# New Features for Discriminative Keyword Spotting

**Punnoose A K**
Flare Speech Systems
Bangalore, India

## Abstract

This paper presents new feature functions and an efficient training approach to discriminative keyword spotting. A 5-dimensional feature function is derived from a frame-based phoneme classifier and a pitch detector. The mechanism by which each feature function finds a correspondence between the keyword-specific variables and the speech segment is discussed. Multiple aspects of the keyword, that the feature functions capture, are explored. The keyword-scoring function along with the positive and negative data associated with a keyword is defined. A computationally intensive operation in keyword training is identified and an approach is developed to reduce the training more tractable. The proposed approach is implemented with a set of 10 keywords and benchmarked against a traditional lattice-based keyword search on a real-world dataset, and the results are discussed.

## 1 Introduction

Keyword spotting refers to the detection of specific words in a speech stream. This is different from automatic speech recognition(ASR) in the sense that models are built exclusively for the keywords. Keyword spotting is used in large-scale audio indexing and retrieval, wake-up word for devices, etc. Traditionally hidden Markov model(HMM) based approaches have been used in keyword spotting (Bahl et al., 1986; Rohlicek et al., 1989; Rose and Paul, 1990; Szöke et al., 2005). Keywords are modelled using keyword-specific HMMs and the rest of the speech is modelled using background models (Wilpon et al., 1990). The likelihood ratio of a speech segment running through a keyword HMM to the background speech HMM is used to detect the keyword. Phone lattice keyword search (Young et al., 1997) searches for the sequence of phonemes of the keyword in the phone lattice generated during N-best Viterbi decoding. The lattice search can be improved by dynamic programming and minimum edit distance (James and Young, 1994; Thambiratnam and Sridharan, 2005). Optimizations like unique arc per-second pruning and posting-list merging (Gales et al., 2017) are employed to prune the lattice for faster keyword search, especially in low-resource keyword spotting. In a low-resource environment, the lattice search has been further optimized using web data for language model training (Mendels et al., 2015) and stimulated training (Ragni et al., 2017).

A phone lattice can be rescored using different approaches to make the path scores more relevant. Acoustic word embeddings (Piunova et al., 2019), lattice context information (Chen and Wu, 2017), word-burst (Ma et al., 2014) have been used to rescore the lattices. Future word contexts have been exploited using recurrent neural language models to rescore the lattices (Chen et al., 2019). Rather than using a single feature, multiple features like hierarchical bottleneck features (Riedhammer et al., 2013), smoothened posteriors (Chen et al., 2014), and multilingual bottleneck features (Menon et al., 2018b) are also used for keyword spotting. To increase robustness, especially in noisy and channel degraded environments, feature fusion (Mitra et al., 2014) is used. Generally, the keyword spotter works on top of ASR. Recently, ASR-free approaches (Menon et al., 2018b; Audhkhasi et al., 2017; Menon et al., 2018a) are also proposed for keyword spotting. This is especially useful in low-resource settings.

Neural networks are often used in conjunction with HMMs for keyword spotting (Rath et al., 2014; Chen and Lee, 2013). Different neural network architectures like deep neural networks (Chen et al., 2014), convolutional neural networks (Sainath and Parada, 2015), compressed time delay neural networks (Sun et al., 2017), recurrent networks (Li et al., 1992; Fernández et al., 2007), long short-term memory(lstm) (Wollmer et al., 2009) have

been employed for keyword detection in speech. Recently transformers have been used for keyword spotting in speech (Berg et al., 2021). In this paper, we follow the framework defined in (Keshet et al., 2009), where a set of feature functions are defined and a weight vector, corresponding to each keyword, is trained in an online discriminative manner. While training, a minimum margin is specified between the keywords and all the other words. In the original setting, the authors derive most of the feature functions directly from the spectral level features, while we use an intermediate layer of a frame classifier and a pitch detector, sitting atop the spectral layer, and deriving feature functions out of it.

In section 2, we review the discriminative keyword spotting framework. The training algorithm, feature functions, and positive and negative vectors are defined. A computationally efficient approach to reduce the individual keyword training time is discussed in detail. In section 3, the experimental details of benchmarking the proposed approach against a traditional lattice-based keyword search are discussed. Section 4 concludes the paper.

