ICSI SRE2006 System Description

Nikki Mirghafori, Andrew Hatch, Lara Stoll, Howard Lei

International Computer Science Institute
1947 Center Street, Suite 600
Berkeley, CA 94704
nikki@icsi.berkeley.edu

1. Overview of the Submission

We are submitting scores for four conditions for two systems. The conditions are:

- 1conv4w-1conv4w
- 8conv4w-1conv4w
- 1conv4w-1convmic
- 8conv4w-1convmic

The primary submission (ICSI_1) is a combination of the following ICSI sub-systems:

1. Word conditional HMM system (WordHMM)
2. Lexically-Conditioned Phone Lattice N-grams System + WCCN (LC-PhoneNgram)
3. Phone Lattice N-gram system + WCCN (PhoneNgram)
4. POS Word N-gram System (POS/Word)
5. Lexical Statistics System (LexStats)
6. Baseline cepstral GMM system (GMM) [courtesy of SRI]

The secondary submission (ICSI_2) is a collaborative submission with SRI. The systems 1-5 above are combined with the following SRI sub-systems below:

- Cepstral GMM System
- Cepstral SVM System
- MLLR SVM System
- MLLR SVM System (*with ICSI's WCCN applied*)
- Word-Ngram SVM System
- Duration GMM System
- Grammar, Word, and Syllable SNERF System

For ICSI_1, the combination of sub-systems 1-6 is used to obtain scores for the English trials. The combination of PhoneNgram and GMM sub-systems is used to obtain scores for the non-English trials. The system were combined using an SVM, trained as a classifier.

For ICSI_2, the combination of all sub-systems is used to obtain scores for the English trials for the 1conv4w test condition. For the English trials for the 1convmic test condition, ICSI systems 1-5 and the SRI Cepstral, MLLR (no WCCN) and Word-Ngram systems are used to obtain scores. The combination of Cepstral, MLLR (no WCCN) and PhoneNgram systems is used to obtain scores for the non-English trials for both the 1conv4w and 1convmic test conditions. The systems were combined using an SVM, trained as a classifier.

In the following sections, we briefly describe the systems, as well as the WCCN normalization strategy, which was applied to some of the systems.

2. Development Data Sets

2.1. Development

SRE05 data was used as the development set to estimate the SVM classifier model parameters and optimal operating point (corresponding to the minimum DCF), as described in Combination Strategy below.

2.2. Background*

Background data for all sub-systems was selected from Fisher and SRE 2003 (Switchboard II) data sets. The background set comprised 1128 utterances from unique speakers in Fisher (5 minutes each), and 425 utterances from unique speakers from SRE 2003 (subsets 4, 5, 6, and 8) extended data (2.5 minutes each). Subsystems 2 and 3 used additional conversation sides from SRE 2003 for background training, detailed in the sub-system-specific sections.

2.3. TNORM*

249 TNORM models were constructed from 2.5 minutes of each 5-minute Fisher conversation. These TNORM models were chosen from unique speakers and were roughly gender-balanced. The data comprised a similar number of electret and cellphone channels, and a handful of carbon-button channels.

*For more details in the construction of the background and TNORM sets, please refer to the system description by SRI.

3. Combination Strategy

The system combination was performed using an SVM classifier, implemented with the SVMLite package [1]. The English trials were combined separately from the non-English trials, and used different sets of systems, as explained in the Overview section. For ICSI_1, all 6 sub-systems were combined for the English trials, while the non-English trials only used scores from the baseline cepstral GMM system and the phone lattice n-gram system. In order to train the SVM combiner and to estimate the operating threshold, SRE05 data was used. Specifically, to train the SVM model, SRE05 1conv4w-1conv4w, 8conv4w-1conv4w,

1conv4w-1convmic, and 8conv4w-1convmic train-test conditions were used for the corresponding conditions of SRE06. For both the 1conv4w and 1convmic test conditions, only the (non-)English trials in SRE05 were used to estimate the SVM parameters and operating threshold for (non-)English trials in SRE06. However, since there were no non-English true speaker trials in 8conv4w-1convmic condition of SRE05, English trials from SRE05 were used to train the SVM classifier and estimate the operating threshold for *both* English and non-English trials in SRE06.

