HIGH-LEVEL SPEAKER VERIFICATION WITH SUPPORT VECTOR MACHINES[*]

W. M. Campbell, J. P. Campbell, D. A. Reynolds, D. A. Jones, and T. R. Leek

MIT Lincoln Laboratory
Lexington, MA 02420
E-mail: wcampbell@ll.mit.edu

ABSTRACT

Recently, high-level features such as word idiolect, pronunciation, phone usage, prosody, etc., have been successfully used in speaker verification. The benefit of these features was demonstrated in the NIST extended data task for speaker verification; with enough conversational data, a recognition system can become "familiar" with a speaker and achieve excellent accuracy. Typically, high-level-feature recognition systems produce a sequence of symbols from the acoustic signal and then perform recognition using the frequency and co-occurrence of symbols. We propose the use of support vector machines for performing the speaker verification task from these symbol frequencies. Support vector machines have been applied to text classification problems with much success. A potential difficulty in applying these methods is that standard text classification methods tend to "smooth" frequencies which could potentially degrade speaker verification. We derive a new kernel based upon standard log likelihood ratio scoring to address limitations of text classification methods. We show that our methods achieve significant gains over standard methods for processing high-level features.

1. INTRODUCTION

We consider the problem of text-independent speaker verification. That is, given a claim of identity and a voice sample (whose text content is *a priori* unknown), determine if the claim is correct or incorrect. Traditional approaches to speaker verification use spectral content (e.g., cepstral coefficients) of the speech and Gaussian mixture models to perform recognition [1].

An exciting area of recent development pioneered by Doddington [2] is the use of "high-level" features for speaker verification. In Doddington's idiolect work, word N-grams from conversations were used to characterize a particular speaker. More recent systems have used a variety of approaches involving phone sequences, pronunciation modeling and prosody. For this paper, we concentrate on the use of phone and word sequences. The processing for this type of system uses acoustic information to obtain sequences of phones or words for a given conversation and then discards the acoustic waveform. Thus, processing is done at the level of terms (symbols) consisting of, for example, phones or phone N-grams.

This paper is organized as follows. In Section 2, we discuss the NIST extended data speaker verification task. In Section 3, we discuss methods for obtaining phone and word sequences. Section 4 shows the structure of the SVM speaker verification system. Section 5 discusses how we construct a new likelihood ratio based kernel for speaker verification using term weighting techniques for document classification. Finally, Section 6 shows the significant improvements of the new methods over traditional N-gram methods for phone and word sequences.

2. THE NIST EXTENDED DATA TASK

Speaker verification experiments were performed based upon the NIST 2003 extended data task. The corpus was a combination of phases 2 and 3 of the Switchboard-2 corpora. Each training utterance in the NIST extended data corpus consisted of a conversation side that was nominally of length 5 minutes (approximately 2.5 minutes of speech) recorded over a land-line telephone.

For training and testing, a jackknife approach was used to increase the number of tests. The data was divided into 10 splits with disjoint speakers. When processing a split, the remaining splits were used to construct a "background" model. For example, when conducting tests on split 1, splits 2-10 could be used to construct a background.

Speaker models were trained using 1, 2, 4, 8, or 16 conversation sides to examine the situation where the system could become more "familiar" with the individual. A large number of speakers and tests were available; for instance, for 8 conversation training, 739 distinct target speakers were used and $11,171$ true trials and $17,736$ false trials were performed. For additional information on the training/testing structure we refer to the NIST extended data task description, see http://www.nist.gov/speech/tests/spk/2003/.

3. HIGH-LEVEL FEATURE EXTRACTION

In this work, we used both word and phone sequences as the base features to characterize a speaker.

Word Sequence Extraction. Transcripts for conversations were provided by NIST. ASR transcripts were generated at NIST using a real-time version of BBN's Byblos system. Potentially, lower speaker verification error rates could be obtained with better transcripts.

Phone Sequence Extraction. Phone sequence extraction for the speaker verification process was performed using the phone recognition system from the PPRLM language identification system [3]. PPRLM uses a MFCC front end with delta coefficients. Each phone is modeled in a gender-dependent

[*] This work was sponsored by the United States Government Technical Support Working Group under Air Force Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

context-independent (monophone) manner using a three-state hidden Markov model (HMM). Phone recognition is performed with a Viterbi search using a fully connected null-grammar network of monophones.