## 2 Problem Setting

First, we define an acoustic tuple $(X, P, F)$ to be a set of vectors derived from passing a speech segment through a frame classifier and a pitch detector. $X$ is the decoded phoneme string sequence with softmax probability sequence $P$ for an utterance of length $T$ frames. $F$ is the sequence of pitch values. A keyword k is associated with a set of positive acoustic tuples $(X_{\mathrm{k}}^{+}, P_{\mathrm{k}}^{+}, F_{\mathrm{k}}^{+})$ and negative acoustic tuples $(X_{\mathrm{k}}^{-}, P_{\mathrm{k}}^{-}, F_{\mathrm{k}}^{-})$. The superscript $+$ implies the presence of the keyword in the speech segment and $-$ indicates its absence. Next, we define a feature function $\phi$ that takes the form

$$\phi : (V_{\mathrm{k}}, X, P, F) \rightarrow R^{5} \tag{1}$$

A feature function broadly takes 2 classes of arguments and maps them into a feature space. $V_{\mathrm{k}}$ is associated with the orthographic representation of the keyword k and $(X, P, F)$ is associated with a given speech segment. $V_{\mathrm{k}}$ involves the variables like the distribution of the ideal duration of the keyword, dictionary entries of the keyword, number of voiced segments in the keyword, distribution of the ideal duration of the voiced segments in the keyword, etc. The feature function must output similar vector values in some distance sense for the same keyword with similar speech segments. The scoring function for keyword k takes the form

$$f = w_{\mathrm{k}}.\phi(V_{\mathrm{k}}, X, P, F) \tag{2}$$

where $w_{\mathrm{k}}$ is the keyword-specific weight vector.

### 2.1 Training

Given a set of positive and negative acoustic tuples, we wish to learn the weight vector $w_{\mathrm{k}}$ for the keyword k. The algorithm operates in an online manner. Let $w_{\mathrm{k}_{i-1}}$ be the weight at the $i-1$th iteration. To find $w_{\mathrm{k}_i}$, we first calculate $(X_{\mathrm{k}}^{*}, P_{\mathrm{k}}^{*}, F_{\mathrm{k}}^{*})$ as,

$$(X_{\mathrm{k}}^{*}, P_{\mathrm{k}}^{*}, F_{\mathrm{k}}^{*}) = \max_{(X_{\mathrm{k}}, P_{\mathrm{k}}, F_{\mathrm{k}})} w_{\mathrm{k}_{i-1}}.\phi(V_{\mathrm{k}}, X_{\mathrm{k}}^{-}, P_{\mathrm{k}}^{-}, F_{\mathrm{k}}^{-}) \tag{3}$$

$(X_{\mathrm{k}}^{*}, P_{\mathrm{k}}^{*}, F_{\mathrm{k}}^{*})$ is the worst possible negative acoustic tuple for $w_{\mathrm{k}}$ at the $i-1$th stage. In the $i$-th step, this has to be penalized. Define $\Delta_i$ as,

$$\Delta_i = \frac{1}{|X^{+}||X^{-}|}\left[\phi(V_{\mathrm{k}}, X_{\mathrm{k}}^{+}, P_{\mathrm{k}}^{+}, F_{\mathrm{k}}^{+}) - \phi(V_{\mathrm{k}}, X_{\mathrm{k}}^{*}, P_{\mathrm{k}}^{*}, F_{\mathrm{k}}^{*})\right] \tag{4}$$

To find $w_{\mathrm{k}_i}$ from $w_{\mathrm{k}_{i-1}}$, the following optimization problem has to be solved.

$$\begin{aligned} w_{\mathrm{k}_i} = \min_{w} & ||w - w_{\mathrm{k}_{i-1}}|| \\ \text{s.t.} \;\; & w.\Delta_i \geq 1 \end{aligned} \tag{5}$$

The solution to this optimization problem is

$$w_{\mathrm{k}_i} = w_{\mathrm{k}_{i-1}} + \delta_i \Delta_i \tag{6}$$

$$\text{where } \delta_i = min\left\{A, \frac{1 - w_{\mathrm{k}_{i-1}}.\Delta_i}{||\Delta_i||^2}\right\} \tag{7}$$

where $A$ is a complexity-accuracy tradeoff parameter. The optimization in equation (5) ensures that the new weight vector $w_{\mathrm{k}_{i-1}}$ is close to the current weight vector $w_{\mathrm{k}_i}$, yet obeying the margin requirement. The correctness of this online update is proved in (Crammer et al., 2006).

