Applications of Keyword-Constraining in Speaker Recognition

Howard Lei

hlei@icsi.berkeley.edu

July 2, 2007

Contents

1 Introduction

The goal of speaker recognition is to associate a speaker identity to a given speech unit (spoken by a single speaker). This speaker recognition task can be broken down into the text-independent and text-dependent domains. In the text-independent domain, the lexical content of the speech is unknown. That is, the recognition is performed using lexically unconstrained speech from an individual. In the text-dependent domain, the speech is lexically constrained, such that speaker recognition is performed in a highly controlled manner, typically employing only certain words or phrases. The text-constancy assumptions used in text-dependent speaker recognition can increase the accuracy of the task, as it reduces the effect that lexical variability may have when attempting to associate an identity to a given speech unit (text-independent speaker recognition suffers from this lack of consistency). In addition, certain words or phrases may have very high inter-speaker variability in its pronunciation and acoustics, further enhancing the motive for lexical constraining. However, because text-independent speaker recognition systems do not rely on lexical constraints, its usage is also unconstrained in this manner, and the system can be more broadly employed compared to text-dependent systems.

Speaker recognition is a research problem that requires the application of various signal processing, statistical, and machine learning techniques. Since the early to mid 1990s, a standard and effective approach to speaker recognition uses acoustic features capturing characteristics of the spectral content of speech, and using GMMs in a bag-of-frames approach to model the acoustic tendencies of certain speakers [1]. In this speaker recognition approach, a speaker-independent, or background model is trained using acoustic features from a large pool of speech units, while speaker-specific acoustic models are trained using only those speech units spoken by particular speakers. These speaker-specific models are obtained via adaptation from the background model.

Since 2001, there has been more attention paid to using high-level features, such as words [2] and phonetic sequences [3] [4], for speaker recognition. The purpose of using words and phones is to attempt to capture idiolect-based tendencies of speakers, and to use the inter-speaker variability of such tendencies for speaker discriminative purposes. Such high-level features, whose data is more sparse compared to low-level acoustic features, have been shown to provide good speaker discriminative power.

The concept of using keywords is another advancement in speaker recognition. I use keywords to refer to any particular word N-gram, typically of orders 1 and 2. The concept of keyword-constraining is that of utilizing only portions of speech where certain keywords are spoken. While discarding much of the speech data, the use of keyword-constraining does introduce text-dependence in a text-independent domain, and focuses speaker modeling power on more informative regions of speech. Thus, if certain keywords have high inter-speaker variability of pronunciation, a system can be built using only portions of speech containing those particular keywords, while reducing the undesirable effect that lexical variability may have. Sturim et al., in 2002, introduced a system utilizing GMMs trained on keyword-constrained bag-of-frames acoustic feature sequences [5]. The best keyword-constrained result for this approach exceeded the result for the same acoustic modeling approach without the use of keywords.

One keyword-based system particularly relevant to some of the systems discussed in this paper is the keyword HMM system, introduced in 2005 by Boakye [6]. In this system, HMM speaker models are trained on keyword-constrained MFCC features, [6], where each HMM trained in this manner is referred to as a keyword HMM. Note that this system's approach uses not only keywords, but also time-dependent information among the acoustic features, in contrast to the

bag-of-frames approach relying on GMMs.

In this work, I explore the keyword HMM system, along with three of my own keyword-based speaker recognition systems involving both acoustic and high-level features. One system, which I introduced in [7], uses the means of the Gaussian mixture components of keyword HMMs as features in an SVM classifier. This approach, which is an extension of the keyword HMM approach, is inspired by Campbell et al.'s [8], which uses the Gaussian mixture means of a GMM-based system in an SVM classifier. The next system uses keyword HMMs trained on phone sequence data. This approach, also based on the keyword HMM approach, hypothesizes that the time-dependent relationships among phones in keyword-constrained sequences provide adequate speaker discriminative power. The last system, which I introduced in [9], uses keyword-constrained phone N-gram counts in an SVM classifier. This approach is based on a non keyword-constrained phone N-grams system [4]. This paper will discuss the methodology behind each approach in turn, followed by discussing and comparing the results among the approaches.

2 The keyword HMM system

Because two of my three systems are based on the approach for the basic keyword HMM system, it is worthwhile to give an overview of the system to provide the reader with adequate background. Figure 1 shows the paradigm used to implement this system.

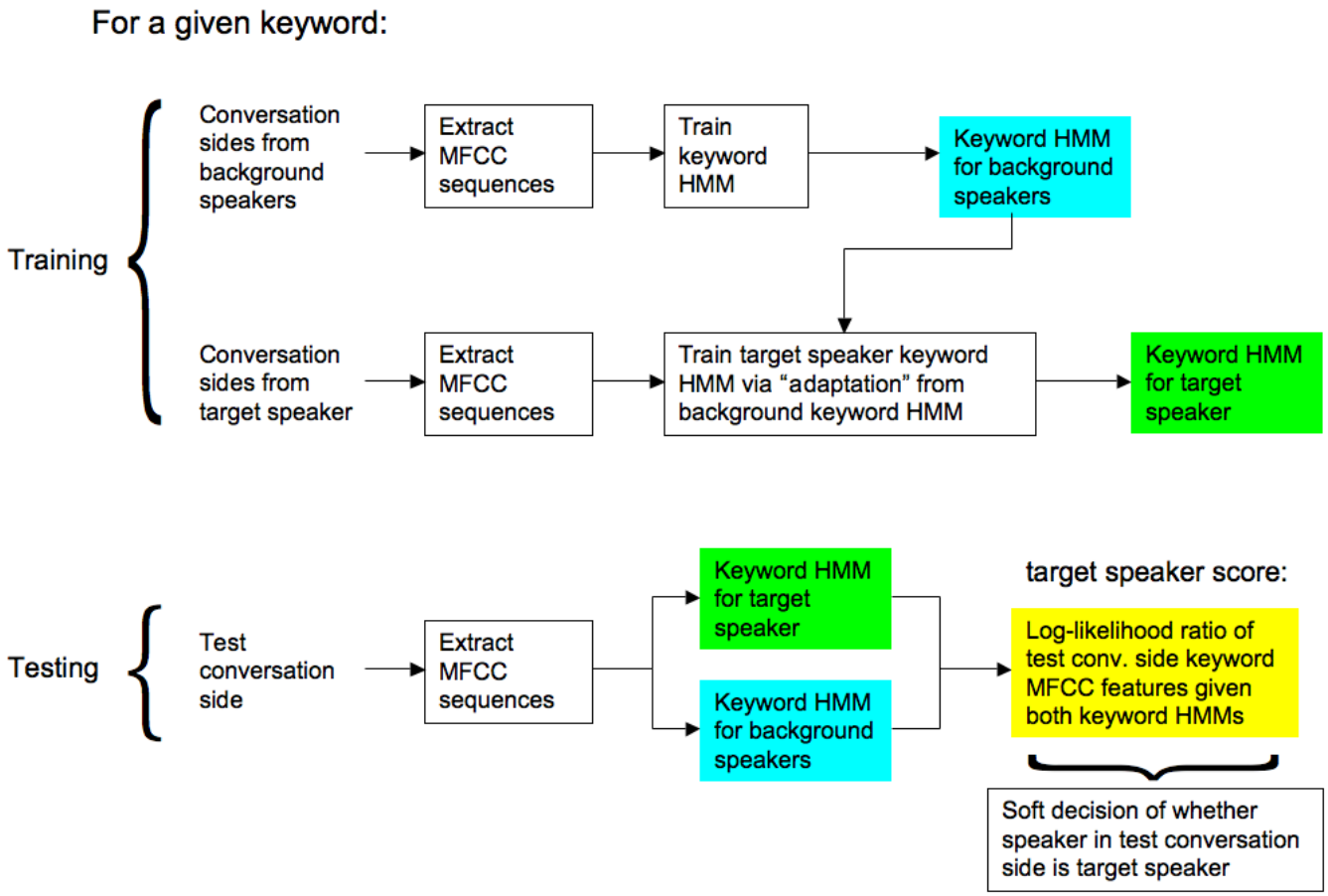


Figure 1: *Speaker recognition paradigm using keyword HMM approach*

In particular, for a given keyword, MFCC feature sequences from a pool of large speech units, or conversation sides (2.5 or 5 minutes of conversational speech from one speaker), are used to train a speaker-independent, or background keyword HMM. Next, MFCC sequences from conversation sides belonging to a particular target speaker are used to train a speaker-specific keyword HMM via adaptation from the corresponding background keyword HMM. Lastly, sequences from a test conversation side are scored against a particular target speaker keyword HMM via the standard log-likelihood ratio to give a soft decision of whether the speaker in the test conversation side is the target speaker.