SRE05 data for each system was also used to estimate the mean and standard deviation of the scores for that system; these estimates were then used to normalize both the SRE05 and SRE06 scores for each system, by subtracting the mean and dividing by the standard deviation, before the scores were put through the combiner. The SVM classifier used a linear kernel. In order to account for the fact that there are significantly more impostor trials than true speaker trials, the SVM used a weighting factor, equal to the ratio of the number of impostor trials to the number of true speaker trials (calculated from the SRE05 training scores only), as a cost-factor such that the training errors on positive examples were weighted more heavily than errors on negative examples.

The operating threshold for the (non-)English subsets were subtracted from (non-)English trial scores, to produce an effective operating threshold of zero for both sets. The sets of scores were then appended for submission. The submitted scores are normalized log likelihood scores, and NOT a posteriori probabilities. A large negative value (-100) was submitted as scores for what were deemed to be empty speech files.

4. Sub-System Descriptions

In the following sections, each ICSI subsystem is described. For SRI sub-systems, please see the SRI system description for more detail.

4.1. Word Conditional HMM System (WordHMM)

This sub-system is an implementation of that described in [2]. Please refer to the paper for more detail. The system uses background and target keyword models generated by Hidden Markov Models (HMMs) for 19 select keywords (where a "keyword" is a word or common word-pair) drawn primarily from the discourse marker, backchannel and filled pause categories. The system employs speaker-independent keyword-specific HMMs which are then adapted to the target training data to create target models, and computes test scores using the usual likelihood ratio of target to background. Only intervals of speech corresponding to the 19 keywords are scored.

4.1.1. Feature Extraction

The HMM feature vectors consist of 19 mel cepstra, the zeroth cepstrum, and their first differences, for a total of 40 features per vector. Cepstral Mean Subtraction was performed over the union of speech-rich segments for each conversation side (segmentation provided by SRI).

4.1.2. Background Model

Keyword UBMs were obtained by training on background data from the SRE 2003 Extended Data set (Switchboard II, phases 2 and 3) and Fisher, as explained in the Data section.

The keyword HMMs were simple left-to-right state sequences with self-loops and no skips. Each state model consisted of a mixture of eight Gaussians and the number of states for each keyword was defined to be the smaller of the number of phones in the standard pronunciation of the word, multiplied by 3, and the median duration in frames, divided by four. All modeling and scoring was performed using the HMM Toolkit, HTK.

4.1.3. Training

Speaker-specific keyword models were obtained by MAP adaptation of the background models by adapting only the means of the Gaussians. In the event that there was no training data for a particular keyword, the UBM was simply copied as the speaker-specific model. This resulted in removing the influence of the keyword, as the contribution to the overall score was zero, due to the cancellation of target and background. Keyword locations within the audio file were determined by word-level alignment information made available from SRI's automatic speech recognition (ASR) system.

4.1.4. Testing

Each keyword appearing in the test segment was scored by taking the difference between the log probabilities obtained from scoring the speaker-specific and UBM models against the test tokens. The final score was obtained by adding these keyword scores and normalizing by the total number of frames.

4.1.5. Score Normalization

TNORM was applied. As explained in the Data section, Fisher TNORM models were constructed from one conversation side and served as TNORM models both for the 8side and the 1side conversation training conditions.

4.1.6. Computational Resources

All computation was performed on a fleet of Intel 2.8GHz Xeon processors with 2-3GB of RAM and Dual Core AMD Opteron 2.2 GHz processor with 4GB of RAM. Note that the reported processing times throughout this system description do not include the time required to generate the ASR word and/or phone output used in the high-level systems. This information can be found in the SRI system description.

Feature Extraction:
Telephone: total elapsed: 51204s, total cputime: 234414s
Altmic: total elapsed: 15238s, total cputime: 6511s

Telephone train/test:
1side-1side: total elapsed: 59527s, total cputime: 27929s
8side-1side: total elapsed: 42929s, total cputime: 18450s

Altmic train/test:
1side-1side: total elapsed: 48731s, total cputime: 36357s
8side-1side: total elapsed: 10488s, total cputime: 6273s

Tnorm normalization:
Tel 1side-1side: total elapsed: 1182740s, total cputime: 901345s
Tel 8side-1side: total elapsed: 764172s, total cputime: 618765s
Altmic 1side-1side: total elapsed: 1352400s, total cputime: 1130205s
Altmic 8side-1side: total elapsed: 1090330s, total cputime: 911358s

4.2. Lexically-Conditioned Phone Lattice N-grams System (LC PhoneNgram)

This sub-system used ASR word and phone lattice output to obtain phone lattice n-gram counts in target and test conversation sides, conditioned on the set of 52 most frequent word unigrams in 1128 conversation sides of the Fisher corpus and 425 conversation sides of the Switchboard II corpus. The motivation for this system was that the way different speakers speak and pronounce each of the 52 words via phone usage provides discriminative information among speakers.

