Non-ASR Based Keyword Spotting in Continuous Speech

A Project Report

submitted in partial fulfillment of the

requirements for the degree of

Master of Engineering

in

Signal Processing

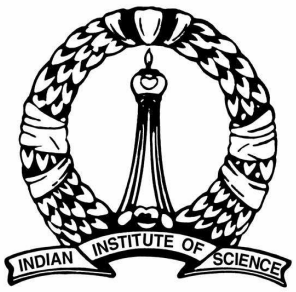
submitted by

Samik Sadhu

SR No. : 04-03-01-10-41-15-1-12199

Under the guidance of

Prof. Prasanta Kumar Ghosh



Department of Electrical Engineering

Indian Institute of Science

Bangalore - 560 012

June 2017

Acknowledgments

I am thankful towards Prof. Prasanta Kumar Ghosh for his critical guidance in my project-work. I would also thank Srinivasa Raghavan K M (Project Assistant at SPIRE Lab, IISc) for his help with Kaldi Automatic Speech Recognition system and Department of Science and technology, Government of India as well as Robert Bosch Centre for Cyber-Physical Systems (RBCCPS) at the Indian Institute of Science, Bangalore for financial support.

Contents

List of Figures

List of Tables

Abstract

In this report, we talk about Non-Automatic Speech Recognition (ASR) based Keyword Spotting (KWS) which has gained popularity almost in parallel with all ASR based keyword spotting algorithms because of their simplicity and high decoding speed. A state-of-the-art non-ASR keyword spotting system can search for keywords at 150,000 x real-time speed. A majority of the present works in non-ASR keyword spotting are aimed towards developing algorithms which require fewer training data (moving towards an unsupervised paradigm), have very fast keyword searching capabilities, yet, have a performance (w.r.t. the standard performance measures like area under ROC curve or Figure of Merit) as close as possible to an ASR based keyword spotting algorithm.

Keywords: Keyword Spotting, Point Process Model, On-line Learning, Unsupervised Learning, Discriminative Training.

Chapter 1

Introduction

Motivation towards non-ASR KWS

ASR systems are not designed for spotting keywords, but rather, they have been developed for decoding continuous speech. As a result, the language model tends to substitute rare words with frequently occurring words and the acoustic models for rare words, or the words of importance can be poorly trained [11]. The following points would further motivate research on non-ASR KWS:

- From our day to day experience, we can understand that the information distribution over different words in a sentence is not uniform, and hence, it is important to decipher important "keywords" in a sentence and the other "filler" words can, in many cases be almost automatically determined and are redundant. **Hence, KWS can act as the basis of an ASR system rather than building ASR based KWS systems.**
- There are applications where detection of keywords is the primary focus, for example, **indoor automation system, voice command systems etc.**
- Because of the speed of non-ASR based KWS systems, they can be used to search over several thousands of hours of speech data (eg. From **You-tube**) to spot keywords, in a similar way as we search large text files for certain words.

Our Contributions

In this project, we work towards

- **Improvement of non-ASR KWS in terms of keyword spotting accuracy measures:** One of the primary goals of non-ASR KWS research, at present, is to achieve a KWS accuracy

at par with the state-of-the-art ASR based KWS systems. In our work we have developed two such methods to improve KWS accuracy above the present non-ASR methods

1. **Discriminative Training of Point Process Models**
2. **Posteriorgram Filtering based Keyword Spotting**

- **Non-ASR KWS under low training resource availability:** ASRs are extremely resource hungry systems, requiring several hours of training data as well as heavy computational machinery for acoustic model training. There are only a handful of languages in the world which have enough language resource to train ASRs[]. Keyword spotting systems, on the other hand, are required to detect keywords rather than to decode entire sentences. Hence, a simpler and less resource hungry solution for keyword spotting has become a sought after research topic. In this regard, we have also developed two algorithms

  1. **Low Resource Point Process Models for Keyword Spotting Using Unsupervised Online Learning**
  2. **Adaptive Matched Filtering Based Unsupervised Keyword Spotting**

Organization of document

In chapter 2, we discus about a few important works in the literature on various aspects of KWS. In chapter 3 we talk about one of our contribution to non-ASR KWS, namely, Discriminative Training of PPM models which is an improvement over Maximum Likelihood (ML) training of PPM models. Then in chapter 4, we go on to describe one of our contributions to low resource KWS, namely, low resource PPM for KWS using unsupervised on-line learning. Alongside these works, we have also recently worked on posteriorgram filter based KWS (chapter 5) and completely unsupervised KWS (chapter 6). Finally, we end with a summary of the report and a discussion on future research prospects in non-ASR KWS in chapter 7.

Chapter 2

A Brief Literature Survey

A summary of non-ASR based KWS

Based on the different types of real-life situations that arise in practice, different directions of KWS research has emerged. In this section, we briefly discus about some of these research works -

HMM based acoustic modeling of keywords

Derived form the concept of HMM based acoustic modeling in typical ASR systems, these KWS systems are based on a keyword HMM model and a filler HMM model. A keyword filler network is then generated to be used to decode a given sentence to generate a sequence of occurrence of keywords and fillers. Bose et. al. in [22] describes a novel method to derive a **likelihood ratio score** based on a **keyword acoustic model, a filler acoustic model and a background acoustic model**. They use different filler models based on **word units, sub-word units (monophones and triphones) as well as unsupervised clustering based units** to show the variation in performance of the KWS system. The system typically requires $\approx$ 100 training instances of the keyword and $\approx$ 3 to 4 hours of annotated data for background/filler model training.

DNN based acoustic modeling of keywords

Deep Neural Network (DNN) based acoustic modeling of the keywords has recently become a popular topic of research [1, 23, 2] with the development of better DNN training algorithms. These systems generally consist of (a) A Voice Activity Detector (b) A feature extractor (c) A DNN/CNN/RNN for acoustic modeling (d) a posterior handling unit. The DNN/RNN/CNN is trained to perform a frame-wise classification into"keyword" and"filler" frames. The posterior handling module takes the softmax output and smoothen it out over time. These methods are

in essence mostly same as the HMM based acoustic modeling methods. However, they require a few thousand of keyword samples for effective modeling and a huge repository of negative keyword examples ($\approx$ 100-200 thousand) for filler/background modeling. Although these systems have been shown to outperform HMM based acoustic modeling of keywords, the improvement is achieved with the requirement of a large annotated training corpus which might not be available under certain circumstances.

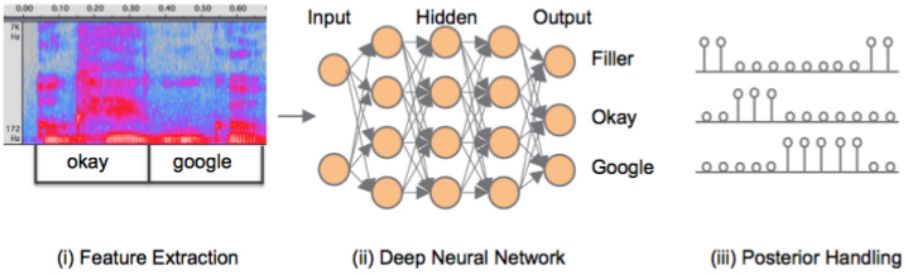


Fig. 2.1 Deep KWS System

Template based keyword detection

Template based KWS or **Query by Example(QbE)** KWS, is another KWS paradigm where only a few samples of the keyword (or any other acoustic unit) are provided by an user to an algorithm to search for similar patterns in running speech or in a speech database. There have been several notable works in this domain of KWS. Amongst these, a considerably important work has been conducted in [26], where the authors have adopted a completely unsupervised approach to template based keyword spotting. A Gaussian Mixture Model (GMM) with k number of mixtures is first trained on a few hours of un-annotated speech. Given a keyword with N frames, the probability contributions from each mixture of the GMM forms a vector of features (dimension=k) called the Gaussian Posteriorgram. The Gaussian Posteriorgram matrix ($k \times N$) for the N frames obtained for a keyword is used as a template and Segmental Dynamic Time Warping (sDTW) [14] is used to find segments in continuous speech that are acoustically similar to the provided example(s).

Another work on template based KWS [20] poses KWS as a dictionary based subspace classification problem. The proposed algorithm learns two dictionaries for the posteriorgrams obtained from the available examples of a keyword and background keywords (competing keywords), $D^{(k)}$

and $D^{(b)}$ respectively. A frame-wise error calculation is performed using the keyword and background keyword dictionaries as

$$e_k(y_t) = \|y_t - D^k \alpha_t^k\| \tag{2.1}$$

$$e_b(y_t) = \|y_t - D^b \alpha_t^b\| \tag{2.2}$$

where y_t denotes a feature vector at the t-th frame and α_t^k and α_t^b are the sparse vectors obtained by projecting the feature vector y_t into the respective sparse domains using the dictionaries $D^{(k)}$ and $D^{(b)}$ respectively. Accordingly, the measure of detection of a keyword for a frame is computed as

$$\Delta(y_t) = e_b(y_t) - e_k(y_t) \tag{2.3}$$

Similar to the DNN based approach, a posterior handling unit is used to smoothen the detector output $\Delta(y_t)$ to obtain a detector plot which is searched for probable keyword locations (high values of Δ).

Point Process Models for KWS

The Point Process Model (PPM) based KWS was introduced as an event based acoustic modeling of keywords [5]. The idea behind such an **event based** approach has been derived from the fact that humans communicate by uttering a string of phonemes, and hence, any other information used for KWS apart from such phonetic events are excess information. This excess information results in additional computation and storage costs for the KWS algorithm. A phonetic event based algorithm aims at cutting down on these costs to preserve and process only the required phonetic information. In this section, we describe in detail, the PPM based KWS which has been the basis for much of the work done in this project.

Phonetic Posteriorgrams

The PPM is based on a mapping function from frame level features to a posterior probability distribution over a set of phonemes $\mathcal{F}$. We train a DNN to generate the posterior probabilities

given a feature vector using the Kaldi ASR Toolkit [17] with speaker adaptive fMLLR features [4] as input to the DNN and a softmax output generating the posterior probabilities over the set phonemes $\mathcal{F}$.

Given N_w examples for the keyword w, we obtain obtain N_w posteriorgrams. Local maximas of the posteriorgrams are extracted by assigning a value of one to locations crossing the threshold value $\delta = 0.5$ and zero to the remaining points. Hence, we obtain a realization of a point process for each phoneme ϕ as $N_\phi = \{t_1, t_2, ...t_{n_\phi}\}$ where $\phi \in \mathcal{F}$, $\mathcal{F}$ being the set of phonemes. We use **Poisson Process** to model the observation for each row considering each row of the observation N_ϕ to be independent of the other phonemes in $\mathcal{F}$ (other rows).

Homogeneous PPM of a keyword

A homogeneous Poisson process is a point process with stationary and independent increments. The probability of n arrivals coming in the interval from a to b is given by

$$P_{a,b}(n) = \frac{(\lambda_\phi(b-a))^n \exp^{-\lambda_\phi(b-a)}}{n!} \tag{2.4}$$

where λ_ϕ is called the rate parameter. It can be easily derived from this that the probability of the first arrival coming at time t is given by

$$f(t) = \lambda_\phi \exp(-\lambda_\phi t) \tag{2.5}$$

Hence, given an observation N_ϕ as described in section 2.2.1, the likelihood of that observation can be calculated as follows

$$P(N_\phi) = f(t_1) \times \prod_{i=2}^{n_\phi} f(t_i - t_{i-1}) \times P_{t_{n_\phi},T}(0) = \lambda_\phi^{n_\phi} \exp(-\lambda_\phi T) \tag{2.6}$$

where, T is the duration of the observation. For the set of observations $R = \{N_\phi\}_{\phi \in \mathcal{F}}$,

$$P(R) = \prod_{\phi \in \mathcal{F}} \lambda_\phi^{n_\phi} \exp(-\lambda_\phi T) \tag{2.7}$$

A maximum likelihood training of the Poisson process model of a keyword would result in the

problem of

$$\lambda_1, \lambda_2, ..., \lambda_\tau = \arg \max_{\lambda_1, \lambda_2, ..., \lambda_\tau} P(R) \tag{2.8}$$

where $\tau = |\mathcal{F}|$. The solution to (2.8) is

$$\lambda_\phi = \frac{K_\phi}{NT} \tag{2.9}$$

where, K_ϕ is the number of observations of phoneme type ϕ over all the training keywords, N is the total number of training keywords and T is the assumed uniform time duration of all training keywords.Given N_w examples for the keyword w, we obtain N_w posteriorgrams. Local maximas of the posteriorgrams are extracted by assigning a value of one to locations crossing the threshold value $\delta = 0.5$ and zero to the remaining points. Hence, we obtain a realization of a point process for each phoneme ϕ as $N_\phi = \{t_1, t_2, ...t_{n_\phi}\}$ where $\phi \in \mathcal{F}$, $\mathcal{F}$ being the set of phonemes. We use a Poisson process to model the observation for each row considering each row of the observation N_ϕ to be independent.

