Discriminative Keyword Spotting

Joseph Keshet[1], *David Grangier*[2] *and Samy Bengio*[2]

[1] School of Computer Science & Engineering, The Hebrew University, Jerusalem, Israel
jkeshet@cs.huji.ac.il
[2] IDIAP Research Institute, Martigny, Switzerland
{grangier,bengio}@idiap.ch

Abstract

This paper proposes a new approach for keyword spotting, which is not based on HMMs. The proposed method employs a new discriminative learning procedure, in which the learning phase aims at maximizing the area under the ROC curve, as this quantity is the most common measure to evaluate keyword spotters. The keyword spotter we devise is based on non-linearly mapping the input acoustic representation of the speech utterance along with the target keyword into an abstract vector space. Building on techniques used for large margin methods for predicting whole sequences, our keyword spotter distills to a classifier in the abstract vector-space which separates speech utterances in which the keyword is uttered from speech utterances in which the keyword is not uttered. We describe a simple iterative algorithm for learning the keyword spotter and discuss its formal properties. Experiments with the TIMIT corpus show that our method outperforms the conventional HMM-based approach.

1. Introduction

Keyword (or word) spotting refers to a proper detecting of any occurrence of a given word in a speech signal. Most previous work on keyword spotting has been based on hidden Markov models (HMMs). See for example [1, 2, 3] and the references therein. Despite their popularity, HMM-based approaches have several known drawbacks such as convergence of the training algorithm (EM) to a local maxima, conditional independence of observations given the state sequence and the fact that the likelihood is dominated by the observation probabilities, often leaving the transition probabilities unused. However, the most acute weakness of HMMs for keyword spotting is that they do not aim at maximizing the detection rate of the keywords.

In this paper we propose an alternative approach for keyword spotting that builds upon recent work on discriminative supervised learning and overcomes some of the inherent problems of the HMM-based approaches. The advantage of discriminative learning algorithms stems from the fact that the objective function used during the learning phase is tightly coupled with the decision task one needs to perform. In addition, there is both theoretical and empirical evidence that discriminative learning algorithms are likely to outperform generative models for the same task (see for instance [4, 5]). One of the main goals of this work is to extend the notion of discriminative learning to the task of keyword spotting.

Our proposed method is based on recent advances in kernel machines and large margin classifiers for sequences [6, 7], which in turn build on the pioneering work of Vapnik and colleagues [4, 5]. The keyword spotter we devise is based on non-linear mapping the speech signal along with the target keyword into a vector-space endowed with an inner-product. Our learning procedure distills to a classifier in this vector-space which is aimed at separating the utterances in which the keyword is uttered from those in which the keyword is not uttered. On this aspect, our approach is hence related to support vector machine (SVM), which has already been successfully applied in speech applications [8, 9]. However, the model proposed in this paper is different from a classical SVM since we are not addressing a simple decision task such as binary classification or regression. Our algorithm is in fact closer to recent work on kernel machine methods for sequence prediction, such as [10, 6, 11], with the main difference that we avoid the costly optimization problems introduced by such models. Instead, we propose an efficient iterative algorithm for learning a discriminative keyword spotter by traversing the training set a single time.

This paper is organized as follows. In Sec. 2 we formally introduce the keyword spotting problem. We then present an iterative algorithm for keyword spotting in Sec. 3. The implementation details of our learning approach and the non-linear set of feature functions we use are presented in Sec. 4. Next, we present experimental results in Sec. 5. Finally, concluding remarks and future directions are discussed in Sec. 6.

2. Problem Setting

Any keyword (or word) is naturally composed of a sequence of phonemes. In the keyword spotting task, we are provided with a speech utterance and a keyword and the goal is to decide whether the keyword is uttered or not, namely, whether the sequence of phonemes is articulated in the given utterance.