4.2.1. Feature Extraction

The features for this system consisted of phone lattice uni, bi, and tri-gram counts conditioned on each of the 52 word unigrams. Phone lattice counts for different word unigrams in a conversation side were concatenated to form the final conversation side feature vector. Only phones with counts greater than 10 were kept.

4.2.2. Training

Feature vectors from speaker model conversation sides were used to train speaker models against the background model via the support vector machine (SVM) algorithm. Feature vectors from speaker model conversation sides were used as positive training examples, while those from background conversation sides were used as negative training examples. We used 6117 Fisher and Switchboard II conversation sides for the background model. Of those, 1128 were from the Fisher corpus. SVM training was done using the SVMLite package. WCCN normalization (see 5) was applied to the feature vectors before SVM training.

4.2.3. Scoring

Features vectors from test conversation sides were scored against speaker models using the SVM. TNORM (trained with Fisher speaker models) was applied to every test conversation.

4.2.4. Computational resources

Word conditioning the phone lattices – total elapsed: 3102600 s, total cputime: 1726175 s
Generating counts – total elapsed: 11410 s, total cputime: 9918 s

4.3. Phone Lattice N-grams System (PhoneNgram)

The ICSI phone n-gram system is similar to the system described in [3]. The phone n-gram system uses an open-loop phone recognizer to generate phone lattices, which are then used to compute expected counts of phone n-grams. These expected counts are converted into relative frequencies, which are then used as feature vectors for training SVM-based speaker models.

4.3.1. Phone Recognition

The phone n-gram system uses the DECIPHER speech recognizer [3] developed by SRI International to generate phone lattices for every conversation side. Our particular realization of DECIPHER uses gender-dependent, monophone acoustic models, where each monophone is modeled by a 3-state hidden Markov model (HMM). The acoustic model was trained on the Switchboard I corpus using MFCC features. Phone decoding was performed in open-loop mode (i.e. we used a unigram phone language model with uniform probabilities) with a vocabulary of 46 phone units.

4.3.2. SVM Features and Training

The phone ngram system extracts one feature vector for every conversation side, where the features represent relative frequencies of the 8500 most frequent phone bigrams and trigrams. We use a linear kernel to train an SVM-based model for every target speaker (see [4]). The SVMs are trained using a one-versus-all approach, where the conversation sides from the target speaker's training data are used as positive training examples, and the conversation sides in a set of background data are used as negative training examples. For our system, we used a background dataset composed of 1128 conversation sides taken from the Fisher corpus and 425 conversation sides taken from the Switchboard II corpus, as explained in the Data section. WCCN normalization (see 5) was applied to the feature vectors before SVM training and scoring, which were done using the SVMLite package.

4.3.3. Scoring

To score a given test-target pair, we simply applied the feature vector of the test conversation to the SVM output function of the target model. We then used TNORM to normalize the scores for every test conversation.

4.3.4. Computational Resources

Each of the four submitted conditions ran in roughly two hours on 25 CPUs. Generating the TNORMed scores took about an extra hour per experiment.

4.4. POS Word N-gram System (POS/Word)

Eric Brill's Supervised Part of Speech Tagger (www.cs.jhu.edu/~brill) was used to supply POS tags on the word hypotheses. The relative frequencies of the joint POS/word tags were calculated and used in an SVM system. See [3] for a similar word-only system, except that the tokens of the POS Word Ngram system represent POS and word information jointly, such as: "but/CC i/NN see/VB ". Relative frequencies for a total of 125,700 uni-, bi-, and tri-grams were calculated and an SVM with a linear kernel was trained using SVMLite.

Computational resources: the system for all the four submitted conditions ran in roughly two hours on 12 CPUs.