Inhomogeneous PPM of keyword

Modeling a keyword with a fixed value of λ for the entire duration of the keyword is not satisfactory, because one phoneme almost certainly will not occur uniformity over the entire duration of the keyword. Different phonemes are expected to be concentrated over different temporal regions of the keyword. Hence, instead of using a fixed λ parameter per keyword, per phoneme, it would be better to use a piecewise constant approximation for λ over the duration of the keyword. The entire duration of a keyword is divided into D number of segments, and we model a keyword with segment-specific λ values. This segmentation results in $\tau \times D$ number of parameters that need to be estimated during training. Hence, the likelihood $P(N_\phi)$ can be obtained as

$$P(N_\phi) = \prod_{d=1}^{D} P(N_{\phi,d}) \tag{2.10}$$

Using equation (2.6), we compute $P(N_{\phi,d})$ as

$$P(N_{\phi,d}) = (\lambda_{\phi,d})^{n_{\phi,d}} \exp(-\lambda_{\phi,d} \Delta T) \tag{2.11}$$

where, $\Delta T = \frac{T}{D}$, $\lambda_{\phi,d}$ is the rate parameter for d^{th} segment for phoneme ϕ. Accordingly, the

likelihood function $P(R)$ changes to

$$P(R) = \prod_{\phi \in \mathcal{F}} \prod_{d=1}^{D} (\lambda_{\phi,d})^{n_{\phi,d}} \exp(-\lambda_{\phi,d} \Delta T) \tag{2.12}$$

Definition of the detector function

The **detector function** $d_w(t)$ is a function of time that represents the presence of keyword w at time t. A high $d_w(t)$ indicates a high probability of occurrence of the keyword w. We define the detector function as follows

$$d_w(t) = \log \frac{P(O|\theta_w(t) = 1)}{P(O|\theta_w(t) = 0)} \tag{2.13}$$

where, O denotes the observations for the test utterance and $\theta_w(t)$ is the identity function that indicates the presence of a keyword w at time t. However, the duration of a keyword can vary in a certain range and, hence, an extra variable T is introduced and the numerator and denominator of the detector equation can be written as

$$P(O|\theta_w(t) = 1) = \int_0^t P(O|T, \theta_w(t) = 1) P(T|\theta_w(t) = 1) dT \tag{2.14}$$

$$P(O|\theta_w(t) = 0) = \int_0^t P(O|T, \theta_w(t) = 0) P(T|\theta_w(t) = 0) dT \tag{2.15}$$

where the distribution $P(T|\theta_w(t) = 1)$ is called the **word duration model** and is learnt empirically from the training data set.

Decoding : Calculating $d_w(t)$

Consider a candidate keyword of duration T at time t, for an utterance of total duration L. We partition the observation into three subsets $R_1 = R|_{(0,t-T]}$, $R_2 = R|_{(t-T,t]}$ and $R_3 = R|_{(t,L]}$. Since, $P(R|\theta_w(t) = 0)$ can be broken down as

$$P(R|\theta_w(t) = 0) = P(R_1|T, \theta_w(t) = 1) \times P(R_2|T, \theta_w(t) = 0) \times P(R_3|T, \theta_w(t) = 1) \tag{2.16}$$

$$P(R|\theta_w(t)=1) = P(R_1|T,\theta_w(t)=1) \times P(R_2|T,\theta_w(t)=1) \times P(R_3|T,\theta_w(t)=1) \tag{2.17}$$

The detector function takes the following form

$$d_w(t) = log\left[\int_0^t \frac{P(R_2|T,\theta_w(t)=1)}{P(R_2|T,\theta_w(t)=0)} P(T|\theta_w(t)=1)dT\right] \tag{2.18}$$

We make a simplifying assumption for the denominator []

$$P(R_2|T,\theta_w(t)=1) = \frac{1}{T^{|R_2|}} P(R_2'|\theta_w(t)=1) \tag{2.19}$$

where, R_2' is the observation R_2 normalized to unit duration, to remove probabilistic dependence on T of the background model. Also, from equation (2.7) we calculate the likelihood value of $P(R_2'|\theta_w(t)=1)$ as

$$\prod_{\phi\in\mathcal{F}}\prod_{d=1}^{D}(\lambda_{\phi,d})^{n_{\phi,d}} \exp(\frac{-\lambda_{\phi,d}}{D}) \tag{2.20}$$

On the other hand, $P(R_2|T,\theta_w(t)=0)$ is obtained from the background model and is given by

$$P(R_2|T,\theta_w(t)=0) = \prod_{\phi\in\mathcal{F}} \mu^{n_\phi} \exp(-\mu_\phi T) \tag{2.21}$$

The rate parameter μ_ϕ with $\phi \in \mathcal{F}$ is estimated using a small annotated database and represents the general frequency of occurrence of the different phonemes.

The detector function $d_w(t)$ can hence be simplified into

$$d_w(t) = \log\left[\sum_{T\in\gamma} \frac{P(R_2'|\theta_w(t)=1)P(T|\theta_w(t)=1)\Delta}{T^{|R_2|}P(R_2|T,\theta_w(t)=0)}\right] \tag{2.22}$$

where, for computation purposes, the integration is approximated by a summation over duration T belonging to the set γ of word-durations containing values starting from T_{min} to T_{max} with an incremental factor of Δ.

Variants of PPM in KWS

Piece-wise constant to continuous λ parameters

In the paper [8], Hermansky et. al. construct a MAP parameter estimation framework for PPM. The paper also models the phonetic event distribution using Gaussian distributions rather than assuming a piece-wise constant nature, also providing sufficient motivation for this change in assumption. The paper makes complete utilization of the MAP training framework by proposing a low resource PPM based on **dictionary priors** which makes a seamless transition into a **posterior model** based on the number of training examples observed by the algorithm. They propose the Normal-Gamma distribution as the prior distribution over the set of Gaussian parameters which gets updated with more number of training examples.

Towards generating oracle (or ideal) phonetic events

The phonetic events generated using the standard posteriorgram thresholding technique will almost never consists of **oracle phonetic events** or ideal phonetic events. **Oracle phonetic events** refer to one phonetic event detection per phoneme being spoken. Logically, an oracle phoneme detector is sufficient for detection of a keyword and any other excess phonetic information are redundant. However, since a DNN based phonetic posteriorgram generator

1. might show confusion amongst different phonemes in a phonetic group
2. might lead to deletion of a phoneme completely (based on the classification accuracy of the DNN)
3. might have a higher likelihood for a wrong phoneme, resulting in wrong classification

, simplistic threshold based phonetic event generation might not be fruitful under all circumstances. In [7], the authors propose a set of **phonetic matched filters** based on **oracle phonetic detectors** to find regions in the phonetic posteriorgram trajectory closest to the oracle phonetic detections. The matched filters outputs are then thresholded to generate **near oracle** phonetic events which are then used for PPM training and decoding. Phonetic event extracted using this method results in a leap in the performance measure of the PPM KWS algorithm.

Senome event based PPM

In the paper [12], the phonetic events used in PPM is extended to senome events. A phonetic decision tree, questions are asked at each node to determine the left and right phonemes and hence a sequence of senomes is generated from a given speech signal. The paper shows that the ATWV and OTWV scores [3] improve when the monophone PPM models are replaced with senome PPM models for KWS.

Chapter 3

Discriminative Training of PPM

Basis and summary of work

For keyword spotting (KWS) task, point process models (PPM) are used to represent a target keyword by capturing the rate of various phonetic events. PPM are usually trained by the maximum likelihood estimation of piecewise-constant rate parameters or maximum a posteriori estimation of continuous rate parameters when limited keyword samples are available. In this work, we propose a discriminative training algorithm for obtaining the rate parameters of PPM by maximizing the ratio of the likelihood of observations of the target keyword samples to that of a set of competing words for a given target keyword. The set of competing words is selected from the false alarms caused by the PPM in a development set. KWS experiments are performed with 14 keywords in the Boston University Radio Speech (BURS) and TIMIT corpora with two evaluation metrics, namely, the figure of merit (FOM) score and area under receiver operating characteristic curve (AROC). We find that the proposed discriminatively trained PPM combined with PPM results in a 0.9% and 2.21% (absolute) increase in FOM and AROC respectively compared to original PPM in the BURS corpus. Similarly, FOM and AROC improve by 0.05% and 0.08% (absolute) in the case of TIMIT corpus.

Introduction

Keyword Spotting (KWS) with Point Process Models (PPM) was introduced by Jansen et al.[5] in the sliding model paradigm of KWS as a phonetic landmark based algorithm. The model has been refined through subsequent works on extracting better phonetic events [7], using context dependent phonetic events [12] and MAP estimation of rate parameters [8] and other improvements [9, 6, 10]. In this work, we propose a discriminative paradigm for training PPM. Discriminative training meth-

ods have been shown to enhance the performance of Automatic Speech Recognition (ASR) systems in comparison to maximum likelihood (ML) training [25, 15, 18, 19, 16]. Discriminative training has also been applied to KWS applications [24]. The primary idea behind discriminative training of model parameters is to obtain a model that maximally discriminates between the desired class against a set of competing classes. In this work, we use the idea of discriminative training to train the PPM parameters to discriminate the target keyword from a set of competing words. The set of competing words for a given keyword is obtained by running a ML trained PPM of the given keyword over a development set. Our problem formulation results in an objective function yielding a closed form solution, and hence, eliminating the need for rigorous optimization algorithms. Although the proposed Discriminative Point Process Model (DPPM) fails to perform better than PPM for most of the keywords, we propose a suitable combination of DPPM and PPM that results in a boost in the KWS performance.

Discriminative Training of Point Process Models

For a given threshold ζ, a keyword is considered to be detected if the detector function as given in eq. (2.22) crosses ζ. For a given choice of ζ, we obtain a set of correct detections (correct locations of the target keyword) as well as a set of false alarms (wrongly located occurrences of the target keyword). The performance of an algorithm depends on how many correct detections are achieved with the least possible number of false alarms. Hence, suppression of false alarms is an important aspect to all KWS algorithms. Motivated by this idea, we formulate a disriminative training procedure for PPM.

Problem Formulation

For discriminative training we consider the objective function as shown in eq. (3.1),

$$\underset{\substack{p\in\mathcal{P},\\ 1\le d\le D}}{\hat{\lambda}^{(w)}_{p,d}} = \arg\max_{\lambda^{(w)}_{p,d}} \log \frac{\left(\prod_{K_w=1}^{\hat{K}_w}\prod_{p\in\mathcal{P}}\prod_{d=1}^{D}(\lambda^{(w)}_{p,d})^{n^{(K_w)}_{p,d}}\exp\left(-\lambda^{(w)}_{p,d}\frac{T^{(K_w)}}{D}\right)\right)^{\frac{1}{\hat{K}_w}}}{\left(\prod_{y\in w_c}\left(\prod_{K_y=1}^{\hat{K}_y}\prod_{p\in\mathcal{P}}\prod_{d=1}^{D}(\lambda^{(w)}_{p,d})^{n^{(K_y)}_{p,d}}\exp\left(-\lambda^{(w)}_{p,d}\frac{T^{(K_y)}}{D}\right)\right)^{\frac{1}{\hat{K}_y}}\right)^{\frac{1}{|w_c|}}} \tag{3.1}$$

where $\hat{K}_x$ is the number of keyword training samples for the keyword x, $n_{p,d}^{(K_x)}$ is the count of phoneme p in the d^{th} segment of the training sample number K_x for keyword x, $T^{(K_x)}$ is the length of the K_x-th training sample of the keyword x and w_c is the set of competing words for the keyword w. The numerator of eq. (3.1) is the likelihood of all the training observations of the target keyword w given the model parameters $\theta_w = \{\lambda_{p,d}^{(w)} | p \in \mathcal{P}, d = 1 \leq d \leq D\}$. The denominator term is the likelihood of occurrence of all the competing words $y \in w_c$ with $\hat{K}_y$ number of training samples. We hypothesize that the model parameters obtained by maximizing the ratio of likelihoods would amplify the detector function at the correct locations of the keywords and suppress it at the locations of the competing words. From the first order optimality condition, the solution to the above maximization problem is given by eq. (3.2)

$$\hat{\lambda}_{p,d}^{(w)} = \frac{\frac{1}{\hat{K}} \sum_{K_w=1}^{\hat{K}_w} n_{p,d}^{(K_w)} - \frac{1}{|w_c|} \sum_{y \in w_c} \left[\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} n_{p,d}^{(K_y)} \right]}{\frac{1}{D} \left[\frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} T^{(K_w)} - \frac{1}{|w_c|} \sum_{y \in w_c} \left(\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} T^{(K_y)} \right) \right]} \tag{3.2}$$

which can be derived as shown in eq. (3.3).

$$\begin{aligned}
\mathcal{L}\left(\lambda_{\substack{p,d \\ p \in \mathcal{P}, d \in \{1,2,..D\}}}^{(w)}\right) &= \frac{1}{K_w} \sum_{K_w=1}^{\hat{K}_w} \sum_{p \in \mathcal{P}} \sum_{d=1}^{D} \left[n_{p,d}^{(K_w)} \log(\lambda_{p,d}^{(w)}) - \lambda_{p,d}^{(w)} \frac{T^{(K_w)}}{D} \right] \\
&- \frac{1}{|w_c|} \sum_{y \in w_c} \left[\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} \sum_{p \in \mathcal{P}} \sum_{d=1}^{D} \left(n_{p,d}^{(K_y)} \log(\lambda_{p,d}^{(w)}) - \lambda_{p,d}^{(w)} \frac{T^{(K_y)}}{D} \right) \right] \\
\frac{\partial \mathcal{L}}{\partial \lambda_{p,d}} &= 0 \\
&\implies \frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} \frac{n_{p,d}^{(K_w)}}{\lambda_{p,d}^{(w)}} - \frac{1}{|w_c|} \sum_{y \in w_c} \left[\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} \frac{n_{p,d}^{(K_y)}}{\lambda_{p,d}^{(w)}} \right] = \frac{1}{D} \frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} T^{(K_w)} - \frac{1}{D} \frac{1}{|w_c|} \sum_{y \in w_c} \left(\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}} T^{(K_y)} \right) \\
&\implies \lambda_{p,d}^{(w)} = \frac{\frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} n_{p,d}^{(K_w)} - \frac{1}{|w_c|} \sum_{y \in w_c} \left[\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} n_{p,d}^{(K_y)} \right]}{\frac{1}{D} \left[\frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} T^{(K_w)} - \frac{1}{|w_c|} \sum y \in w_c \left(\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} T^{(K_y)} \right) \right]}
\end{aligned} \tag{3.3}$$