The phone recognition system encompassed multiple languages—English (EG), German (GE), Japanese (JA), Mandarin (MA), and Spanish (SP). In earlier phonetic speaker recognition work [4], it was found that these multiple sequences were useful for improving accuracy. The phone recognizers were trained using the OGI multilanguage corpus which had been hand labeled by native speakers.

After a "raw" phone sequence was obtained from PPRLM, additional processing was performed to increase robustness. First, speech activity detection marks were used to eliminate phone segments where no speech or crosstalk was present. Second, silence labels of duration greater than 0.5 seconds were replaced by "end start" pairs. Third, extraneous silence was removed at the beginning and end of the resulting segments. Finally, all phones with short duration were removed (less than 3 frames).

4. SVMS FOR HIGH-LEVEL SPEAKER RECOGNITION

4.1. Support Vector Machines

A support vector machine (SVM) is a two-class classifier constructed from sums of a kernel function $K(\cdot,\cdot)$,

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i t_i K(\mathbf{x}, \mathbf{x}_i) + b; \tag{1}$$

where the t_i are targets, and $\sum_{i=1}^{N} \alpha_i t_i = 0$. The vectors $\mathbf{x}_i$ are support vectors and obtained from the training set by an optimization process [5]. The target values are either 1 or -1 depending upon whether the corresponding support vector is in class 1 or class 2. For classification, a class decision is based upon whether the value, $f(\mathbf{x})$, is above or below a threshold.

The kernel $K(\cdot,\cdot)$ is constrained to have certain properties (the Mercer condition), so that $K(\cdot,\cdot)$ can be expressed as

$$K(\mathbf{x},\mathbf{y}) = \mathbf{b}(\mathbf{x})^t \mathbf{b}(\mathbf{y}) \tag{2}$$

where $\mathbf{b}(\mathbf{x})$ is a mapping from the input space (where $\mathbf{x}$ lives) to a possibly infinite dimensional space.

4.2. Phone SVM System

Our system for speaker verification using phone sequences is shown in Figure 1. The scenario for its usage is as follows. An individual makes a claim of identity. The system then retrieves the SVM models of the claimed identity for each of the languages in the system. A test utterance is then collected. A phone sequence is derived using each of the language phone recognizers and then post-processing is performed on the sequence as discussed in Section 3. The phone sequence is then vectorized by computing frequencies of N-grams—this process will be discussed in Section 5. We call this term calculation since we compute term types (unigram, bigram, etc.), term probabilities and weightings in this step [6]. This vector is then processed by a SVM using the speaker's model in the appropriate language, and a score per language is produced. These scores are then linearly fused to produce a final score. The final score is compared to a threshold and a reject or accept decision is made based upon whether the score was below or above the threshold, respectively.

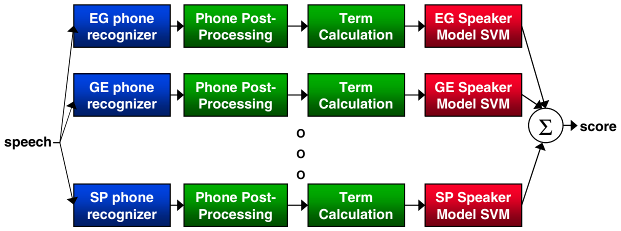

Fig. 1. Phonetic speaker verification using SVMs.

4.3. Word SVM System

The word-based SVM speaker recognition system is the same structure as one "branch" of the phone system shown in Figure 1. The input acoustic test utterance is converted to an English word sequence. The word sequence is then converted to a vector of probabilities (term calculation). This vector is then put into an SVM of the form (1) along with support vectors (from training) representing the speaker model. The resulting output score is compared to a threshold and an accept or reject decision is made.

4.4. Training

Training for the system in Figure 1 is based upon the structure of the NIST extended data corpus. We treat each conversation side in the corpus as a "document." From each of these conversation sides we derive a single (sparse) vector of weighted probabilities. To train a model for a given speaker, we use a one-versus-all strategy. The speaker's conversation sides are trained to a SVM target value of $+1$. The conversations sides not in the current split are used as a background and trained to an SVM target value of -1. Note that this strategy ensures that speakers that are used as impostors are "unseen" in the training data.

