Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams

Unsupervised Spoken Keyword Spotting via Segmental DTW on Gaussian Posteriorgrams

Yaodong Zhang and James R. Glass

MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar Street, Cambridge, Massachusetts 02139, USA
{ydzhang, glass}@csail.mit.edu

***Abstract*—In this paper, we present an unsupervised learning framework to address the problem of detecting spoken keywords. Without any transcription information, a Gaussian Mixture Model is trained to label speech frames with a Gaussian posteriorgram. Given one or more spoken examples of a keyword, we use segmental dynamic time warping to compare the Gaussian posteriorgrams between keyword samples and test utterances. The keyword detection result is then obtained by ranking the distortion scores of all the test utterances. We examine the TIMIT corpus as a development set to tune the parameters in our system, and the MIT Lecture corpus for more substantial evaluation. The results demonstrate the viability and effectiveness of our unsupervised learning framework on the keyword spotting task.**

I. Introduction

Automatic speech recognition (ASR) technology typically requires large quantities of language-specific speech and text data that is used to train complex statistical acoustic and language models. Unfortunately, such valuable linguistic resources are unlikely to be available for the majority of languages in the world, especially for less frequently used languages. For example, commercial ASR engines typically support 50-100 (or fewer) languages [1]. Despite substantial development efforts to create annotated linguistic resources that can be used to support ASR development [2], the results fall dramatically short of covering the nearly 7,000 human languages spoken around the globe [3]. For this reason, there is a need to explore ASR training methods which require significantly less language-specific data than conventional methods.

There has been past work which has addressed issues related to sparse amounts of annotated training data. The work of Lamel et al. for example showed the potential for training with lightly annotated data [4]. The more recent work of Novotney et al. showed the potential for leveraging small amounts of annotated data to train initial models, which can then be used to automatically annotate much larger quantities of unannotated data [5]. The approach by Gish et al. requires even less supervision by creating initial acoustic models using unsupervised techniques on unannotated audio data [13]. This latter work is most similar to our ongoing research directed towards rapid portability of ASR technology to languages with limited linguistic resources. Our initial efforts have focused on techniques that lend themselves to spoken term and key phrase detection, although our ultimate goal is to develop methods that would be useful for general ASR.

The problem of keyword spotting in audio data has been explored for many years, and researchers typically use ASR technology to detect instances of particular keywords in a speech corpus [7]. Although large-vocabulary ASR methods have been shown to be very effective [6], a popular method incorporates parallel filler or background acoustic models to compete with keyword hypotheses [11], [12]. These keyword spotting methods typically require large amounts of transcribed data for training the acoustic model. For instance, the classic filler model requires hundreds of minutes of speech data transcribed at the word level [12], while in some phonetic lattice [9], [17] or phone N-gram [15] matching based approaches, the training of an appropriate recognizer needs annotated data in phone/word level. The annotation work is not only time consuming, it also requires linguistic expertise for providing necessary annotations which can be a barrier to new languages.

As we enter an era where digital media can be created and accumulated at a rate that far exceeds our ability to annotate it, it is natural to question how much can be learned from the speech data alone, without any supervised input. A related question is what techniques can be performed well using unsupervised techniques in comparison to more conventional supervised training methods. These two questions are the fundamental motivation of our research.

In this paper, we present a completely unsupervised learning framework to address the problem of audio keyword spotting. Without any transcription information, a Gaussian mixture model (GMM) is trained to represent each speech frame with a Gaussian posteriorgram. Given one or more spoken examples of an input keyword, a segmental dynamic time warping (SDTW) technique is used to compare the Gaussian posteriorgrams between keyword examples and unseen test data. Keyword spotting is achieved by processing the distortion scores on the test utterances and selecting the best matching candidates as keyword hits. We use the TIMIT dataset as a development set to tune the parameters in our system, and a large vocabulary experiment is conducted on a corpus of academic lectures. The results demonstrate the feasibility and effectiveness of our unsupervised learning framework for the task of keyword spotting.