From the second order optimality condition, we get

$$\frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} n_{p,d}^{(K_w)} > \frac{1}{|w_c|} \sum_{y \in w_c} \left(\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} n_{p,d}^{(K_y)} \right) \tag{3.4}$$

for existence of the maxima as derived in eq. (3.5).

$$\frac{\partial^2 \mathcal{L}}{\partial \lambda_{p,d}^2} = -\frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} \frac{n_{p,d}^{(K_w)}}{{\lambda_{p,d}^{(w)}}^2} + \frac{1}{|w_c|} \sum_{y \in w_c} \left[\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} \frac{n_{p,d}^{(K_y)}}{{\lambda_{p,d}^{(w)}}^2} \right] < 0 \tag{3.5}$$

$$\implies \frac{1}{{\lambda_{p,d}^{(w)}}^2} \left[-\frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} n_{p,d}^{(K_w)} + \frac{1}{|w_c|} \sum_{y \in w_c} \left(\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} n_{p,d}^{(K_y)} \right) \right] < 0$$

It can be seen from eq. (3.4) that the existence of a maxima depends on the observations, and hence, is not guaranteed. Thus we modify the objective function eq. (3.1) as in eq. (3.6),

$$\underset{p \in \mathcal{P}, d \in \{1,2,..D\}}{\hat{\lambda}_{p,d}^{(w)}} = \arg\max_{\lambda_{p,d}^{(w)}} \log \frac{\left(\prod_{K_w=1}^{\hat{K}_w} \prod_{p \in \mathcal{P}} \prod_{d=1}^{D} (\lambda_{p,d}^{(w)})^{(n_{p,d}^{(K_w)} + \hat{K}_w \gamma_{p,d})} \exp\left(-\lambda_{p,d}^{(w)} \frac{T^{(K_w)}}{D} \right) \right)^{\frac{1}{\hat{K}_w}}}{\left(\prod_{y \in w_c} \left(\prod_{K_y=1}^{\hat{K}_y} \prod_{p \in \mathcal{P}} \prod_{d=1}^{D} (\lambda_{p,d}^{(w)})^{n_{p,d}^{(K_y)}} \exp\left(-\lambda_{p,d}^{(w)} \frac{T^{(K_y)}}{D} \right) \right)^{\frac{1}{\hat{K}_y}} \right)^{\frac{1}{|w_c|}}} \tag{3.6}$$

where $\gamma_{p,d}$ is the stabilizing factor required for the optimal solution to satisfy the second order optimality condition. The optimal solution for the objective function eq. (3.6) is given by equation eq. (3.7)

$$\hat{\lambda}_{p,d}^{(w)} = \frac{\frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} n_{p,d}^{(K_w)} - \frac{1}{|w_c|} \sum_{y \in w_c} \left[\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} n_{p,d}^{(K_y)} \right] + \gamma_{p,d}}{\frac{1}{D} \left[\frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} T^{(K_w)} - \frac{1}{|w_c|} \sum y \in w_c (\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} T^{(K_y)}) \right]} \tag{3.7}$$

which can be derived in a similar manner to eq. (3.3). On the other hand, the second order condition for the optima to exist is given by

$$\gamma_{p,d} > \frac{1}{|w_c|} \sum_{y \in w_c} \left(\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} n_{p,d}^{(K_y)} \right) - \frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} n_{p,d}^{(K_w)}. \tag{3.8}$$

which can be proved following eq. (3.5). A suitable value of $\gamma_{p,d}$ is selected for each phoneme p and segment d for each keyword such that the solution eq. (3.7) is the optimal solution. The stabilizing factor $\gamma_{p,d}$ can be interpreted as a boost in the number of phonetic event count $n_{p,d}^{(K_w)}$ by an extra $\gamma_{p,d}$ number of phonetic events.

Obtaining w_c

For a target keyword w, a PPM is trained using ML estimation as described in [5] and the detector functions are obtained over a development set. We obtain the maximum value of all the corresponding detector functions and select a threshold ζ_q which is $q\%$ below that value. w_c is obtained by selecting all the words that has occurred nearest to the peaks in all detector plots from the development set above the threshold ζ_q. The set w_c is obtain using the value of q that yields the best performance for the DPPM model on the development set. In our experiments (Section 3.4.2) we obtained w_c for each keyword w based on the performance of the DPPM model with four values of q - 70, 75, 80 and 85.

PPM and DPPM combination

We observe that DPPM is effective in suppressing false alarms, unlike PPM; however, given a threshold, DPPM also misses some true keyword locations. This results in DPPM not performing better than PPM for many keywords. Because PPM and DPPM both have their individual merits, we propose a combination of the detector functions obtained from PPM and DPPM ($d_w^{(PPM)}(t)$ and $d_w^{(DPPM)}(t)$ respectively) for a keyword in such a way that the combined model has a better overall performance than both DPPM as well as PPM individually. Keeping in mind the utilities of DPPM and PPM, we define a combined detector function $d_w^{(PPM-DPPM)}$ as

$$d_w^{(PPM-DPPM)}(t) = \begin{cases} d_w^{(PPM)}(t) \text{ for } d_w^{(DPPM)}(t) \geq \alpha_w \\ d_w^{(DPPM)}(t) \text{ for } d_w^{(DPPM)}(t) < \alpha_w \end{cases} \tag{3.9}$$

The value of α_w is chosen according to the best performance achieved on a development set for each keyword w.

Experimental Setup

To evaluate the proposed DPPM algorithm and the PPM-DPPM combination, we use 14 keywords chosen from two datasets, namely, TIMIT [27] and the BURS [13] corpora. The keywords chosen from the TIMIT are **dark, suit, greasy, wash, water, year, carry, oily, always, about,**

through, enough, every, children. We use 4620 sentences in the TIMIT training set for training PPM as well as DPPM. On the other hand, we use a test and development set of 740 sentences consisting of all the sentences of 24 speakers from TIMIT core test set ($24 \times 10 = 240$ sentences) as well as all the speakers in the development set used by Kaldi [17] TIMIT recipe ($50 \times 10 = 500$ sentences). Half of these sentences is used for development and the remaining half is used for testing purposes. From the BURS corpus, we use the keywords **boston, city, committee, government, hundred, massachusetts, official, percent, president, program, public, thousand, year, yesterday** to test our algorithm. We use all sentences spoken by the speakers F1A, F2B, M1B, M2B, M3B as the train set. The development set and the test set consist of the sentences spoken by the speakers F3A and M4B respectively. The DNNs used for generating posteriorgrams have been trained using Kaldi [17] with 40 dimensional feature-space maximum likelihood linear regression (fMLLR) input features [4] and a context of five frames on either side.

We use two measures to quantify the performance of the KWS algorithms. We define the percentage area under the ROC curve as $AROC = \frac{100 \times A}{f}$, where A is the area under the ROC curve upto a false alarm rate of f. The reason behind taking the normalized area is to make the $AROC$ independent of the support of the curve, which is necessary to compare the performance of different algorithms. The other performance measure that we use to compare the performance of the algorithms is the Figure of Merit (FOM) score [21] which is the average of detection probabilities at $1, 2, \ldots 10$ false alarms/keyword/hour.

Results

The value of α_w, for keyword w is chosen such that the average of the two evaluation metrics AROC and FOM for the detector plots as defined in eq. (3.9) for a development set. For different keywords in TIMIT, α_w lies in the range -80 to -300 and for BURS, it lies in the range of 0 to -130. On the other hand, the value of $\gamma_{p,d}$ is chosen as

$$\gamma_{p,d} = \frac{1}{|w_c|} \sum_{y \in w_c} \left(\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} n_{p,d}^{(K_y)} \right) - \frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} n_{p,d}^{(K_w)} + 10^{(-6)}. \tag{3.10}$$

Table 3.1 provides a detailed comparison for all the 14 keywords in both TIMIT and BURS. It can be seen that AROC for DPPM is less than that of PPM for most words. However, for

TIMIT							BURS						
	FOM			AROC				FOM			AROC		
Keyword	PPM	DPPM	PPM-DPPM	PPM	DPPM	PPM-DPPM	Keyword	PPM	DPPM	PPM-DPPM	PPM	DPPM	PPM-DPPM
about	60	0	60	98.54	81.53	98.93	boston	68.96	63.75	68.96	92.63	89.91	92.63
always	50	60	50	94.35	70.69	94.35	city	7.86	11.43	7.86	90.98	77.68	90.98
carry	96.58	96.84	96.58	95.65	88.54	95.68	committee	61.67	65	61.67	97.57	96.74	97.57
children	100	66.67	100	98.64	65.63	98.64	government	35.41	16.49	35.41	77.27	64.91	77.27
dark	97.11	97.11	97.11	94.76	83	94.82	hundred	54.58	41.88	54.58	86.7	71.62	85.65
enough	100	100	100	100	100	100	massachusetts	80.26	80.26	80.26	97.54	97.54	97.54
every	63.33	83.33	63.33	92.09	91.85	92.09	official	44.8	57.6	57.4	61.75	90.51	90.31
greasy	97.3	97.3	97.3	94.63	77.29	94.63	percent	56.67	56.25	56.67	95.89	95.95	95.89
oily	95.41	94.05	95.41	97.34	93.45	97.34	president	46.88	3.75	46.88	90.15	51.52	90.15
suit	94.86	93.24	94.86	96.45	86.61	96.48	program	67.61	53.89	67.61	94.15	88.26	94.09
through	33.33	86.67	33.33	96.82	90.2	96.97	public	14	9.2	14	74.79	52.74	74.79
wash	97.3	96.76	97.3	95.45	77.6	95.45	thousand	43.89	40.56	43.89	81.45	87.19	81.45
water	96.84	97.11	96.84	95.63	82.31	95.64	year	77.83	77.83	77.83	96.12	96.12	96.12
year	94.1	94.62	94.87	89.92	84.08	90.44	yesterday	33.18	45	33.18	85.06	88.64	88.56
Average	84.01	83.12	84.07	95.73	83.77	95.82	Average	49.54	44.49	50.44	87.29	82.09	89.49

Table 3.1 Comparison of FOM and AROC values using PPM, DPPM and PPM-DPPM for 14 keywords from BURS and TIMIT. The bold entry in row indicate the best performing scheme for the respective keyword. The last row shows the FOM and AROC averaged across all keywords.

PPM-DPPM, it is in most cases greater than or equal to that for PPM, which shows that the PPM-DPPM combination is better than PPM or DPPM individually. A similar observation can be made regarding the FOM performance measure for PPM-DPPM in comparison to PPM or DPPM individually. Although for some words such as **official**, **yesterday**, **always**, **water**, FOM values for DPPM are higher than PPM, it fails to achieve the overall average performance as that of PPM. The reason behind this decrease in performance is that although DPPM, due to its discriminative nature, brings down the number of false alarms, it also results in missing some correct detections of the keyword. This is illustrated in Fig. 3.1 for two randomly chosen keywords **dark** and **greasy**. It is clear that, for each of these keywords, the difference between the false alarm curves from PPM and DPPM is higher than their correct detection curves. It is also observed that for a given threshold, DPPM false alarms are considerably less compared to PPM false alarms at the expense of loosing out on a few correct detections. In fact, as the threshold decreases, the growth in false alarms for lower values of thresholds can be seen to be almost exponential for PPM, while that for DPPM, the rise in false alarms is more gradual. As an example, for the keyword **greasy**, a threshold of -50 results in 100% (38 in total) detection for both PPM and DPPM, yet results in 169 false alarms for PPM but only 59 false alarms for DPPM. However, the performance measures AROC and FOM both do not directly capture this characteristic of DPPM.

We illustrate the benefit of DPPM in suppressing false alarms using detection functions for two exemplary keywords. Fig. 3.2 shows the detector functions $d_w^{(PPM)}(t)$ and $d_w^{(DPPM)}(t)$ obtained

using the PPM and DPPM for the keyword **greasy** on a SA1 sentence from the TIMIT test set. It can be seen that DPPM suppresses some of the false alarm peaks that appear in PPM, while maintaining the amplitude at the location of the keyword to almost the same level as that of the PPM. A similar observation can be made in Fig. 3.3 for the detector functions obtained using the PPM and DPPM for keyword **dark** for another SA1 sentence from the TIMIT test set. Hence, these observations support the benefits of DPPM as shown in the plots in Fig. 3.1.