In the original setting (Keshet et al., 2009), the authors use feature functions that map a tuple of an acoustic vector, a phoneme sequence, and an alignment sequence to a real vector. For a keyword, each training unit consists of the phoneme sequence corresponding to the keyword along with a positive and negative acoustic vector. At any stage in training, the worst possible acoustic negative tuple is

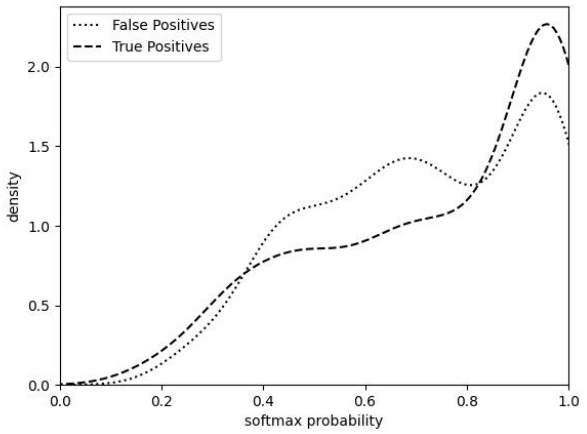


Figure 1: Density of softmax probability of /f/

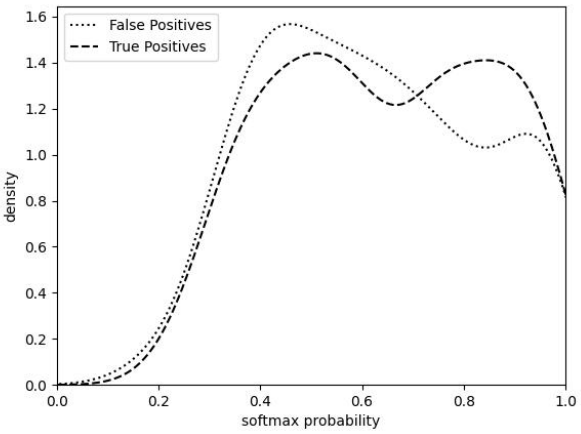


Figure 2: Density of softmax probability of /aa/

the tuple that consists of the keyword phoneme sequence, negative acoustic vector, and the worst possible alignment sequence. In our framework, the acoustic tuples and the feature functions inherently capture the alignment sequence, so that we can get away without having an explicit alignment sequence variable.

### 2.2 Feature Functions

We define foreign frames of a keyword as the frames that belong to the phonemes which do not fall in any dictionary entry of the keyword. $\phi_1$ is the percentage of relevant phonemes detected in sequence in $X$, in the total number of phonemes in the dictionary entry of the keyword, expressed in decimal. $\phi_1$ captures the relevance of phonemes in the speech segment to the keyword. $\phi_2$ is the percentage of the relevant frames detected in sequence, in the total number of frames in $X$, expressed in decimal.

$\phi_3$ is based on the observation that, for a frame based phoneme classifier with a softmax output, the density of softmax probability for true positive and false positive phoneme detections are different for different phonemes. $\phi_3$ is the average ratio of the density of frame softmax probability of true positives to the false positives of the frames detected correctly in sequence. Fig.1 and Fig.2 plots the density of the softmax probability of true positive and false positive detections of the phonemes /f/ and /aa/. The plot is generated from the frame labelled data $d_{train2}$, which is explained in section 3. At the peak, the density of true positives is greater than that of false positives. If a set of consecutive frames are detected as /f/, and if all the frames have a softmax probability greater than 0.95, it is more probable that all the frames are true positives. $\phi_3$ is calculated as follows.

$$\phi_3(V_{\mathrm{k}}, X, P, F) = \begin{cases} \frac{1}{N}\sum_i \frac{f_{\mathrm{tp}}(p_i;x_i)}{f_{\mathrm{fp}}(p_i;x_i)} & 50\% \text{ rec.} \\ 0 & \text{else} \end{cases} \tag{8}$$

where $f_{\mathrm{tp}}(x_i; p_i)$ and $f_{\mathrm{fp}}(x_i; p_i)$ are the densities of the softmax probability $p_i$ evaluated on the true positive and the false positive softmax probability density curve of the phoneme $x_i$. $N$ is the number of relevant frames detected in sequence. $\phi_3$ returns 0 if less than 50% phonemes in the best possible dictionary entry of the keyword are detected in sequence.