II. Related Work

Over the past few years, there has been related research investigating the use of unsupervised methods to perform keyword spotting [13], [14], [16]. In [13], the authors proposed a subword-unit modeling approach that consisted of three main components. First, a segmental GMM was used to label speech segments by the index of the most probable Gaussian component. Given a word level transcription, a Joint Multigram model ([10]) was trained to build a mapping between letters to the labeled Gaussian component indices. When a new keyword was given, the previously trained Multigram model could be used to convert the keyword to a sequence of Gaussian component indices. Finally, a string matching algorithm was employed to locate the possible occurrence of the keyword in the test set by calculating the edit distance.

The research in [14] presented an HMM-based keyword spotting system using filler/background-model approach. They used a single 128-state ergodic HMM to represent the keyword, filler and background model. In the modeling stage, the ergodic HMM was trained on all speech frames in an unsupervised manner. For keyword detection they required one or more spoken instances of a keyword and used the ergodic HMM to convert each instance into a series of HMM states. These HMM state sequences were then combined to create a conventional HMM to represent the keyword. For keyword spotting, each test utterance was decoded by the ergodic HMM as well as the keyword model and a putative hit was produced for high confidence scores.

The recent work of [16] made use of some prior knowledge in the form of phonetic posteriorgrams. A phonetic posteriorgram is defined by a probability vector representing the posterior probabilities of a set of pre-defined phonetic classes for a speech frame. By using an independently trained phonetic recognizer, each input speech frame can be converted to its corresponding posteriorgram representation. Given a spoken sample of a keyword, the frames belonging to the keyword are converted to a series of phonetic posteriorgrams by a full phonetic recognition. Then, they use dynamic time warping to calculate the distortion scores between the keyword posteriorgrams and the posteriorgrams of the test utterances. The detection result is given by ranking the distortion scores.

In sum, [13], [14] require no transcription for training either the segmental GMM or the ergodic HMM. [13] requires transcription for training the multigram model, while [14] and [16] need spoken instances of a keyword. [16] also requires no transcription for the working data but needs an independently trained phonetic recognizer.

III. System Design

Our approach is most similar to the research explored in [16]. However, instead of using an independently trained phonetic recognizer, we directly model the speech using a GMM without any supervision. As a result, the phonetic posteriorgram effectively becomes a Gaussian posteriorgram. Given spoken samples of a keyword, we apply the segmental dynamic time warping (SDTW) that we have explored previously [18] to compare the Gaussian posteriorgrams between keyword samples and the test utterances. We output the keyword detection result by ranking the distortion scores of the most reliable warping paths. We give a detailed description of each procedure in the following sections.

A. Gaussian Posteriorgram Definition

Posterior features have been widely used in template-based speech recognition systems [8], [20]. In a manner similar to the definition of the phonetic posteriorgram in [16], a Gaussian posteriorgram is a probability vector representing the posterior probabilities of a set of Gaussian components for a speech frame. Formally, if we denote a speech utterance with n frames as $S = (s_1, s_2, \cdots, s_n)$, then the Gaussian posteriorgram (GP) is defined by:

$$GP(S) = (q_1, q_2, \cdots, q_n) \tag{1}$$

Each q_i vector can be calculated by

$$q_i = (P(C_1|s_i), P(C_2|s_i), \cdots, P(C_m|s_i))$$

where C_i represents i-th Gaussian component of a GMM and m denotes the number of Gaussian components.

B. Gaussian Posteriorgram Generation

The generation of a Gaussian posteriorgram is divided into two phases. In the first phase, we train a GMM on all the training data and use this GMM to produce a raw Gaussian posteriorgram vector for each speech frame. In the second phase, a discounting based smoothing technique is applied to each posteriorgram vector.