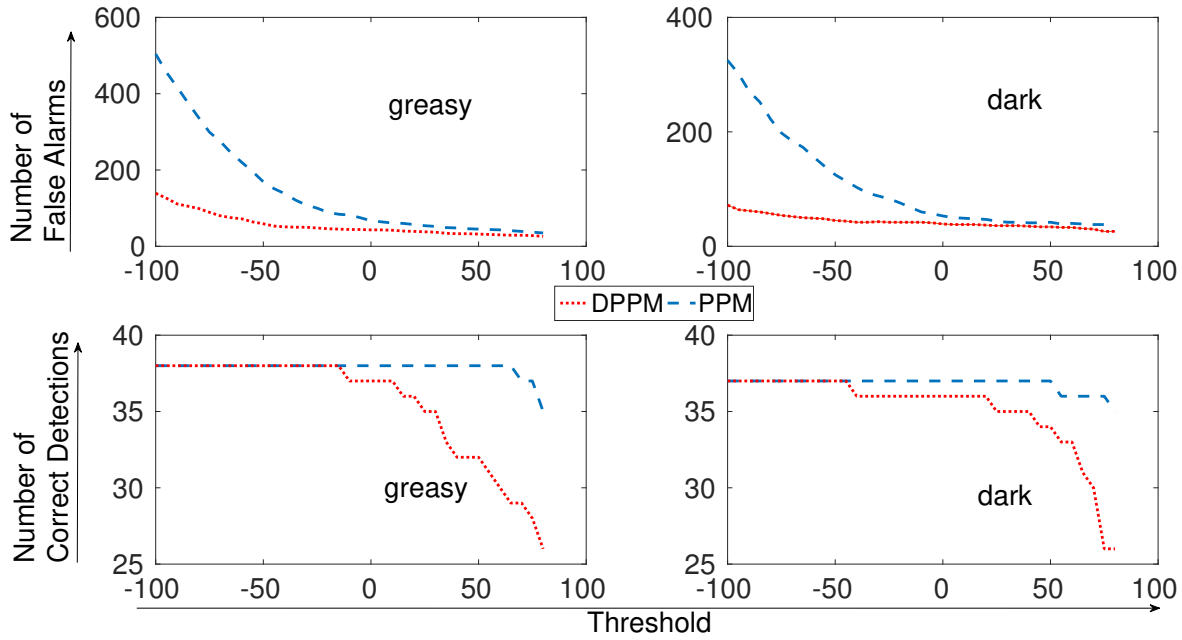


Fig. 3.1 Comparison of false alarm and correct detection of PPM and DPPM for different thresholds for keywords **dark** and **greasy**

Conclusion

Through our experiments, we show that the DPPM-PPM combination performs better than PPM, by utilizing the false alarm rejection capability of DPPM. We also show the specialty of the proposed discriminative training algorithm through sample detector functions showing how DPPM can be better than PPM at suppressing false alarms. This key feature of the DPPM algorithm was emphasized by providing plots showing the number of false alarms and correct detections for PPM and DPPM at different thresholds. In future work, other keyword spotting algorithms can also be trained in the discriminative training paradigm and better discriminative objective functions can

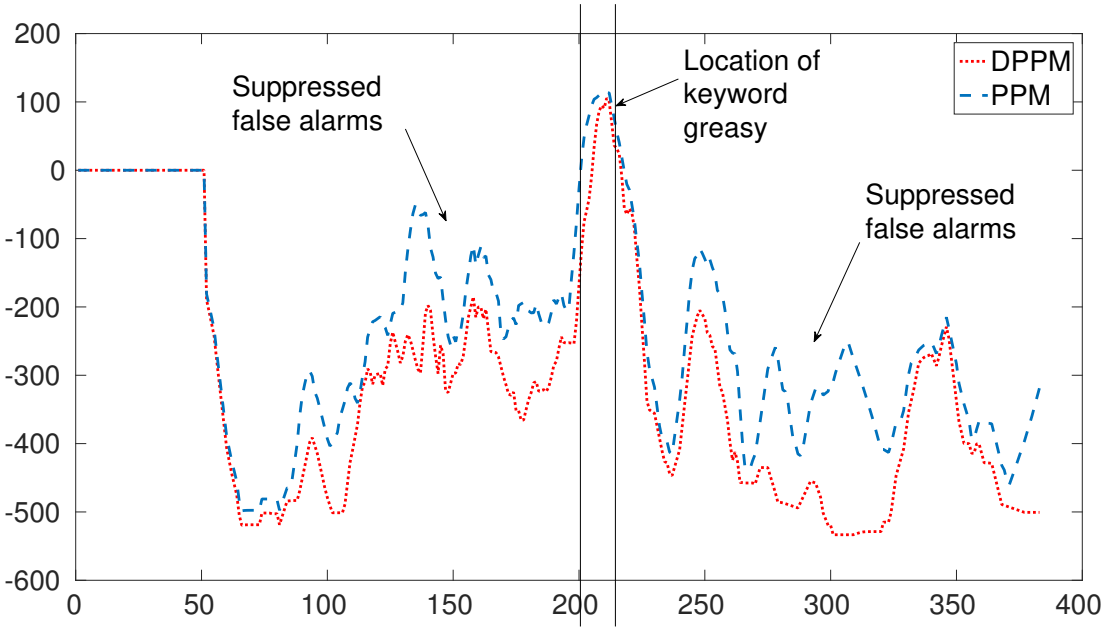


Fig. 3.2 Comparison of detector function of PPM and DPPM for keyword **greasy**

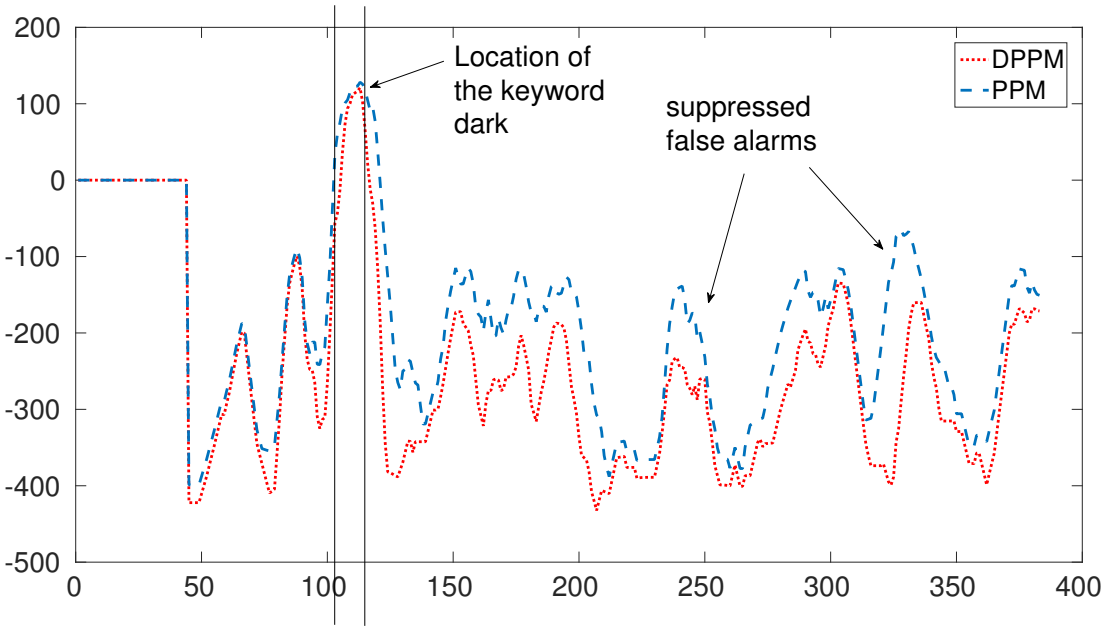


Fig. 3.3 Comparison of detector function of PPM and DPPM for keyword **dark**

be proposed in order to improve their keyword spotting performance.

Chapter 4

Low Resource Point Process Models for Keyword Spotting Using Unsupervised Online Learning

Basis and summary of work

Point Process Models (PPM) have been widely used for keyword spotting applications. Training these models typically requires a considerable number of keyword examples. In this work, we consider a scenario where very few keyword examples are available for training. The availability of a limited number of training examples results in a PPM with poorly learnt parameters. We propose an unsupervised online learning algorithm that starts from a poor PPM model and updates the PPM parameters using newly detected samples of the keyword in a corpus under consideration and uses the updated model for further keyword detection. We test our algorithm on eight keywords taken from the TIMIT database, the training set of which, on average, has 469 samples of each keyword. With an initial set of only five samples of a keyword (corresponds to $\sim 1\%$ of the total number of samples) followed by the proposed online parameter updating throughout the entire TIMIT train set, the performance on the TIMIT test set using the final model is found to be comparable to that of a PPM trained with all the samples of the respective keyword available from the entire TIMIT train set.

Introduction

Keyword Spotting (KWS) using Poisson Process Models (PPM) performs poorly when trained with limited number of training samples [5]. This degradation is detrimental for a situation where not many training samples for the keyword are available, but a good keyword spotting performance is in demand. An example scenario can be the one where lots of intercepted voice communications from a secretive group need to be searched for keywords with very few voice examples. Another application would be to detect keywords in languages with limited linguistic resources, because typical automatic speech recognition (ASR) systems do not support more than 50-100 languages [26]. The idea behind the present work is to initiate a PPM with few available keyword samples and then use carefully chosen newly detected samples using this initial PPM to update the PPM model parameters. Hence, the proposed approach is, in principle, unsupervised in nature and works with a small set of annotated keywords. While there are several unsupervised approaches to KWS [26][14], to the best of our knowledge, there is no work that incorporates an online model updating scheme to enhance the performance over the course of an online learning corpus. KWS experiments with eight keywords from the TIMIT corpus show that a PPM, initialized with only five samples and updated using the proposed online learning algorithm performs as good as a PPM trained with all annotated samples in the onilne learning corpus (approximately 469 samples, on average, per keyword). We begin with a brief description of PPM for KWS.

Unsupervised Online Learning in PPM based KWS

The steps of the proposed unsupervised online learning algorithm are illustrated in Fig. 4.1. At the beginning of the algorithm, we initialize a PPM with parameters $\theta_w^{(K_{start})}$ learnt from K_{start} training samples of a keyword as described in section 4.3.1 and an initial estimate of keyword detection threshold $\gamma(K_{start})$. At any point of the online learning, we use the current model to determine the detector plot $d_w(t)$ on the speech from an online learning corpus as described in the section 2.2.5 (indicated by [A] in Fig. 1). Given the speech, the proposed algorithm detects new occurrences of the keyword using a procedure outlined in section 4.3.2 (indicated by [B] in Fig. 1). Once the k^{th} ($k > K_{start}$) sample of the keyword is detected, we update the threshold value

to $\gamma(k)$ as described in section 4.3.3 and the learning factor to $\alpha(k)$ as described in section 4.3.4 (indicated by [C] and [D] in Fig. 1 respectively). The PPM model is updated using the new value $\alpha(k)$ (indicated by [E] in Fig. 1). If no keyword is detected in the given speech, the PPM does not undergo any update.

Initial Model

We assume a scenario where not many annotated training samples of the keyword are available. Suppose only K_{start} training samples of the keyword are available to begin with and we train a PPM and a word duration model with these K_{start} samples following the steps outlined in [5]. The parameters from the resulting model are used as the initial estimate for the proposed online learning algorithm. However, the word duration model is not updated and remains fixed throughout the learning process.

New location and duration hypotheses

Given the speech from the online learning corpus, we obtain the detector plot $d_w(t)$ using equation (2.22). We estimate the location and duration of a new keyword sample by the following steps:

Location $t^{(k)}$ of the k^{th} keyword

Let $\gamma(k-1)$ be the threshold after $(k-1)$ keywords have been detected. Suppose the k^{th} sample is detected in the region $[\tau_1^{(k)}, \tau_2^{(k)}]$ where $d_w(\tau_1^{(k)} + \epsilon) > \gamma(k-1)$ and $d_w(\tau_1^{(k)} - \epsilon) < \gamma(k-1)$ for a small value $\epsilon > 0$ and $\tau_2^{(k)}$ is the first time instant after $\tau_1^{(k)}$ where $d_w(\tau_2^{(k)} + \epsilon) < \gamma(k-1)$ and $d_w(\tau_2^{(k)} - \epsilon) > \gamma(k-1)$ for a small value $\epsilon > 0$. The end location of the k^{th} keyword is hypothesized to occur at