5. KERNEL CONSTRUCTION

Our first step of kernel construction is the selection of probabilities to describe the symbol sequence. We follow the work of [2, 4] and use a "bag of N-grams" approach. For a symbol sequence, we produce N-grams by the standard transformation of the stream; e.g., for bigrams, the sequence of symbols from a conversation side, t_1, t_2, ..., t_n, is transformed to the sequence of bigrams of symbols $t_1\_t_2$, $t_2\_t_3$, ..., $t_{n-1}\_t_n$. We then find probabilities of N-grams with N fixed. That is, suppose we are considering unigrams and bigrams of symbols, and the unique unigrams and bigrams of the corpus are designated d_1, ..., d_M and $d_1\_d_1$, ... $d_M\_d_M$, respectively; then we calculate probabilities

$$\begin{aligned} p(d_i) &= \frac{\#(t_k = d_i)}{\sum_l \#(t_k = d_l)} \\ p(d_i\_d_j) &= \frac{\#(t_k\_t_{k+1} = d_i\_d_j)}{\sum_{l,m} \#(t_k\_t_{k+1} = d_l\_d_m)} \end{aligned} \tag{3}$$

where $\#(t_k = d_i)$ indicates the number of symbols in the conversation side equal to d_i, and an analogous definition is used for bigrams. These probabilities then become entries in a vector $\mathbf{v}$ describing the conversation side

$$\mathbf{v} = \begin{bmatrix} p(d_1) & \ldots & p(d_M) & p(d_1\_d_1) & \ldots & p(d_M\_d_M) \end{bmatrix}^t. \tag{4}$$

In general, the vector $\mathbf{v}$ will be sparse since the conversation side will not contain all potential unigrams, bigrams, etc.

A second step of kernel construction is the selection of the term weighting for the entries of the vector $\mathbf{v}$ in (4) and the normalization of the resulting vector. By term weighting, we mean that for each entry, v_i, of the vector $\mathbf{v}$, we multiply by a factor w_i for that entry determined from the background. In the information retrieval literature, this is referred to as the "collection" component of the weighting [6]. We tried two distinct approaches for term weighting.

TFIDF weighting. The first is based upon the standard term-frequency inverse-document-frequency (TFIDF) weighting approach [6]. From the background section of the corpus, we compute the frequency of a particular N-gram using conversation sides as the item analogous to a document. I.e., if we let $DF(t_i)$ be the number of conversation sides where a particular N-gram, t_i, is observed, then our resulting term-weighted vector has entries

$$v_i \log\left(\frac{\#\text{ of conversation sides in background}}{DF(t_i)}\right). \tag{5}$$

We follow the weighting in (5) by a normalization of the vector to unit length $\mathbf{x} \mapsto \mathbf{x}/\|\mathbf{x}\|_2$.

Log likelihood ratio weighting. An alternate method of term weighting may be derived using the following strategy. Suppose that we have two conversation sides from speakers, spk_1 and spk_2. Further suppose that the sequence of N-grams (for fixed N) in each conversation side is t_1,t_2, ..., t_n and u_1, u_2, ..., u_m respectively. We denote the unique set of N-grams in the corpus as d_1, ..., d_M. We can build a "model" based upon the conversation sides for each speaker consisting of the probability of N-grams, $p(d_i|\text{spk}_j)$. We then compute the likelihood ratio of the first conversation side as is standard in verification problems [1]; a linearization of the likelihood ratio computation will serve as the kernel. Proceeding,

$$\frac{p(t_1, t_2, \ldots, t_n|\text{spk}_2)}{p(t_1, \ldots, t_n|\text{background})} = \prod_{i=1}^{n} \frac{p(t_i|\text{spk}_2)}{p(t_i|\text{background})} \tag{6}$$

where we have made the assumption that the probabilities are independent. We then consider the log of the likelihood ratio normalized by the number of observations,

$$\begin{aligned} \text{score} &= \frac{1}{n}\sum_{i=1}^{n} \log\left(\frac{p(t_i|\text{spk}_2)}{p(t_i|\text{background})}\right) \\ &= \sum_{j=1}^{M} \frac{\#(t_i = d_j)}{n} \log\left(\frac{p(d_j|\text{spk}_2)}{p(d_j|\text{background})}\right) \qquad (7) \\ &= \sum_{j=1}^{M} p(d_j|\text{spk}_1) \log\left(\frac{p(d_j|\text{spk}_2)}{p(d_j|\text{background})}\right). \end{aligned}$$