4.5. Lexical Statistics (LexStats)

The Lexical Statistics system attempts to capture high-level sentence and conversation information by utilizing features such as: number of conversation turns, number of words (per conversation and per turn), number of characters (per conversation and per turn), and speaking rate. Eight such features were calculated for each conversation side and used in an SVM which was trained with SVMLite and utilized a linear kernel.

Computational resources: the system for all the four submitted conditions ran in roughly 10 minutes on 12 CPUs.

5. Within-Class Covariance Normalization (WCCN)

We applied within-class covariance normalization (WCCN) [5, 6, 7] to the following three systems: SVM-based Lattice Phone N-grams, SVM-based Word-Conditioned Lattice Phone N-grams, and SRI MLLR-SVM system (used in the joint system submissions, ICSI_2).

The WCCN approach implements a generalized linear kernel of the form, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{W}^{-1} \mathbf{x}_2$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are two input feature vectors, and $\mathbf{W}$ is the *expected within-class covariance matrix* over all classes (i.e. speakers) in the training set. We define $\mathbf{W}$ as follows:

$$\mathbf{W} \triangleq \sum_{i=1}^{M} p(i) \cdot \mathbf{C}_i,$$

$$\mathbf{C}_i \triangleq \mathbb{E}\,(\mathbf{x}_i - \bar{\mathbf{x}}_i)(\mathbf{x}_i - \bar{\mathbf{x}}_i)^T \quad \forall i \in \{1, \ldots, M\}.$$

Here, $\mathbf{x}_i$ represents a random draw from class (i.e. speaker) i, M represents the total number of classes, and $\bar{\mathbf{x}}_i$ represents the mean of $\mathbf{x}_i$. The terms, $\mathbf{C}_i$ and $p(i)$, represent the covariance matrix and the prior probability of class i. (Note that we use the term, "class," to refer to a given speaker.)

Given $\mathbf{W}$, where $\mathbf{W}$ is full-rank, we can perform WCCN on the input feature vectors by using the following feature transformation, Φ:

$$\Phi(\mathbf{x}) \triangleq \mathbf{A}^T \mathbf{x}. \tag{1}$$

Here, $\mathbf{A}$ is defined as the Cholesky factorization of $\mathbf{W}^{-1}$:

$$\mathbf{A}\mathbf{A}^T \triangleq \mathbf{W}^{-1}.$$

The WCCN feature transformation emphasizes "directions" in feature space that are informative (i.e. directions where the within-class variance is small compared with the overall variance), while attenuating directions that are noisy. Under various conditions, it can be shown that WCCN minimizes a particular upper bound on classification error when used for SVM training [5, 6].

5.1. Smoothing

We used the following smoothing model to compute estimates of $\mathbf{W}$:

$$\hat{\mathbf{W}}_s \triangleq (1 - \alpha) \cdot \hat{\mathbf{W}} + \alpha \cdot \mathbf{I}, \qquad \alpha \in [0, 1]. \tag{2}$$

Here, $\hat{\mathbf{W}}_s$ represents a smoothed version of the empirical expected within-class covariance matrix, $\hat{\mathbf{W}}$, and $\mathbf{I}$ represents an $N \times N$ identity matrix where N is the dimensionality of the feature space. The α parameter represents a tunable smoothing weight whose value is between 0 and 1.

5.2. Experimental Procedure

We followed the procedure described in [7] for applying WCCN to large feature sets. The procedure is summarized as follows:

1. Perform per-feature within-class *variance* normalization on all of the input features (i.e. scale each features to have an average within-class variance of one on the training data).
2. Use kernel principal component analysis (KPCA) with a linear kernel to reduce the dimensionality of the input feature space. For linear kernels, the KPCA transformation can be expressed in terms of a projection matrix, $\mathbf{U}$:

$$\Phi_{PCA}(\mathbf{x}) \triangleq \mathbf{U}^T \mathbf{x},$$

   We use Φ_{PCA} to represent the KPCA transformation of feature vector $\mathbf{x}$. Further details on how to compute $\mathbf{U}$ are provided in [7]. For a general overview of KPCA, see (for instance) [8, 9].
3. Compute the so-called *PCA-complement* for every input feature vector. We use $\Phi_{\overline{PCA}}(\mathbf{x})$ to represent the PCA-complement of feature vector $\mathbf{x}$. Given the $\mathbf{U}$ projection matrix from step 2, $\Phi_{\overline{PCA}}(\mathbf{x})$ is computed as follows:

$$\Phi_{\overline{PCA}}(\mathbf{x}) \triangleq (\mathbf{I} - \mathbf{U}\mathbf{U}^T)\mathbf{x}.$$