$$t^{(k)} = \arg \max_{\tau_1^{(k)} < t < \tau_2^{(k)}} d_w(t). \tag{4.1}$$

We also define

$$\beta_k = \max_{\tau_1^{(k)} < t < \tau_2^{(k)}} d_w(t). \tag{4.2}$$

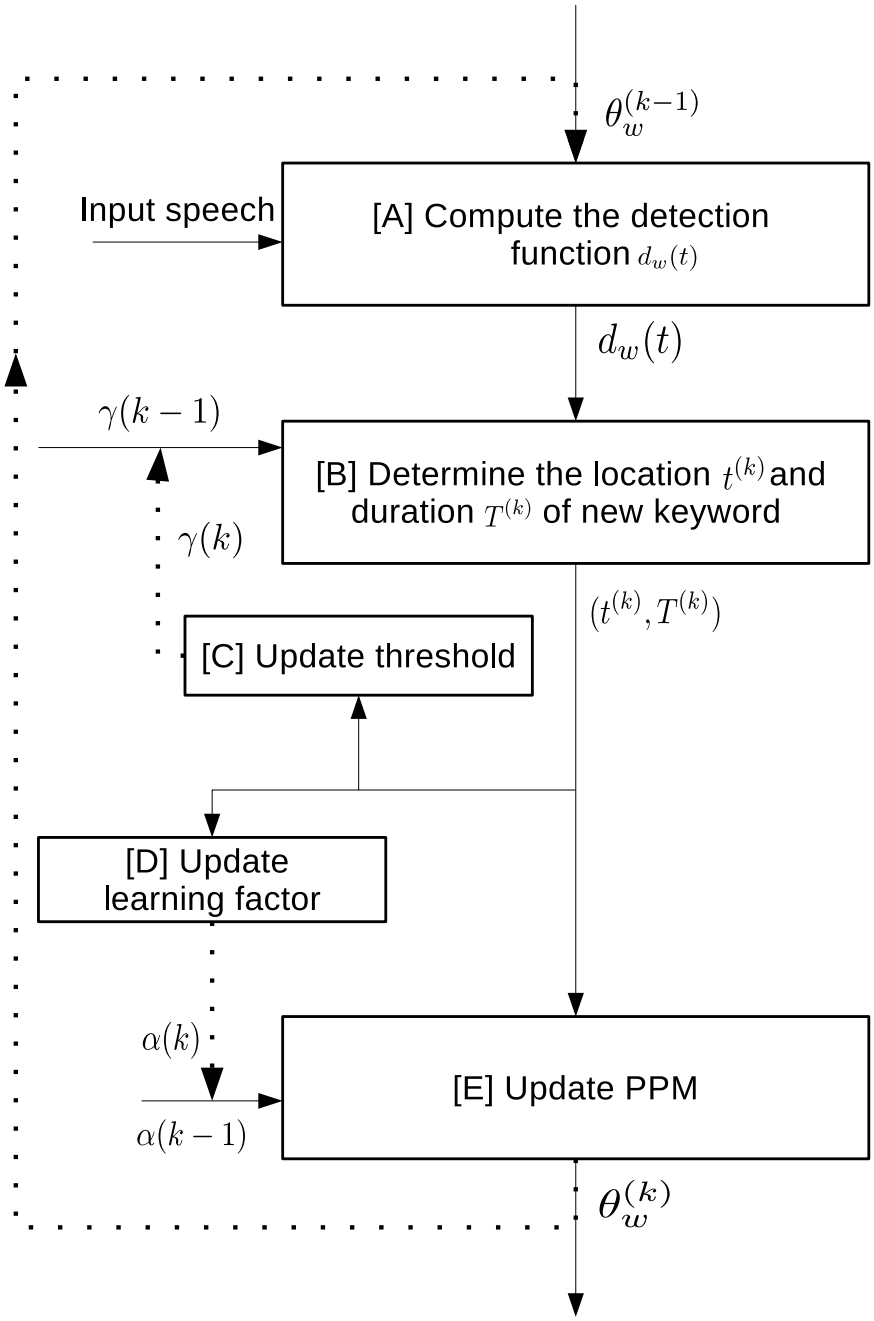


Fig. 4.1 Block diagram summarizing the online learning steps for updating the PPM.

Duration $T^{(k)}$ of the k^{th} keyword occurring at time $t^{(k)}$

From the word duration model $\beta(T|w)$, we consider four potential durations of the keyword and the duration of the k^{th} newly detected sample is estimated using equation (4.3).

$$T^{(k)} = \arg \max_{n \in \{-1,0,1,2\}} P\left(O^{24}_{\mu_w + n\sigma_w}(t^{(k)}) \,\middle|\, \mu_w + n\sigma_w, \theta_w^{(k-1)}\right) \tag{4.3}$$

Updating $\gamma(k)$ after detection of the k^{th} sample of a keyword

A low $\gamma(k)$ value can potentially give rise to a lot of false alarms, which may, in turn, result in a poor PPM. On the other hand, a very high value of $\gamma(k)$ may result in true rejections leading to a poor model too. In our algorithm, after determining k keyword locations and durations, we derive the set

$$M^{(k)}(w) = \{\beta_{\hat{k}} | 1 \leq \hat{k} \leq k\} \tag{4.4}$$

We propose that the value of $\gamma(k)$ after detecting the k^{th} sample of a keyword be assigned to

$$\gamma(k) = \begin{cases} 0.1 \times median(M^{(k)}(w)) \text{ for } k = K_{start} \\ 0.5 \times median(M^{(k)}(w)) \text{ for } k > K_{start} \end{cases} \tag{4.5}$$

The set $M^{(K_{start})}(w)$ consists of the values of β_k corresponding to the initial K_{start} samples. We take 10% of the median value at the beginning of the algorithm to encourage accurate detection of keywords as the initial model might not be rich enough to give a high response at new keyword locations. The purpose of choosing the median instead of mean is to avoid $\gamma(k)$ to be influenced by outlier values in $M^{(k)}(w)$ due to false alarms.

Model updating

Once the k^{th} sample of w is detected at location $t^{(k)}$ with duration $T^{(k)}$, we update the parameter set θ_w by the following operations.

Calculate rate parameter set $\hat{\theta}_w$ for the new example

Consider the current set of rate-parameters at the end of detecting $k-1$ samples of keyword w

$$\theta_w^{(k-1)} = \left\{ \lambda_{p,d}^{(k-1)} \middle| \, p \in \mathcal{P}, 1 \leq d \leq D \right\} \tag{4.6}$$

A new set of piece-wise constant rate-parameters is obtained for the newly detected keyword by maximizing the likelihood function as

$$\hat{\theta}_w = \arg\max_{\theta_w} P\left(O_{T^{(k)}}(t^{(k)}) | T^{(k)}, \theta_w \right) \tag{4.7}$$

The solution to the above optimization problem is obtained as

$$\hat{\theta}_w = \left\{ \hat{\lambda}_{p,d} = \frac{n_{p,d} D}{T^{(k)}} \middle| \, p \in \mathcal{P}, 1 \leq d \leq D \} \right\} \tag{4.8}$$

where, $n_{p,d}$ is the number of phonetic events for the p^{th} phoneme in the d^{th} segment of the newly detected keyword.

Obtaining the updated parameter set $\theta_w^{(k)}$

The updated set of rate parameters is obtained by a convex combination of the elements from the above two sets using a learning factor $\alpha(k)$.

$$\begin{aligned} \theta_w^{(k)} = \{ \lambda_{p,d}^{(k)} =& (\alpha(k)) \lambda_{p,d}^{(k-1)} + (1 - \alpha(k)) \hat{\lambda}_{p,d} \\ & | \, p \in \mathcal{P}, 1 \leq d \leq D \} \end{aligned} \tag{4.9}$$

The choice of a proper value of $\alpha(k)$ is essential for arriving at a good set of model parameters after the algorithm runs over the entire online learning corpus. We choose the value of $\alpha(k)$ to be

$$\alpha(k) = \frac{\sum_{\hat{k}=1}^{k-1} T^{(\hat{k})}}{\sum_{\hat{k}=1}^{k} T^{(\hat{k})}}. \tag{4.10}$$

The ML solution for the set $\theta_w^{(k-1)}$ by maximizing the likelihood function (4.11)

$$\begin{aligned} P(\mathcal{O}_M^{(w)} | T_w, \theta_w) &= \prod_{m=1}^{M} P(O_{T^{(m)}} | T^{(m)}, \theta_w) \\ &= \prod_{m=1}^{M} \prod_{d=1}^{D} \prod_{p \in \mathcal{P}} (\lambda_{p,d})^{n_{p,d}^{(m)}} \exp\left(\frac{-\lambda_{p,d} T^{(m)}}{D} \right) \end{aligned} \tag{4.11}$$

with the observations $\mathcal{O}_{(k-1)}^{(w)}$ is obtained as

$$\theta_w^{(k-1)} = \left\{ \lambda_{p,d}^{(k-1)} = \frac{\sum_{\hat{k}=1}^{(k-1)} n_{p,d}^{(\hat{k})} D}{\sum_{\hat{k}=1}^{(k-1)} T^{(\hat{k})}} \middle| \, p \in \mathcal{P}, 1 \leq d \leq D \right\} \tag{4.12}$$

where, $T^{(\hat{k})}$ is the duration of the $\hat{k}^{th}$ detected sample and $n_{p,d}^{(\hat{k})}$ is the number of events for the p^{th}

phoneme in the d^{th} segment in the $\hat{k}^{th}$ detected sample. Inclusion of one more training sample to $\mathcal{O}_k^{(w)}$ would modify the solution of the parameter as

$$\begin{aligned}
\theta_w^{(k)} &= \left\{ \lambda_{p,d}^{(k)} = \frac{\sum_{\hat{k}=1}^{k} n_{p,d}^{(\hat{k})} D}{\sum_{\hat{k}=1}^{k} T^{(\hat{k})}} \middle| \, p \in \mathcal{P}, 1 \leq d \leq D \right\} \\
&= \left\{ \lambda_{p,d}^{(k)} = \frac{\sum_{\hat{k}=1}^{k-1} T^{(\hat{k})}}{\sum_{\hat{k}=1}^{k} T^{(\hat{k})}} \frac{\sum_{\hat{k}=1}^{k-1} n_{p,d}^{(\hat{k})} D}{\sum_{\hat{k}=1}^{k-1} T^{(\hat{k})}} + \frac{T^{(k)}}{\sum_{\hat{k}=1}^{k} T^{(\hat{k})}} \frac{n_{p,d}^{(k)} D}{T^{(k)}} \right\} \\
&= \left\{ \lambda_{p,d}^{(k)} = \frac{\sum_{\hat{k}=1}^{k-1} T^{(\hat{k})}}{\sum_{\hat{k}=1}^{k} T^{(\hat{k})}} \lambda_{p,d}^{(k-1)} + \frac{T^{(k)}}{\sum_{\hat{k}=1}^{k} T^{(\hat{k})}} \hat{\lambda}_{p,d} \right\}
\end{aligned} \tag{4.13}$$

Hence, from equation (4.13), we can see that choosing $\alpha(k)$ as given is equation (4.10) ensures that the updated parameter at every detected sample of a keyword exactly matches with the respective ML solution.

Experiments and Results

To evaluate our proposed algorithm, we use eight keywords obtained from the TIMIT [27] SA1 and SA2 sentences, namely **greasy, water, dark, wash, carry, oily, suit, year**. We use the TIMIT training set consisting of 4620 sentences for training as well as the online learning corpus for learning the model parameters for each of these eight keywords using the proposed algorithm. The number of keywords in the TIMIT train and test as a pair are (462,74), (479,75), (473,75), (469,74), (463,75), (470,74), (462,74), (473,79) for eight keywords respectively. Out of these 4620 sentences, five sentences containing a keyword are used to train the initial model PPM_{init}. The remaining 4615 sentences are used as the online learning corpus for updating the model using the proposed online learning approach denoted by $PPM_{online}(\zeta)$ where $\zeta \in \{1, 2, \ldots 4615\}$ denotes the index of sentences seen by the algorithm. Hence, $PPM_{online}(\zeta)$ is the new model updated from the previous model $PPM_{online}(\zeta - 1)$ by incorporating the keywords detected in the ζ^{th} sentence. Similarly, we also train a PPM $PPM_{all}(\zeta)$, $\zeta \in \{1, 2, \ldots 4615\}$, using the original keyword locations provided in the word transcriptions upto ζ^{th} sentence available in the TIMIT corpus. We use PPM_{online}^{F} and PPM_{all}^{F} to denote the final models $PPM_{online}(4615)$ and $PPM_{all}(4615)$ respectively. The DNN

used to generate the posteriorgrams in our experiments is obtained from the Kaldi [17] TIMIT recipe . The features used as input to the DNN are 40 dimensional feature-space maximum likelihood linear regression (fMLLR) features [4] with a context of five frames on either side.