2.1 Background keyword HMM training

One background keyword HMM is obtained for each keyword using MFCCs sequences with C0-C19 plus deltas (40 dimensions total), extracted every 10 ms with 25 ms frames using the HTK software [10]. The output distribution at each HMM state is a mixture of Gaussian components. Ideally, there should be enough components to represent a wide range of distributions necessary to model the data, and not too many such as to pose a risk for over-training in addition to being computationally expensive. Eight Gaussian mixture components are experimentally chosen to satisfy both criteria. The HMMs are left-to-right with self-loops at each state and no skips [6]. The HMMs begin in the first state and finish in the last state, and the first and last states are non-emitting. The number of states for each keyword HMM is the following [6]:

$$NumStates = min\left(3P, \frac{1}{4}D\right) \tag{1}$$

where P is the average number of phones comprising the keyword (obtained from a dictionary), and D is the median number of MFCC frames for the keyword.

Background keyword HMM parameters are obtained via the Viterbi alignment and EM algorithms using HTK. The parameters are first assigned uniform distributions and updated using the Viterbi alignment algorithm. These updated values then act as initial values for EM training.

2.2 Target speaker keyword HMM training

For each target speaker, an HMM is trained for each keyword, using keyword-constrained MFCC features from eight conversation sides of target speaker data. Eight conversation sides are used to provide sufficient keyword HMM training data, because not all keywords may exist in a single conversation side. Training is done via MAP adaptation from the background keyword HMMs, where target speaker keyword HMM parameters are adapted from the corresponding background keyword HMM. MAP adaptation provides a certain consistency between the background and target speaker HMMs, such that if a keyword does not exist among any target speaker conversation sides (or if there is insufficient target speaker keyword data), the target speaker keyword HMM is the same as the background keyword HMM.

Only the Gaussian mixture means are altered, via HTK, as follows: for state j and Gaussian mixture m [6][10]:

$$\hat{\mu}_{jm} = \frac{N_{jm}}{N_{jm} + \tau}\bar{\mu}_{jm} + \frac{\tau}{N_{jm} + \tau}\mu_{jm} \tag{2}$$

where τ is the weight of a priori knowledge to the adaptation data, N_{jm} is the occupation likelihood of the adaptation data, $\bar{\mu}_{jm}$ is the Gaussian mean of the adaptation data, μ_{jm} is the Gaussian mean of the background keyword HMM, and $\hat{\mu}_{jm}$ is the updated Gaussian mean. Figure 2 illustrates MAP adaptation using MFCC features for a given keyword.

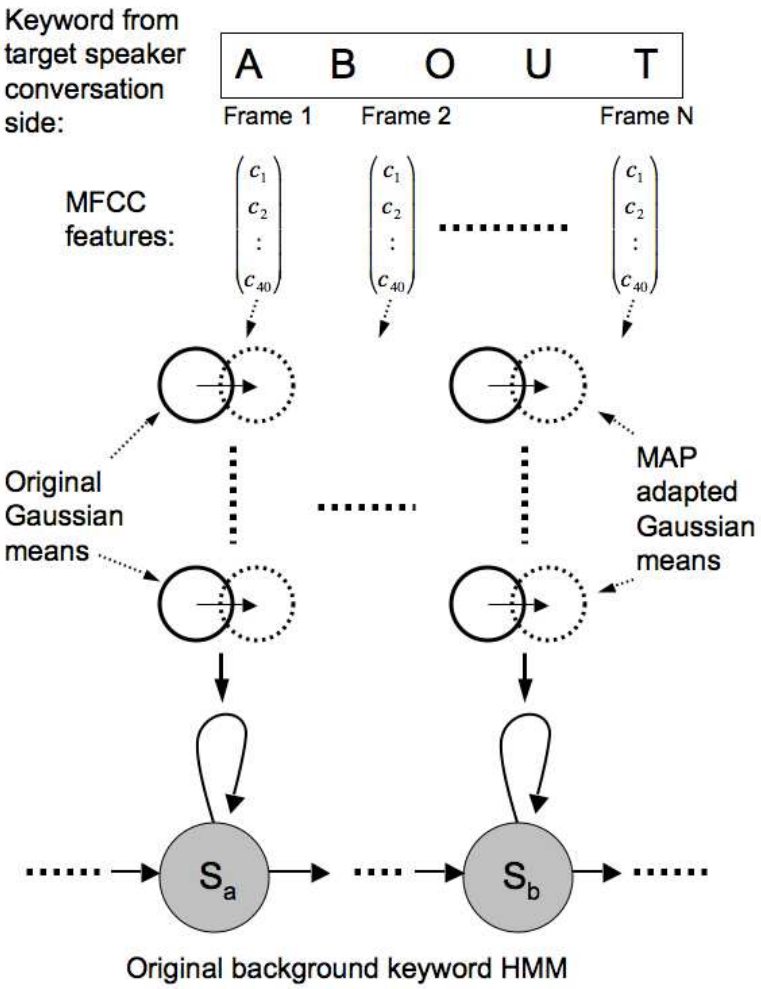


Figure 2: *MAP adaptation of background keyword HMM to create target speaker keyword HMM.*

2.3 Testing and scoring

A sequence of feature vectors $(f_1, ..., f_N)$ belonging to instance i of keyword W in test conversation side t is scored against target speaker model M_{TS} as follows:

$$Score(i, W, t, M_{TS}) = \log\left(\frac{p(f_1, ..., f_N|M_{TS})}{p(f_1, ...f_N|M_{BKG})}\right) \tag{3}$$

where M_{BKG} is the background keyword HMM, and

$$\log(p(f_1, ..., f_N|M)) = \log\left(\sum_x p(f_1, ..., f_N|x, M)p(x|M)\right) \tag{4}$$

where x is the sequence of allowable states.

An overall keyword score for a trial (a test conversation side scored against a target speaker model) is obtained by adding the scores for all instances of the keyword, and dividing by the total number of acoustic feature frames of those keyword instances. A keyword-combined score is obtained by adding the scores for all instances of all keywords, and dividing by the total number of acoustic feature frames. If the speaker in the test conversation side is the same as the target speaker, the trial is known as a true speaker trial; otherwise, the trial is an impostor trial. A scoring threshold is established to separate the true speaker trial scores from the impostor trial scores. For a given threshold, a false accept (FA) occurs when an impostor score is classified as a true speaker score, and a miss (MI) occurs when a true speaker score is classified as an impostor score. The equal error rate (EER) occurs at a threshold at which $FA = MI$. The lower the EER, the better the system.

3 Approach 1: The supervector keyword HMM system

The supervector keyword HMM system is an extension of the keyword HMM system. Instead of computing the log-likelihoods and scoring each test conversation side as in the keyword HMM approach, the MAP adapted Gaussian mixture means of each target speaker keyword HMM are used as features in an SVM classifier. The 40-dimensional Gaussian mixture mean vectors of each component of each state (excluding the first and last states) are concatenated to form a high-dimensional supervector. The supervector concept is introduced by Campbell et al. [8] in a system using GMMs instead of HMMs as statistical models. Figure 3 illustrates this process.

3.1 SVM training with supervectors

As in [8], an SVM classifier is trained for each target speaker. All SVM training is done using the SVM^{light} software package [11], and a linear kernel is used. For each target speaker, the supervector obtained from its keyword HMM acts as the positive SVM training example, supervectors from keyword HMMs trained from example impostor speakers [8] act as negative training examples, while those from keyword HMMs trained using data from single test conversation sides act as SVM test examples. The same MAP adaptation is used to train keyword HMMs for example impostor speakers (using eight conversation sides) and test conversation sides.

The SVM training approach as described above implies that each target speaker SVM is trained with only one positive training example (the supervector from the target speaker keyword HMM). To increase the number of positive training examples, different subsets of the eight conversation sides for a target speaker can be used (in a round-robin) to train a keyword HMM per subset, and supervectors from keyword HMMs from all subsets can be used as positive training examples. Hence, this round-robin training gives as many positive training examples as the number of subsets. Note that this is in contrast to the SVM training approach of [8], which uses one target speaker supervector per conversation side (the possible absence of keyword instances in single conversation sides prevents us from doing the same). Figure 4 illustrates the round-robin training process.