Formally, we represent a speech signal as a sequence of acoustic feature vectors $\bar{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$, where $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^d$ for all $1 \le t \le T$. We denote a keyword by $k \in \mathcal{K}$, where $\mathcal{K}$ is a lexicon of words. Each keyword k is composed of a sequence of phonemes $\bar{p}^k = (p_1, \ldots, p_L)$, where $p_l \in \mathcal{P}$ for all $1 \le l \le L$ and $\mathcal{P}$ is the domain of the phoneme symbols. We denote by $\mathcal{P}^*$ the set of all finite length sequences over $\mathcal{P}$. Our goal is to learn a *keyword spotter*, denoted f, which takes as input the pair $(\bar{\mathbf{x}}, \bar{p}^k)$ and returns a real value expressing the confidence that the targeted keyword k is uttered in $\bar{\mathbf{x}}$. That is, f is a function from $\mathcal{X}^* \times \mathcal{P}^*$ to the set $\mathbb{R}$. The confidence score outputted by f for a given pair $(\bar{\mathbf{x}}, \bar{p}^k)$ can then be compared to a threshold b to actually determine whether $\bar{p}^k$ is uttered in $\bar{\mathbf{x}}$. Let us further define the *alignment* between a keyword phoneme sequence and a speech signal. We denote by $s_l \in \mathbb{N}$ the start time of phoneme p_l (in frame units), and by $e_l \in \mathbb{N}$ the end time of phoneme p_l. We assume that the start time of phoneme p_{l+1} is equal to the end time of phoneme p_l, that is, $e_l = s_{l+1}$ for all

$1 \leq l \leq L-1$. The alignment sequence $\bar{s}^k$ corresponding to the phonemes sequence $\bar{p}^k$ is a sequence of start-times and an end-time, $\bar{s}^k = (s_1, \ldots, s_L, e_L)$, where s_l is the start-time of phoneme p_l and e_L is the end-time of the last phoneme p_L.

Our construction is based on a set of predefined non-linear feature functions $\{\phi_j\}_{j=1}^n$. Each feature function is of the form $\phi_j : \mathcal{X}^* \times \mathcal{P}^* \times \mathbb{N}^* \to \mathbb{R}$. That is, each feature function takes as input an acoustic representation of a speech utterance $\bar{\mathbf{x}} \in \mathcal{X}^*$, together with a keyword phoneme sequence $\bar{p}^k \in \mathcal{P}^*$, and a candidate alignment sequence $\bar{s}^k \in \mathbb{N}^*$, and returns a scalar in $\mathbb{R}$ which represents the confidence in the suggested alignment sequence given the keyword $\bar{p}^k$. For example, one element of the feature function can sum the number of times phoneme p comes after phoneme p', while other elements of the feature function may extract properties of each acoustic feature vector $\mathbf{x}_t$ provided that phoneme p is pronounced at time t. The complete set of the non-linear feature functions we use is described in Sec. 4.

As mentioned above, our goal is to learn a keyword spotter f, which takes as input a sequence of acoustic features $\bar{\mathbf{x}}$, a keyword $\bar{p}^k$, and returns a confidence value in $\mathbb{R}$. The form of the function f we use is

$$f(\bar{\mathbf{x}}, \bar{p}^k) = \max_{\bar{s}} \ \mathbf{w} \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{p}^k, \bar{s}) \ , \qquad (1)$$

where $\mathbf{w} \in \mathbb{R}^n$ is a vector of importance weights that should be learned and $\boldsymbol{\phi} \in \mathbb{R}^n$ is a vector function composed out of the feature functions ϕ_j. In other words, f returns a confidence prediction about the existence of the keyword in the utterance by maximizing a weighted sum of the scores returned by the feature function elements over all possible alignment sequences. The maximization defined by Eq. (1) is over an exponentially large number of all possible alignment sequences. Nevertheless, as in HMMs, if the feature functions $\boldsymbol{\phi}$ are decomposable, the maximization in Eq. (1) can be efficiently calculated using a dynamic programming procedure.

