**Piush Karmacharya**

**Department of Electrical and Computer Engineering, Temple University**

**A short Literature Review on**

**Discriminative Keyword Spotting**

***tub86339@temple.edu***

1. **Summary**

This report is a review of paper *Discriminative Keyword Spotting* by *Joseph Keshet,. David Grangier and Sany Bengio*. In this paper a new discriminative learning approach to keyword spotting is proposed unlike traditional HMM based systems. This approach aims at maximizing objective function i.e. the area under the ROC curve rather than developing a generalized model. The paper claims that this method outperforms conventional context-dependent HMM-based approach when tested on TIMIT corpus. The rest of the report is organized as follows: The next section is of elaboration of keyword spotting problem followed by introduction to some general approaches used to solve the problem in section III. Section IV contains introduction to kernel based approach and Support Vector Machines (SVM) used in this experiment. Section V explains the experimental setup, elaboration of the algorithm and implementation details which are pretty much similar to that described in the paper followed by experimental results in section VI. Finally in section VII covers discussion over the approach and personal comments.

1. **Introduction**

Keyword spotting (KWS) is a branch of speech processing that deals with identifying certain keywords in a long utterance. In present context when people are trying to make human-machine interaction seamlessly natural, the scope of this field is even bigger because human conversation contains not only irrelevant words but also non-intentional sounds like cough, exclamations and noise. If we can extract only the embedded information, computation can be much efficient and robust. Some examples of these kinds of systems can be voice dialing, voice command systems, audio search engines etc. Keyword spotting also has many direct applications such as audio document retrieval and covert speech surveillance systems etc.

Efficiency of a keyword spotting system is defined in terms of percentage of keyword detected for given false alarm. Optimal design aims to maximize detection rate while keeping false alarm rate as low as possible. The plot of keyword detection vs. false alarm called the Receiver Operating Characteristic (ROC) curves are generally used for measuring performance of KWS. Intuitively, System with maximum area under the ROC curve (AUC) gives best accuracy.

1. **Hidden Markov Based Systems**

Hidden Markov Model is a statistical model in which the system is assumed to be a Markov process with unobservable states. The output of each state however is observable and each state has a probability distribution over the possible outcomes. A simple form of two states HMM is shown in Figure 1 below.

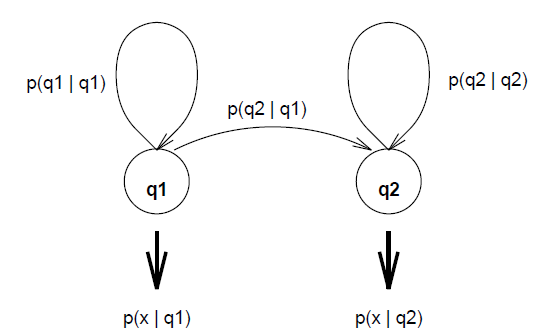


Figure 1.Two state Hidden Markov Model.

HMMs are “hidden” because the state of the model, q, is not observed; rather the output, x, of a stochastic process attached to that state is observed. This is described by a probability distribution P(x|q). The other set of visible probabilities are the state transition probabilities, P(qi|qj). Assuming that the system is first order Markov process, the probability of next state depends only on the current state.

From the observable sequence of outputs, we can infer the most likely system. The result is a model for the underlying process. Alternatively, given a sequence of outputs, we can infer the most likely sequence of states.

HMM are most widely used these days for speech recognition as well as keyword spotting. Assuming speech signal to be piecewise stationary, each word is modeled as a sequence of stationary units with certain acoustic feature parameter. Although there are several linguistic arguments for choosing units such as syllables or semi-syllables, the unit most commonly used units are the phonemes. Isolated word recognition using HMM models can be broken down into two steps.

1. For each word in the vocabulary build an HMM, i.e. estimate the model parameters that optimizes the likelihood of the training set.
2. Observation sequence is obtained for each word to be recognized by feature extraction and likelihood for all possible models is calculated. The word with highest likelihood is selected as a match. This is done using Viterbi algorithm for optimizing the computation.

For large vocabulary continuous speech recognition (LVSCR), syntactical models are also used along with word model to apply grammatical constraints. This additional linguistic constraint makes recognition task easier and also increases the performance of the system. The objective is to output the most probable sentence given the acoustic data X. Thus we choose sentence S, for which the probability P(S|X) is maximum. In order to use HMM, the sentence is represented by a particular state sequence, QN, and the probability we require is P(QN | X) which is not directly computable. So we re-express this probability using Bayes’ rule as

i.e. we divide the estimation process into two parts: acoustic modeling, in which the data probability density P(X|QN) / P(X) are estimated; and language modeling in which prior probabilities of state sequences, P(QN) are estimated.