Different weights can be assigned to SVM training errors from positive and negative training examples. Because there are still many more negative training examples than positive training

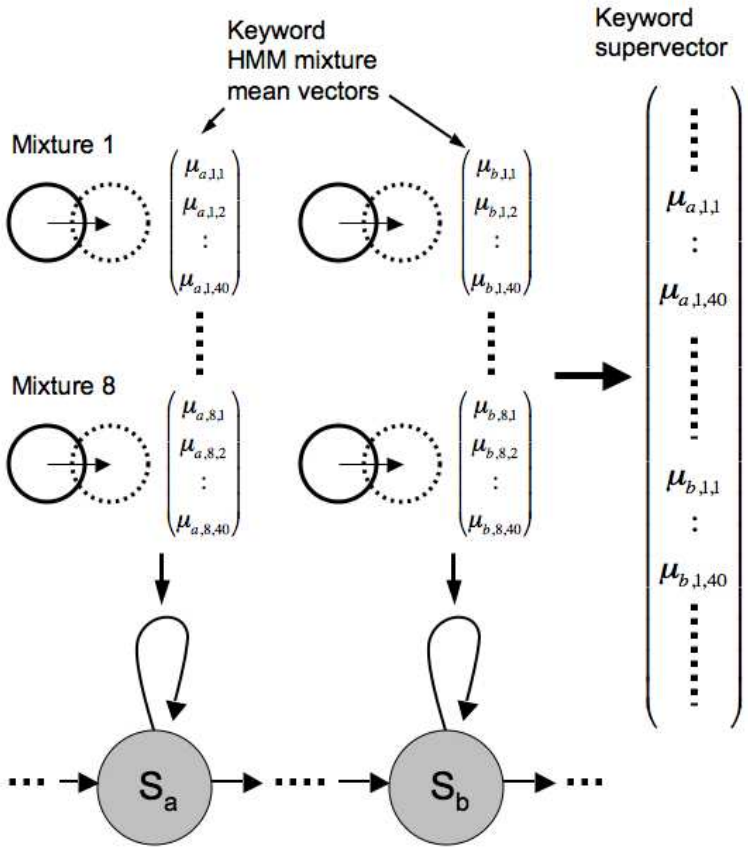


Figure 3: *Obtaining supervector from MAP adapted Gaussian mixture means.*

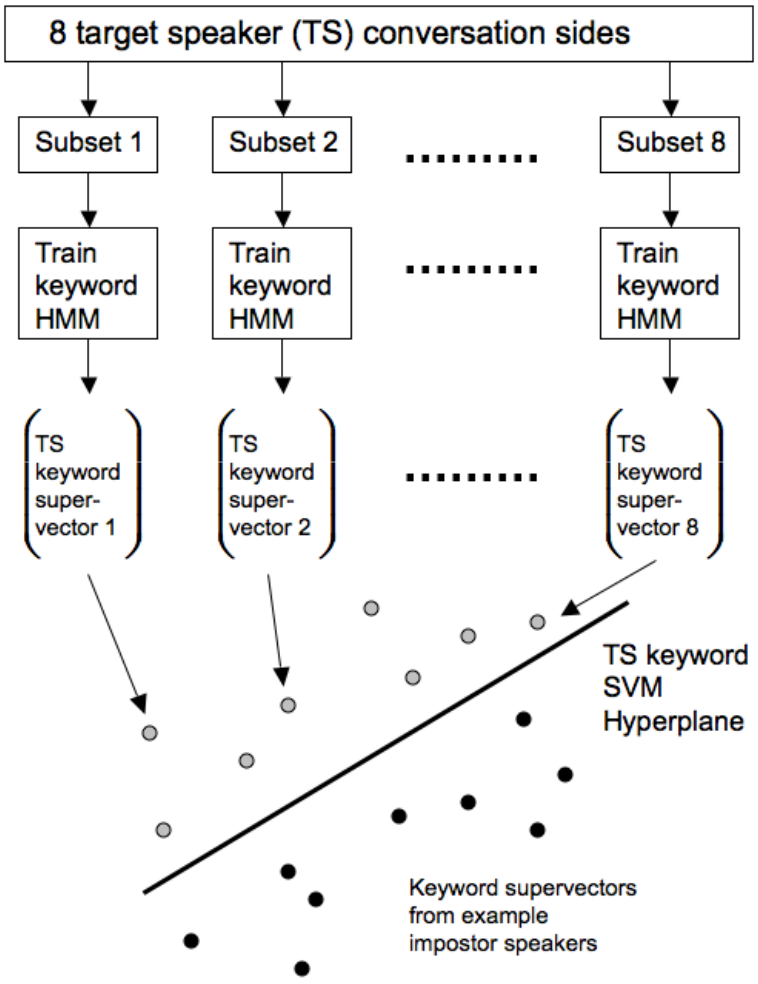


Figure 4: *Target speaker round-robin training for SVMs.*

examples for each target speaker even after subset selection, giving the positive example training errors more weight compared to negative example training errors seems desirable.

3.2 Testing and scoring

Once an SVM is trained for each target speaker, they are used to classify supervectors from keyword HMMs trained from single test conversation sides. If a keyword is missing in a test conversation side, the supervector from the corresponding background keyword HMM is used as a substitute. The classification score for a test conversation side supervector against a target speaker SVM model is the score for that particular trial (where the test conversation side is scored against the target speaker model), and the EER can likewise be computed given the scores from all trials.

The above approach trains one SVM for each target speaker from the corresponding keyword HMM, such that the speaker discriminative power of each keyword can be determined separately. To combine the keywords, supervectors obtained from all keyword HMMs for a target speaker can be concatenated into one higher-dimensional supervector for the target speaker (the same must be done for each example impostor speaker and test conversation side). SVM training and testing, as previously described, can be performed using the higher-dimensional supervectors to determine the combined speaker discriminative power of all keywords.

4 Approach 2: The phone lattice keyword HMM system

This system is another extension of the keyword HMM system, and uses the same training and testing paradigm - for a given keyword, one background keyword HMM is trained, and for each target speaker, a keyword HMM is trained by adapting from the background keyword HMM. However, instead of using MFCC feature sequences to train each keyword HMM, phone sequences are used. The nature of the keyword HMMs, such as the number of states, ergodicity, and output probability distributions, also differ. Because phone sequence data is far more sparse than the 40-dimensional MFCC feature sequences from before, this approach uses considerably less data and model parameters compared to the other approaches.

Both word and open-loop phone recognition decodings are needed. They are obtained from SRI, performed using the DECIPHER recognizer [12]. DECIPHER uses gender-dependent 3-state HMMs, trained using MFCC features of order 13 plus deltas and double deltas, for phone recognition [4].

4.1 Extraction of phone sequences

Each keyword HMM is trained using a set of keyword-constrained phone sequences, extracted from phone lattice decodings. Each phone lattice decoding for a conversation side contains a set of nodes and edges. Each edge represents a particular phone, along with its probability given the acoustic features used to perform the decoding; each node represents a particular time in the conversation side. Since each edge connects two nodes, each edge has a start and end time. For each keyword instance and a given conversation side, the top N most probable phone sequences formed by phone lattice edges falling within the time boundaries of the keyword (a keyword-constrained lattice segment) are considered.

It is possible to simply multiply the probabilities of the keyword-constrained edges of every possible phone sequence and take the top N. However, a sequence may have keyword-constrained edges of high probability, but the edges leading up to and from the keyword-constrained edges might have very low probability such that the overall probability of the sequence is low. Hence, the global path probabilities of each keyword-constrained phone sequence must be computed. These probabilities are estimated by finding the most probable paths (via Dijkstra's algorithm) from the first lattice node (in terms of time) to the nodes at the beginning of the keyword-constrained lattice segment, and from the last lattice node to the nodes at the end of the segment. Because the phones corresponding to edges outside of the keyword-constrained segment are not considered, these most probable paths have no phones associated with them. This entire procedure is shown in figure 5.