The performance of a keyword spotting system is often measured by the Receiver Operating Characteristics (ROC) curve, that is, a plot of the true positive (spotting a keyword correctly) rate as a function of the false positive (mis-spotting a keyword) rate (see for example [1, 3, 2] and the references therein). The points on the curve are obtained by sweeping the decision threshold b from the most positive confidence value outputted by the system to the most negative one. Hence, the choice of b represents a trade-off between different operational settings, corresponding to different cost functions weighing false positive and false negative errors. Assuming a flat prior over all these cost functions, a criterion to identify a good keyword spotting system that would be good on average for all these settings could be to select the one maximizing the area under the ROC curve (AUC). In the following we propose an algorithm which directly aims at maximizing the AUC.

Recall that we would like to obtain an algorithm that maximizes the AUC on unseen data. In order to do so, we will maximize the AUC over a large set of training examples. Let us consider two sets of examples. Denote by $\mathcal{X}_k^+$ a set of speech utterances in which the the keyword k is uttered. Similarly, denote by $\mathcal{X}_k^-$ a set of speech utterances in which the keyword k is not uttered. The AUC for the keyword k can be written in the form of the Wilcoxon-Mann-Whitney statistics [12] as

$$A_k = \frac{1}{|\mathcal{X}_k^+||\mathcal{X}_k^-|} \sum_{\substack{\bar{\mathbf{x}}^+ \in \mathcal{X}_k^+ \\ \bar{\mathbf{x}}^- \in \mathcal{X}_k^-}} \mathbb{1}_{\{f(\bar{\mathbf{x}}^+, \bar{p}^k) > f(\bar{\mathbf{x}}^-, \bar{p}^k)\}},$$

where $\mathbb{1}_{\{\cdot\}}$ refers to the indicator function and $\bar{p}^k$ refers to the phoneme sequence corresponding to keyword k. Thus, A_k estimates the probability that the score assigned to an utterance in which the keyword is uttered is greater than the score assigned to an utterance in which the keyword is not uttered. The average AUC over the set of keywords $\mathcal{K}$ can be written as $A = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} A_k$. In the next section we describe an iterative algorithm for learning the weight vector $\mathbf{w}$, which aims at maximizing the average AUC.

3. An Iterative Algorithm

We now describe a simple iterative algorithm for learning the weight vector $\mathbf{w}$ based on a training set of examples. Each example in the training set S is composed of a keyword phoneme sequence $\bar{p}^k$, an utterance in which the keyword k is uttered $\bar{\mathbf{x}}^+ \in \mathcal{X}_k^+$, an utterance in which the keyword k is not uttered $\bar{\mathbf{x}}^- \in \mathcal{X}_k^-$, and an alignment sequence $\bar{s}^k$ that corresponds to the location of the keyword in $\bar{\mathbf{x}}^+$. The algorithm receives as input a set of training examples $S = \{(\bar{p}^{k_i}, \bar{\mathbf{x}}_i^+, \bar{\mathbf{x}}_i^-, \bar{s}_i^{k_i})\}_{i=1}^m$. and examines each of them sequentially. Initially, we set $\mathbf{w} = \mathbf{0}$. At each iteration i, the algorithm updates $\mathbf{w}$ according to the current example $(\bar{p}^{k_i}, \bar{\mathbf{x}}_i^+, \bar{\mathbf{x}}_i^-, \bar{s}_i^{k_i})$ as we now describe. Denote by $\mathbf{w}_{i-1}$ the value of the weight vector before the ith iteration. Let $\bar{s}'$ be the predicted alignment for the negative utterance, $\bar{\mathbf{x}}_i^-$, according to $\mathbf{w}_{i-1}$,

$$\bar{s}' = \arg\max_{\bar{s}} \ \mathbf{w}_{i-1} \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}_i^-, \bar{p}^{k_i}, \bar{s}) \ . \qquad (2)$$

Let us define the difference between the feature functions of the acoustic sequence in which the keyword is uttered and the feature functions of the acoustic sequence in which the keyword is not uttered as $\Delta\boldsymbol{\phi}_i$, that is,

$$\Delta\boldsymbol{\phi}_i = \frac{1}{|\mathcal{X}_{k_i}^+||\mathcal{X}_{k_i}^-|} \left(\boldsymbol{\phi}(\bar{\mathbf{x}}_i^+, \bar{p}^{k_i}, \bar{s}^{k_i}) - \boldsymbol{\phi}(\bar{\mathbf{x}}_i^-, \bar{p}^{k_i}, \bar{s}') \right) \ .$$

