by individual systems to a combined score in order to reduce the equal error rate (EER) or the NIST-defined Detection Cost Function (DCF), the Bayesian risk with a specific set of parameters, both performance metrics summarizing the ultimate measure of performance, the ROC curve or the NIST-defined variation of the ROC curve, the Detection Error Tradeoff (DET) curve.

Many different techniques have been tried for score combination, the most common including linear combination, neural networks, perceptrons [1,2], and support vector machines [3]. Each of these approaches estimates the parameters of a classifier for detecting true vs. impostor speech given the claimed target identity in the vector space of system scores being combined. The final score is either a (possibly non)linear regression on system scores or a distance from the separating boundary.

In designing a classifier in the score space, there is an alternative rationale based on the observation of distinct classes of target and test speakers or acoustical conditions, and so on, which motivates designing separate classifiers for each class. The class-based classifier scores are subsequently combined and/or selected to generate the final score. For example, in [4], trials are clustered according to the quality of the test files to train multiple classifiers during score combination of the cepstral and the higher-level systems.

In this work, we are presenting a class-based score combination technique that relies on clustering of both the target models and the test utterances in a vector space defined by a set of speaker-specific transformation parameters estimated during transcription of the talker's speech by ASR. Specifically, we use the parameters of the maximum likelihood linear regression (MLLR) transform [9], which is estimated for each speaker during ASR. We map the components of the transform to a lower dimensional space via principal component analysis (PCA) and perform the clustering in this space. For each class in the product set, that is, (target, test) pair of clusters, we allocate a separate combiner trained to fit the data in the class. We show that significant performance gains are obtained by using the first few principal components of the MLLR transform for clustering all the speakers into a small number of classes for (target, test) pairs.

The paper is organized as follows. We describe the main infrastructure, i.e. the task, corpora, component systems, and the baseline combination setup in Section 2. The proposed method of class-based score combination is detailed in Section 3. Experiments on the NIST SRE task and analysis of the results are presented in Section 4, with a discussion concluding the paper in Section 5.

2. Baseline Setup

2.1. Task and corpora

The task on which we report performance of the class-based combination techniques is the speaker verification task as defined in the NIST SRE [5]. We report results in the "common" condition, which has one conversation side (of a 5-minute conversation) as enrollment/training data for a target speaker, and one side as the testing utterance. The main criterion in the NIST SRE is the Detection Cost Function, defined as the Bayesian risk with P_{target} = 0.01, C_{fa}=1, and C_{miss}=10. This is the measure of performance we aim to optimize in this paper.

The FISHER development set is created from the FISHER database, which is collected and distributed by LDC for the

DARPA EARS program. We selected two nonoverlapping sets of speakers from this data [6]. Each set is balanced with respect to different genders and handsets. The first set and a part of the EVAL03 dataset involving 425 unique speakers were used to create the background models. The second FISHER set was divided into two equal splits, FISHER1 and FISHER2, and used to build train and development datasets. The 2004 NIST SRE dataset (referred to as EVAL04) is part of the conversational speech data recorded in the Mixer Project and was used for the final evaluation of the method.

2.2. ASR for speaker recognition features

The transcriptions and time alignments used for the long-term, higher-level features were generated with SRI's 5xRT CTS recognition system, using improved models developed for the NIST RT-03F evaluation [6]. The WER on RT-03 evaluation data was 21%.

2.3. Component systems

Following is a description of the systems included in the combination reporting performance on EVAL04 (100*DCF / EER%).

Cepstral GMM system. (Perf: 3.37/8.01) This is a traditional background adapted cepstral GMM system that uses a 2048-component GMM and is described in detail in [6].

Cepstral SVM system. (Perf: 3.13/8.01) This system uses the baseline cepstral feature vector with CMS and transform-based channel normalization [7]. Four different systems modeling different projections of PCA-transformations of mean polynomial vectors (two of which are variance normalized) are combined with an equal weight to produce the final score.