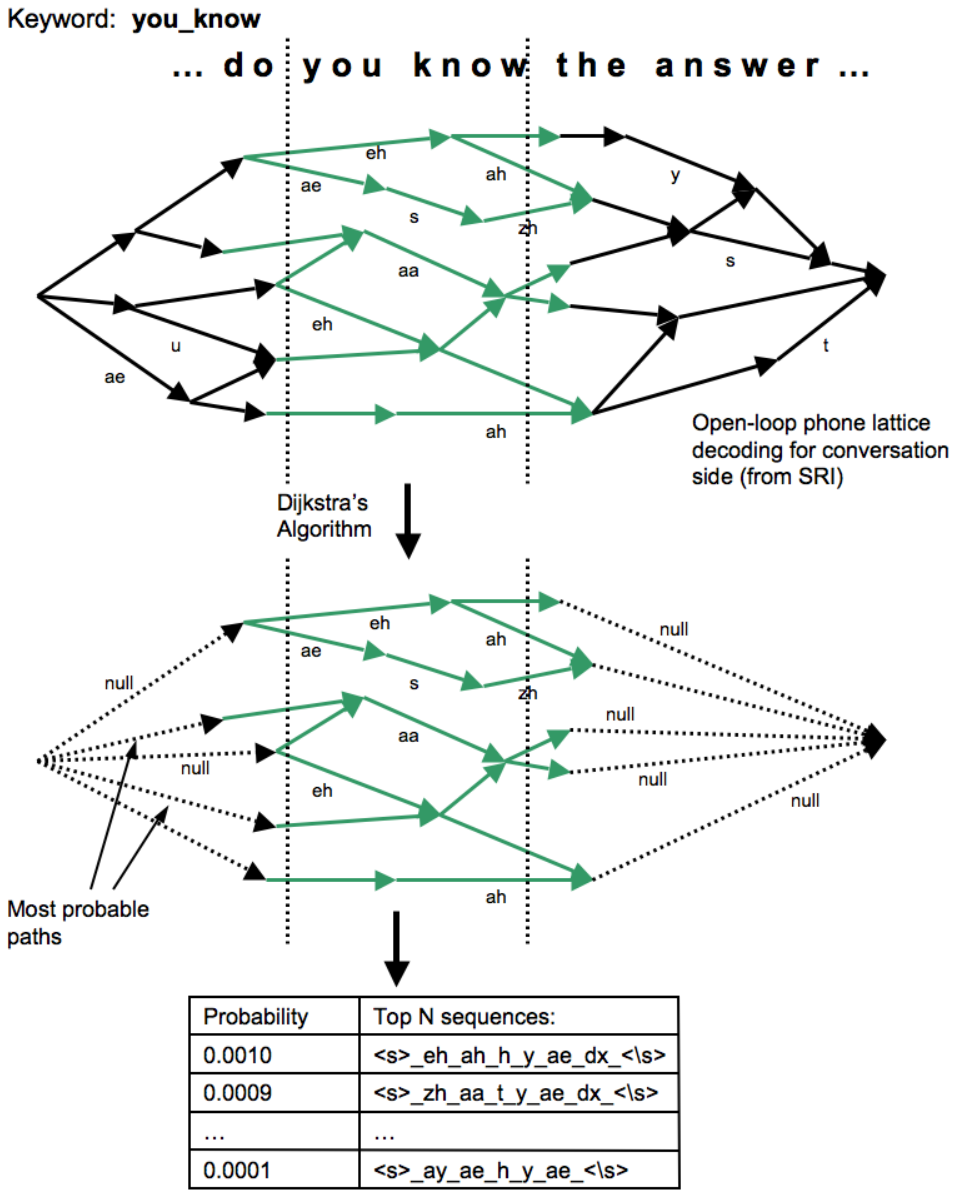


Figure 5: *Extraction of top N keyword-constrained phone sequences*

The global probability of a particular keyword-constrained phone sequence is the product of the probabilities of each edge along its keyword-constrained edge sequence, and the probabilities of

the most probable paths extending from the beginning and end of that edge sequence. If there are multiple instances of the same phone sequence among the keyword-constrained edges, global probabilities for each instance are summed to obtain an overall global probability for the sequence. Keyword-constrained phone sequences are ranked according to their overall global probabilities, and the top N sequences are used for keyword HMM training. Note that this entire procedure of phone sequence extraction is only for a particular keyword instance and conversation side. The procedure must be repeated for all keywords instances and conversation sides.

The following 48 phones, obtained from SRI, are used: *aa, ae, ah, ao, aw, ax, ay, b, ch, d, dh, dx, eh, er, ey, f, fip, g, hh, ih, iy, jh, k, l, lau, m, n, ng, ow, oy, p, pau, puh, pum, r, s, sh, t, th, uh, uw, v, w, y, z, zh,* $< s >$, $< \backslash s >$. Note that some of the phones, such as *pau, lau,* $< s >$, $< \backslash s >$ are not actual phones, but symbols that represent various aspects of conversational speech. *pau* represents a pause, or silence, *lau* represents laughter, $< s >$ represents the start of a phone sequence, and $< \backslash s >$ represents the end of a phone sequence.

4.2 Keyword HMM training

Once the phone sequences for a particular keyword are obtained, keyword HMMs are trained. Each of the 48 phones are assigned an integer value from 1 through 48, such that each phone sequence represents a sequence of discrete values. Hence, keyword HMM states have multinomial output probability distributions with 48 outputs.

All HMMs use 4 states, which is the minimum number of states necessary to capture time-dependent information among the phones (the first and last states are non-emitting; hence, there are only two "real" states). Because each phone sequence is typically only 6 to 7 phones in length (extremely short), HMMs with more than the minimum number of states would result in over-training, and decreased performance (experimentally determined). As with the keyword HMM system, each HMM must begin in the first state and end in the last state. Because those states are non-emitting, the first and last phones of each sequence (which correspond to the first and last states) are discarded. Since each phone sequence begins with $< s >$ and ends with $< \backslash s >$, these phones are not directly used to train the keyword HMMs. Note that the HMMs are ergodic. This differs from the keyword HMM approach, in which the HMMs have self-loops with no skips at each state. The reason for using ergodic HMMs is that intuitively, in a given phone sequence, any phone can transition to any other phone. If one HMM state has a high output probability for a particular phone, and another state has high output probability for another phone, then the states should not be restricted from transitioning to each other.

For a given keyword, the top N phone sequences (in terms of global probabilities) from each background conversation side are used to train the background keyword HMM (using the HTK software). As with the keyword HMM system, the HMM parameters are trained using the EM algorithm, with initializations obtained using the Viterbi alignment algorithm. This time, however, the EM and Viterbi alignment algorithms are implemented for HMMs with discrete multinomial output probability distributions, as opposed to a mixture of Gaussians. If a keyword does not exist among the target speaker conversation sides, the target speaker keyword HMM is the same as the corresponding background keyword HMM (like the keyword HMM approach).

Target speaker keyword HMM training is done in almost the same fashion as background keyword HMM training. For each keyword, one keyword HMM is trained for each target speaker via the EM algorithm. Instead of using the Viterbi alignments algorithm to initialize the HMM parameters, the background keyword HMM parameters (for the same keyword) act as the initial

parameters. Like the keyword HMM system, eight conversation sides of target speaker data are used to train each HMM.

4.3 Handling multiple keyword instances in a conversation side

When a keyword appears more than once in a conversation side (as is often the case), the top N sequences from each keyword instance can be obtained, and the entire set of top N sequences can be used to train a keyword HMM. Note that this method implicitly gives higher HMM training weight to phone sequences that appear more than once among the keyword instances, since those sequences appear more than once in the HMM training data.

4.4 Testing and scoring

For a given keyword W, top N phone sequences from each test conversation side t are scored against a target speaker model M_{TS}. The score for phone sequence i is computed via the standard log-likelihood ratio (like the keyword HMM approach):

$$LLR(i, t, M_{TS}, W) = \log\left(\frac{p(ph_1, \ldots, ph_T|M_{TS})}{p(ph_1, \ldots, ph_T|M_{BKG})}\right)$$

where M_{BKG} is the background keyword HMM, T is the length of the phone sequence, and

$$\log\left(p(ph_1, \ldots, ph_T|M)\right) = \log\left(\sum_x p(ph_1, \ldots, ph_T|x, M)p(x|M)\right)$$

where x is the sequence of allowable states. Note that keyword W, as used in the above equations, encapsulates all instances of the keyword, such that phone sequence i can be any phone sequence belonging to any instance of keyword W.

Similar to the keyword HMM approach, the overall score for a trial for keyword W is computed by taking an average of the likelihood ratios of the phone sequences, and the keywords can be combined by averaging the likelihood ratios across all keywords. Once scores are obtained from all trials, the EER can be obtained as before.

5 Approach 3: The keyword phone N-grams system

This system steps away from the HMM realm, and examines the speaker discriminative power of keyword-constrained phone N-gram counts using an SVM classifier. Unlike the previous system, time-dependent relationships among keyword-constrained phones are removed, and only their counts are used for training and testing. This approach is inspired by an approach that uses non keyword-constrained phone N-gram counts in a similar SVM classifier [4]. This approach uses the same word and open-loop phone lattice decodings as the previous approach.

5.1 Phone N-gram counts extraction

For a given keyword, the phone N-gram counts extraction procedure is similar to the extraction of top N keyword-constrained phone sequences in the previous system, in that a phone lattice segment consisting of edges falling within the time boundaries of the keyword are obtained, and the global path probabilities from beginning and ends of the lattice to the segment are estimated via Dijkstra's algorithm. However, instead of computing the global probabilities of entire phone sequences within the lattice segment, the probabilities of phone N-grams are desired. An estimate of the global probability $P(N_i|W,C)$ for phone N-gram N_i constrained by keyword W in conversation side C is computed as follows:

$$P(N_i|W,C) = \frac{\sum_j \sum_k p(S_k|W_j,C) count(N_i|S_k)}{count(W|C)} \tag{5}$$

where S_k is a phone sequence constrained by keyword W, $count(N_i|S_k)$ is the number of occurrences of phone N-gram N_i along phone sequence S_k, and $count(W|C)$ is the number of instances of keyword W in conversation side C. The phone N-grams extraction procedure is illustrated in figure 6.

The estimated probabilities of all phone N-grams (typically of orders 1, 2, and 3) for a particular keyword and conversation side is used as a feature vector in the SVM classifier (after a minor weighting of the features).