$\phi_4$ captures how well the total duration of the detected voiced regions in the speech segment matches the ideal total voiced duration of the keyword. The ideal total voiced duration of a keyword is the sum of all the voiced durations in the keyword and is represented by a normal density, $\mathcal{N}(\mu_{\text{voiced}}, \sigma^2_{\text{voiced}})$. Likewise, $\phi_5$ captures how well the duration between the voiced regions at the starting and the ending of the speech segment matches the ideal duration between the voiced regions at the boundaries. $\phi_5$ is also represented by a normal density $\mathcal{N}(\mu_{\text{between voiced}}, \sigma^2_{\text{between voiced}})$. $\phi_5$ in a sense captures the length of the keyword. $\phi_5$ mandates the keywords to have atleast 2 voiced segments. If there are no 2 voiced regions detected in a speech segment, $\phi_5$ outputs 0.

### 2.3 Negative Data

Negative data corresponds to the speech segments where a given keyword is absent. For a big dataset,

the number of speech segments that can be selected where a particular keyword is absent is enormous. We employ a simple heuristic to cut short the number of such speech segments. We compute the average duration $d$ of a keyword from the positive instances of that keyword. From an arbitrary starting frame $t$, all the speech segments from $(t, t + d/2)$ to $(t, t + 3d/2)$ are treated as separate negative instances of the keyword k, and the corresponding negative acoustic tuples are computed. This makes it extremely skewed to negative instances compared to the positive instances of a keyword.

For a keyword, the calculation of the worst possible acoustic tuple involves the dot product computation of all the negative instances with the present weight vector $w$ as expressed in equation (3). This is computationally expensive as the number of negative instances is often much more than the number of positive training instances of a keyword. To solve this problem, we first compute the convex hull of all the negative vectors of a keyword. Then, take the dot product of the current weight vector with all the extremal points in the convex hull and assign the extremal point with the highest dot product as the worst possible acoustic tuple for the current weight vector.

### 2.4 Analysis

Consider a finite set of points $P$ in $R^d$. Let $C$ be the convex hull constructed on the set $P$. Let $E$ be the set of extremal points of the convex hull. $E \subset P$. Let $x$ be a point outside $C$. Let $\cdot$ denote the dot product.

**Proposition 1** *There exists atleast one point $e \in E$ with the condition, $x \cdot e \geq x \cdot i \quad \forall\, i \in P - E$.*

**Proof** Let us consider $R^2$. Let $b$ be a point in $C$ such that $x \cdot b \geq x \cdot p$. Further, there are 2 cases.

1. $b \in E$. In this case, Proposition 1 is proved.

2. $b \notin E$. In this case, $b$ is a point in a line segment $\overline{b_1 b_2}$ with endpoints in $E$. Assume a hyperplane, $H = \{z | x \cdot z = x \cdot p\}$. In $R^2$, $H$ is a line. We loosely label a point $c$ inside $H$, if $x \cdot c < x \cdot p$. Now there are 4 possible cases.

   (a) The line segment $\overline{b_1 b_2}$ is such that $x \cdot b_1 > x \cdot p$ and $x \cdot b_2 > x \cdot p$. Both $b_1$ and $b_2$ are outside points. Proposition 1 stands true.

   (b) The line segment $\overline{b_1 b_2}$ intersects $H$. For $b_1$,

   $$\begin{aligned} & x \cdot b_1 = x \cdot p + x \cdot (b_1 - p) \\ & x \cdot (b_1 - p) < 0 \qquad b_1 \text{ is inside } H \\ & x \cdot b_1 < x \cdot p \end{aligned} \tag{9}$$

   Similarly for $b_2$,

   $$\begin{aligned} & x \cdot b_2 = x \cdot p + x \cdot (b_2 - p) \\ & x \cdot (b_2 - p) > 0 \qquad b_2 \text{ is outside } H \\ & x \cdot b_2 > x \cdot p \end{aligned} \tag{10}$$

   Proposition 1 stands true.