The GMM training in the first phase is a critical process of our system. Without any transcription information, we train a GMM by assuming the labels for all speech frames are the same. This might introduce a problem that, without any guidance, it is easy to generate an unbalanced GMM. Specifically, it is possible to have a GMM with a small number of Gaussian components that dominate the probability space, with the remainder of the Gaussian components representing only a small number of training samples. We found this to be particularly problematic for speech in the presence of noise and other non-speech artifacts, due to their large variance. The unfortunate result of such a condition was a posteriorgram that did not discriminate well between phonetic units. Our initial solution to this problem was to apply a speech/non-speech detector to extract speech segments, and to only train the GMM on these segments.

The GMM consists of diagonal variance Gaussian components, initialized by the K-means algorithm. No variance floor is applied. After a GMM is trained, we use Equation (1) to calculate a raw Gaussian posteriorgram vector for each speech frame and the given spoken keyword samples. To avoid approximation errors, a probability floor threshold P_{min} is set to eliminate dimensions (i.e., set them to zero) with posterior probabilities less than P_{min}. The vector is re-normalized to set the summation of each dimension to one. Since this threshold would create many zeros in the Gaussian posteriorgram vectors, we apply a discounting based smoothing strategy to move a small portion of probability mass from non-zero dimensions to zero dimensions. Formally, for each Gaussian posteriorgram vector q, each zero dimension z_i is assigned by $z_i = \frac{\lambda \cdot 1}{Count(z)}$ where $Count(z)$ denotes the number of zero dimensions. Each non-zero dimension v_i is changed to $v_i = (1-\lambda)v_i$.

C. Modified Segmental DTW Search

After extracting the Gaussian posteriorgram representation of the keyword samples and all the test utterances, we perform a simplified version of the segmental dynamic time warping (SDTW) to locate the possible occurrences of the keyword in the test utterances.

SDTW has demonstrated its success in unsupervised word acquisition [18]. To apply SDTW, we first define the difference between two Gaussian posterior vectors p and q:

$$D(p, q) = -\log(p \cdot q)$$

Since both p and q are probability vectors, the dot product gives the probability of these two vectors drawing from the same underlying distribution [16].

SDTW defines two constraints on the DTW search. The first one is the commonly used adjustment window condition [21]. In our case, formally, suppose we have two Gaussian posteriorgram $GP_i = (p_1, p_2, \cdots, p_m)$ and $GP_j = (q_1, q_2, \cdots, q_n)$, the warping function $w(\cdot)$ defined on a $m \times n$ timing difference matrix is given as $w(k) = (i_k, j_k)$ where i_k and j_k denote the k-th coordinate of the warping path. Due to the assumption that the duration fluctuation is usually small in speech [21], the adjustment window condition requires that $|i_k - j_k| \leq R$. This constraint prevents the warping process from going too far ahead or behind in either GP_i or GP_j.

The second constraint is the step length of the start coordinates of the DTW search. It is clear that if we fix the start coordinate of a warping path, the adjustment window condition restricts not only the shape but also the ending coordinate of the warping path. For example, if $i_1 = 1$ and $j_1 = 1$, the ending coordinate will be $i_{end} = m$ and $j_{end} \in (1+m-R, 1+m+R)$. As a result, by applying different start coordinates of the warping process, the difference matrix can be naturally divided into several continuous diagonal regions with width $2R+1$, shown in the Figure 1. In order to avoid the redundant computation of the warping function as well as taking into account warping paths across segmentation boundaries, we use an overlapped sliding window moving strategy for the start coordinates ($s1$ and $s2$ in the figure). Specifically, with the adjustment window size R, every time we move R steps forward for a new DTW search. Since the width of each segmentation is $2R+1$, the overlapping rate is 50%.