MLLR transform SVM system. (Perf: 3.09/8.92) The MLLR-SVM system uses speaker adaptation transforms used in SRI's speech recognition system as features for speaker verification. The transform coefficients are modeled by SVMs [9].

Phone n-gram system. (Perf: 5.41/12.09) Phone bigram and trigram frequencies for each speaker are extracted from phone recognition lattices. The frequencies for the most frequent N-grams are combined into one feature vector and modeled with SVMs [8].

Word N-gram SVM system. (Perf: 8.06/23.05) This system uses a SVM with a linear kernel with first-, second-, and third-order word N-gram frequencies as features [6].

SNERF system. (Perf: 6.69/16.16) This system uses a set of prosodic features where the extraction region is defined by automatically estimated syllable boundaries. The modeling is done using SVMs [11].

Duration system. Three sets of duration features – state (Perf: 7.16/15.81), phone (Perf: 8.73/19.75), and word level (Perf: 8.62/21.50) – modeled by GMMs are used in this system [10].

2.4. Neural network combiner

The baseline combiner for the SRI system is a single-layer feed-forward network that uses a sigmoid output node during training and a linear output for the final predictions. The linear output allows better combination of these predictions with the ones from our class-dependent combiner. The scores from the component systems are normalized with respect to the statistics in the training set, and then used as inputs to the perceptron, which is trained by minimum squared error with training output labels, 0 (impostor), and 1 (target).

3. Class-based Score Combination

3.1. Main idea

The hypothesis that motivates the class-based approach is that there are distinct classes of pairs of (target, test) trials, and for each class, the combination of short-term spectral and long-term higher-level features may be optimized separately. We cluster target and testing conversation sides with identical schemes as described below, resulting in trial pairs specified by (target, test) clusters. Separate combiners are trained for each class, and during verification probabilities for each class are assigned to the trials and an appropriate mixture of combiners is used.

3.2. Features for clustering

For clustering the speakers we use as features the parameters of the two-class version of the MLLR transform as described in [9]. For each conversation side two affine transforms are estimated, one for the obstruents and one for the nonobstruents, each transform containing a translation vector and a rotation matrix. These transforms can be viewed as a text-independent encapsulation of the speaker's acoustic properties. These values are concatenated to form a long vector of dimension 3120. Half of the background model data are then used to compute the principal components (pc's) of this vector. Finally, the vectors corresponding to the conversation sides on the other half of the background data and the ones used for train and test are transformed into the principal component space. Only the first few pc's are used during clustering.

Due to the speaker-dependent nature of the MLLR features, the obtained clusters contain speakers that are found by the recognizer to be acoustically similar. In particular, given that the MLLR transforms are computed in a gender-dependent manner by the recognizer, we expect this automatically determined gender to be strongly reflected in the value of these features. In fact, we observe that the first pc simply conveys the information of the gender as detected by the recognizer.

3.3. Clustering

We use the remaining half of the background data to obtain the clusters by taking as feature vectors the first N pc's of the MLLR features and training a Gaussian mixture model (GMM) with diagonal covariance, equal volume, and equal shape for each Gaussian. The GMM determines the probability of each conversation side belonging to each of the clusters determined by the Gaussians.

The obtained clusters group the conversation sides into a small set of "prototypical speakers". We use these clusters to assign classes to each of the trials in the train and test databases. The class is determined by the cluster to which the target and the test conversation sides belong. For example, if the target conversation side has the highest probability of belonging to cluster 1 and the test side has the highest probability of belonging to cluster 2, then the class for that trial is given by (1,2). Rather than making hard decisions, we assign probabilities to each of the classes for each trial. The probability for class (N,M) is given by the probability of the target conversation side belonging to cluster N times the probability of the test conversation-side belonging to cluster M.