We set the next weight vector $\mathbf{w}_i$ to be the minimizer of the following optimization problem,

$$\begin{aligned} &\min_{\mathbf{w} \in \mathbb{R}^n, \xi \geq 0} \ \frac{1}{2} \|\mathbf{w} - \mathbf{w}_{i-1}\|^2 + C\,\xi \\ &\text{s.t.} \ \ \mathbf{w} \cdot \Delta\boldsymbol{\phi}_i \geq 1 - \xi \ , \end{aligned} \qquad (3)$$

where C serves as a complexity-accuracy trade-off parameter (see [13]) and ξ is a non-negative slack variable, which indicates the loss of the ith example. Intuitively, we would like to minimize the loss of the current example, i.e., the slack variable ξ, while keeping the weight vector $\mathbf{w}$ as close as possible to our previous weight vector $\mathbf{w}_{i-1}$. The constraint makes the projection of the utterance in which the keyword is uttered onto $\mathbf{w}$ higher than the projection of the utterance in which the keyword is not uttered onto $\mathbf{w}$ by at least 1. It can be shown (see [13]) that the solution to the above optimization problem is

$$\mathbf{w}_i = \mathbf{w}_{i-1} + \alpha_i \Delta\boldsymbol{\phi}_i \ . \qquad (4)$$

The value of the scalar α_i is based on the different scores that $\bar{\mathbf{x}}^+$ and $\bar{\mathbf{x}}^-$ received according to $\mathbf{w}_{i-1}$, and a parameter C. Formally,

$$\alpha_i = \min\left\{ C, \ \frac{[1 - \mathbf{w}_{i-1} \cdot \Delta\boldsymbol{\phi}_i]_+}{\|\Delta\boldsymbol{\phi}_i\|^2} \right\} \ . \qquad (5)$$

The optimization problem given in Eq. (3) is based on ongoing work on online learning algorithms appearing in [13].

```
Input: training set $S = \{(\bar{p}^{k_i}, \bar{\mathbf{x}}_i^+, \bar{\mathbf{x}}_i^-, \bar{s}^{k_i})\}_{i=1}^m$; validation set $S_{\mathrm{val}}$; parameter $C$
Initialize: $\mathbf{w}_0 = \mathbf{0}$
For $i = 1, \ldots, m$
  Predict: $\bar{s}' = \arg\max_{\bar{s}} \ \mathbf{w}_{i-1} \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}_i^-, \bar{p}^{k_i}, \bar{s})$
  Set: $\Delta\boldsymbol{\phi}_i = \frac{1}{|\mathcal{X}_{k_i}^+||\mathcal{X}_{k_i}^-|} \left(\boldsymbol{\phi}(\bar{\mathbf{x}}_i^+, \bar{p}^{k_i}, \bar{s}^{k_i}) - \boldsymbol{\phi}(\bar{\mathbf{x}}_i^-, \bar{p}^{k_i}, \bar{s}')\right)$
  If $\mathbf{w}_{i-1} \cdot \Delta\boldsymbol{\phi}_i < 1$
    Set: $\alpha_i = \min\left\{C \,,\ \frac{1 - \mathbf{w}_{i-1} \cdot \Delta\boldsymbol{\phi}_i}{\|\Delta\boldsymbol{\phi}_i\|^2}\right\}$
    Update: $\mathbf{w}_i = \mathbf{w}_{i-1} + \alpha_i \cdot \Delta\boldsymbol{\phi}_i$
Output: The weight vector $\mathbf{w}^*$ which achieves best AUC performance on the validation set $S_{\mathrm{val}}$.
```

Figure 1: An iterative algorithm.