   (c) Same case as above, where $b_2$ is inside $H$ and $b_1$ outside $H$. Proposition 1 stands true.

   (d) Both $x \cdot b_2 < x \cdot p$ and $x \cdot b_1 < x \cdot p$. i.e, both $b_1$ and $b_2$ are inside $H$. If this is the case,

   $$x \cdot b = x \cdot [\lambda b_1 + (1 - \lambda) b_2] \tag{11}$$
   $$= \lambda x \cdot b_1 + (1 - \lambda) x \cdot b_2 \tag{12}$$
   $$< x \cdot p \tag{13}$$

   which is against the initial assumption $x \cdot b \geq x \cdot p$. Hence this case is infeasible.

Note that the convex hull computation is a one-time operation for a keyword, and is computed in the feature space. Once the convex hull of the negative points of a keyword, i.e, $E$ is computed, the equation (3) becomes

$$\begin{aligned} & (X_{\mathrm{k}}^*, P_{\mathrm{k}}^*, F_{\mathrm{k}}^*) = \\ & \max_{(X_{\mathrm{k}}, P_{\mathrm{k}}, F_{\mathrm{k}})} w_{\mathrm{k}_{i-1}} . \phi(V_{\mathrm{k}}, X_{\mathrm{k}}^-, P_{\mathrm{k}}^-, F_{\mathrm{k}}^-) \\ & \text{where } \phi(V_{\mathrm{k}}, X_{\mathrm{k}}^-, P_{\mathrm{k}}^-, F_{\mathrm{k}}^-) \in E \end{aligned} \tag{14}$$

## 3 Experimental Details & Results

Voxforge dataset (Voxforge.org) is used in the experiments. The Voxforge dataset is a real-world non-curated dataset that perfectly captures the real-world noise and the acoustic characteristics of different recording equipment. Approximately 40 hours of data($\mathrm{d}_{train1}$) is forced aligned using Kaldi (Povey et al., 2011) and a baseline multilayer perceptron(MLP) frame classifier, with the architecture 351x1000x1000x1000x41, is built using ICSI Quicknet (Johnson) with perceptual linear prediction(plp) coefficients as the feature input. Standard English phonemes are used as the target. Given a

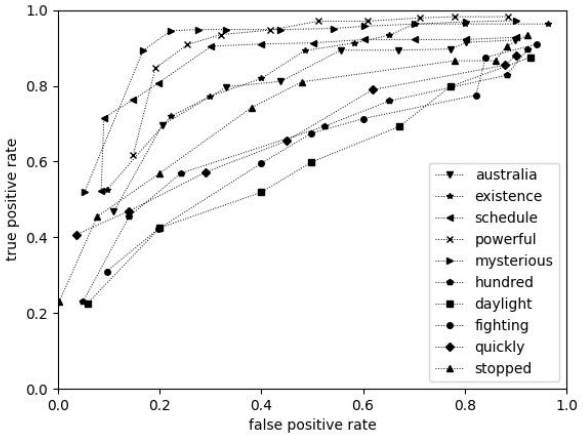


Figure 3: ROC of discriminative keyword spotting

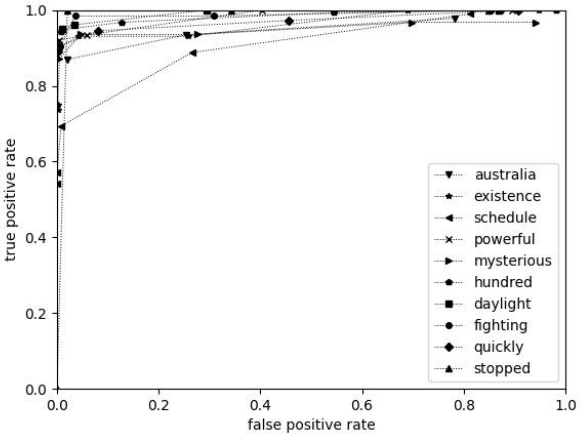


Figure 4: ROC of lattice keyword spotting

9 frame plp input, with each frame corresponding to 25ms with 15ms overlap, the classifier outputs a probability vector of size 41, each component corresponding to a phoneme, and the phoneme corresponding to the highest probability is treated as the recognized phoneme. For identifying the voiced regions, an autocorrelation-based pitch detector (Huckvale) is used. A pitch range of 40-600Hz is used and any value outside is discarded. Pitch values are computed on a 50ms time window and boundary adjusted with the output of the frame classifier. A pitch segment with a pitch difference between the adjacent frames within a 20Hz threshold is treated as a voiced region.