3.4. Class-dependent combiners

Our goal is to train a model that is dependent on the class of the trial and generates a unique score for each trial given the nine-dimensional vector of individual scores. We have trained one combiner for each of the resulting classes (e.g., 4x4=16 of them if we are using four clusters), using all the samples to train all of the combiners, weighting the samples by the probability of belonging to each class times a factor that depends on the type of trial (true-speaker vs impostor) as explained below. This approach proved to be slightly more robust than using the training trials for which the probability for that class is the highest, especially when using more than two clusters. Linear combiners are trained to predict the labels of the trials (-1 for impostor and 1 for true-speaker) using weighted least squares (WLS) where the weight for each sample *i* when training the combiner corresponding to class j is given by

$$w_{j,i} = \begin{cases} p_{j,i}\ p_{tsp} / s_{j,tsp} & \text{for true speaker trials} \\ p_{j,i}\ p_{imp} / s_{j,imp} & \text{for impostor trials} \end{cases} \quad (1)$$

where $p_{j,i}$ is the probability of trial *i* belonging to class *j*, p_{sp} is the true-speaker prior probability, p_{imp} is the impostor prior probability and $s_{j,imp}$ and $s_{j,sp}$ are the sum of the probabilities for all the impostor and true speaker trials for class *j*. Defining the weights in this way allows us to balance the data used to train each of the combiners to see an "equivalent" frequency of true speaker trials equal to the prior p_{sp}. This prior is set to 0.09, as is usually done when trying to optimize for DCF with the NIST parameters. In our case the training database was built with these priors, hence no compensation needs to be done when training the class-independent combiners.

During testing, all the combiners are used to score each of the trials. A weighted average of the resulting predictions is used, where the weight for combiner *j* is given by the probability for the trial of belonging to class *j*. The resulting scores determine the final predictions of our class-dependent combiner.

3.5. Combination with neural network scores

The resulting prediction can be further averaged with the predictions obtained by the neural network. To this end, we first (Z-)normalize (substract mean and divide by standard deviation) the scale of the scores given by both systems so that equal weights can be used for the combination. The statistics used for Z-norm are obtained on the development test scores. As we will see in the next section, averaging the two combiners leads to additional performance improvement.

4. Experiments and Results

The system was developed with FISHER data. FISHER1 was used for training each of the combiners (including the neural network combiner), and FISHER2 was used for development/parameter tuning. The data used to train the background models for the individual systems was split into two and used for computation of the principal components for the MLLR features and training of the GMM clustering. EVAL04 data was used for the final evaluation of the method.

We tried different numbers of clusters for both the training conversation side and the test conversation side and found that most of the improvement is achieved by using 2x2 clusters. We will use this case to interpret the way the class-dependent combiner is working. First, we observe that the clusters are almost exclusively composed of same-gender (as determined by the recognizer, which agrees most, but not all, of the time with the gender detected by a GMM-based gender classification system) conversation sides. This means that the 2x2 case is almost the same as labeling the trials by the gender obtained by the recognizer for both train and test conversations and training a classifier for each of the four cases. In theory there should not be any "cross-cluster" trials because the task is designed to have only same-gender trials. However, since the gender as determined by the recognizer is not the same as the actual gender, there are enough trials in the cross-cluster classes to warrant separate linear classifiers.

Table 1 shows several statistics for the data in each of the classes and on the complete dataset. First and second rows show the number of impostor and true-speaker trials in each class. Note that the cross-cluster classes ((1,2) and (2,1)) have one order of magnitude less data than the same-cluster classes. As mentioned before, this is because there are no same-gender trials in the task. Only those trials for which the recognizer's gender label does not match the true gender fall in those classes. Hence, these classes correspond to the cases where the true-speaker trials were especially hard for the cepstral features that form the basis of the gender detectors to the degree that the recognizer thought that the conversation sides corresponded to two different genders, while they were actually from the same speaker.

Table 1: The breakdown of data and results for the class-independent and class-dependent combiners.