KWS can be directly implemented using Large Vocabulary Continuous Speech Recognizer (LVSCR). But this requires training the model for large dictionary of words and huge language model which is rather impractical. An alternative approach is to consider the speech signal to be composed to keywords and non-keywords and using garbage or filler models to characterize non-keywords. Some systems add extra models for representing non-speech events such as silence or coughing. Both keyword model and garbage model are built from concatenation of phoneme sub-models. Likelihood of test utterance matching the keyword model gives the confidence measure of the keyword.

The major drawback of HMM based approaches is that it does not tie up with the final spotting objective of maximizing the area under the ROC curve. Besides HMM approach requires huge amount of training data and thus favors most occurring events because of the lack of training data to model rare events. Also they are known for poorly modeling long temporal dependencies, which needs to be circumvented with refined features or adaptation techniques. In order to overcome these issues, significant effort has been made towards discriminative approaches to keyword spotting.

1. **Large Margin Methods and Support Vector Machines**

The standard Support Vector Machine SVM is a non-probabilistic binary linear classifier that categorizes given input date into one of the two possible classes. SVN constructs a hyperplane or set of hyperplanes in a high dimensional space, which can be used for classification, regression or other tasks. A good separation is achieved by a hyperplane that has the largest distance to the nearest training data points of any class, since in general the larger the margin the lower the generalization error of the classifier.

But in most practical cases, the data are not linearly separable. In these cases, one solution could be use non-linear planes for separation. A more efficient method involves transforming the original data into higher dimensional vector space and use linear pattern analysis algorithms.

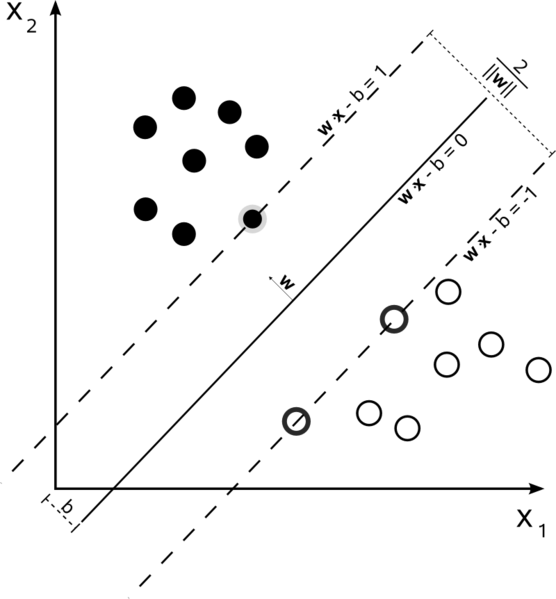


Figure 2.Maximum-margin hyperplane and margins for an SVM trained with samples from two classes

Consider the problem of separating the set of *m* training vectors belonging to two different classes, {(x1,y1),…,(xm,ym)} where xiϵRN is a feature vector and yiϵ{-1,1} a class label, with a hyper-plane of equation *w***.**x + b = 0 where {**.**} denotes the dot product and *w* is a normal vector perpendicular to the hyperplane. Of all the boundaries determined by w and b, the one that maximizes the margin would generalize better, as compared to other possible separating hyper-planes. These hyperplanes can be described by the equations *w***.** x – b = 1 and *w***.** x + b = -1. The separating hyperplane satisfies the constraint yi(*w*. xi + b) >1.

The optimal separating hyperplane is given by maximizing the margin M given by equation: M = 1/||w||. This is equivalent to minimizing

As mentioned earlier, linear boundary is inappropriate in some cases. In such cases, the SVM replaces the inner product xi**.**xj by a kernel function *K*(*xi****.****xj*), and then constructs an optimal hyperplane in the mapped space. The main advantage of transforming the data into higher feature space is that the data then are linearly separable.

If there exists no hyperplane that can split all given data into distinct class, a modified method called the Soft Margin will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. The method introduces slack variables, i, which measure the degree of misclassification of the datum x*i.* The optimization problem is now treated as a minimization of the classification error. The separating hyperplanes must satisfy the following inequality.

The generalized optimal separating hyperplane is determined by vector *w*, that maximizes the function:

where and C are constants.

1. **Problem Settings**

The objective of this research was to identify if a keyword was uttered in given speech utterance and also the position if present. Since a keyword can be considered as a sequence of phonemes, the objective can be stated as to detect if the corresponding phoneme sequence is articulated in the given utterance and where. The author claims that this method can be applied to detect any keyword and not just the keyword already seen in the training phase.