If we now "linearize" the log function in (7) by using $\log(x) \approx x - 1$ (Taylor series expansion around $x = 1$), we get

$$\begin{aligned} \text{score} &\approx \sum_{j=1}^{M} p(d_j|\text{spk}_1) \frac{p(d_j|\text{spk}_2)}{p(d_j|\text{background})} - \sum_{j=1}^{M} p(d_j|\text{spk}_1) \\ &= \sum_{j=1}^{M} p(d_j|\text{spk}_1) \frac{p(d_j|\text{spk}_2)}{p(d_j|\text{background})} - 1 \\ &= \sum_{j=1}^{M} \frac{p(d_j|\text{spk}_1)}{\sqrt{p(d_j|\text{background})}} \frac{p(d_j|\text{spk}_2)}{\sqrt{p(d_j|\text{background})}} - 1 \end{aligned} \tag{8}$$

Thus, (8) suggests we use a term weighting given by $1/\sqrt{p(d_j|\text{background})}$. This strategy for constructing a kernel is part of a general process of finding kernels based upon training on one instance and testing upon another instance [7].

6. EXPERIMENTS

Experiments were performed using the NIST extended data task protocol using the "v1" lists (which encompass the entire Switchboard 2 phases 2 and 3 corpora). Tests were performed for 1, 2, 4, 8, and 16 training conversations. Scoring was performed using the SVM system shown in Figure 1 and the word system described in Section 4.3. Both word and phone sequences were vectorized as unigram and bigram probabilities (4). Both the standard TFIDF term weighting (5) and the log likelihood ratio (TFLLR) term weighting (derived in (8)) methods were used. We note that when a term did not appear in the background, it was ignored in training and scoring. A linear kernel was used $\mathbf{x} \cdot \mathbf{y} + 1$ to compare the vectors of weighted terms. Training was performed using the SVMTorch [5] with a margin and error tradeoff of $c = 1$.

Results were compared via equal error rates (EERs) and DET curves. Table 1 shows results for two different weightings, TFIDF (5) and TFLLR (8), using English phones only and 8 training conversations. The table shows that the new TFLLR weighting method is more effective. This may be due to the fact the IDF is too "smooth"; e.g., for unigrams, the IDF is approximately 1 since a unigram almost always appears in a given 5 minute conversation. Also, alternate methods of calculating the TF component of TFIDF may yield gains; we will explore this in future work.

We next considered the effect on performance of the language of the phone stream for the 8 conversation training case. Figure 2 shows a DET plot with results corresponding to the 5 language phone streams. The best performing system in the figure is an equal fusion of all scores from the SVM outputs for each language and has an EER of 3.5%; other fusion weightings were not explored in detail. As expected, the best performing language is English. Note, though, as we indicated in Section 3 that other languages do provide significant speaker verification information.

EERs were then found for different training conversation lengths for the fused phone SVM system. Results for 1, 2, 4, 8, and 16 conversations were 13.6%, 8.6%, 5.3%, 3.5% and 2.5%, respectively. Note that even for 1 training conversation, the SVM system provides significant speaker discrimination.

Figure 3 shows DET plots comparing the performance of the standard N-gram log likelihood ratio method [4] to our new SVM method using the TFLLR weighting. We show N-gram log likelihood results based on both bigrams and trigrams; in addition, a slightly more complex model involving discounting of probabilities is used. One can see the dramatic reduction in error, especially apparent for low false alarm probabilities. The EERs of the standard system are 8.75% (trigrams) and 10.4% (bigrams), whereas our new SVM system produces an EER of 3.5%; thus, we have reduced the error rate by 60%.

Table 1. Comparison of different term weighting strategies, English only scores, 8 conversation training.

Term Weighting Method	EER
TFIDF	7.4%
TFLLR	5.2%

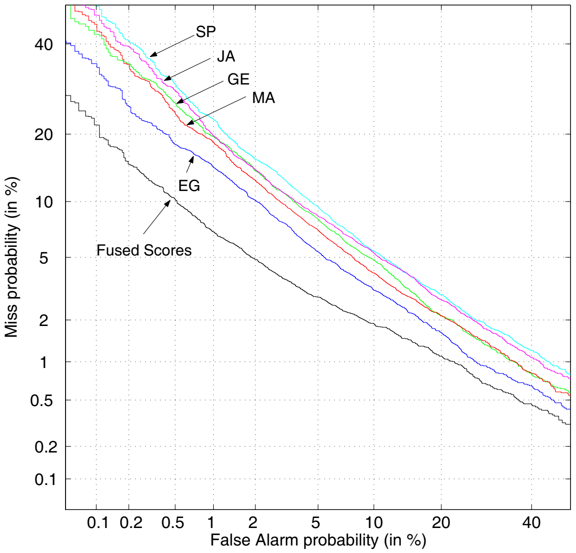

Fig. 2. DET plot for the 8 conversation training case for the SVM approach with varying languages, TFLLR weighting, and phone unigrams and bigrams. The plot shows in order of increasing EER—fused scores, EG, MA, GE, JA, SP.

