# THE MIT-LL/IBM 2006 SPEAKER RECOGNITION SYSTEM: HIGH-PERFORMANCE REDUCED-COMPLEXITY RECOGNITION*

*W. M. Campbell*•, *D. E. Sturim*•, *W. Shen*•, *D. A. Reynolds*•, *J. Navrátil*†

•MIT Lincoln Laboratory, †IBM

## ABSTRACT

Many powerful methods for speaker recognition have been introduced in recent years—high-level features, novel classifiers, and channel compensation methods. A common arena for evaluating these methods has been the NIST speaker recognition evaluation (SRE). In the NIST SRE from 2002-2005, a popular approach was to fuse multiple systems based upon cepstral features and different linguistic tiers of high-level features. With enough enrollment data, this approach produced dramatic error rate reductions and showed conceptually that better performance was attainable. A drawback in this approach is that many high-level systems were being run independently requiring significant computational complexity and resources. In 2006, MIT Lincoln Laboratory focused on a new system architecture which emphasized reduced complexity. This system was a carefully selected mixture of high-level techniques, new classifier methods, and novel channel compensation techniques. This new system has excellent accuracy and has substantially reduced complexity. The performance and computational aspects of the system are detailed on a NIST 2006 SRE task.

***Index Terms***— speech processing, speaker recognition

## 1. INTRODUCTION

In recent years, several novel techniques have proven effective for text-independent speaker recognition and have reduced error rates dramatically. These techniques have addressed some of the main issues in speaker recognition: speaker modeling, channel compensation, and multiple linguistic cues to speaker identity. Our goal in this paper is to discuss a successful fusion of these techniques and tradeoffs in complexity in providing low error rates.

One key area of improvement in speaker recognition has been in direct modeling of the spectral content of speech. These systems have been a driver for low error rates in speaker recognition. Two significant innovations have been the introduction of discriminative techniques and advanced channel compensation methods. For discriminative techniques, support vector machines (SVMs) were found to fuse well with standard Gaussian mixture models [1]. For channel compensation, methods such as latent factor analysis (LFA) [2] and nuisance attribute projection (NAP) [3] have significantly reduced error rates by supervised modeling of channel and session variation.

Another key area of improvement in speaker recognition has been the fusion of high-level features. The 2002 JHU SuperSID workshop [4] showed that significant reduction in speaker error rates could be achieved by including features modeling high-level cues in speech (as opposed to low-level cepstral modeling). These cues included information about prosody, phonotactics, idiolect, and dialog.

A drawback of the approaches that evolved from fusion of multiple cepstral and high-level systems is that many systems were run independently without regard for system complexity. Also, focusing on many high-level systems creates a system which may be vulnerable to multilanguage variation. A re-evaluation of system components in 2006 resulted in a streamlined system with an emphasis on cepstral system performance and a single high-level tokenizer. This approach proved powerful on several NIST SRE 2006 tasks.

We outline in this paper the systems, techniques, and experimental results for the new systems. Section 2 describes the cepstral systems. Section 3 described the high-level tokenizer and the multiple modeling methods from this system. Finally, in Section 5, we describe experimental evaluation of the system on the NIST SRE 2006 data.

## 2. CEPSTRAL SYSTEMS

### 2.1. Front end processing for cepstral systems

The cepstral-based systems used a common set of speech activity detection marks from a GMM-based speech activity detection (SAD) system and an adaptive energy-based SAD.

Two sets of features are used for recognition—MFCCs and LPCCs. For MFCCs, 19 cepstral coefficients and deltas were computed to produce a 38 dimensional feature vector. The feature vector stream is processed through SAD to eliminate non-speech vectors. RASTA, CMS, and variance normalization are then applied to the feature stream.

For linear prediction (LP) based processing, 18 LPCCs are obtained from 12 LP coefficients extracted using the standard Levinson-Durbin recursion. For LPCCs, windowing, frame rate, RASTA, CMS, and variance normalization are performed in the same manner as for MFCCs.

### 2.2. GMM ATNorm

The MITLL GMM UBM (universal background model) speaker detection system is the basis for three of the cepstral systems implemented for the 2006 NIST SRE. The first system, GMM ATNorm (adaptive TNorm), uses a standard GMM UBM implementation with MFCC features. The GMM ATNorm system MAP adapts a 2048 mixture component GMM UBM with a relevance factor of 16 [5]. ATNorm uses speaker-dependent cohort speakers in the TNorm calculation [6]. The speaker-dependent TNorm set is determined by the TNorm models which have the highest similarity to the speaker model on a set of imposter utterances. Similarity is calculated by arranging the per utterance scores into a vector and using a Euclidean distance to compare vectors. The closest set of $p$ cohort models are used for TNorm during run time; $p$ was empirically chosen to be 55. Cohorts are selected from the NIST SRE 2004 corpus.