For this, the speech signal is represented as a sequence of acoustic feature vectors *ẋ*={x1,x2,…,xT}, where xt ϵ *K С* ***Rd*** for all 1 ≤ t ≤ T, where ***Rd*** is the domain of feature vectors. Each keyword *k* ϵ *K*, whereK is the lexicon of words, is composed of a sequence of phonemes pk = (p1, p2,,.., pL, where pl ϵ ***P*** for all 1 ≤ l ≤ L, where ***P*** is the domain of phoneme symbols. Let ***P\****denote the set of all finite length sequences over ***P***. The objective is to learn a keyword spotter, denoted ***f****,* which takes as input the pair (*ẋ*,pk) and returns a real value expressing the confidence that the targeted keyword *k* is uttered in *ẋ*. The confidence returned by ***f*** can then be compared to a threshold *b* todetermine if the keyword is uttered in the speech.

Let *sl* ϵ *N* denote the start time of phoneme *pl* (in terms of frame units) and *el* ϵ N denote the end time. If we assume start time of phoneme pl+1 is equal to the end time of phoneme *pl*, then *el = sl+1* for all 1 ≤ l ≤ L. This means we do not need to store end time for all the phonemes but the last. The alignment sequence *sk* corresponding to the phoneme sequence *pk* is then a sequence of start-times and a end-time, sk = (s1, …. ,sL, eL).

We define a set of predefined non-linear functions Фj : *X\** \* ***P****\** \* *N\** → **R**. That is, each feature function takes as input an acoustic feature of speech *ẋ* ϵ *X*\*, together with a keyword phoneme sequence *pk* ϵ ***P****\**, and a candidate sequence alignment *sk* ϵ *N\**, and returns a scalar in **R** which is the measure of confidence in the suggested alignment.

The objective is to learn a keyword spotter ***f***, which takes sequence of acoustic features *ẋ,* a keyword *pk*, and returns a confidence value in **R**. The form of function ***f*** used is given as:

Where *w* ϵ **Rn** is avector of importance weights that needs to be learned and ϵ **Rn**is a vector function composed of the feature functions i. ***f***returns a confidence measure of the keyword being present in the utterance by maximizing a weighted sum of the feature function elements over all possible alignment sequences. If the feature functions are decomposable, the equation can be calculated using efficient dynamic programming procedure.

The performance of spotting system can be measure in terms of Receiver Operating Characteristics (ROC). A criterion to identify a good keyword spotting system that would be good on average for all the settings would be to select the one maximizing the area under the ROC curve or AUC. The algorithm proposed in this paper aims at maximizing the AUC.

To train the system to maximize the AUC, let us consider two sets of utterances with and without the keyword of interest. Let *X+* denote the set in which keyword *k* is uttered and *X-* denote the set in which the keyword *k* is not uttered. The AUC can be written in the form of Wilcoxon-Mann-Whitney statistics as

Where, 1{.} refers to the indicator function and *pk* refers to the phoneme sequence corresponding to the keyword *k*. Thus, Ak is the measure of the probability that the score assigned to an utterance is greater than the score assigned to an utterance in which the keyword is not uttered. The average AUC over the set of keywords ***K***can be written as

In order to achieve high AUC, we need to pick a function ***f*** from a set of linear functions defined above for which ***f***(*x+ , pk*) > ***f***(*x- , pk*), for every keyword k ϵ ***K*** and for as much utterance pairs x+ ϵ *X*+ and x- ϵ *X*- as possible. Finding the function is realized by learning the weight vectors *w.*

The weight vectors can be learned by using large margin techniques. Each training set is composed of sequence of phonemes representing the keyword *k*, an utterance ϵ in which the keyword *k* is uttered, and an utterance ϵ in which the keyword *k* is not uttered and the time span of the phoneme sequence.

The idea of choosing weight vector *w* is based on the idea of large-margin separation. Theoretically the approach can be described as a two-step procedure: first construct the vectors Ф(, , ) and Ф(, , ) in the vector space **Rn** based on each instance (, , ), and each possible time span s for the negative sequence . Second, find a vector *w* ϵ **Rn**, such that the projection of vectors onto *w* ranks the constructed vectors according to their quality. Ideally for any keyword *ki,* we would like the following constraint to hold

*w* . , , ) – max *w* . , , ) ≥ 1

That is, *w* should rank the utterance with keyword above any utterance without keyword by at least q. The SVN algorithm solves this problem by selecting weights *w* minimizing |*w*|2/2 subject to the constraints given above. But in practice the constraints cannot be satisfied. To overcome this obstacle, we need to follow the soft SVN approach and choses the vector *w\** which minimizes the following optimization problem:

*w\** = arg min + C , )

Where C serves as a complexity-accuracy trade-off parameter; a low value of C favors simple model, while a large value of C favors a model which solves all training constraints. The optimization problem is expensive since it involves maximization for each training example, most solvers iterate over the whole dataset several times until it converges. So this paper proposes an iterative algorithm that is comparative to the large margin approach and attains high AUC over the training examples and over unseen examples.