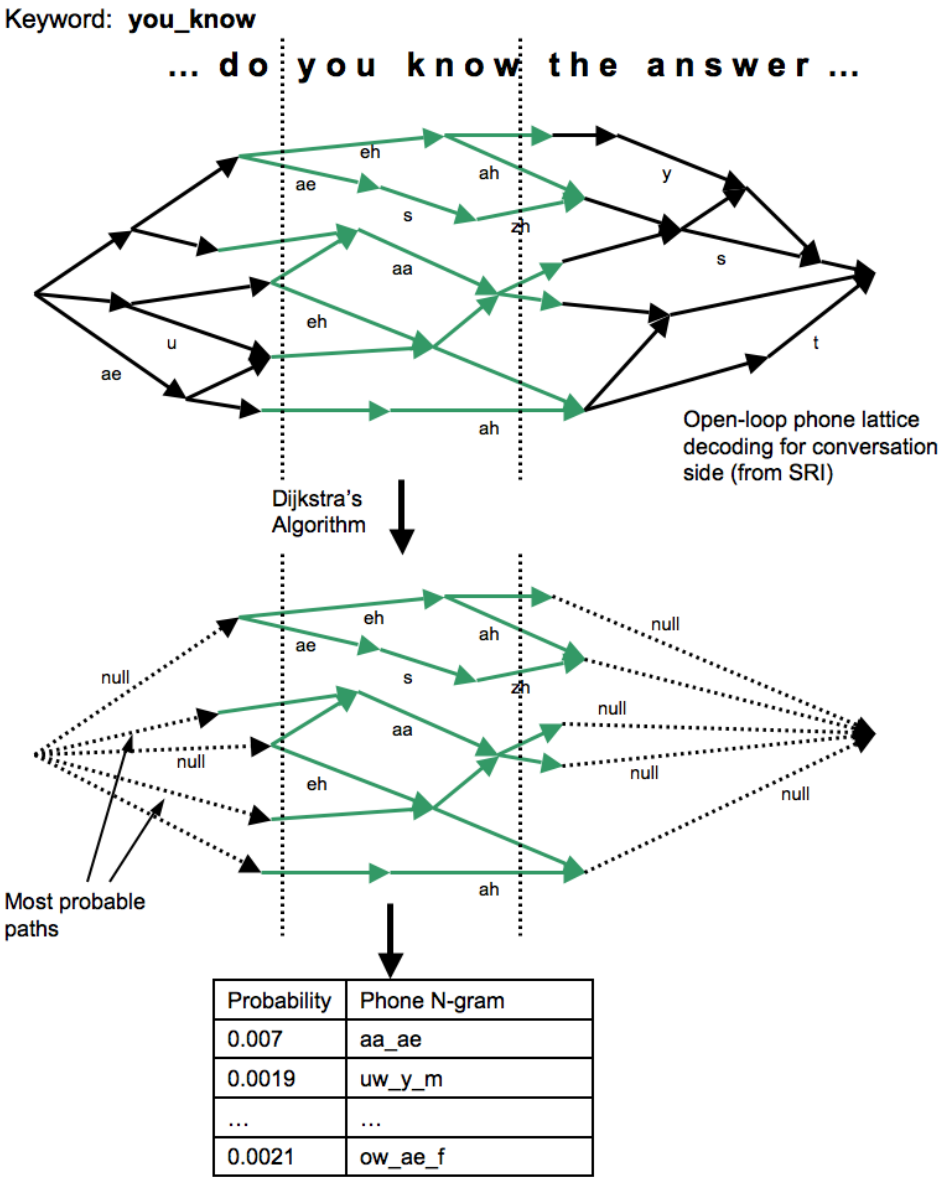
5.2 Keyword combination

As with the previous approaches, results can be obtained for each keyword separately by using only the phone N-gram features constrained by each particular keyword. Different keywords can be combined at the feature level by concatenating the feature vectors for the different keywords within a conversation side (similar to the keyword-combination approach for the supervector keyword HMM system). Training, testing and scoring are completed on these larger feature vectors. This combination approach is shown in figure 7.

Because not all keywords appear in all conversation sides, phone N-gram data for a particular keyword in a conversation side may not exist, and are assigned feature values of 0. This is undesirable, since the values of 0 do not accurately reflect phone N-gram probabilities should the keyword exist in the conversation side. This missing data problem can be avoided by choosing high frequency keywords, with the majority of conversation sides containing most or all of the keywords. Note that a second way to address the missing data problem is by substituting the missing values with existing background values. This approach, however, has been experimentally determined to perform poorly.

5.3 SVM training, testing, and scoring

As with the supervector keyword HMM approach and the approach for the non keyword-constrained phone N-grams system [4], an SVM with a linear kernel is trained for each target speaker (via the SVM^{light} software package [11]). For each target speaker, the the feature vectors belonging to 8 target speaker conversation sides are used as positive training examples, while the vectors belonging to all background conversation sides are used as negative training examples. To obtain a score



Probability	Phone N-gram
0.007	aa_ae
0.0019	uw_y_m
…	…
0.0021	ow_ae_f

Figure 6: *Extracting phone N-gram counts from a conversation side*

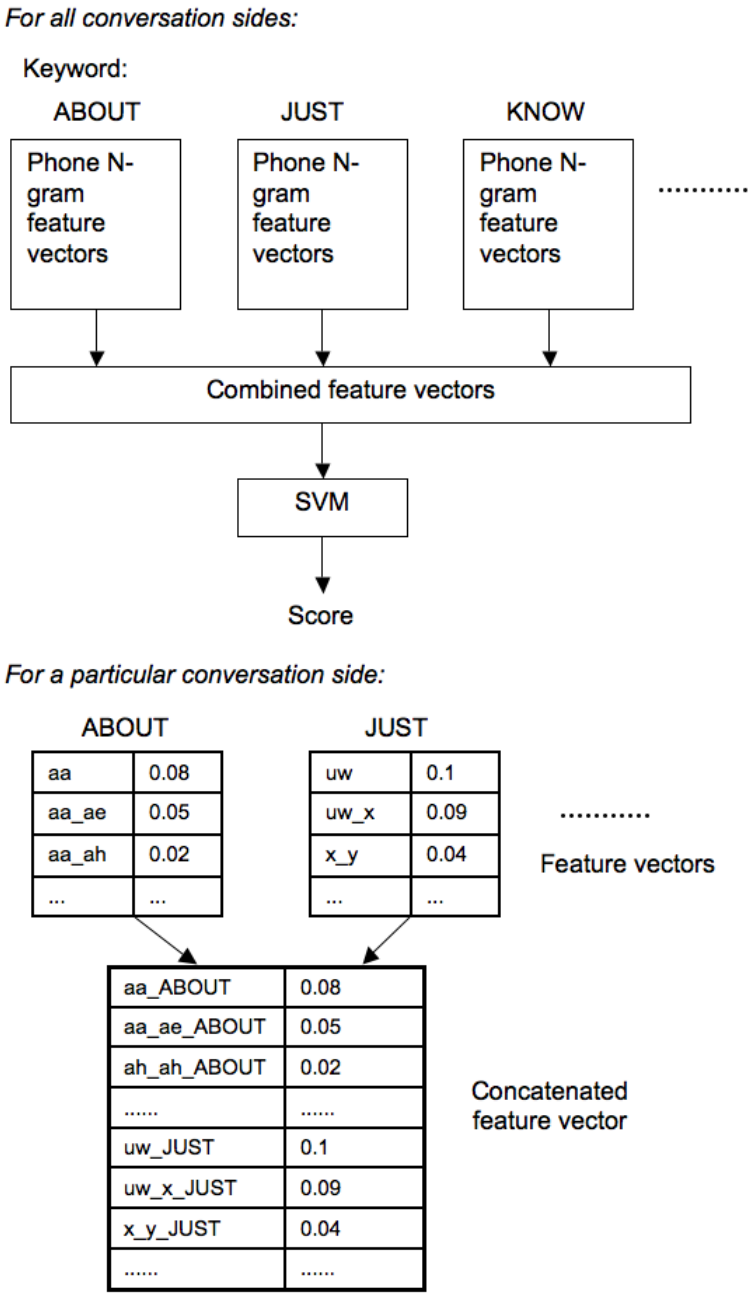


Figure 7: *Keyword combination for the keyword-constrained phone lattice approach*

for a particular trial, the test conversation side feature vector is scored against the target speaker SVM model, as is done in the supervector keyword HMM system.