Note that in our case, since the keyword sample is fixed, we only need to consider the segment regions in the test utterances. For example, if GP_i represents the keyword posteriorgram vector and GP_j is the test utterance, we only need to consider the regions along with the j axis. Formally, given the adjustment window size R and the length of the test utterance n, the start coordinate is

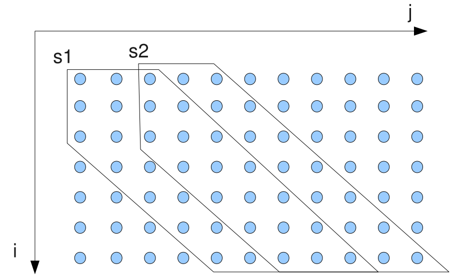

Fig. 1. The first two start coordinates of the warping path with $R = 2$

$$(1, (k-1) \cdot R + 1), \ 1 \leq k \leq \left\lfloor \frac{n-1}{R} \right\rfloor$$

As we keep moving the start coordinate, for each keyword, we will have a total of $\lfloor \frac{n-1}{R} \rfloor$ warping paths, each of which represents a warping between the entire keyword sample and a portion of the test utterance.

D. Voting Based Score Merging and Ranking

After collecting all warping paths with their corresponding distortion scores for each test utterance, we simply choose the warping region with the minimum distortion score as the candidate region of the keyword occurrence for that utterance. However, if multiple keyword samples are provided and each sample provides a candidate region with a distortion score, we need a scoring strategy to calculate a final score for each test utterance, taking into account the contribution of all keyword samples.

In contrast to the direct merging method used [16], we considered the reliability of each warping region on the test utterance. Given multiple keyword samples and a test utterance, a reliable warping region on the test utterance is the region where most of the minimum distortion warping paths of the keyword samples are aligned. In this way a region with a smaller number of alignments to keyword samples is considered to be less reliable than a region with a larger number of alignments. Therefore, for each test utterance, we only take into account the warping paths pointing to a region with alignments to multiple keyword samples.

An efficient binary range tree is used to count the number of overlapped alignment regions on a test utterance. After counting, we consider all regions with only one keyword sample alignment to be unreliable, and thus the corresponding distortion scores are discarded. We are then left with regions having two or more keyword samples aligned. We then apply the same score fusion method as in [16]. Formally, if we have $k \geq 2$ keyword samples s_i aligned to a region r_j, the final distortion score for this region is:

$$S(r_j) = -\frac{1}{\alpha} \log \frac{1}{k} \sum_{i=1}^{k} \exp(-\alpha S(s_i)) \qquad (2)$$

TABLE I
TIMIT 10 KEYWORD LIST

age(3:8)	warm(10:5)	year(11:5)	money(19:9)
artists(7:6)	problem(22:13)	children(18:10)	surface(3:8)
development(9:8)	organizations(7:6)		

where varying α between 0 and 1 changes the averaging function from a geometric mean to an arithmetic mean. Note that since one test utterance may have several regions having more than two keyword alignments, we choose the one with the smallest average distortion score. An extreme case is that some utterances may have no warping regions with more than one keyword alignment (all regions are unreliable). In this case we simply set the distortion score to a very big value.

After merging the scores, every test utterance should have a distortion score for the given keyword. We rank all the test utterances by their distortion scores and output the ranked list as the keyword spotting result.

IV. EVALUATION

We have evaluated this unsupervised keyword spotting framework on two different corpora. We initially used the TIMIT corpus for developing and testing the ideas we have described in the previous section. Once we were satisfied with the basic framework, we performed more thorough large vocabulary keyword spotting experiments on the MIT Lecture corpus [22].

The evaluation metrics that we report follow those suggested by [16]: 1) P@10 : the average precision for the top 10 hits; 2) P@N : the average precision of the top N hits, where N is equal to the number of occurrences of each keyword in the test data; 3) EER : the average equal error rate at which the false acceptance rate is equal to the false rejection rate. Note that we define a putative hit to be correct if the system proposes a keyword that occurs somewhere in an utterance transcript.

A. TIMIT Experiments

The TIMIT experiment was conducted on the standard 462 speaker training set of 3,696 utterances and the common 118 speaker test set of 944 utterances. The total size of the vocabulary was 5,851 words. Each utterance was segmented into a series of 25 ms frames with a 10 ms window shifting (i.e., centi-second analysis); each frame was represented by 13 Mel-Frequency Cepstral Coefficients (MFCCs). Since the TIMIT data consists of read speech in quiet environments, we did not apply the speech detection module in the TIMIT experiments. All MFCC frames in the training set were used to train a GMM with 50 components. We then used the GMM to decode both training and test frames to produce the Gaussian posteriorgram representation. For testing, we randomly generated a 10-keyword set and made sure that they contained a variety of numbers of syllables. Table I shows the 10 keywords and their number of occurrences in both training and test sets (# training : # test).