A separate 20 hours of data $\mathrm{d}_{train2}$ is used to learn the parameters in $\phi_3$, $\phi_4$ and $\phi_5$. This goes into the keyword-specific variable $V_k$. Another 20 hours of speech data is run through the baseline classifier and the pitch detector to get the $\mathrm{d}_{meta}$. Keyword present areas in $\mathrm{d}_{meta}$ are manually labelled at the word level, and the positive and negative training segments $(X, P, F)$ of each keyword are generated. The positive and negative segments of each keyword are converted to points in the feature space. The convex hull is computed for all the negative data points and the weight vector is trained as described in subsection 2.1 for all the keywords. The value of the complexity-accuracy tradeoff parameter $A$ is set to 1. Ten keywords are chosen, depending on the frequency, duration, and presence of voiced regions in the boundaries. Table 1 shows the number of extremal points on the convex hull computed from a random sample of 20000 points of negative instances of each keyword. Although the number of extreme points of the convex hull of a set is dependent on how the points are distributed, from Table 1, it is clear that atleast an order of magnitude reduction in the computation of dot product is attained. The convex hull is computed as explained in (Barber et al., 1996).

Table 1: Number of extremal points on the convex hull

| Keyword | # Extremal Points |
|---|---|
| australia | 126 |
| existence | 133 |
| schedule | 178 |
| powerful | 124 |
| mysterious | 204 |
| hundred | 172 |
| daylight | 264 |
| fighting | 151 |
| quickly | 181 |
| stopped | 152 |

20 hours of data $\mathrm{d}_{test}$ is used for testing. The occurrence of all the keywords is manually time labelled and positive and negative points are generated for all keywords in the feature space. Speech segments, for creating the negative points in the feature space, are generated in the manner specified in subsection 2.4. The scoring function for a speech instance against a keyword takes the form as expressed in equation (2).

The discriminative keyword spotter is benchmarked against a lattice-based keyword search using Kaldi. Kaldi is trained using the same $\mathrm{d}_{train}$ and is decoded into a lattice using $\mathrm{d}_{test}$. Multiple phoneme paths through the lattice are searched for keywords. While searching through the lat-

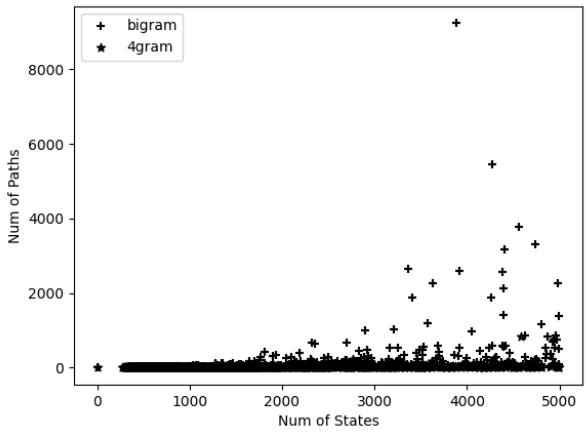


Figure 5: Number of states vs number of all possible paths in the lattice

tice, constraints like reducing the number of cycles and pruning the maximum length of paths, are employed to reduce the search space. We discard the lattice with states more than 500000. The acoustic score in the lattice is ignored. A bigram language model is used for decoding. The phoneme path through the lattice is divided into chunks of the size of the dictionary entry, and the length of the longest subsequence with the dictionary entry of the keyword is found. We consider a keyword to be detected if the longest subsequence length between a phoneme chunk in the lattice path and the dictionary entry is above a certain number of phonemes.