*This work was sponsored by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

### 2.3. GMM LFA

The GMM latent factor analysis system (LFA) was based directly on the work presented in [7]. MFCC features are used as inputs to the system. The GMM LFA approach reduces session variability through a low-dimensional subspace compensation in both training and testing. The session variability is modeled as an bias to the GMM model means

$$\mathbf{m}_i(s) = \mathbf{m}(s) + U\mathbf{x}(s) \quad (1)$$

where $\mathbf{m}_i(s)$ and $\mathbf{m}(s)$ are supervectors of stacked means GMM means [8], and $U$ is a low-rank matrix. The $\mathbf{m}_i(s)$ is the supervector from the $i$-th session of talker $s$, whereas $\mathbf{m}(s)$ is the session independent term of talker $s$. The GMM UBM supervectors are generated with a standard GMM system using MAP adaptation.

One of the challenges of implementing this system was selecting an appropriate algorithm from the many variations available for training the hyperparameters and implementation of speaker enrollment and verification [7, 8, 9]. Training of the $U$ matrix is accomplished by finding the GMM supervectors on a per utterance basis for a large corpus, forming delta vectors on a per speaker basis, and then computing eigenvectors using the kernel trick [9, 10]. The data sets used to train the low-rank transformation matrix are the Switchboard II phase 1-5 corpora. Enrollment and verification are performed using the method of Vogt, et. al. [7]. This method involves computing estimates of the channel factors $\mathbf{x}$ during both enrollment and verification. Only one Gauss-Seidel type iteration is used in the estimation process.

Z-norm followed by T-normalization was performed on the scores. The Z-norm imposter test messages are drawn from the Switchboard II phases 1-5 corpora. TNorm cohorts are drawn from the SRE04 and Fisher corpora.

### 2.4. SVM GSV NAP

The SVM GMM supervector (GSV) system uses a novel kernel based upon an approximation to the KL divergence. This method is described in detail in [10]. Feature extraction is performed using a standard MFCC front end.

The SVM GSV kernel is computed as follows. A 2048 mixture GMM UBM is adapted using one MAP iteration on a per utterance basis; the means of this GMM UBM are then stacked to form a GMM supervector. The means in the supervector are then weighted appropriately using the inverse covariance matrices and mixture weights from the GMM UBM, see [10].

NAP is applied to the GMM supervectors to reduce session effects. NAP uses a linear projection, $\mathcal{P}$, applied to the SVM expansion space, $\mathbf{b}(\mathbf{x})$, to obtain a session compensated vector, $\mathcal{P}\mathbf{b}(\mathbf{x})$. The effect of NAP is to excise dimensions from the SVM expansion that are related to channel and session variability [3].

Training for the SVM GSV is performed by constructing a set of pseudo-impostors from the LDC Fisher corpora to form a *background*. This background is labeled as $-1$ for SVM training, and enrollment data for a speaker was labeled as $+1$. Training is then performed using SVMTorch using pre-computed kernel inner products to reduce memory requirements.

### 2.5. SVM GLDS NAP

The SVM GLDS system uses the polynomial-based sequence kernel described in [11]. A degree 3 basis of monomials is used for the SVM expansion.

The SVM GLDS system uses a novel front end combination. One front end is based upon MFCCs, and the other is based on LPCCs as described in Section 2.1. Speaker models and scores are calculated independently for both feature sets, and the resulting scores are fused with a linear combination.

The novelty of the front end combination is based on the use of NAP for session compensation. Before NAP, LPCC features were found to have significantly higher error rates on common NIST cross-channel tasks. By applying NAP to the SVM GLDS LPCC system, error rates are decreased dramatically and this enabled beneficial fusion with an SVM GLDS MFCC system.

NAP projection is trained on the Switchboard 2 phases 1-5 corpora using session variability as the nuisance variable [3]. The SVM system also requires a set of pseudo-impostors to use as a background for training speaker models; this background is taken from the LDC Fisher corpora.

## 3. HIGH-LEVEL SYSTEMS

As mentioned in the introduction, one of our main goals was to minimize system complexity through reduction of high-level systems. With this in mind, we explored fusion of multiple high-level systems—pitch-based, phone-based, word-based—and found that focusing on the output of a single high-level word system proved the most powerful in terms of fusion performance and multiple system possibilities. We used the BBN Byblos English conversational telephone speech system [12] and its by-products for our high-level speaker recognition system. From this one high-level tokenizer, we were able to produce five different speaker recognition systems which we detail in the following sections.