The performance of the algorithm is quantified by the percentage area under the Receivers Operating Curves (ROC) obtained by running the models $PPM_{online}(\zeta)$ and $PPM_{all}(\zeta)$ for $\zeta \in \{1, 2, \ldots, 4615\}$ on the TIMIT test set consisting of 740 sentences. These 740 sentences comprise of all the sentences of 24 speakers from TIMIT core test set (24x10=240 sentences) as well as all sentences of 50 speakers from the development set used by the Kaldi TIMIT recipe (50x10=500 sentences). If the area under the ROC is A upto a false alarm rate of f, then the percentage area under the ROC curve is given by $PA_{ROC} = \frac{100 \times A}{f}$

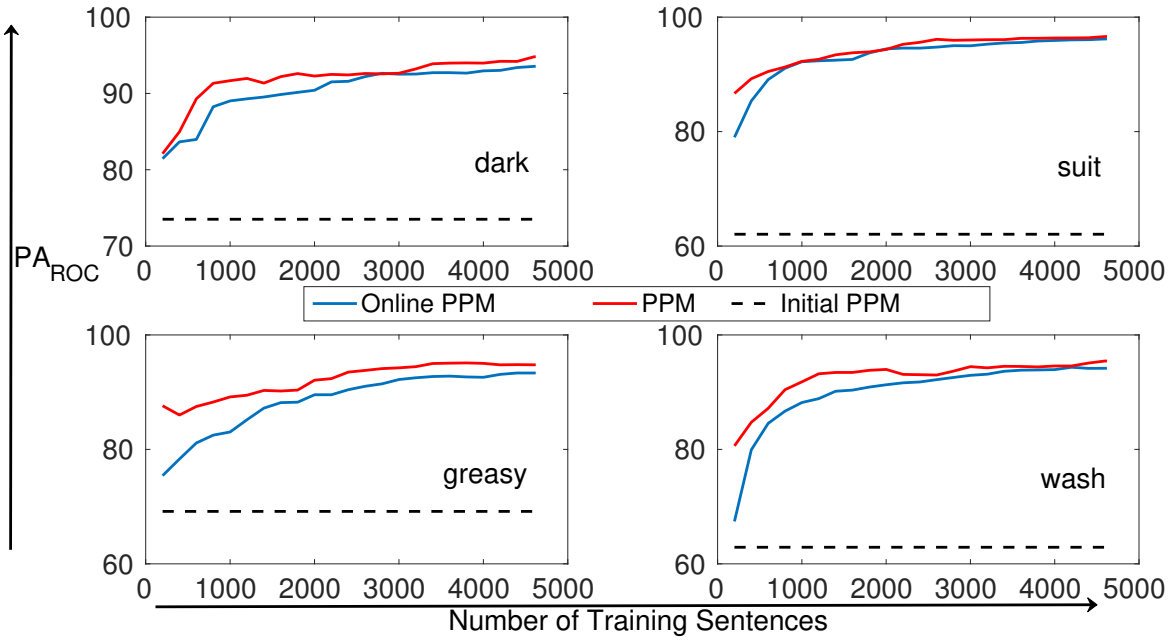


Fig. 4.2 Variation of PA_{ROC} on TIMIT test set as a function of number of observed sentences in the online learning corpus for the four keywords dark, suit, greasy and wash

The purpose of doing this normalization is to get rid of variations of area under the ROC because of differences in the support of the ROC curves. Another performance measure typically used to evaluate KWS performance is the Figure of Merit (FOM) [21] score, which is the mean of a modified ROC curve sampled at ten points. We have found PA_{ROC} to be better than FOM in capturing gradual improvements of the model because FOM takes the value of the ROC curve at certain number of points and does not quantify the overall change (improvement or deterioration)

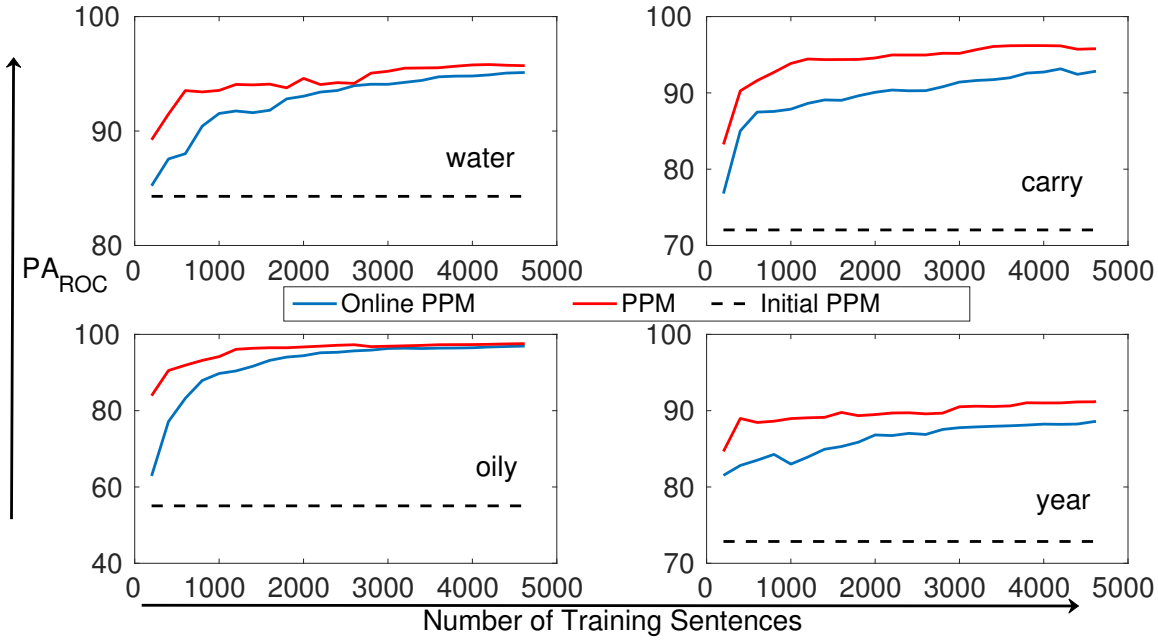


Fig. 4.3 Variation of PA_{ROC} on TIMIT test set as a function of number of observed sentences in the online learning corpus for the four keywords water, carry, oily and year

of the curve. Since each of the models $PPM_{online}(\zeta)$ where $\zeta \in \{1, 2, \ldots 4615\}$, does not change significantly from the previous model, the FOM measure fails to capture the fine change in performance. Hence, we rely on the measure PA_{ROC} to assess the performance of the proposed algorithm. Figs. 4.2 and 4.3 show the variation of PA_{ROC} for the models $PPM_{online}(\zeta)$ and $PPM_{all}(\zeta)$ for $\zeta \in \{1, 2, \ldots 4615\}$. It is clear from these figures that the PA_{ROC} from PPM_{online} gets closer to that from PPM_{all} as the ζ increases.

On the other hand, Figs. 4.4 and 4.5 show a comparison of the actual number of keywords present in the online learning corpus as well as the number of correctly detected keywords and the number of false alarms as a function of ζ. It can be observed that the number of correctly detected keywords as well as the false alarms vary depending on the keyword. This, in turn, determines the quality of the updated PPM. It also suggests that as the number of correctly detected keyword increases, the updated PPM_{online} matches closely with the actual PPM although the online update includes several false alarms.

Table 4.1 provides a comparison of the performance of the final model in terms of FOM score. It can be seen that the average FOM scores for the models PPM^F_{online} and PPM^F_{all} on the TIMIT test are comparable and are 3.9% and 4% better respectively than the FOM score of the initial model

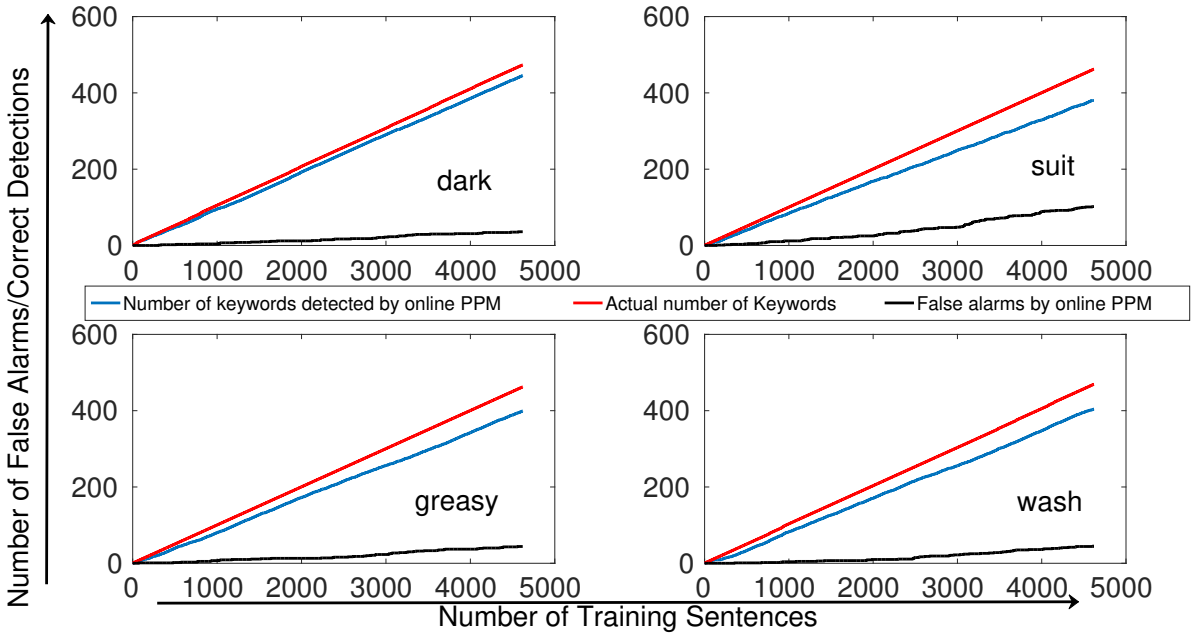


Fig. 4.4 Number of actual, detected keywords and false alarms by online PPM for dark, suit, greasy and wash on the online learning corpus

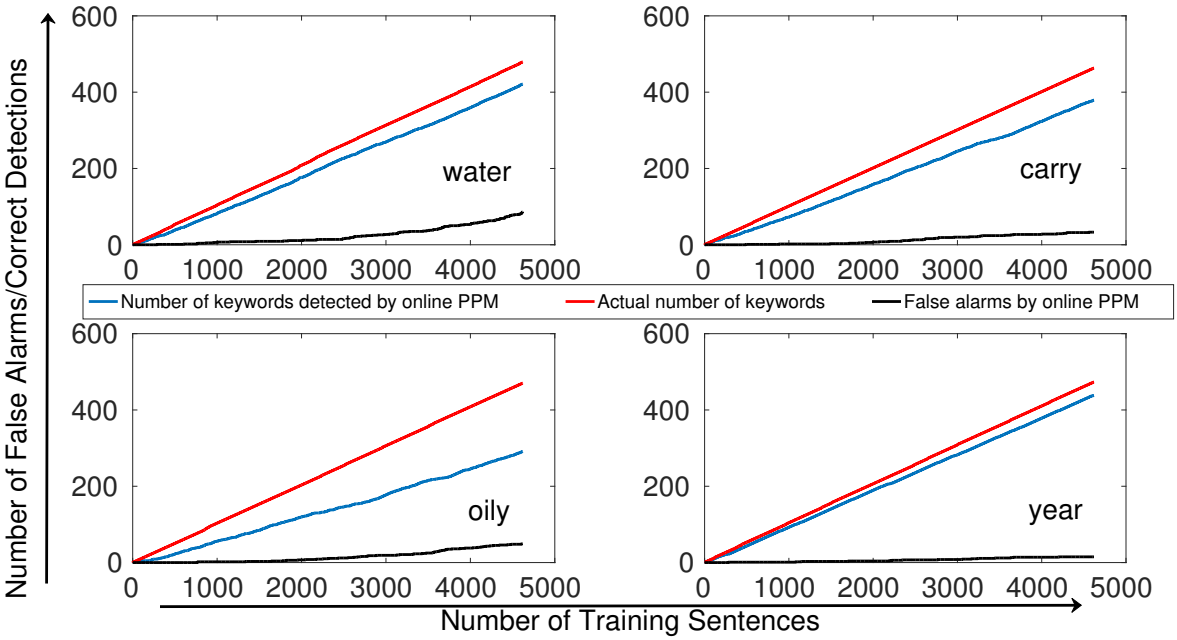


Fig. 4.5 Number of actual, detected keywords and false alarms by online PPM for water, carry, oily and year on the online learning corpus

PPM_{init} on the TIMIT test set. On the other hand, the improvements in PA_{ROC} are 36% and 38% for PPM^{F}_{online} and PPM^{F}_{all} respectively over the initial model PPM_{init} which also indicates

Keyword	PPM_{init}	PPM^{F}_{online}	PPM^{F}_{all}
dark	94.67	96.93	96.80
suit	89.59	96.23	95.27
greasy	94.59	97.03	97.03
wash	95.81	96.76	97.03
water	96.80	96.67	96.93
carry	92.40	96.93	96.67
oily	85.54	96.49	96.08
year	91.39	92.66	95.06
Average	92.60	96.19	96.36

Table 4.1 Comparison of FOM values on TIMIT test set using PPM_{init}, PPM^{F}_{online} and PPM^{F}_{all}

that the performance of PPM^{F}_{online} and PPM^{F}_{all} are similar. This shows that starting with five examples, the proposed online learning algorithm has updated the parameter set such that it results in a performance similar to that of the parameters obtained from original PPM algorithm [5] which is trained on all the available annotated keywords from the entire TIMIT train database. Hence, the proposed algorithm reduces the required number of training samples to approximately 1% of that required by the original PPM algorithm at the expense of a negligible loss in performance

Conclusions

Using experiments with eight keywords from the TIMIT database we show that the proposed unsupervised online PPM training algorithm gives a comparable performance to the supervised PPM algorithm. This algorithm is useful in scenarios where limited number of annotated keyword samples is available for training, for example, in low-resource languages where the transcription data might not be available, also in situations where the keyword is only used by a secretive group for communication. The key features of the proposed approach is the requirement of as less as 1% of the amount of data required for PPM training to achieve a performance similar to that of PPM. In future work, such unsupervised online learning schemes can be extended to other KWS algorithms and also for unsupervised online training of ASR systems.