Fig.3 and Fig.4 plots the ROC curve of the discriminative keyword spotter and the lattice keyword spotter for various keywords. The number of phonemes detected in sequence is used as the threshold in Fig.4. It is clear that lattice keyword spotting is superior compared to discriminative keyword spotting. Lattice keyword spotting operates with a language model, while the discriminative spotter is purely acoustic in nature. Moreover, the number of possible paths through a lattice can be large, whereas the discriminative keyword spotter selects speech segments linearly in a recording for locating the keywords. Fig.5 plots the number of all possible paths against the number of states in the lattice, for a small subset of testing data. The maximum number of possible states is limited to 5000. As the number of states in the lattice increases, the number of possible paths also increases. A lattice decoded with a 4-gram language model constrains the number of possible paths compared to that with a bigram.

## 4 Conclusion

A set of 5 robust feature functions derived from a frame classifier and a pitch detector, for the discriminative approach to the keyword spotting problem, is presented. The feature functions along with the positive and negative training instances of the keyword are defined. The mechanism by which each feature function finds a correspondence between the keyword-specific variables and the speech segment is discussed in detail. Multiple aspects of the keyword, that the feature functions capture, are explored. A computationally intensive operation in keyword training is identified. An approach that makes keyword training more efficient, is presented, proved, and discussed in detail. The proposed approach is implemented with a set of 10 keywords with the Voxforge dataset. To benchmark our approach, a lattice-based keyword search is implemented in Kaldi with the same dataset and the results are compared. ROC curves are plotted for each keyword separately.

The approach shows how feature functions derived from multiple aspects of speech, can be combined to predict the keyword. In the future, the same framework can be used to incorporate more features like formants, acoustic-phonetic characteristics, phoneme-specific spectrogram features, etc for better keyword recognition.

## References

Kartik Audhkhasi, Andrew Rosenberg, Abhinav Sethy, Bhuvana Ramabhadran, and Brian Kingsbury. 2017. End-to-end asr-free keyword search from speech. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1351–1359.

Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. 1986. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Proc. ICASSP*, volume 11, pages 49–52.

C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483.

Axel Berg, Mark O'Connor, and Miguel Tairum Cruz. 2021. Keyword transformer: A self-attention model for keyword spotting. In *Interspeech 2021*. ISCA.

Guoguo Chen, Carolina Parada, and Georg Heigold. 2014. Small-footprint keyword spotting using deep neural networks. In *Proc. ICASSP*, pages 4087–4091.

I.-F Chen and Chin-Hui Lee. 2013. A hybrid hmm/dnn approach to keyword spotting of short words. In *Proc. INTERSPEECH*, pages 1574–1578.

Xie Chen, Xunying Liu, Yu Wang, Anton Ragni, Jeremy H. M. Wong, and Mark J. F. Gales. 2019. Exploiting future word contexts in neural network language models for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(9):1444–1454.

Zhipeng Chen and Ji Wu. 2017. A rescoring approach for keyword search using lattice context information. In *Proc. INTERSPEECH*, pages 3592–3596.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. 2007. An application of recurrent neural networks to discriminative keyword spotting. In *Proc. ICANN*.

M.J.F. Gales, Kate Knill, and Anton Ragni. 2017. Low-resource speech recognition and keyword-spotting. pages 3–19.

Mark Huckvale. Ampitch. `https://www.speechandhearing.net/laboratory/ampitch/`.

D.A. James and S.J. Young. 1994. A fast lattice-based approach to vocabulary independent wordspotting. In *Proc. ICASSP*, volume i, pages I/377–I/380.

D Johnson. Icsi quicknet software package. `http://www.icsi.berkeley.edu/Speech/qn.html`.

Joseph Keshet, David Grangier, and Samy Bengio. 2009. Discriminative keyword spotting. *Speech Communication*, 51(4):317–329.

K.P. Li, J.A. Naylor, and M.L. Rossen. 1992. A whole word recurrent neural network for keyword spotting. In *Proc. ICASSP*, volume 2, pages 81–84.

Min Ma, Justin Richards, Victor Soto, Julia Hirschberg, and Andrew Rosenberg. 2014. Strategies for rescoring keyword search results using word-burst and acoustic features. In *Proc. INTERSPEECH*, pages 2769–2773.

Gideon Mendels, Erica Cooper, Victor Soto, Julia Hirschberg, M.J.F. Gales, Kate Knill, Anton Ragni, and Haipeng Wang. 2015. Improving speech recognition and keyword search for low resource languages using web data. In *Proc. INTERSPEECH*.