We first examined the effect of changing the smoothing factor, λ in the posteriorgram representation when fixing the SDTW window size to 6 and the score weighting factor α to 0.5, as illustrated in Figure 2. Since the smoothing factor

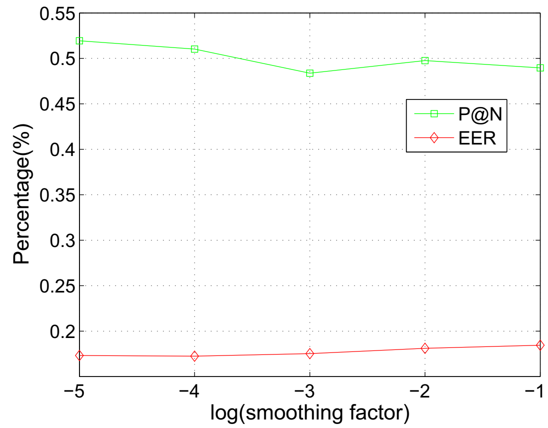

Fig. 2. Effect of different smoothing factors

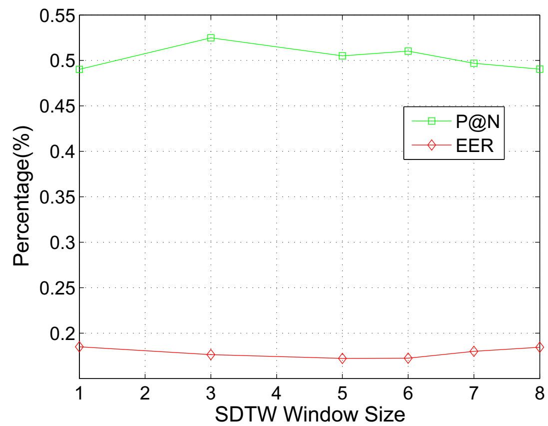

Fig. 3. Effect of different SDTW window sizes

ranges from 0.1 to 0.00001, we use a log scale on the x axis. Note that we do not plot the value for P@10 because as we can see in Table I, not all keywords occur more than ten times in the test set. In the figure, $\lambda = 0.0001$ was the best setting for the smoothing factor mainly in terms of EER, so this value was used for all subsequent experiments.

As shown in Figure 3, we next investigated the effect of setting different adjustment window sizes for the SDTW when fixing the smoothing factor to 0.0001 and the score weighting factor α to 0.5. The results shown in the figure confirmed our expectation that an overly small DTW window size could overly restrict the warp match between keyword references and test utterances, which could lower the performance. An overly generous DTW window size could allow warping paths with an excessive time difference, which could also affect the performance. Based on these experiments, a window size equal to 6 was the best considering both P@N and EER.

We also ran keyword spotting experiments with different settings of the score weighting factor α while fixing the smoothing factor to 0.0001 and SDTW window size to 6. As shown in Figure 4, P@N prefers the arithmetic mean metric, while the EER metric is relatively steady. By considering both metrics, we chose 0.5 as the best setting for α.