Based on that work, it can be shown that, under some mild technical conditions, the cumulative performance of the iterative procedure, i.e., $\frac{1}{m}\sum_{i=1}^m \mathbb{1}_{\{\mathbf{w}_i \cdot \Delta\boldsymbol{\phi}_i > 0\}}$ is likely to be high. Moreover, it can further be shown that if the cumulative performance of the iterative procedure is high, there exists at least one weight vector among the vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$ which attains high averaged performance on the test examples as well, that is, there exists a vector which attains high averaged AUC over a set of test examples. To find this weight vector, we simply calculate the averaged loss attained by each of the weight vectors on a validation set. A pseudo-code of our algorithm is given in Fig. 1.

In the case the user would like to select a threshold b that would ensure a specific requirement in terms of true positive rate or false negative rate, a simple cross-validation procedure (see [14]) would consist in selecting the confidence value given by our model at the point of interest over the ROC curve plotted for some validation utterances of the targeted keyword.

4. Non-Linear Feature Functions

In this section we present the implementation details of our learning approach for the task of keyword spotting. Recall that our construction is based on a set of non-linear feature functions, $\{\phi_j\}_{j=1}^n$, which maps an acoustic-phonetic representation of a speech utterance as well as a suggested alignment sequence into an abstract vector-space. In order to make this section more readable we omit the keyword index k.

We introduce a specific set of feature functions, which is highly adequate for the keyword spotting problem. We utilize seven different feature functions ($n = 7$). These feature functions are used for defining our keyword spotting function $f(\bar{\mathbf{x}}, \bar{p})$ as in Eq. (1).

Our first four feature functions aim at capturing transitions between phonemes. These feature functions are the distance between frames of the acoustic signal at both sides of phoneme boundaries as suggested by an alignment sequence $\bar{s}$. The distance measure we employ, denoted by d, is the Euclidean distance between feature vectors. Our underlying assumption is that if two frames, $\mathbf{x}_t$ and $\mathbf{x}_{t'}$, are derived from the same phoneme then the distance $d(\mathbf{x}_t, \mathbf{x}_{t'})$ should be smaller than if the two frames are derived from different phonemes. Formally, our first four feature functions are defined as

$$\phi_j(\bar{\mathbf{x}}, \bar{p}, \bar{s}) = \sum_{i=2}^{|\bar{p}|-1} d(\mathbf{x}_{-j+s_i}, \mathbf{x}_{j+s_i}), \ \ j \in \{1, 2, 3, 4\} \ . \quad (6)$$

If $\bar{s}$ is the correct timing sequence then distances between frames across the phoneme change points are likely to be large. In contrast, an incorrect phoneme start time sequence is likely to compare frames from the same phoneme, often resulting in small distances. Note that the first four feature functions described above use only the start time of the ith phoneme and do not use the values of s_{i-1} and s_{i+1}.

The fifth feature function we use is built from a frame-wise phoneme classifier described in [15]. Formally, for each phoneme event $p \in \mathcal{P}$ and frame $\mathbf{x} \in \mathcal{X}$, there is a confidence, denoted $g_p(\mathbf{x})$, that the phoneme p is pronounced in the frame $\mathbf{x}$. The resulting feature function measures the cumulative confidence of the complete speech signal given the phoneme sequence and their start-times,

$$\phi_5(\bar{\mathbf{x}}, \bar{p}, \bar{s}) = \sum_{i=1}^{|\bar{p}|} \sum_{t=s_i}^{s_{i+1}-1} g_{p_i}(\mathbf{x}_t) \ . \quad (7)$$

The fifth feature function uses both the start time of the ith phoneme and the $(i+1)$th phoneme but ignores s_{i-1}.

Our next feature function scores timing sequences based on phoneme durations. Unlike the previous feature functions, the sixth feature function is oblivious to the speech signal itself. It merely examines the length of each phoneme, as suggested by $\bar{s}$, compared to the typical length required to pronounce this phoneme. Formally,

$$\phi_6(\bar{\mathbf{x}}, \bar{p}, \bar{s}) = \sum_{i=1}^{|\bar{p}|} \log \ \mathcal{N}(s_{i+1} - s_i; \hat{\mu}_{p_i} \,, \hat{\sigma}_{p_i}) \ \ , \quad (8)$$

where $\mathcal{N}$ is a Normal probability density function with mean $\hat{\mu}_p$ and standard deviation $\hat{\sigma}_p$. In our experiments, we estimated $\hat{\mu}_p$ and $\hat{\sigma}_p$ from the training set (see Sec. 5).