Raghav Menon, Herman Kamper, John Quinn, and Thomas Niesler. 2018a. Fast asr-free and almost zero-resource keyword spotting using dtw and cnns for humanitarian monitoring. In *Proc. INTERSPEECH*, pages 2608–2612.

Raghav Menon, Herman Kamper, Emre Yılmaz, John Quinn, and Thomas Niesler. 2018b. Asr-free cnn-dtw keyword spotting using multilingual bottleneck features for almost zero-resource languages. In *Proc. SLTU*, pages 20–24.

Vikramjit Mitra, Julien van Hout, Horacio Franco, Dimitra Vergyri, Yun Lei, Martin Graciarena, Yik-Cheung Tam, and Jing Zheng. 2014. Feature fusion for high-accuracy keyword spotting. In *Proc. ICASSP*, pages 7143–7147.

Anna Piunova, Eugen Beck, Ralf Schlüter, and Hermann Ney. 2019. Rescoring keyword search confidence estimates with graph-based re-ranking using acoustic word embeddings. In *Proc. INTERSPEECH*, pages 4205–4209.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

A. Ragni, C. Wu, M. J. F. Gales, J. Vasilakes, and K. M. Knill. 2017. Stimulated training for automatic speech recognition and keyword search in limited resource conditions. In *Proc. ICASSP*, pages 4830–4834.

Shakti Rath, Kate Knill, A. Ragni, and M.J.F. Gales. 2014. Combining tandem and hybrid systems for improved speech recognition and keyword spotting on low resource languages. In *Proc. INTERSPEECH*, pages 835–839.

Korbinian Riedhammer, Van Hai Do, and James Hieronymus. 2013. A study on lvcsr and keyword search for tagalog. In *Proc. INTERSPEECH*.

Jan Robin Rohlicek, William Russell, Salim Roukos, and Herbert Gish. 1989. Continuous hidden markov modeling for speaker-independent word spotting. In *Proc. ICASSP*, volume 1, pages 627–630.

Richard C. Rose and Douglas B. Paul. 1990. A hidden markov model based keyword recognition system. In *Proc. ICASSP*, volume 1, pages 129–132.

Tara Sainath and Carolina Parada. 2015. Convolutional neural networks for small-footprint keyword spotting. In *Proc. INTERSPEECH*, pages 1478–1482.

Ming Sun, David Snyder, Yixin Gao, Varun K. Nagaraja, Mike Rodehorst, Sankaran Panchapagesan, Nikko Strom, Spyridon Matsoukas, and Shiv Naga Prasad Vitaladevuni. 2017. Compressed time delay neural network for small-footprint keyword spotting. In *Proc. INTERSPEECH*, pages 3607–3611.

Igor Szöke, Petr Schwarz, Pavel Matejka, Lukás Burget, Martin Karafiát, and Jan Honza Cernocký. 2005. Phoneme based acoustics keyword spotting in informal continuous speech. In *TSD*.

K. Thambiratnam and S. Sridharan. 2005. Dynamic match phone-lattice searches for very fast and accurate unrestricted vocabulary keyword spotting. In *Proc. ICASSP*, volume 1, pages I/465–I/468.

Voxforge.org. Free speech... recognition (linux, windows and mac) - voxforge.org. http://www.voxforge.org/. Accessed 06/25/2014.

Jay G. Wilpon, Lawrence R. Rabiner, Chin-Hui Lee, and E. R. Goldman. 1990. Automatic recognition of keywords in unconstrained speech using hidden markov models. *IEEE Trans. Acoust. Speech Signal Process.*, 38:1870–1878.

Martin Wollmer, Florian Eyben, Joseph Keshet, Alex Graves, Bjorn Schuller, and Gerhard Rigoll. 2009. Robust discriminative keyword spotting for emotionally colored spontaneous speech using bidirectional lstm networks. In *Proc. ICASSP*, pages 3949–3952.

Steve Young, M.G. Brown, J.T. Foote, Gareth Jones, and K. Jones. 1997. Acoustic indexing for multimedia retrieval and browsing. In *Proc. ICASSP*, volume 1, pages 199 – 202.