6 The keywords

A total of 62 keywords are involved in the three systems, along with the keyword HMM system. Some are among the common discourse markers, back-channels, and filled pauses [6]. Certain keywords are chosen based on their perceived inter-speaker variability in pronunciation, while others are chosen based on frequency (to address the missing data problem in the keyword-constrained phone N-gram counts system). The keywords are the following – *a, about, actually, all, and, anyway, are, be, because, but, do, for, get, have, i, if, i_know, i_mean, in, is, i_see, it, i_think, just, know, like, mean, my, no, not, now, of, oh, okay, on, one, or, people, really, right, see, so, that, the, there, they, think, this, to, uh, uhhuh, um, was, we, well, what, with, would, yeah, yep, you, you_know*.

Among these, the following 18 keywords – *actually, anyway, i_know, i_mean, i_see, i_think, like, now, okay, right, see, uh, uhhuh, um, well, yeap, yep, you_know* – are used in the keyword HMM system [6], and represent ∼15% of background total conversation side duration. Note that the keyword HMM system originally used 19 keywords. Because one keyword (*you_see*) is extremely rare and resulted in degeneracy among the Gaussian mixture components of the HMM states in the keyword HMM system, it is not used here.

Also, the following 52 unigrams among the 62 keywords – *a, about, all, and, are, be, because, but, do, for, get, have, i, if, in, is, it, just, know, like, mean, my, no, not, of, oh, okay, on, one, or, people, really, right, so, that, the, there, they, think, this, to, uh, uhhuh, um, was, we, well, what, with, would, yeah, you* – represent high-frequency unigrams, each occurring more than 4,000 times in the background conversation sides. Note that the 62 keywords represent ∼42% of background conversation side duration, with the 52 unigrams representing the majority (∼41%) of this duration. While most of the data is unused in obtaining all results, it is interesting to note that given all the words in the English language, the 52 keyword unigrams represent slightly less than half of the conversation side durations.

Since different keywords are typically better suited for different approaches, different subsets of the 62 keywords are examined for each system. A major goal of these experiments is to examine the performance of keywords, especially as it relates to their lengths and frequencies, in different settings.

7 Data

The Switchboard II and Fisher corpora are used for background model training, and the SRE06 corpora for target speaker model training and testing for all systems. For the supervector keyword HMM system, 3,879 conversation sides from Switchboard II and 1,792 from the SRE04 corpora are used to train 1,300 example impostor speakers for all SVM experiments. 7,598 conversation sides from SRE06 are used for target speaker model training and testing for all systems. 1,553 Fisher and Switchboard II conversation sides, where each speaker is represented by no more than one conversation side, are used for background model training.

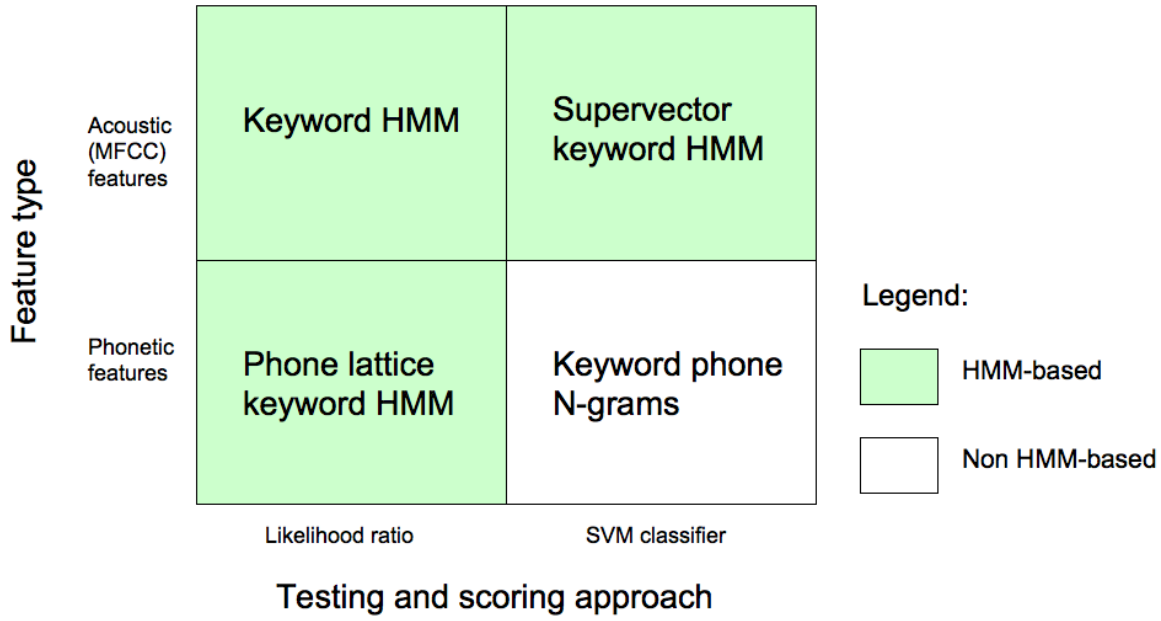


Figure 8: *Summary of similarities and differences among the approaches*

8 Experiments and results

The three new approaches along with the keyword HMM approach are similar to one another in methodology. In particular, the keyword HMM approach resembles the supervector keyword HMM approach, since both approaches use keyword-constrained acoustic features to train HMMs, and the phone lattice keyword HMM approach, since both approaches use the same log-likelihood based scoring in the keyword HMM framework. The phone lattice keyword HMM approach also resembles the keyword phone N-grams approach, since both approaches utilize only high-level keyword-constrained phone features. Lastly, the keyword phone N-grams and supervector keyword HMM approaches utilize the SVM classifier for testing and scoring. Figure 8 summarizes the similarities and differences among the approaches. Comparisons using various subsets of the 62 keywords are made for the similar approaches.

8.1 Keyword combination results

The keyword-combination results examine the effectiveness of the collective power of a set of keywords for each system. The set of 18 keywords (19 minus *you_see*) of the keyword HMM system, along with 20 of the 52 high-frequency unigrams – *about, all, because, but, have, just, know, mean, no, not, one, people, really, so, that, there, think, this, was, what* – are examined for the HMM-based systems (a total of 38 keywords are used). These 20 unigrams are selected based on their length, as each resulted in at least 5 HMM states for the acoustic keyword HMM systems. Note that these 38 keywords represent ∼26% of background conversation side duration. Because the keyword phone N-grams system requires keywords of high frequency due to the missing data problem, the 52 high-frequency unigrams are chosen. The 52 keywords are also used for the phone lattice keyword HMM approach.

The keyword-combination results for each system are for its optimal set of tunable parameters. For

the supervector keyword HMM system, a weight of 1 is given to weigh the positive versus negative SVM training errors, and round-robin training using subsets of 3 conversation sides is used to give 8 positive training examples per target speaker. For the keyword phone N-grams system, a weight of 500 (for the positive versus negative SVM training errors) is used, phone N-grams of order 1, 2, and 3 are extracted as features, and the top ∼33K features, in terms of frequency among the background conversation sides, are used. For the phone lattice keyword HMM system, the 40 most probable phone sequences in terms of global probability for each keyword instance are used to train the 4-state, ergodic keyword HMMs. Table 1 shows the keyword-combination results for the various systems, on the SRE06 corpus using 16,831 trials with 2,010 true speaker trials. Also shown are results for a baseline cepstral GMM system [13] and a baseline non keyword-constrained phone N-grams system [4], both with T-norm added [14]. Note that these last two systems use more data by requiring entire speech conversation sides, unlike the 4 keyword-based systems.

System	# of keywords	EER (%)
Keyword HMM	18	5.5
Keyword HMM	38	5.0
Approach 1: Supervector keyword HMM	18	4.9
Approach 1: Supervector keyword HMM	38	4.3
Approach 2: Phone lattice keyword HMM	18	20.7
Approach 2: Phone lattice keyword HMM	38	16.9
Approach 2: Phone lattice keyword HMM	52	13.7
Approach 3: Keyword phone N-grams	52	5.0
Non keyword-constrained phone N-grams baseline	–	5.2
GMM baseline	–	4.6

Table 1: *Keyword combined results.*

The supervector keyword HMM system out-performs all other systems. The 4.3% EER achieved by using 38 keywords in combination is the best overall result, with a 6.5% improvement over the GMM baseline (4.6% EER), and a 17.3% improvement over the non keyword-constrained phone N-grams baseline (5.2% EER). This is quite interesting, in that only ∼26% of speech data is being used to give improvements over baseline systems using 100% of the data and T-norm.

The 4.9% EER achieved by using the set of 18 keywords is better than the results of all other systems except the GMM baseline (4.6% EER). In particular, with 18 keywords, the supervector keyword HMM system (4.9% EER) achieves a 10.9% improvement over the keyword HMM system (5.5% EER), and a 76.3% improvement over the phone lattice keyword HMM system (20.7% EER). With 38 keywords, the supervector keyword HMM system (4.3% EER) achieves a 14.0% improvement over the keyword HMM system (5.0% EER), and a 74.6% improvement over the phone lattice keyword HMM system (16.9% EER).