4. Perform WCCN on the PCA feature set (i.e. the feature vectors produced by the Φ_{PCA} transformation) using the smoothing model shown in equation (2). The smoothing parameter α is tuned on a set of held-out cross-validation data.
5. For every input feature vector, $\mathbf{x}$, concatenate a scaled version of the WCC-normalized PCA features with a scaled version of the PCA-complement features to arrive at the final feature representation, Φ:

$$\Phi(\mathbf{x}) \triangleq \begin{bmatrix} (1 - \sigma) \cdot \mathbf{A}^T \Phi_{PCA}(\mathbf{x}) \\ \sigma \cdot \Phi_{\overline{PCA}}(\mathbf{x}) \end{bmatrix}, \qquad \sigma \in [0, 1].$$

   Here, $\mathbf{A}^T$ represents the transformation matrix derived in step 4 from equation (1) to perform WCCN on the PCA features. Thus, $\mathbf{A}^T \Phi_{PCA}(\mathbf{x})$ represents the WCC-normalized PCA component of feature vector $\mathbf{x}$. We use the parameter σ to control the relative weight applied to the two feature sets (i.e. the PCA set and the PCA-complement set). This parameter is tuned on a held-out cross-validation set.
6. Use the final feature representation Φ to train and test SVM-based speaker models.

5.3. Training Sets

We used ∼3600 conversation sides from the SRE2003 task to train the KPCA transformation described in step 2 of the experimental procedure. Approximately 7200 conversation sides from SRE2003 were used to train $\hat{\mathbf{W}}$ and to train the per-feature within-class variances for step 1. The α and σ weights were tuned sequentially (i.e. first α, then σ given α) on the SRE2005 dataset. For every feature set where WCCN was applied, the optimal values for α and σ were found to be approximately 0.92 and 0.4, respectively.

5.4. Computational Resources

We used 25 computers in parallel to train the full WCCN feature transformation (i.e. Φ from the experimental procedure) for each of the following systems: phone n-grams, word-conditioned phone n-grams, and SRI International's MLLR-SVM system. The training time took approximately 10, 22, and 20 hours, respectively.

6. Acknowledgments

We thank our advisor on this project, George Doddington, and our collaborators at SRI, especially Sachin "extraordinaire" Kajarekar, for myriad exchanges of resources and insights, which not only have been technically fruitful, but FUN!

7. References

[1] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European Conference on Machine Learning*, 1998.

[2] K. Boakye and B. Peskin, "Text-constrained speaker recognition on a text-independent task," in *Proc. Odyssey Speaker Recognition Workshop*, Toledo, Spain, 2004, pp. 129–34.

[3] S. Kajarekar, L. Ferrer, A. Venkataraman, K. Sonmez, E. Shriberg, A. Stolcke, and R. R. Gadde, "Speaker recognition using prosodic and lexical features," in *ASRU*, St. Thomas, US Virgin Islands, 2003.

[4] A. Hatch, B. Peskin, and A. Stolcke, "Improved phonetic speaker recognition using lattice decoding," in *ICASSP*, 2005.

[5] A. Hatch and A. Stolcke, "Generalized linear kernels for one-versus-all classification: application to speaker recognition," in *proc. of ICASSP*, Toulouse, France, 2006.

[6] A. Hatch and A. Stolcke, "Adaptive linear kernels for binary classification of multiclass data," in *Technical Report*, 2006, www.icsi.berkeley.edu/∼ahatch/alk.pdf.

[7] A. Hatch and A. Stolcke, "Within-class covariance normalization for SVM-based speaker recognition," in *submitted to ICSLP*, 2006, icsi.berkeley.edu/∼ahatch/papers/interspeech2006/finalPaper.pdf.

[8] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, Cambridge, 2004.

[9] N. Cristianini and J. Shawe-Taylor, *Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, Cambridge, 2000.