### 3.1. Byblos processing

We extracted word lattices and MLLR parameters using BBN's Byblos 1xRT recognizer trained with 2000+ hours of telephone speech. Audio files are first segmented into chunks of 15 seconds or less using a two-class HMM (speech/non-speech) trained on a small selection (approx. four hours) of Switchboard II and Fisher data. Word lattices are generated for each segment using Byblos (with SCTM + VTLN and HLDA adaptation). MLLR parameters are obtained after the un-adapted decode pass of the recognizer.

### 3.2. SVM MLLR NAP

We used the MLLR transforms from the BBN Byblos STT recognizer as features for an SVM-based speaker recognition system. The approach we took is based on the work described in [13] with a number of differences:

- We use two gender-independent regression classes and a global transform for features. The final dimensionality of each feature vector is 10980.
- We apply 0-1 normalization to the each feature vector using statistics derived from the background model.
- We apply NAP as described in [3].

During testing T-norm is applied.

### 3.3. SVM word and LM word

Expected counts of $n$-grams from the lattices produced from the adaptive decode pass of Byblos were calculated using SRI's language modeling tool [14]. These expected counts were used for both the SVM word and N-gram word systems.

The SVM word system uses a kernel for comparing conversation sides based upon methods from information retrieval. Sequences of tokens are converted to a vector of probabilities of occurrences of terms and co-occurrences of terms (bag of unigram and bag of bigrams). This method was first used in the NIST SRE 2003 evaluation and is documented in [15].

The SVM uses a weighted linear kernel. Individual entries in the vector of probabilities are weighted by $\log(1/p_{\mathrm{bkg}}(t_i)) + 1$, where $p_{\mathrm{bkg}}(t_i)$ is the probability of an $n$-gram over all conversations in the background. The background set for training the SVM is derived from the Fisher corpora. SVM training was performed using SVM Light. Verification is performed using a linear kernel with the target speaker model.

For standard language modeling scoring, LM word, expected counts are used to compute joint probabilities for a speaker model and a background using standard $n$-gram modeling. These probabilities are then used to compute the cross perplexity between the test message and the target and background models. The final score is the ratio of the target model score and the background score. ZT-norm was applied to these scores using models and non-target messages trained from NIST SRE 04 data set.

### 3.4. BT word

We utilize models with a binary-decision tree (BT) structure to describe token sequences generated by the speakers in terms of the STT transcripts. In essence, the BT models can be viewed as variable-length N-grams with trainable context clustering and they were applied in speaker and language recognition previously [16, 17].

In this evaluation, the STT transcripts of the speaker speech were used to generate sequences of tokens with an inventory defined as the 512 top frequent words plus an additional "other" class representing the remaining STT vocabulary. The tree structures in this evaluation were trained using a fast flip-flop algorithm to minimize the prediction entropy in terminal nodes as described in [18]. The flip-flop sped up the search by a factor of 30, as compared to the previous search algorithm, and allowed for building models with the relatively sizable token vocabulary of 513 in an efficient manner.

First, on data from background speakers, a common BT model was created resulting in a BT with about 15k terminal nodes using up to 2 predictors (i.e. exploiting a context of 3 words at a time). Subsequently, individual target speaker models were created using an adaptive BT training algorithm from the common BT model as described in [16].

The probability of a token $a_t$ in a sequence generated by the STT tokenizer and given a speaker hypothesis $S_j$, is retrieved from the corresponding BT model in a way described above (traversing the tree). In addition, a recursive parental-node smoothing is applied to the probability as described in [16]. The resulting BT score is $S(a) = \sum_t \log p_{BT}(a_t|\mathrm{Pred}(a_t))/T$, where $a = a_1, ..., a_T$ is the token sequence. A C-norm followed by the T-Norm standardization is applied to the scores. The C-norm is based on an automatic gender-dependent 5-channel detector.

### 3.5. SVM word duration

The word duration system models the expected duration of phones in words and is based on the work in [19]. Each utterance is represented as a feature vector of approximately 56,000 phone durations in word contexts (those seen in our background training set, an approximately 3,000 utterance subset of the Fisher Corpus).

During training, vectors from a target speaker are used in conjunction with vectors for a background model to train an SVM. We apply a relative expected duration kernel, normalizing by the expected phone duration of each phone (in word context) from our background model.

Each message model pair is scored as a weighted inner product between model support vectors and the test message. T-norm is then optionally applied to the resulting scores using models trained from NIST SRE 04.