Using the 52 high-frequency keywords, the keyword phone N-grams system (5.0% EER) achieves a respectable result, and a 3.8% improvement over the non keyword-constrained phone N-grams baseline (5.2% EER). This result directly demonstrates the effectiveness of focusing modeling power on the informative regions of speech containing the keywords, since the keyword phone N-grams system uses the methodology of the non keyword-constrained system, but applied to each keyword separately. The keyword phone N-grams system also achieves a 63.5% improvement over the phone lattice keyword HMM system (13.7% EER). However, it uses ∼41% of speech data compared to the maximum amount of data (∼26%) used for the keyword HMM and supervector keyword HMM systems.

Recall that both the keyword phone N-grams and supervector keyword HMM systems are SVM-based, with feature vectors having ∼33K and ∼55K dimensions respectively for the SVM classifier. Hence, with the exception of the keyword HMM system using 38 keywords, which has the same EER (5.0%) as the keyword phone N-grams system, SVM-based approaches relying on high-dimensional feature vectors outperform the other approaches.

Unlike the keyword phone N-grams system, which uses ∼264K parameters to represent each target speaker model (∼33K x 8 positive training examples), the phone lattice keyword HMM system uses only 98 model parameters (2 emitting states x 48 outputs per state minus 1 + 4 transition probability values). Also, unlike the keyword HMM system, the phone lattice keyword HMM system uses a very sparse form of speech data (phone sequences versus 40-dimensional MFCC feature sequences) to train its keyword HMMs. Hence, it is not surprising to observe a drop-off in performance for this system.

The results also demonstrate the benefits of using more keywords. Increasing the number of keywords from 18 to 38 results in an 12.2% improvement for the supervector keyword HMM system (4.9% EER to 4.3% EER), a 9.1% improvement for the keyword HMM system (5.5% EER to 5.0% EER), and a 18.4% improvement for the keyword phone N-grams system (20.7% EER to 16.9% EER). Increasing the number of keywords from 18 to 52 (albeit with only 8 common keywords) achieves a 18.9% improvement for the latter system (16.9% EER to 13.7% EER).

However, as more and more keywords are used, one advantage of using keyword-constraining, namely, reducing the amounts of speech data required, diminishes. The 38 keywords (∼26% of background data) represent a ∼60% increase in data over the 18 keywords (∼15% of background data), while the 52 keywords (41% of background data) represent a ∼173% increase. This increase in data usage greatly increases the need for memory and computation when implementing the systems.

8.2 Individual keyword results

The individual keywords are examined to see how each performs in the various approaches, under the optimal set of parameters as previously described. Because not all keywords exist in all conversation sides, individual keyword results only involve trials where the keyword exists in both the target speaker model conversation sides, and test conversation side. Table 2 shows the individual keyword results for 34 of the 38 keywords which have at least 1,000 occurrences among the background conversation sides (note that the other 4 keywords have too few instances resulting in too few trials for statistical significance), for the supervector keyword HMM (SVHMM), keyword HMM (HMM), and phone lattice keyword HMM systems (PLHMM). Table 3 shows results for the 52 keyword unigrams, for the phone lattice keyword HMM (PLHMM) and keyword phone N-grams (PLN) systems. Also shown are the number of occurrences of the keyword in background conversation sides, and the number of keyword HMM states for the MFCC feature-based HMM systems. Note that for the 38 keywords, if a keyword unigram is part of a keyword bigram, those unigram instances are ignored. For the 52 unigrams, however, no keyword instances are discarded. The results in table 2 are sorted according to EER for the keyword HMM system, which has the overall best individual keywords results. Results in table 3 are sorted according to EER for the keyword phone N-grams system.

According to both tables, keywords that perform relatively well for one system tend to perform well for the others. Results show that individual keywords generally perform significantly better in the acoustic features framework (the keyword HMM and supervector keyword HMM systems),

	EER (%) results			Other statistics		
Keyword	HMM	SVHMM	PLHMM	# trials (true spkr trials)	# of occurrences	HMM states
yeah	11.4	17.0	29.7	16,419 (1,951)	26,530	6
you know	11.9	17.5	26.0	15,775 (1,882)	17,349	8
i think	14.7	23.5	34.0	13,181 (1,564)	6,288	9
right	14.7	22.7	30.2	13,606 (1,654)	8,021	8
um	14.8	19.3	30.7	14,494 (1,713)	11,962	6
that	14.9	19.2	30.1	16,661 (1,986)	26,277	5
because	15.2	24.1	32.3	13,151 (1,573)	5,164	8
like	15.2	21.7	26.7	15,651 (1,869)	18,058	5
i mean	15.8	26.8	34.0	11,773 (1,416)	5,470	8
but	16.6	22.9	32.6	15,945 (1,899)	12,766	5
people	17.2	26.5	34.6	12,389 (1,490)	4,906	8
so	17.4	24.7	34.5	16,116 (1,922)	14,291	6
have	18.0	25.4	35.2	15,273 (1,828)	9,610	5
just	18.1	28.4	35.3	13,938 (1,660)	8,660	6
not	18.3	26.0	36.2	14,603 (1,739)	6,817	5
really	18.6	28.5	32.2	12,077 (1,444)	6,674	7
uhhuh	20.1	26.4	37.0	12,494 (1,485)	8,371	9
think	20.3	30.2	39.3	9,724 (1,174)	3,179	6
okay	20.4	28.1	38.8	8,960 (1,038)	4,322	9
about	20.7	30.1	37.8	12,858 (1,543)	5,769	7
uh	21.3	23.6	37.7	15,968 (1,897)	18,065	3
now	22.5	34.2	40.9	8,532 (1,060)	2,851	6
actually	23.1	31.5	37.9	5,240 (663)	2,240	9
this	24.8	33.9	43.7	12,032 (1,418)	5,408	5
was	25.0	34.4	38.3	13,654 (1,651)	9,888	5
what	25.7	31.7	39.1	14,824 (1,780)	8,088	5
i know	25.8	33.1	42.4	8,136 (966)	2,142	9
one	26.2	33.3	41.4	11,618 (1,389)	4,559	5
no	26.3	33.0	42.1	11,313 (1,341)	4,245	6
there	26.5	33.7	40.3	11,467 (1,384)	4,716	5
know	27.8	34.8	39.9	11,489 (1,362)	4,767	5
see	28.0	35.7	43.1	9,075 (1,080)	2,006	6
all	28.4	35.5	41.7	12,665 (1,502)	4,681	5
well	29.7	34.3	39.7	14,163 (1,676)	7,590	6
average EER	20.4	28.0	36.3	–	–	–

Table 2: Individual keyword results for keyword HMM, supervector keyword HMM, and phone lattice keyword HMM systems

Keyword	EER (%) results PLA	PLHMM	Other statistics # trials (true spkr trials)	# of occurrences
know	24.1	26.1	16,485 (1,962)	24,258
i	24.7	25.2	16,790 (2,005)	59,622
and	24.8	28.7	16,745 (1,998)	37,878
yeah	26.1	29.7	16,419 (1,951)	26,530
like	26.1	26.7	15,651 (1,869)	18,058
you	26.9	28.9	16,735 (1,999)	39,093
that	27.3	30.1	16,661 (1,986)	26,277
think	28.3	32.9	14,748 (1,754)	9,467
right	29.0	30.2	13,606 (1,654)	8,021
because	29.1	32.3	13,151 (1,573)	5,164
but	29.8	32.6	15,945 (1,899)	12,766
people	30.5	34.6	12,389 (1,490)	4,906
really	30.7	32.2	12,077 (1,444)	6,674
um	32.2	30.7	14,494 (1,713)	11,962
to	32.2	33.2	16,635 (1,986)	27,224
mean	32.4	35.0	12,292 (1,467)	5,724
the	32.7	34.6	16,714 (1,998)	31,606
so	33.3	34.5	16,116 (1,922)	14,291
have	33.4	35.2	15,273 (1,828)	9,610
they	34.7	35.9	15,628 (1,862)	12,436
about	35.0	37.8	12,858 (1,543)	5,769
okay	35.1	38.8	8,960 (1,038)	4,322
just	35.4	35.3	13,938 (1,660)	8,660
not	35.8	36.2	14,603 (1,739)	6,817
uhhuh	36.7	37.0	12,494 (1,485)	8,371
of	37.1	36.3	16,337 (1,956)	18,541
do	37.6	39.1	14,661 (1,758)	7,754
uh	37.6	37.7	15,968 (1,897)	18,065
for	37.8	38.0	13,542 (1,642)	7,169
was	37.8	38.3	13,654 (1,651)	9,888
on	37.9	36.8	13,487 (1,611)	6,229
what	37.9	39.1	14,824 (1,780)	8,088
it	38.4	38.8	16,581 (1,983)	19,958
well	38.8	39.7	14,163 (1,676)	7,590
in	38.9	38.6	15,903 (1,911)	14,799
is	39.1	38.4	15,798 (1,881)	11,599
oh	39.5	39.2	14,270 (1,679)	10,316
get	39.6	38.6	11,303 (1,360)	4,385
my	39.7	37.1	12,259 (1,465)	7,597
there	39.9	40.3	11,467 (1,384)	4,716
with	40.1	40.8	12,510 (1,519)	5,014
if	40.2	40.8	12,832 (1,528)	5,499
a	40.2	39.3	16,557 (1,978)	21,705
one	41.0	41.4	11,618 (1,389)	4,559
no	41.0	42.1	11,313 (1,341)	4,245
be	41.4	40.4	14,064 (1,689)	6,869
all	41.7	41.7	12,665 (1,502)	4,681
or	41.7	36.0	12,349 (1,516)	5,625
this	42.0	43.7	12,032 (1,418)	5,408
would	42.1	42.1	10,736 (1,305)	5,222
are	42.9	41.4	12,528 (1,525)	5,581
we	43.7	42.5	11,872 (1,406)	7,270
average EER	35.4	36.2	–	–