The number of Gaussian components in the GMM is another key parameters in our system. With fixed smoothing factor

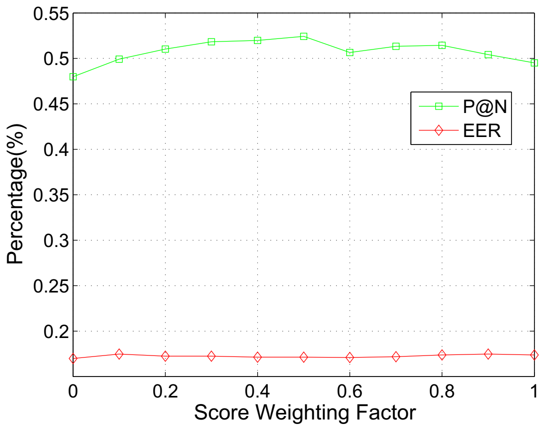

Fig. 4. Effect of different score weighting factors

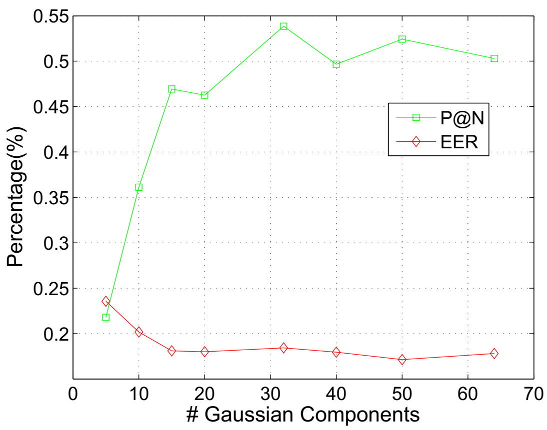

Fig. 5. Effect of different numbers of Gaussian components

(0.0001), SDTW window size (6) and score weighting factor (0.5), we ran several experiments with GMMs with different numbers of Gaussian components, as illustrated in Figure 5. Due to the random initialization of the K-means algorithm for GMM training, we ran each setting five times and reported the best number. The result indicated that the number of Gaussian components in the GMM training has a key impact on the performance. When the number of components is small, the GMM training may suffer from an underfitting problem, which causes a low detection rate in P@N. In addition, the detection performance is not monotonic with the number of Gaussian components. We think the reason is that the number of Gaussian components should approximate the number of underlying broad phone classes in the language. As a result, using too many Gaussian components will cause the model to be very sensitive to variations in the training data, which could result in generalization errors on the test data. Based on these results, we chose 50 as the best number of GMM components.

B. MIT Lecture Experiments

The MIT Lecture corpus consists of more than 300 hours of speech data recorded from eight different subjects and over 80

TABLE II
MIT LECTURE 30 KEYWORD LIST

zero (247:77)	space (663:32)	solutions (33:29)
examples (137:29)	performance (72:34)	matter (353:34)
molecule (28:35)	pretty (403:34)	results (121:35)
minus (103:78)	computer (397:43)	value (217:76))
situation (151:10)	therefore (149:46)	important (832:47)
parameters (21:50)	negative (50:50)	equation (98:61)
distance (58:56)	algorithm (35:36)	direction (214:37)
maximum (20:32)	responsible (92:10)	always (500:37)
likelihood (13:31)	mathematical (37:15)	never (495:21)
membrane (19:27)	problems (270:23)	course (847:76)

TABLE III
EFFECT OF DIFFERENT NUMBERS OF KEYWORD SAMPLES

# Examples	P@10	P@N	EER
1	27.0%	17.3%	27.0%
5	61.3%	33.0%	16.8%
10	68.3%	39.3%	15.8%

general seminars [22]. In most cases, the data is recorded in a classroom environment using a lapel microphone. For these experiments we used a standard training set containing 57,351 utterances and a test set with 7,375 utterances. The vocabulary size of both the training and the test set is 27,431 words.

Since the data was recorded in a classroom environment, there are many non-speech artifacts that occur such as background noise, filled pauses, laughter, etc. This non-speech data could cause serious problems in the unsupervised learning stage of our system. Therefore, prior to GMM training, we ran a speech detection module [23] to filter out non-speech segments. GMM learning was performed on frames within speech segments. Note that the speech detection module was trained independently from the Lecture data and did not require any transcription of the Lecture data. 30 keywords were randomly selected; all of them occur more than 10 times in both the training and test sets. All keywords occur less than 80 times in the test set to avoid using keywords that are too common in the data. Table II shows all the keywords and the number of their occurrences in the training and test sets.