## 4. FUSION

The scores from the systems are fused with a perceptron classifier using LNKnet. Development testing and fusion parameters are obtained using the NIST SRE 05 test sets. The perceptron architecture chosen has classifier scores fed to input nodes, no hidden layers, and two output nodes. Input values to the perceptron are normalized to zero mean and unit standard deviation using parameters derived from the training data. The perceptron weights are trained using the entire development data with a mean-squared error criteria. The classifier corresponding to the number of training conversations is then used to fuse scores from systems. The fusion classifier is trained to minimize the DCF by using prior probability for the target class in training and testing set to 0.09 corresponding to the costs and priors $(C_{\mathrm{miss}} * P_{\mathrm{tgt}}/(C_{\mathrm{miss}} * P_{\mathrm{tgt}} + C_{\mathrm{fa}} * (1 - P_{\mathrm{tgt}})))$. The score for the test file is then remapped to the application prior of 0.01. The minDCF threshold from cross-validation experiments on the development data are used to make hard decision for the submissions.

## 5. EXPERIMENTS

### 5.1. Experimental Setup

Experiments were performed on the NIST 2006 SRE data set. Enrollment/verification methodology and the evaluation criterion (minDCF) were based on the NIST SRE evaluation plan [20]. Both the 1 and 8 conversation enrollment 4-wire (4w) tasks, referred to as 1c and 8c respectively, were considered. Verification was performed on 1 conversation 4w data. All systems as described in Sections 2 and 3 used background, T-Norm, and Z-Norm sets from the LDC Fisher and Switchboard corpora. Fusion was trained from scores on the NIST SRE 2005 evaluation set. Results were obtained for both the English only task (ENG) and for all trials (ALL) which includes speakers that enroll/verify in different languages.

### 5.2. Results

We break out system results for the 8c enrollment task in Figure 1. The figure clearly shows the low error rates of the spectral only systems highlighted in the left portion of the figure. The figure also illustrates the varying performance of different high-level systems. The SVM MLLR approach performs the best of all high-level systems; it can be viewed as a hybrid approach that uses both high-level word and low-level spectral information.

The fusion results are broken out according to three categories. Fuse all is a fusion of all systems. Fuse best is a fusion of a subset of the systems with the criteria of finding the best performing. Fuse cep is a fusion of the cepstral only systems. Note that "fuse cep" has excellent performance and is one of the main performance drivers. The high-level systems add the additional performance gains to obtain the best fusion results.

Figure 2 breaks out the computational complexity of various systems when they are run independently. The computational complexity is approximate and reflects the real-time factor from timing tests of the various systems on a standard task—lower numbers are better. The figure shows 3 basic clusters. For low computation complexity and low error rates, cepstral systems perform very well. In the middle of the plot, for higher computational complexity, various high-level systems can be implemented. These results can be fused to show the contrasting situations of best cesptral and best overall.

As a final summary of our results, Table 1 shows the performance of our best combination of systems for all systems and cepstral only systems. This table shows excellent progress from prior NIST data sets and substantiates our claim that this system is a high-performance system.

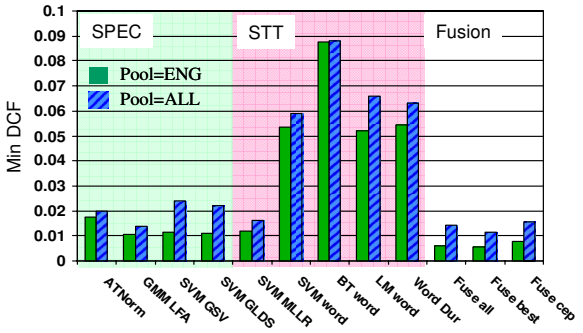


**Fig. 1**. Breakout of different systems for the NIST SRE 8c task

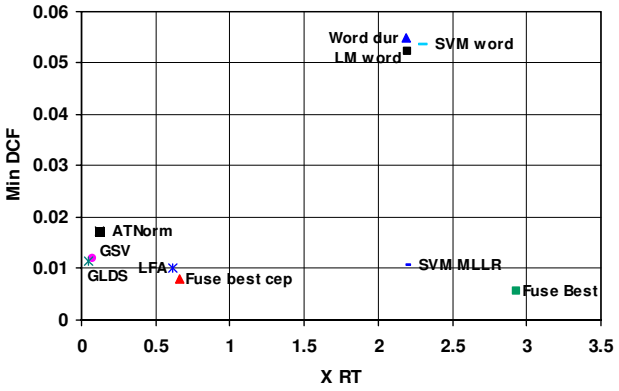


**Fig. 2**. Computational complexity (RT=real-time factor) versus DCF for various systems on the 8c English-only task

**Table 1**. Summary of best performance on the NIST SRE 2006 task (post-evaluation)

| | Eng | | All | |
|---|---|---|---|---|
| Task | EER (%) | minDCF (×100) | EER (%) | minDCF (×100) |
| 1c - fuse best | 2.7 | 1.4 | 4.4 | 2.2 |
| 8c - fuse best | 1.5 | 0.55 | 2.6 | 1.1 |
| 1c - fuse cep | 3.4 | 1.7 | 4.7 | 2.3 |
| 8c - fuse cep | 2.1 | 0.76 | 3.0 | 1.4 |