	Class (1,1)	Class (1,2)	Class (2,1)	Class (2,2)	All data
Number of impostor trials	8456	687	402	5525	15070
Num. of true-speaker trials	794	38	38	638	1508
EER for class-indep. Classifier	1.97%	25.9%	29.6%	0.65%	3.09%
EER for class-dep. Classifier	1.81%	18.5%	14.8%	0.65%	2.56%
Rel. weight for noncep. systems	8.1%	46.9%	42.0%	11.1%	11.3%

Third and fourth rows show the EER for the data in each of the classes, when using a class-independent classifier (trained using least squares regression on the complete data) and the class-dependent classifier. Analyzing the performance of each of the classifiers in the case of the class-independent combiner we can observe that the cross-cluster cases are the hardest ones by far: with EER of more than 20% compared to less than 2% for the same-cluster cases, reflecting the observation above that the cross-cluster classes correspond to the hardest trials for the cepstral systems. For the same reason, note that it is in the cross-cluster classes where the class-dependent combiner makes the biggest difference, improving performance in those classes by up to 50% relative. Having a class-dependent combiner allows adaptation of combination weights by assigning more import to the noncepstral systems. In fact, the noncepstral features (duration, snerf and word-ngram) receive a much higher weight relative to the class-independent case for those classes as can be seen in the last row in Table 1, which shows the percent of weights assigned to the noncepstral systems compared to the total sum of the weights for the classifiers trained using the weights for the corresponding class (in the case of all-data, the weights are one for all samples). Finally,

the performance in the same-cluster classes does not degrade with respect to the class-independent classifier.

Although the interpretation of what the clusters and the classes are is not as clear as for the two-cluster case, using three and four clusters slightly improves the performance. Table 2 shows the results for 4x4 and 2x2 clusters for comparison (performance with the 3x3 clusters is in between). In all cases, no more pc's than the first three were found to be useful. The measure of performance we are optimizing is the DCF. In FISHER2, a relative improvement of 12% in DCF with respect to the neural network performance is achieved by averaging the neural network prediction with the class-dependent prediction. On EVAL04, the relative improvement is around 7%. For this data, the number of false rejections (true-speaker trials misclassified as impostor trials) for the averaged combiner at the DCF point is significantly smaller (at 95% level) than for the neural network, while the number of false alarms is not significantly different, reaffirming our observation that it is mainly the misclassified true-speaker trials that are corrected by this class-dependent combination method.

*Table 2: Performance (100*DCF/EER%) using neural network combiner and class-dependent combiner for different parameter settings.*

Test DB	Neural Net Performance		Class-dependent combiner			Neural Net + Class-dep Performance	
			Params	Performance			
FISH 2	0.564	2.94	Gen 2x2	0.646	2.56	0.602	2.86
			Ran 2x2	0.668	3.09	0.609	3.01
			Hnd 3x3	0.745	3.24	0.639	3.01
			MLLR 2x2	0.541	2.56	0.511	2.94
			MLLR 4x4	0.556	2.64	0.496	2.86
EVAL 04	2.370	5.27	MLLR 2x2	2.472	5.20	2.271	4.99
			MLLR 4x4	2.579	5.55	2.194	5.13

It is interesting to note that the performance of the 4x4 class-dependent combiner alone is not better than that of the 2x2 case, but in combination with the neural network the relative gain is greater. This indicates that the two combiners, the neural network and the class-dependent combiner, are less correlated when using more classes. In the 4x4 case, some of the classes end up with very few samples (less than a hundred in many cases). This indicates that a back-off technique could improve the performance of this combiner. One way to do this is to use fewer clusters for either the train or the test conversation, but this does not result in significant improvements. Using the same number of clusters for both the train and the test conversation sides seems to be optimal.