Chapter 5

Posteriorgram filters for KWS

Introduction

PPM based KWS works with phonetic/senome event based modeling of a keyword obtained by thresholding posteriorgrams. However, ideal **oracle events** can almost never be achieved to exactly determine which phoneme has been uttered at what time instant. On the other hand, by thresholding the posteriorgram, phoneme confusion based information are excluded which can be useful for phoneme classification or keyword modeling and to avoid inaccurate oracle event generation. Hence, in this work, we define 2-D posteriorgram filters which are characterized by a higher output at the locations of the keywords and a lower output elsewhere and study a few these filters and their utility in KWS. In this chapter, we talk about two such posteriorgram filters

1. Matched Filters
2. Level Discriminative Optimal (LDO) Filters

Matched Filter

Training

For a given keyword w, let there be N number of training example. A set of posteriorgrams is obtained from the N training instances $\mathcal{X}^{(w)} = \{X_i^{(w)} | i = 1, 2, \dots N\}$. The posteriogram $X_i^{(w)}$ has a dimension $p \times K_i^{(w)}$, where p is the number of phonemes (typically 48 or 61) and $K^{(w)}$ is the number of frames for the i-th training example. Since training a matched filter using variable $K_i^{(w)}$ is not feasible, we normalize the posteriorgrams to a common dimension $p \times \hat{K}^{(w)}$ where $\hat{K}^{(w)} \geq max\{K_1^{(w)}, K_2^{(w)}, \dots K_N^{(w)}\}$ and hence obtain a normalized set of posteriorgrams $\mathcal{X}_{norm}^{(w)} =$

$\{\hat{X}_i^{(w)} | i = 1, 2, \ldots N\}$. Hence, using these normalized keyword templates, an average matched filter can be obtained as

$$M^{(w)} = \frac{1}{|\mathcal{X}_{norm}^{(w)}|} \sum_{X \in \mathcal{X}_{norm}^{(w)}} X \tag{5.1}$$

Word Duration Model

We require a word duration model for decoding and keyword searching since we need to hypothesize durations of the keyword in test sentences and their probability values for use in decoding. Such a word duration model is obtained in the same way as that of PPM from the set of training posteriorgrams $\mathcal{X}^{(w)} = \{X_i^{(w)} | i = 1, 2, \ldots N\}$. The set of durations $K_i^{(w)}$ is fitted to a **Gaussian distribution** $\mathcal{N}(k|\mu, \sigma)$ to obtain an empirical word duration distribution which is used as the word duration model.

Decoding

We sample the word duration model $\mathcal{N}(k|\mu, \sigma)$ at four points, $\mathcal{S} = \{\mu + n\sigma | n = -1, 0, 1, 2\}$ to obtain a set of sampled values $\mathcal{L} = \{\mathcal{N}(\mu + n\sigma|\mu, \sigma) | n = -1, 0, 1, 2\}$. The detector function using matched filters is defined as follows

$$d_w(t) = \max_{n \in \{-1,0,1,2\}} \left(\sum_{i=1}^{p} \sum_{j=t-\mu-n\sigma+1}^{t} X_{test}[i][j] M_{[\mu+n\sigma]}^{(w)}[i][j - t + \mu + n\sigma] \mathcal{N}(\mu + n\sigma|\mu, \sigma) \right) \tag{5.2}$$

where X_{test} is the posteriorgram of a given test sentence and $M_{[\mu+n\sigma]}^{(w)}$ is the matched filter obtained for keyword w, normalized from $(p \times \hat{K}^{(w)})$ to $(p \times (\mu + n\sigma))$.

Level Discriminative Optimal Filter

Level Discriminative Optimal (LDO) filters are obtained by maximizing a suitable objective function with respect to the filter parameters that assigns a **higher level** to the keyword locations and a **lower value** to a set of competing keyword locations.

The objective function

For a given keyword w, we maximize the following objective function with respect to the filter $M^{(w)}$

$$M^{(w)} = \arg\min_{M^{(w)}} \left[\frac{1}{|L^{(w)}|} \sum_{w_L \in L^{(w)}} \left(\sum_{x=1}^{p} \sum_{y=1}^{\hat{K}} X^{w_L}_{[p\times\hat{K}]}[x,y] M^{(w)}[x,y] - V \right)^2 + \frac{1}{|L_c^{(w)}|} \sum_{\hat{w}_L \in L_c} \left(\sum_{x=1}^{p} \sum_{y=1}^{\hat{K}} X^{\hat{w}_L}_{[p\times\hat{K}]}[x,y] M^{(w)}[x,y] \right)^2 + \sum_x \sum_y (M^{(w)}[x,y])^2 \right] \tag{5.3}$$

where, $X^{w_L}_{[p\times\hat{K}]}$ is the posteriorgram of the w_L-th keyword example from a set of $L^{(w)}$ keyword examples, p is the number of phonemes, senomes or states. Similarly, $X^{\hat{w}_L}_{[p\times\hat{K}]}$ is the posteriorgram of the $\hat{w}_L$-th competing keyword example. $\hat{K}$ is the normalized filter dimension and V is the parameter which sets to a high level (50-100) for the filter output at the keyword locations. The filter output at the competing keyword locations are, by-default, set as close to 0 as possible through minimization of the objective function 5.3.

Obtaining the LDO solution

The LDO objective function 5.3 is convex in nature, and hence, the first order optimality condition is sufficient to prove the existence of an optimal solution and to generate the optimal solution. The first order optimality condition is obtained as

$$\begin{aligned} & \frac{\partial \mathcal{O}}{\partial M[\alpha,\beta]} \\ = \; & 2\left[\frac{1}{|L^{(w)}|} \sum_{w_L \in L^{(w)}} \left(\sum_{x=1}^{p} \sum_{y=1}^{\hat{K}} X^{w_L}_{[p\times\hat{K}]}[x,y] M^{(w)}[x,y] - V \right) X^{w_L}_{[p\times\hat{K}]}[\alpha,\beta] + \frac{1}{|L_c^{(w)}|} \sum_{\hat{w}_L \in L_c^{(w)}} \left(\sum_{x=1}^{p} \sum_{y=1}^{\hat{K}} X^{\hat{w}_L}_{[p\times\hat{K}]}[x,y] M^{(w)}[x,y] \right) X^{\hat{w}_L}_{[p\times\hat{K}]}[\alpha,\beta] + (M^{(w)}[\alpha,\beta]) \right. \\ = \; & 0 \end{aligned} \tag{5.}$$

Equation 5.4 forms a linear equation in terms of the filter coefficients as

$$\sum_{a=1}^{p} \sum_{b=1}^{\hat{K}} \zeta(a,b,\alpha,\beta) M^{(w)}[a,b] = \gamma(\alpha,\beta) \tag{5.5}$$

where

$$\zeta(a,b,\alpha,\beta) = \frac{1}{|L^{(w)}|} \sum_{w_L \in L^{(w)}} X^{(w_L)}_{[p\times\hat{K}]}[a,b] X^{(w_L)}_{[p\times\hat{K}]}[\alpha,\beta] + \frac{1}{|L_c^{(w)}|} \sum_{\hat{w}_L \in L_c^{(w)}} X^{(\hat{w}_L)}_{[p\times\hat{K}]}[a,b] X^{(\hat{w}_L)}_{[p\times\hat{K}]}[\alpha,\beta] + \delta(a-\alpha, b-\beta) \tag{5.6}$$

$$\gamma(\alpha,\beta) = V \sum_{w_L \in L} X^{(w_L)}_{[p\times\hat{K}]}[\alpha,\beta] \tag{5.7}$$

$$\delta(a-\alpha, b-\beta) = \begin{cases} 1, & \text{if } a=\alpha \text{ and } b=\beta \\ 0 & \text{else} \end{cases} \tag{5.8}$$

Taking the derivative of the objective function $\mathcal{O}$ with respect to all filter coefficients results in a set of $p \times \hat{K}$ linear equations in $p \times \hat{K}$ unknowns which can be easily seen to be **full rank**

$$\sum_{a=1}^{p}\sum_{b=1}^{\hat{K}} \zeta(a,b,\alpha,\beta) M^{(w)}[a,b] = \gamma(\alpha,\beta) \text{ for } 1 \leq \alpha \leq p \text{ and } 1 \leq \beta \leq \hat{K} \tag{5.9}$$

or in matrix form

$$Z_{[L,L_c]} M^{(w)} = \Gamma_L \text{ or } M^{(w)} = Z^{-1}_{[L,L_c]} \Gamma_L \tag{5.10}$$

Hence, an analytical solution for $M^{(w)}$ can be derived and application of approximate convex optimization algorithms is not necessary (assuming that the matrix inversion can be accurately estimated). This is an important outcome of such a posteriogram filter formulation.

Choice of $L_c^{(w)}$

The set of training keywords $L^{(w)}$ can be obtained from any standard database. However, the choice of the competing keyword set $L_c^{(w)}$ for a given keyword w is critical for the performance of the algorithm. We select a proper set of competing keywords as follows

1. Train a set of **non-discriminative** filters for each keyword W by minimizing the objective function

$$M^{(w)}_{no-D} = \arg\min_{M^{(w)}} \left[\frac{1}{|L^{(w)}|} \sum_{w_L \in L^{(w)}} \left(\sum_{x=1}^{p}\sum_{y=1}^{\hat{K}} X^{w_L}_{[p\times\hat{K}]}[x,y] M^{(w)}[x,y] - V \right)^2 + \sum_x \sum_y (M^{(w)}[x,y])^2 \right] \tag{5.11}$$

   Analytical solution for equation 5.11 can be obtained as in the case of 5.9 by excluding all the competing keywords terms and expressions.

2. A development set *dev* consisting a few sentences containing keyword w and other non-keyword containing sentences is defined.

3. $M_{no-D}^{(w)}$ is used to search for the keywords w in *dev* to obtain **a set of keywords and false alarm locations** by declaring the words occurring closest to the location where the detector plot exceeds the threshold to be a detected keyword.

4. The obtained set of false alarm words are then sorted in terms of the frequency of occurrence of the false alarm and the top 10 words are chosen as the **list of competing words** $\hat{L}_c^{(w)}$

5. With different thresholds, different lists $\hat{L}_c^{(w)}$ are generated to train $M^{(w)}$ and the performance of the resulting filter on the set *dev* is determined.

6. The list $\hat{L}_c^{(w)}$ giving the best performance on *dev* is defined to be the list of competing words $L_c^{(w)}$

Word Duration Model

A word duration model is obtained in the same way as that of PPM from the set of training posteriorgrams $\mathcal{X}^{(w)} = \left\{ X^{(w_L)}_{[p\times K_i^{(w)}]} | w_L \in L^{(w)} \right\}$. The set of durations $K_i^{(w)}$ is fitted to a **Gaussian distribution** $\mathcal{N}(k|\mu,\sigma)$ to obtain an empirical word duration distribution which is used as the word duration model for decoding purposes.

Decoding

Decoding using LDO filters is performed in the same manner as with matched filters. Sample the word duration model $\mathcal{N}(k|\mu,\sigma)$ at four points, $\mathcal{S} = \{\mu + n\sigma | n = -1, 0, 1, 2\}$ to obtain a set of sampled values $\mathcal{L} = \{\mathcal{N}(\mu + n\sigma|\mu,\sigma) | n = -1, 0, 1, 2\}$. The detector function using matched filters is defined as in section 5.2.3

$$d_w(t) = \max_{n\in\{-1,0,1,2\}} \left(\sum_{i=1}^{p} \sum_{j=t-\mu-n\sigma+1}^{t} X_{test}[i][j] M^{(w)}_{[p\times(\mu+n\sigma)]}[i][j-t+\mu+n\sigma] \mathcal{N}(\mu+n\sigma|\mu,\sigma) \right) \tag{5.12}$$

where X_{test} is the posteriorgram of a given test sentence and $M^{(w)}_{[p\times(\mu+n\sigma)]}$ is the matched filter obtained for keyword w, normalized from $(p \times \hat{K}^{(w)})$ to $(p \times (\mu + n\sigma))$ and t is the frame number.

Experimental Setup

The experimental setup to test the two posteriogram filters (matched filters and LDO filters) on their KWS performance, we use 14 keywords from the TIMIT database, **dark, suit, greasy, wash, water, year, carry, oily, always, about, through, enough, every and children**. The keywords occurring in all 4620 sentences in the TIMIT training set are used for training the posteriorgram filters. We use a test and development set of 740 sentences consisting of all the sentences of 24 speakers from TIMIT core test set ($24 \times 10 = 240$ sentences) as well as all the speakers in the development set used by Kaldi [17] TIMIT recipe ($50 \times 10 = 500$ sentences). Half of these sentences is used for development and the remaining half is used for testing purposes. The DNNs used for generating posteriorgrams have been trained using Kaldi [17] with 40 dimensional feature-space maximum likelihood linear regression (fMLLR) input features [4] and a context of five frames on either side. We use the Receiver Operating Curve (ROC) as a performance measure for the KWS problem. For accurate quantitative comparison between algorithms, the average area under the ROCs of all the 14 keywords is used in this work.

Results

A comparison of the average Receiver Operating Curves (ROC) for three algorithms, namely **PPM, matched filter based KWS and LDO filter based KWS** is provided in fig. 5.1. The average ROC is obtained by computing the ROCs of the indiavidual keywords and taken an average of the plots on the true positive rate (TPR) axis for each false positive rate (FPR). It can be observed that although for matched filter based based KWS, the ROC appears to be considerably below that of the ROC for PPM, the performance of the LDO filter based KWS appears to be at par with PPM. For a finer comparison, we use the area under the ROC as a metric for comparison of the two algorithm. Table 5.1 shows that LDO-KWS has a higher area under the ROC and hence has an advantage over the PPM-KWS algorithm.