Table 3: Individual keyword results for phone lattice keyword HMM and keyword phone N-grams systems

with a 43.8% improvement in average EER (36.3% to 20.4%) from the phone lattice to the keyword HMM system, and a 22.9% improvement (36.3% to 28.0%) from the phone lattice to the supervector keyword HMM system. In addition, there is a 27.1% improvement in average EER (28.0% to 20.4%) from the supervector to the keyword HMM system, which is in contrast to the 14.0% improvement displayed by the former to the latter in the keyword-combination experiments.

The individual keyword results for the two phone lattice-based systems are significantly worse than MFCC-based systems, while the keyword phone N-grams system shows a modest 2.2% improvement in average EER (36.2% to 35.4%) from the phone lattice keyword HMM system. This is in stark contrast to the 63.5% improvement achieved in the keyword-combination experiments. This suggests that SVMs (used by the keyword phone N-grams system) perform exceedingly well in feature-level combination, such that increasing the dimensionality of the overall feature vector (via concatenation of features) can lead to significantly improved results. Whereas the feature vectors in the keyword-combination experiments are of very high dimension (over 30K), they are typically of 1,000 to 2,000 in dimension in the individual keyword experiments. Overall, however, these results suggest that while systems relying on phone features can match or exceed performances of acoustic feature-based systems in keyword-combination experiments, the sparseness of the high-level phone features compared to the acoustic features leads to significantly worse results when keywords are examined separately.

It is also interesting to observe, from tables 2 and 3, that keywords that perform well (i.e. *yeah, you_know*) generally occur more often in the background conversation sides, and the worst-performing keywords (i.e. *see, anyway*) rank among the bottom in occurrences. This is especially true for the supervector keyword HMM system, where 6 of the top 7 performing keywords (*yeah, you_know, that, um, like, but*) rank among the top 10 in occurrences. This leads to the conclusion that having more keyword training data is generally helpful, especially for the SVM-based supervector keyword HMM approach.

While keyword performance is positively correlated with keyword frequency, keyword length (approximately represented by the number of keyword HMM states), appears to be much less of a factor. However, a couple keywords of high length (*actually, i_know*) have very low frequency, while 4 of the top 7 keywords for the keyword HMM system have eight or more HMM states. Hence, given more data such that more instances of keywords with high numbers of HMM states can be used for training and testing, it may be possible to discover a clear positive correlation between keyword performance and length. Note that one particular keyword – *you_know* – has the best combination of length (8 HMM states) and frequency (17,349 occurrences), and ranks among the top 2 in performance for all three systems. To its benefit, it has 34.6% fewer occurrences than the top-performing keyword – *yeah* – in the keyword HMM and supervector keyword HMM systems, while showing only 4.4% and 2.9% drop-offs in performance respectively for the two systems. For the phone lattice keyword HMM system, the *you_know* keyword even showed a 12.5% improvement in performance in comparison to the *yeah* keyword.

9 Conclusion

In this work, I have discussed four state-of-the-art keyword-based systems for speaker recognition, three of which (the supervector keyword HMM, phone lattice keyword HMM, and keyword phone N-grams systems) I implemented, and the other (the keyword HMM system) I experimented with. In particular, I have examined the performances of 62 keywords (representing $\sim$42% of background conversation side duration) of various lengths and frequencies in both keyword-

combination experiments and individual keyword experiments using the various approaches.

One aspect of my work has shown that the SVM is a very effective classifier for systems with very high dimensional feature vectors. This is typically the case for keyword-combination, where the overall feature vector is a concatenation of feature vectors for each keyword. The effectiveness of the SVM in this framework appears to compensate for features that perform poorly in the individual keyword setting (i.e. phone N-grams). I have shown that acoustic features, on the other hand, perform significantly better in the individual keyword setting, and achieve the best result in conjunction with the SVM (i.e. supervector keyword HMM system) in keyword-combination.

Most importantly, my results show that the supervector keyword HMM system, using $\sim$26% of speech data (in terms of background conversation side duration) in keyword combination, achieves a 0.3% absolute EER improvement (4.6% to 4.3% EER) on SRE06 over the GMM baseline system (the best baseline) that uses 100% of speech data.

Lastly, I have shown that one particular keyword – *you_know* – displays an excellent combination of length, frequency, and performance, and is arguably the best keyword for these systems. Future work can attempt to increase the modeling power of this keyword, along with other high-performing keywords, to examine exactly how much speaker discriminative information can be obtained from individual keywords themselves. One approach, which I am currently working on as an extension, is to augment the supervectors of the supervector keyword HMM system using long-term, prosodic features. In addition, other features can be added to the current MFCC acoustic feature vectors to train keyword HMMs with higher-dimensional Gaussian means and covariances. Different methods to combine the scores for the same keyword for multiple systems can also be examined, in an attempt to improve the individual keyword results.

References

[1] Reynolds, D., "Speaker Identification and Verification using Gaussian Mixture Speaker Models," in Speech Communication, Vol. 171-2, pp. 91-108, 1995.

[2] Doddington, G., "Speaker Recognition based on Idiolectal Differences between Speakers," in Proc. of Eurospeech, pp. 2521-2524, 2001.

[3] Andrews, W., Kohler, M., Campbell, J., "Phonetic Speaker Recognition," in Proc. of Eurospeech, pp. 149-153, 2001.

[4] Hatch, A., Peskin, B., Stolcke, A., "Improved Phonetic Speaker Recognition Using Lattice Decoding," in Proc. of ICASSP, Vol. 1, pp. 169-172, 2005.

[5] Sturim, D., Reynolds, D., Dunn, R., Quatieri, T., "Speaker Verification using Text-Constrained Gaussian Mixture Models," in Proc. of ICASSP, Vol. 1, pp. 677-680, 2002.

[6] Boakye, K., "Speaker Recognition in the Text-Independent Domain using Keyword Hidden Markov Models," master's report, UC Berkeley, 2005.

[7] Lei, H., Mirghafori, N., "Word-Conditioned HMM Supervectors for Speaker Recognition," to appear in Proc. of Interspeech, 2007.

[8] Campbell, W., Sturim, D., Reynolds, D., Solomonoff, A., "SVM Based Speaker Verification using a GMM Supervector Kernel and NAP Variability Compensation," in Proc. of ICASSP, Vol. 1, pp. I-97-100, 2006.

[9] Lei, H., Mirghafori, N., "Word-Conditioned Phone N-grams for Speaker Recognition," in Proc. of ICASSP, 2007.

[10] HMM Toolkit (HTK): http://htk.eng.cam.ac.uk

[11] Joachims, T., "Making Large Scale SVM Learning Practical," in Advances in kernel methods - support vector learning, B. Schoelkopf, C. Burges, and A. Smola, Eds. MIT-press, 1999.

[12] Stolcke, A., Bratth, H., Butzberger, J., Franco, H., Rao Gadde, V., Plauche, M., Richey, C., Shriberg, E., Sonmez, K., Weng, F., Zheng, J., "The SRI March 2000 Hub-5 Conversational Speech Transcription System," Proc. NIST Speech Transcription Workshop, 2000.

[13] Kajarekar, S., Ferrer, L., Venkataraman, A., Sonmez, K., Shriberg, E., Stolcke, A., Gadde, R.R., "Speaker Recognition using Prosodic and Lexical Features," in Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop, pp. 19-24, 2003.

[14] Bimbot, F., Bonastre J.F., Fredouille C., Gravier G., Magrin-Chagnolleau I., Meignier S., Merlin T., Ortega-Garca J., Petrovska-Delacrtaz D., Reynolds D.A., "A Tutorial on Text-independent Speaker Verification, in EURASIP Journal on Applied Signal Processing, Vol. 4, pp. 430-451, 2004.