**Feature Functions**

This section contains the details of feature functions {Ф used in the experiment which maps an acoustic-phoneme representation of a speech as well as suggested time span of a keyword into a vector-space. In order to reduce the equation complexity, we consider only one keyword and omit the keyword index *k.* Seven different feature functions (n=7) were used for defining the keyword spotting function ***f*** (*x,p*) as described earlier.

The first four feature function aims at capturing the transition between phonemes. They are the distance between frames of the acoustic signal at both sides of phoneme boundaries as suggested by alignment sequence *s*. The distance measure is the Euclidean distance between the feature vectors and denoted by *d.* Formally,

The assumption is that if two frames, xt and xt’ are derived from the same phoneme, the distance d(xt,xt’) should be smaller than if the two frames are derived from different phonemes. If s is the correct timing sequence then distances between frames across the phoneme change points are likely to be large. The features use only the start time of the ith phoneme and do not use the values si-1 and si+1.

The fifth feature function used is based on phoneme classifier. It is the measure of confidence denoted as gp(x), that the phoneme *p* is produced in the frame x. The feature function is the measure of the cumulative confidence of the complete speech signal given the phoneme sequence and their start-time,

The next feature function used scores the timing sequences based on the phone durations. It examines the length of each phoneme compared to the typical length required to pronounce this phoneme.

where *N* is a Normal probability density function with mean µp and standard deviation σ estimated from the training set.

The last feature function takes into account the speaking rate of the speaker. Assuming people speak in almost steady rate, timing sequence in which speech rate changes abruptly is probably incorrect. If µp denote the average length of the pth phoneme, the relative speech rate ri = (si+1 – si)/ µp. In practice, speaking rate ratios differ from speaker to speaker and within a given utterance. The local change in speaking rate is measure as (ri – ri-1)2 and the feature function is defined as the local change in speaking rate,

**Iterative Algorithm**

As stated earlier, the paper proposes and iterative approach to learn the weight vector *w* based on training examples. Each set of training data is composed of one utterance with keyword and one without keyword uttered in it. The algorithm receives as input a set of training examples S = {(, , , )},i=1 - m, and examines each of them sequentially. Initially *w* is set to 0. At each iteration, the algorithm updates *w* according to the current training data {(, , , ). If wi-1 denote the weight before ith iteration, sl denote the predicted alignment of the negative utterance, xi-, then

The difference between feature function of the acoustic sequences in which the keyword is present and the one in which the keyword is not present is given as

The weight vector *wi* is then set to be the minimize of the following optimization problem.

where C serves as a complexity-accuracy trade-off parameter ξ and is a non-negative slack variable, which indicates the loss of the ith example. Intuitively, we would like to minimize the loss of the slack variable while keeping the weight vector *w* as close as possible to the previous weight vector wi-1. The constraint makes the projection of the utterance in which the keyword in present onto w higher than the projection of the utterance without the keyword onto w by at least 1. It has been shown that the solution to optimization problem is

The value of the scalar αi is based on the scores that x+ and x- received according to wi-1 , and a parameter C.

Based on current work on online learning algorithms, it can be shown that under some mild technical conditions, the cumulative performance of the iterative procedure is likely to be high. If so, there exists at least one weight vector among the vectors {w1… wm} which attains high averaged performance on the test examples. To find that weight vector, averaged loss attained by each of the weight vectors on the validation set is calculated. The algorithm can be summarized as shown below.

* **Input: training set**
* **Initialize: *w0* = 0**
* **For i = 1,….,m**

**Predict**

**Set**

* **If**

**Set:**

**Update:**

* **Output: The weight vector *w\** which achieves best AUC performance on the validation set Sval.**

1. **Experimental Results**

In order to validate the proposed algorithm, experiments were carried on TIMIT corpus. The training portion was divided into three disjoint parts containing 500, 80 and 3116 utterances. First set was used to learn the phoneme classifier *gp* which defines the feature function Ф5. For that, MFCC+ ∆ + ∆∆ and Gaussian kernel (σ = 6.24 and C = 5.0) were used. The second set of 80 utterances made up the validation set for keyword spotting algorithm. It was composed of 40 keywords, 80 pairs of utterances with keyword uttered in it and that without keyword. The last set of 3116 utterances was used for learning the weight vectors *w*. The value of C was set to 1.