Table III shows the keyword detection performance when different numbers of keyword samples are given. As a result of the TIMIT experiments, we fixed the smoothing factor to 0.0001, the SDTW window size to 6 and the score weighting factor to 0.5. All of the three evaluation metrics improve dramatically from the case in which only one keyword sample is given to the case in which five samples are given. Beyond five examples of a keyword, the trend of the performance improvement slows. We believe the reason for this behavior is that the improvement from one sample to five samples is mainly caused by our voting based score merging strategy. When going from five samples to ten samples, we gain additional performance improvement, but there are always some difficult keyword occurrences in the test data. Table IV gives the list of 30 keywords ranked by EER in the 10-example experiment. We observe that the words with more syllables tended to have better performance than ones with only two or three syllables.

Since the data used in [16] is not yet publicly available, we are unable to perform direct comparisons with their experiments. Nevertheless, we can make superficial comparisons. For example, in the case of 5 keyword samples, the P@10 performance (61.3%) of our system is competitive with their result (63.3%), while for the P@N and EER metrics, we are lower than theirs (P@N : 33.0% vs. 52.8%, EER : 16.8% vs. 10.4%). We suspect that one cause for the superior performance of their work is their use of a well-trained phonetic recognizer. However, this will require additional investigation before we can quantify this judgement.

V. Conclusion and Future Work

In this paper we have presented an unsupervised framework for spoken keyword detection. Without any annotated corpus, a completely unsupervised GMM learning framework is introduced to generate Gaussian posteriorgrams for keyword samples and test utterances. A modified segmental DTW is used to compare the Gaussian posteriorgrams between keyword samples and test utterances. After collecting the warping paths from the comparison of every pair of the keyword sample and the test utterance, we use a voting based score merging strategy to give a relevant score to every test utterance for each keyword. The detection result is determined by ranking all the test utterances with respect to their relevant scores. In the evaluation, due to various system parameters, we first designed several experiments on the smaller TIMIT dataset to have a basic understanding of appropriate parameter settings as well as to verify the viability of our entire framework. We then conducted experiments on the MIT Lecture corpus, which is a much larger vocabulary dataset, to further examine the effectiveness of our system. The results were encouraging and were somewhat comparable to other methods that require more supervised training [16].

While this work represents our first attempt to use unsupervised methods to solve speech related problems such as spoken keyword spotting, there is still much room for further improvement. Specifically, in the unsupervised GMM learning, the current training method needs to manually set the number of Gaussian components. Based on the experiment we presented on TIMIT, it is clear that a good choice for the number of components can significantly improve performance. In the future, we hope this number or the model structure can be found in an unsupervised way [25] such as using ideas from successive state splitting [24]. In the SDTW search, the current system requires spoken samples of a keyword, which may suffer from out-of-vocabulary (OOV) problem. Inspired by the Joint Multigram model used in [10], [13], we also plan to develop a letter to Gaussian posteriorgram model to solve the OOV problem. Furthermore, since this framework is completely language independent and generic, we would like to examine its performance on languages other than English as well as other signal processing pattern matching tasks.

TABLE IV
30 Keywords Ranked by EER

responsible (0.2%)	direction (10.3%)	matter (22.8%)
situation (0.5%)	parameters (10.5%)	always (23.0%)
molecule (4.9%)	algorithm (11.3%)	therefore (23.9%)
mathematical (6.7%)	course (11.4%)	membrane (24.0%)
maximum (7.5%)	space (13.8%)	equation (24.9%)
solutions (8.1%)	problems (17.8%)	computer (25.3%)
important (8.5%)	negative (18.0%)	minus (25.7%)
performance (8.8%)	value (19.4%)	examples (27.0%)
distance (9.0%)	likelihood (19.4%)	pretty (29.1%)
results (9.3%)	zero (22.7%)	never (29.5%)

References

[1] http://www.nuance.com/recognizer/languages.asp.

[2] http://www.ldc.upenn.edu/.

[3] http://www.vistawide.com/languages/language_statistics.htm.