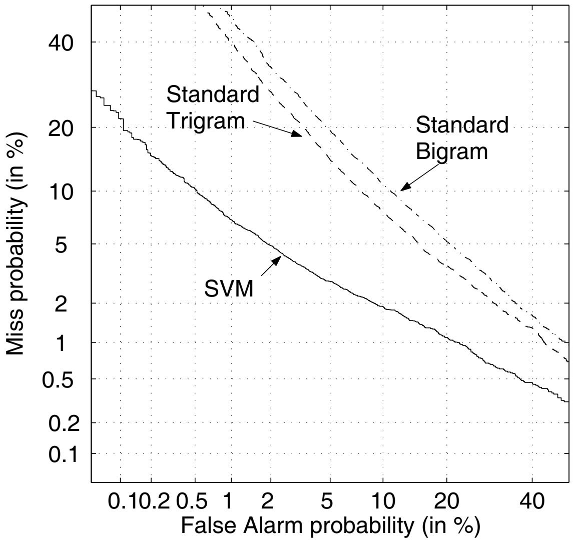

Fig. 3. DET plot for 8 conversation training comparing the phone SVM approach (solid line) to the standard N-gram log likelihood ratio approach using phone bigrams (dash-dot line) and phone trigrams (dashed line).

Figure 4 shows the SVM system applied to the word stream. In this case, the SVM shows significant improvement only at low false alarm rates. A significant factor impacting performance in the word case is the high sparsity of the document vectors. To achieve high performance, both the log likelihood system and SVM system used discounting value of 1 in scoring, see [2]. Discounting with a value of 1 is equivalent to using 0 (not present) or 1 (present) for the term frequency.

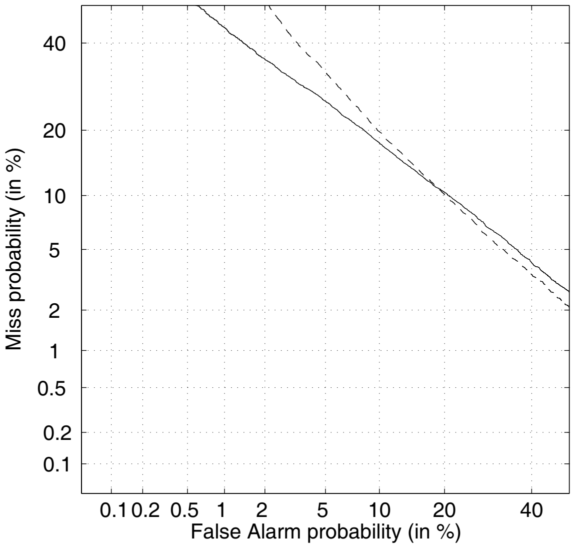

Fig. 4. DET plot for 8 conversation training showing a comparison of the SVM approach (solid line) to the standard N-gram log likelihood ratio approach (dashed line) using word bigrams.

7. CONCLUSIONS

An exciting new application of SVMs to speaker verification was shown. By computing frequencies of phones and words in conversations and using this in an SVM, speaker discrimination was performed. A new kernel was introduced that reduced error rates up to 60% over standard N-gram log likelihood techniques.

8. REFERENCES

[1] 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.

[2] G. Doddington, "Speaker recognition based on idiolectal differences between speakers," in *Proceedings of Eurospeech*, 2001, pp. 2521–2524.

[3] M. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Trans. Speech and Audio Processing*, vol. 4, no. 1, pp. 31–44, 1996.

[4] Walter D. Andrews, Mary A. Kohler, Joseph P. Campbell, John J. Godfrey, and Jaime Hernández-Cordero, "Gender-dependent phonetic refraction for speaker recognition," in *Proceedings of the International Conference on Acoustics Speech and Signal Processing*, 2002, pp. I149–I153.

[5] Ronan Collobert and Samy Bengio, "SVMTorch: Support vector machines for large-scale regression problems," *Journal of Machine Learning Research*, vol. 1, pp. 143–160, 2001.

[6] Thorsten Joachims, *Learning to Classify Text Using Support Vector Machines*, Kluwer Academic Publishers, 2002.

[7] W. M. Campbell, "Generalized linear discriminant sequence kernels for speaker recognition," in *Proceedings of the International Conference on Acoustics Speech and Signal Processing*, 2002, pp. 161–164.