The HMM system for comparison were done using Torch package which is a machine-learning library, developed at IDIAP using C++. The HMM system is composed of two sub HMM models – keyword model and garbage model and the best path was found using Viterbi decoding.

Testing set was composed of 80 keywords that were different from those of training set. For each keyword, around 20 utterance with keyword and 20 without keyword were tested. The results obtained after testing on both the system is shown in the ROC curve below and the AUC is found to be 0.99 for discriminative approach while 0.96 for HMM algorithm. In order to further support the superiority of this method over HMM, Wilcoxon test was performed which is is a non-parametric statistical hypothesis test for the case of two related samples or repeated measurements on a single sample.



Figure 3. ROC curves of the discriminative algorithm and the HMM approach

1. **Discussion**

In this paper the author have proposed a novel approach for keyword spotting which is not based on HMM system widely used today. It aims at maximizing the Area Under Curve (AUC) of the Receiver Operation Characteristic (ROC) which is the primary objective of any keyword spotting system.

The fundamental idea is to derive feature functions that best characterize the keyword and based design a keyword spotter that assign specific weight to each of these feature functions to maximize the difference between utterance with keyword uttered and that without it. In order to achieve optimal weight vector that maximizes the difference, large margin methods and kernel methods like support vector machines (SVN) can be used. But that would be computationally expensive. So an iterative approach is proposed that would learn the weight vectors. The algorithm is based on the hypothesis that the weight that maximizes the cumulative distance between keyword and non-keyword would maximize the AUC.

For that purpose, the training data is composed of sets each with one utterance with keyword uttered and one without keyword uttered. Again there is no specification on the length of the utterance. Also the system requires prior information about the phoneme sequence and keyword location in the positive utterance (positive in the sense it contains keyword). Hence the efficiency of the system relies on the performance of phoneme recognizer that produced the phoneme sequence and there is no clear indication of how that is obtained in the paper. This could highly effect the efficiency of the system overall. Moreover phonetic representation might not be the best way to model a keyword because in practice not all phonemes are pronounced clearly in a keyword especially in casual conversation unlike in the TIMIT corpus which is a read speech database. The experiment does not take into consideration those cases.

There are seven feature functions that make the keyword spotter. The first four feature functions are related to phoneme transition. For this Euclidean distance between feature vectors at both the boundary of phoneme is computed. If the timing sequence is correct, the distance between phoneme change points are likely to be large while small for same phoneme. In my view these feature function depends on the efficiency of the phoneme recognizer might act to compensate for its inefficiency. Again there is no explanation to why four of these frames were chosen for feature functions because these four functions appear to be highly correlated.

The fifth feature function measure the confidence of the speech signal given the phoneme sequence and their alignment. For each phone this is measure of confidence that the phoneme was pronounced in frame *x* given the acoustic features. In my view this feature again relies on the efficiency of the phoneme recognizer and has less significance on maximizing the objective function.

The sixth feature function is the measure of phoneme duration. The seventh and last feature function measures the speaking rate of the speaker. It is based on the assumption that people tend to speak in an almost steady rate and therefore abrupt change in speech rate is possibly incorrect. In my view speaking rate varies for different speaker and sometimes even for same speaker depending on the context and external conditions. So the rate variation need to be accounted for and normalized rather than considering as a measure of confidence.

The algorithm works by lowering the maximum confidence for negative utterance to that of the positive utterance by at least 1. For this there has to be only one false keyword in the negative utterance.

In order to verify the results and compare it with HMM system, the test set of 80 keywords was used. But there is no information in the paper about the properties of keywords in terms of length or phoneme distribution. HMM systems efficiency relies on the amount of training data which might not be enough for the test. Results based on Wilcoxon test might not be as reliable as that on real speech date. Therefore more research has to be done to verify the superiority of this method over HMM methods.

1. **References:**
2. Discriminative Keyword Spotting, Joseph Keshet, David Grangier, Samy Bengio; IDIAP Research Institute
3. Connectionist Probability Estimation in HMM Speech Recognition; Steve Renals, Nelson Morgan
4. Confidence Measures for Keyword Spotting Using Support Vector Machines; Y. Benayed, D. Fohr, J.P. Haton, G. Chollet
5. A Tutorial on Hidden Markov Models and selected applications in Speech Recognition; Lawrence R. Rabiner
6. Learning the Inter-frame Distance for Discriminative Template-based Keyword Detection; David Grangier, Samy Bengio
7. <http://en.wikipedia.org/wiki/Support_vector_machine>