Our last feature function exploits assumptions on the speaking rate of a speaker. Intuitively, people usually speak in an almost steady rate and therefore a timing sequence in which speech rate is changed abruptly is probably incorrect. Formally, let $\hat{\mu}_p$ be the average length required to pronounce the pth phoneme. We denote by r_i the relative speech rate, $r_i = (s_{i+1} - s_i)/\hat{\mu}_{p_i}$. That is, r_i is the ratio between the actual length of phoneme p_i as suggested by $\bar{s}$ to its average length. The relative speech rate presumably changes slowly over time. In practice the speaking rate ratios often differ from speaker to speaker and within a given utterance. We measure the local change in the speaking rate as $(r_i - r_{i-1})^2$ and we define the feature function ϕ_7 as the local change in the speaking rate,

$$\phi_7(\bar{\mathbf{x}}, \bar{p}, \bar{s}) = \sum_{i=2}^{|\bar{p}|} (r_i - r_{i-1})^2 \ \ . \quad (9)$$

5. Experimental Results

To validate the effectiveness of the proposed approach we performed experiments with the TIMIT corpus. We divided the training portion of TIMIT (excluding the SA1 and SA2 utterances) into three disjoint parts containing 500, 80 and 3116 utterances. The first part of the training set was used for learning the functions g_p (Eq. (7)), which define the feature function ϕ_5. Those functions were learned by the algorithm described in [15] using the MFCC+Δ+$\Delta\Delta$ acoustic features and a Gaussian kernel ($\sigma = 6.24$ and $C = 5.0$). The second set of 80 utterances formed the validation set needed for our keyword spotting algorithm. The set was built out of a set of 40 keywords randomly

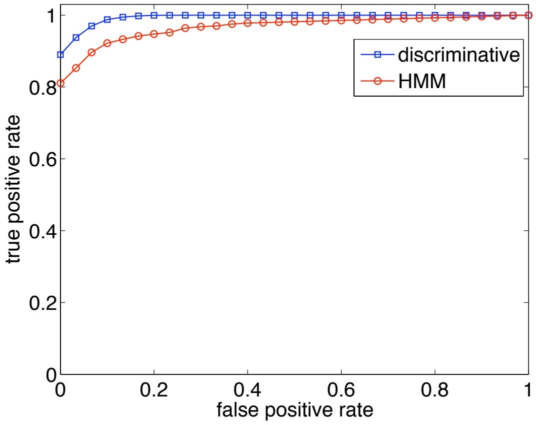

Figure 2: ROC curves of the discriminative algorithm and the HMM approach. The AUC of the ROC curves is 0.99 and 0.96 for the discriminative algorithm and the HMM algorithm, respectively.

chosen from the TIMIT lexicon. The 80 utterances were chosen by pairs: one utterance in which the keyword was uttered and another utterance in which the keyword was not uttered. Finally, we ran our iterative algorithm on the rest of the utterances in the training set. The value of C was set to be 1.

We compared the results of our method to the HMM approach, where each phoneme was represented by a simple left-to-right HMM of 5 emitting states with 40 diagonal Gaussians. These models were enrolled as follows: first the HMMs were initialized using K-means, and then enrolled independently using EM. The second step, often called embedded training, re-enrolls all the models by relaxing the segmentation constraints using a forced alignment. Minimum values of the variances for each Gaussian were set to 20% of the global variance of the data. All HMM experiments were done using the *Torch* package [16]. All hyper-parameters including number of states, number of Gaussians per state, variance flooring factor, were tuned using the validation set.

Keyword detection is performed with a new HMM composed of two sub HMM models, the keyword model and the garbage model. The keyword model is an HMM, which estimates the likelihood of an acoustic sequence given that this sequence represents the keyword phoneme sequence. The garbage model is an HMM composed of phoneme HMMs fully connected with each others, which estimates the likelihood of any acoustic sequence. The overall HMM fully connects the keyword model and the garbage model and the best path found by Viterbi decoding on this overall HMM either passes through the keyword model (in which case the keyword is said to be uttered) or not (in which case the keyword is not in the acoustic sequence).