## 6. CONCLUSIONS

We have demonstrated a novel synthesis of techniques for speaker recognition using discriminative techniques, channel compensation, and high-level speaker recognition. By limiting high-level tokenization to one system and using multiple classifiers, we have achieved excellent speaker recognition performance. Tradeoffs between complexity and recognition accuracy were explored.

## 7. REFERENCES


[1] W. M. Campbell, D. A. Reynolds, and J. P. Campbell, "Fusing discriminative and generative methods for speaker recogntion: Experiments on Switchboard and NFI/TNO field data," in *Proc. Odyssey Workshop*, 2004, pp. 41–44.

[2] P. Kenny and P. Dumouchel, "Experiments in speaker verification using factor analysis likelihood ratios," in *Proc. Odyssey04*, 2004, pp. 219–226.

[3] Alex Solomonoff, W. M. Campbell, and I. Boardman, "Advances in channel compensation for SVM speaker recognition," in *Proceedings of ICASSP*, 2005.

[4] 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," in *Proceedings of ICASSP*, 2003, pp. IV–784–IV–787.

[5] Douglas A. Reynolds, T. F. Quatieri, and R. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[6] D. E. Sturim and D. A. Reynolds, "Speaker adaptive cohort selection for Tnorm in text-independent speaker verification," in *Proceedings of ICASSP*, 2005.

[7] Robbie Vogt, Brendan Baker, and Sridha Sriharan, "Modelling session variability in text-independent speaker verification," in *Proc. Interspeech*, 2005, pp. 3117–3120.

[8] P. Kenny, G. Boulianne, and P. Dumouchel, "Eigenvoice modeling with sparse training data," *IEEE Trans. Speech and Audio Processing*, vol. 13, no. 3, pp. 345–354, 2005.

[9] M. Tipping and C. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.

[10] W. M. Campbell, D. E. Sturim, D. A. Reynolds, and A. Solomonoff, "SVM based speaker verification using a GMM supervector kernel and nap variability compensation," in *Proceedings of ICASSP*, 2006, pp. I–97–I–100.

[11] W. M. Campbell, "Generalized linear discriminant sequence kernels for speaker recognition," in *Proceedings of ICASSP*, 2002, pp. 161–164.

[12] S. Matsoukas, R. Prasad, B. Xiang, L. Nguyen, and R. Schwartz, "The RT04 BBN 1xRT recognition systems for English CTS and BN," in *Proceedings of the Rich Transcription Workshop*, 2004.

[13] A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, and A. Venkataraman, "MLLR transforms as features in speaker recognition," in *Proc. Interspeech*, 2005, pp. 2425–2428.

[14] Andreas Stolcke, "SRILM–an extensible language modeling toolkit," in *International Conference on Spoken Language Processing*, 2002, pp. 901–904.

[15] W. M. Campbell, J. P. Campbell, D. A. Reynolds, D. A. Jones, and T. R. Leek, "High-level speaker verification with support vector machines," in *Proceedings of ICASSP*, 2004, pp. I–73–76.

[16] Jiri Navrátil, Qin Jin, Walter D. Andrews, and Joseph P. Campbell, "Phonetic speaker recognition using maximum-likelihood binary-decision tree models," in *Proceedings of ICASSP*, 2003, pp. IV–796–IV–799.

[17] J. Navrátil, "Automatic language identification," in *Multilingual Speech Processing*, T. Schultz and K. Kirchhoff, Eds., pp. 233–272. Academic Press, 2006.

[18] J. Navrátil, "Recent advances in phonotactic language recognition using binary decisions trees," in *Proc. Interspeech*, 2006.

[19] V. R. R. Gadde, S. Kajarekar, E. Shriberg, K. Sonmez, A. Stolcke, and A. Venkataraman, "Modeling duration patterns for speaker recognition," in *Proc. Eurospeech*, 2003, pp. 2017–2020.

[20] "The NIST year 2006 speaker recognition evaluation plan," http://www.nist.gov/speech/tests/spk/2006/index.htm, 2005.