For comparison, we tried using the gender automatically determined by a cepstral GMM gender detector, which results in predictions that are much closer to the true gender than those of the recognizer-determined gender, to label the trials. In this case, the cross-cluster trials are much fewer than in the case of two clusters given by MLLR features. The class-dependent combiner using these labels is worse than the neural network combiner in DCF, and averaging those two gives only a very slight improvement of EER. Using random clusters or handset dependent clusters (three cases: cell, carbon and electret) does not give any improvement, not even after combining with the neural network result.

Other feature vectors such as mean and standard deviation of the duration of the phones and latent semantic analysis features were tried as clustering features, but none of them showed consistent improvements over the neural network results.

5. Conclusions

We have introduced a novel technique for combining information sources in a speaker recognition system by classifying trials based on the parameters of an ASR speaker adaptation transform and training separate combiners for each trial class. Analysis of the resulting classes indicates that emerging cross-cluster classes contain the hardest trials for the short-term cepstral systems. Through estimation of distinct combiners for these classes, higher-level features are assigned more weight and the performance for the trials in cross-cluster classes improves dramatically, resulting in an overall performance gain. We have reported speaker verification results on the NIST SRE task, and the technique has been shown to provide significant performance gains in combination with a neural network combiner.

6. Acknowledgements

We thank our SRI colleagues Elizabeth Shriberg, Andreas Stolcke and Anand Venkataraman for valuable suggestions and comments.

This work was funded by NSF STIMULATE 9619921 (which supports an offsite RAship through Stanford University), and by an interagency KDD project administered through NSF 9619921. The views herein are those of the authors and do not reflect the views of the funding agencies.

7. References

[1] D. Reynolds, W. Andrews, J. Campbell, J. Navratil, B. Peskin, A. Adami, Q. Jin, D. Klusacek, J. Abramson, R. Mihaescu, J. Godfrey, D. Jones, and B. Xiang, "The SuperSID Project: Exploiting high-level information for high-accuracy speaker recognition", *Proc. IEEE ICASSP*, Hong Kong, 2003

[2] J. Campbell, D. Reynolds, and R. Dunn, "Fusing high- and low-level features for speaker recognition", *Proc. Eurospeech*, Geneva, Switzerland, 2003.

[3] D. Garcia-Romero, J. Fierrez-Aguilar, J. Ortega-Garcia and J. Gonzalez-Rodriguez, "Support vector machine fusion of idiolectal and acoustic speaker information in Spanish conversational speech", in *Proc. IEEE ICASSP*, Hong Kong, April 2003.

[4] Y. Solewicz and M. Koppel, "Enhanced fusion methods for speaker verification", *SPECOM*, Saint-Petersburg, September, 2004

[5] NIST 2004 Speaker Recognition Evaluation plan, http://www.nist.gov/speech/tests/spk/2004/SRE-04_evalplan-v1a.pdf.

[6] S. Kajarekar, L. Ferrer, E. Shriberg, K. Sonmez, A. Stolcke, A. Venkataraman, and J. Zheng, "SRI's 2004 NIST speaker recognition evaluation system," *Proc. IEEE ICASSP*, Philadelphia, March 2005.

[7] W. M. Campbell, "Generalized linear discriminant sequence kernels for speaker recognition," *Proc. ICASSP*, Orlando, May 2002.

[8] A. Hatch, B. Peskin and A. Stolcke, "Improved phonetic speaker recognition using lattice decoding", *Proc. IEEE ICASSP*, Philadelphia, March 2005.

[9] A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, and A. Venkataraman, "MLLR transforms as features in speaker recognition", submitted to *Eurospeech*, 2005

[10] L. Ferrer, H. Bratt, V. R. Gadde, S. Kajarekar, E. Shriberg, K. Sonmez, A. Stolcke, and A. Venkataraman, "Modeling duration patterns for speaker recognition", *Proc. Eurospeech*, Geneva, September, 2003.

[11] E. Shriberg, L. Ferrer, A. Venkataraman, and S. Kajarekar, "SVM modeling of SNERF-grams for speaker recognition", *Proc. ICSLP*, South Korea, September, 2004.