The test set was composed of 80 keywords, distinct from the keywords of the training and validation set. For each keyword, we randomly picked at most 20 utterances in which the keyword was uttered and at most 20 utterances in which it was not uttered. The number of test utterances in which the keyword was uttered was not always 20, since some keywords were uttered less than 20 times in the whole TIMIT test set. Both the discriminative algorithm and the HMM based algorithm have been evaluated against this data and their results are reported as averaged ROC curves in Fig. 2. The AUC of the ROC curves is 0.99 and 0.96 for the discriminative algorithm and the HMM algorithm, respectively. In order to check whether the advantage over the averaged AUC could be due to a few keyword, we ran the Wilcoxon test. At the 95% confidence level, the test rejected this hypothesis, showing that our model indeed brings a consistent improvement on the keyword set.

6. Conclusions

In this work, we introduced a discriminative approach to keyword spotting. We adopted a large-margin formulation of the problem and proposed a model relying on an objective function related the area under the ROC curve, i.e., the most common measure for keyword spotter evaluation. Compared to state-of-the-art approaches which mostly rely on generative HMM models, the proposed model has shown to yield an improvement over the TIMIT corpus.

Acknowledgments: This research was partly conducted while Joseph Keshet was visiting the IDIAP Research Institute. This research was supported by the PASCAL European Network of Excellence and the DIRAC project.

7. References

[1] M.-C. Silaghi and H. Bourlard, “Iterative posterior-based keyword spotting without filler models,” in *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, Keystone, USA, 1999, pp. 213–216.

[2] Y. B. Ayed, D. Fohr, J.-P. Haton, and G. Chollet, “Confidence measure for keyword spotting using support vector machines,” in *Proc. of International Conference on Audio, Speech and Signal Processing*, 2004.

[3] H. Ketabdar, J. Vepa, S. Bengio, and H. Bourlard, “Posterior based keyword spotting with a priori thresholds,” in *Proceeding of Interspeech*, 2006.

[4] V. N. Vapnik, *Statistical Learning Theory*. Wiley, 1998.

[5] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[6] B. Taskar, C. Guestrin, and D. Koller, “Max-margin markov networks,” in *Advances in Neural Information Processing Systems 17*, 2003.

[7] S. Shalev-Shwartz, J. Keshet, and Y. Singer, “Learning to align polyphonic music,” in *Proceedings of the 5th International Conference on Music Information Retrieval*, 2004.

[8] J. Keshet, D. Chazan, and B.-Z. Bobrovsky, “Plosive spotting with margin classifiers,” in *Proceedings of the Seventh European Conference on Speech Communication and Technology*, 2001, pp. 1637–1640.

[9] J. Salomon, S. King, and M. Osborne, “Framewise phone classification using support vector machines,” in *Proceedings of the Seventh International Conference on Spoken Language Processing*, 2002, pp. 2645–2648.

[10] M. Collins, “Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms,” in *Conference on Empirical Methods in Natural Language Processing*, 2002.

[11] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, “Support vector machine learning for interdependent and structured output spaces,” in *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[12] C. Cortes and M. Mohri, “Confidence intervals for the area under the roc curve,” in *Advances in Neural Information Processing Systems 17*, 2004.

[13] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, “Online passive aggressive algorithms,” *Journal of Machine Learning Research*, vol. 7, Mar 2006.

[14] S. Bengio, J. Maréthoz, and M. Keller, “The expected performance curve,” in *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[15] O. Dekel, J. Keshet, and Y. Singer, “Online algorithm for hierarchical phoneme classification,” in *Workshop on Multimodal Interaction and Related Machine Learning Algorithms; Lecture Notes in Computer Science*. Springer-Verlag, 2004, pp. 146–159.

[16] R. Collobert, S. Bengio, and J. Mariéthoz, “Torch: a modular machine learning software library,” IDIAP, IDIAP-RR 46, 2002.