[4] L. Lamel, J. Gauvain, and G. Adda, "Lightly supervised and unsupervised acoustic model training," *Computer Speech and Language,* 16(1), 115–129, 2002.

[5] S. Novotney, R. Schwartz, and J. Ma, "Unsupervised acoustic and language model training with small amounts of labelled data," in *Proc. ICASSP*, 4297–4300, Taipei, 2009.

[6] M. Weintraub, "LVCSR log-likelihood ratio scoring for keyword spotting", in *Proc. ICASSP*, 129–132, 1995.

[7] I. Szoke, P. Schwarz, L. Burget, M. Fapso, M. Karafiat, J. Cernocky and P. Matejka, "Comparison of keyword spotting approaches for informal continuous speech", in *Proc. Interspeech*, 633–636, 2005.

[8] G. Aradilla, H. Bourlard and M. Magimai-Doss, "Posterior features applied to speech recognition tasks with user-defined vocabulary", in *Proc. ICASSP*, Taipei, 2009.

[9] K. Thambiranam and S. Sridharan, "Dynamic match phone-lattice searches for very fast and accurate unrestricted vocabulary keyword spotting", in *Proc. ICASSP*, 2005.

[10] S. Deligne, F. Yvon and F. Bimbot, "Variable-length sequence matching for phonetic transcription using joint multigrams", in *Proc. Eurospeech*, 2243–2246, 1995.

[11] J. Wilpon, L. Miller and P. Modi, "Improvements and applications for keyword recognition using Hidden Markov modeling techniques", in *Proc. ICASSP*, 309–312, Toronto, 1991.

[12] R. Rose and D. Paul, "A Hidden Markov Model based keyword recognition system", in *Proc. ICASSP*, 129–132, Albuquerque, 1990.

[13] A. Garcia and H. Gish, "Keyword spotting of arbitrary words using minimal speech resources", in *Proc. ICASSP*, 123–127, Atlanta, 2006.

[14] P. Li, J. Liang and B. Xu, "A novel instance matching based unsupervised keyword spotting system", in *Proc. Int. Conf. on Innovative Computing, Information and Control*, 550-553, 2007.

[15] U. Chaudhari and M. Picheny, "Improvements in phone based audio search via constrained match with high order confusion estimates", in *Proc. ASRU*, 665–670, 2007.

[16] T. Hazen, W. Shen and C. White, "Query-by-example spoken term detection using phonetic posteriorgram templates", in *Proc. ASRU*, 2009.

[17] H. Lin, A. Stupakov and J. Bilmes, "Improving multi-lattice alignment based spoken keyword spotting," in *Proc. ICASSP*, Taipei, 2009.

[18] A. Park and J. Glass,"Unsupervised pattern discovery in speech", in *IEEE Trans. ASLP*, 6(1), 1558–1569, 2008.

[19] J. Glass, "A probabilistic framework for segment-based speech recognition," in *Computer Speech and Language*, 17, 137–152, 2003.

[20] G. Aradilla, J. Vepa and H. Bourlard, "Using posterior-based features in template matching for speech recognition", in *Proc. Interspeech*, 2006.

[21] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition", in *IEEE Trans. ASSP*, 26(1), 43–49, 1978.

[22] J. Glass, T. Hazen, L. Hetherington and C. Wang, "Analysis and processing of lecture audio data: preliminary investigations", in *Proc. HLT-NAACL*, 9–12, Boston, 2004.

[23] J. Glass, T. Hazen, S. Cyphers, I. Malioutov, D. Huynh, and R. Barzilay, "Recent progress in the MIT spoken lecture processing project", in *Proc. Interspeech*, 2553–2556, 2007.

[24] H. Singer and M. Ostendorf, "Maximum likelihood successive state splitting", in *Proc. ICASSP*, 601–604, Atlanta, 1996.

[25] S. Petrov, A. Pauls and D. Klein, "Learning structured modles for phone recognition", in *Proc. EMNLP*, 897–905, 2007.