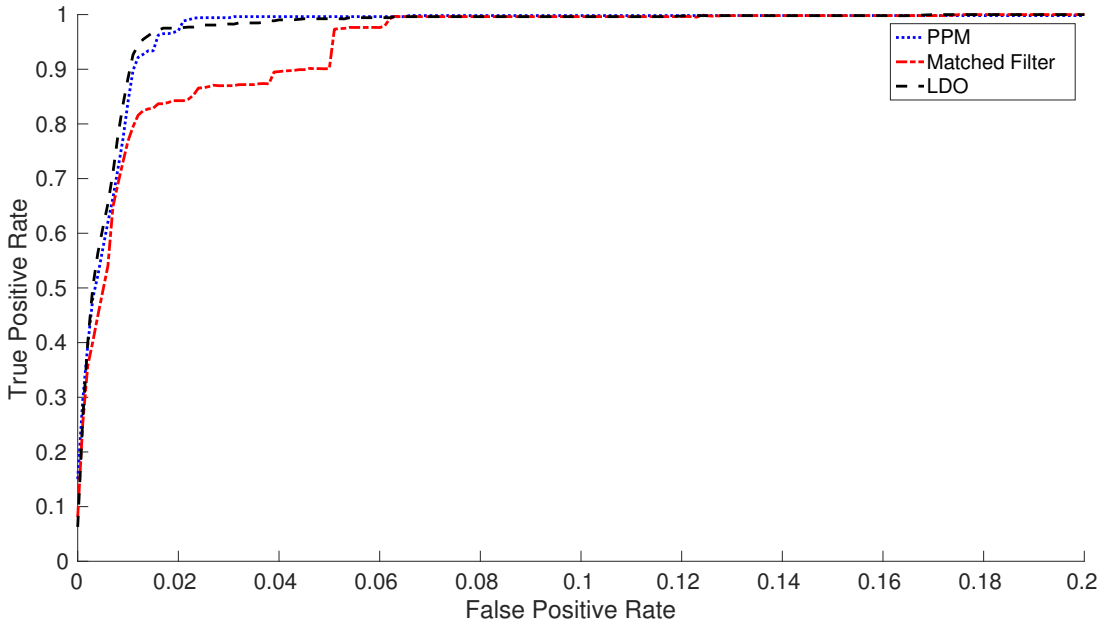


Fig. 5.1 Comparison of the Average Receiver Operating Curves for PPM, Matched Filter based KWS and LDO based KWS for 14 keywords from TIMIT

Algorithm	**Average Area Under ROC**
PPM	0.992990
MF-KWS	0.988951
LDO-KWS	0.994549

Table 5.1 Comparison of average area under ROC for PPM, MF-KWS and LDO-KWS on TIMIT test set for 14 keywords

Chapter 6

Adaptive Matched Filtering Based Fully Unsupervised KWS

Motivation

There are only a small number of languages in the world that have standard databases available for speech recognition and word recognition model training [11]. Research on ASR or KWS technologies have mostly assumed plenty of annotated training data available for system training. Recently, some research has been focused on the low resource problem that is an inevitable bottleneck for further ASR/KWS research mostly for low resource languages. The natural questions that comes to mind immediately are

1. **How well can we train a KWS system with very less annotated training data?**
2. **Can we somehow use the detected keywords for online parameter updating?**

Problem Statement

Given only a **handful ($\approx$ 5) of cut-out snippets** of the acoustic sample (keyword/phrase/sound) to search for and an **un-annotated dataset** consisting of only speech files spoken by a set of speakers, design a KWS algorithm with the limited resources.

Baseline system

We consider a Segmental Dynamic Time Warping (sDTW) based KWS algorithm [26] as the baseline system. The authors move to a completely unsupervised paradigm of KWS exactly in the

same setting as described in section 6.2 with unsupervised **Gaussian Posteriorgram** based feature generation and a modified DTW algorithm to search for pattern matching in the test features. The sDTW algorithm is clearly described in [14] which consists of two modified constraints. There are no global constraints on the DTW, however, there are local constraints, namely

- **Adjustment window condition:** The DTW path is kept restricted to a fat diagonal from the starting point such that the difference in x and y coordinates i_x and i_y do not exceed a parameter R, i.e., $|i_x - i_y| \leq R$
- **Step length of start co-ordinates:** The starting coordinates must be optimally decided. If every frame is considered to be possible starting location for DTW search, the algorithm will take a long time to execute and several redundant overlapping computation will be performed. To optimize the algorithm with respect to computation time, Glass et. al. in [26] proposes a R frame step jump for DTW computation.

Proposed algorithm

Consider a scenario where only **five** templates of a keyword is available for KWS. The algorithm proposed by Glass et. al. [26], is a non-adaptive completely unsupervised algorithm. The algorithm is not benefited by detection of new keyword examples in test scenarios. We propose an adaptive matched filter based algorithm where a initial matched filter is obtained from the five available templates for each keyword with **Gaussian posteriorgram features** and on subsequent detections of probable keywords, it is passed through two levels of screening to reduce false occurrences of keywords to a low value. The matched filter is then adapted to incorporate this new keyword instance as a training example.

Initialization and decoding

Initialization of the filter is done as is described in section 5.2 for a general (Gaussian) posteriorgram based matched filter training but only with 5 samples. Once an intial matched filter MF_{init} is obtained, along with a word duration model (section 5.2.2) training with the five samples, decoding is performed as in section 5.2.3 to search for new keywords.

Detection and selection of new keyword instances

Consider the matched filter for keyword w, $M_k^{(w)}$ after detection of k keywords and let $peak(k)$ be the set of maximum values of the detector output at the keyword locations till the k-th keyword detection. The following steps are followed to obtain the model $M_{k+1}^{(w)}$ form $M_k^{(w)}$.

1. Decode through test sentences using the filter $M_k^{(w)}$.
2. Use a very low threshold $0.2 \times \max(peak(k))$ to get a preliminary location of keyword
3. Once a probable location of the keyword is obtained, the end and duration of the keyword is hypothesized as described in section 4.3.2.
4. The newly detected keyword is verified by a second level of DTW based check where an average DTW distance D_{avg} is computed between the detected keyword and the five keyword templates provided. A threshold $D_{thresh} = 2\times$ the inter DTW score between the five known templates is set and the first level hypothesized keywords having $D_{avg} < D_{thresh}$ is assumed to be surely a keyword location.
5. If the Gaussian posteriorgram features of the detected keyword be GP_{new}, the matched filter $M_{k+1}^{(w)}$ is obtained as
$$M_{k+1}^{(w)} = \frac{k}{k+1} M_k^{(w)} + \frac{1}{k+1} GP_{new} \tag{6.1}$$
Hence, the adaptation is done so as to obtain a filter which would be the same as a matched filter obtained for a set of $k+1$ detected keyword examples.

Experimental Setup

To evaluate our proposed algorithm, we use eight keywords obtained from the TIMIT [27] SA1 and SA2 sentences, namely **greasy, water, dark, wash, carry, oily, suit, year**. We use the TIMIT training set consisting of 4620 sentences for training as well as the **adaptation corpus for learning the model parameters** for each of these eight keywords using the proposed algorithm. The number of keywords in the TIMIT train and test as a pair are (462,74), (479,75), (473,75), (469,74), (463,75), (470,74), (462,74), (473,79) for eight keywords respectively. Out of these 4620 sentences, five sentences containing a keyword are used to train the initial model MF_{init}. The

remaining 4615 sentences are used as the **adaptation corpus** for updating the model using the proposed adaptation method. The performance of the algorithm is quantified by the $P@N$ measure which is the number of correct keyword detections (P) out of the highest scoring N number of detections, where N is the number of ground truth keywords present in the test corpus which has been used to evaluate the baseline system [26]. The test corpus consists of 1680 sentences from the complete TIMIT test corpus.

Results

The Table 6.1 shows a comparison of the $P@N$ values of the 8 selected keywords for the initial model (MF_{init}) and after full adaptation of the matched filter model over the **adaptation corpus** (MF_{adapt}) alongside the baseline system (GP_{base}).

Keyword	MF_{init}	MF_{adapt}	GP_{base}
dark	0.8889	0.5205	0.5088
suit	0.6667	0.3631	0.5774
greasy	0.5298	0.2917	0.6012
wash	0.8631	0.7917	0.7976
water	0.8059	0.5059	0.6000
carry	0.7929	0.5148	0.6391
oily	0.6190	0.5179	0.4940
year	0.8701	0.7797	0.8305
Average	**0.7545**	**0.5356**	**0.6310**

Table 6.1 Comparison of FOM values on TIMIT test set using PPM_{init}, PPM^{F}_{online} and PPM^{F}_{all}

It can be seen that because of the matched filter adaptation on the **adaptation corpus** using the proposed algorithm, the average $P@N$ performance for the 8 keywords improved from 0.5356 to 0.7545. Although the initial average $P@N$ for the baseline system GP_{base} with 5 templates is better than that of MF_{init} trained with 5 templates of the keyword, due to filter adaptation, the $P@N$ performance of MF_{adapt} improves over the baseline GP_{base} system to 0.7545.

Chapter 7

Conclusion and Future Work

Conclusion

In this report, we talk about some of the most prominent research domains in KWS, in particular, low resource KWS and also general improvement of non-ASR KWS algorithms. We proposed **low resource point process models for keyword spotting using unsupervised on-line learning** and **adaptive matched filtering based unsupervised KWS system** towards unsupervised on-line learning or on-line adaptation algorithms that incorporate new keyword detections to enrich the existent model - a concept non-existent in the literature to the best of our knowledge till date. We also worked on **discriminative PPM training** algorithm and **posteriorgram filtering based KWS (matched filters and LDO filters)** as general performance enhancing algorithms over the already existing algorithms in the the domain of non-ASR KWS.

Future Work

In the future, apart from additional work on improving the general KWS performance of these algorithms, and development of KWS algorithms under low-resource setting, some other possible directions of work still exist in non-ASR KWS, namely

1. **Multi-modal KWS** systems and **fusion strategies for keyword informations from multiple modes** (eg. acoustic,visual features, articulatory features etc.)

2. Scope of **system combination of non-ASR and ASR based KWS systems** keeping in mind that ASR systems are usually more accurate and non-ASR KWS systems are simple and fast. A suitable combination of two such systems in a suitable way leads to the possibility of an accurate, yet fast KWS system.

3. Testing of the KWS algorithms developed for low resource scenarios on low resource languages, rather than English.

Bibliography

[1] Guoguo Chen, Carolina Parada, and Georg Heigold. Small-footprint keyword spotting using deep neural networks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4087–4091. IEEE, 2014.

[2] Guoguo Chen, Carolina Parada, and Tara N Sainath. Query-by-example keyword spotting using long short-term memory networks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5236–5240. IEEE, 2015.

[3] Jonathan G Fiscus, Jerome Ajot, John S Garofolo, and George Doddington. Results of the 2006 spoken term detection evaluation. In *NIST Proc. sigir*, volume 7, pages 51–57, 2007.

[4] Mark JF Gales. Maximum likelihood linear transformations for hmm-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998.

[5] Aren Jansen and Partha Niyogi. Point process models for spotting keywords in continuous speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(8):1457–1470, 2009.

[6] Keith Kintzley, Aren Jansen, Kenneth Church, and Hynek Hermansky. Inverting the point process model for fast phonetic keyword search. In *INTERSPEECH*, pages 2438–2441, 2012.

[7] Keith Kintzley, Aren Jansen, and Hynek Hermansky. Event selection from phone posteriorgrams using matched filters. In *INTERSPEECH*, pages 1905–1908, 2011.

[8] Keith Kintzley, Aren Jansen, and Hynek Hermansky. Map estimation of whole-word acoustic models with dictionary priors. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[9] Keith Kintzley, Aren Jansen, and Hynek Hermansky. Text-to-speech inspired duration modeling for improved whole-word acoustic models. In *INTERSPEECH*, pages 1253–1257, 2013.

[10] Keith Kintzley, Aren Jansen, and Hynek Hermansky. Featherweight phonetic keyword search for conversational speech. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 7859–7863. IEEE, 2014.

[11] Mikko Lehtonen. Hierarchical approach for spotting keywords. Technical report, IDIAP, 2005.

[12] Chunxi Liu, Aren Jansen, and Sanjeev Khudanpur. Context-dependent point process models for keyword search and detection-based asr. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 6025–6029. IEEE, 2016.

[13] Mari Ostendorf, Patti J Price, and Stefanie Shattuck-Hufnagel. The boston university radio news corpus. *Linguistic Data Consortium*, pages 1–19, 1995.

[14] Alex S Park and James R Glass. Unsupervised pattern discovery in speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(1):186–197, 2008.

[15] Daniel Povey. *Discriminative training for large vocabulary speech recognition.* PhD thesis, University of Cambridge, 2005.

[16] Daniel Povey. Improvements to fmpe for discriminative training of features. In *Interspeech*, pages 2977–2980, 2005.

[17] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The Kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[18] Daniel Povey, Dimitri Kanevsky, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Karthik Visweswariah. Boosted MMI for model and feature-space discriminative training. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2008.*, pages 4057–4060.

[19] Daniel Povey, Brian Kingsbury, Lidia Mangu, George Saon, Hagen Soltau, and Geoffrey Zweig. fMPE: Discriminatively trained features for speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05).*, volume 1, pages I–961.

[20] Dhananjay Ram, Afsaneh Asaei, Pranay Dighe, and Hervé Bourlard. Sparse modeling of posterior exemplars for keyword detection. In *Proceedings of Interspeech*, number EPFL-CONF-209088, 2015.

[21] J Robin Rohlicek, William Russell, Salim Roukos, and Herbert Gish. Continuous hidden Markov modeling for speaker-independent word spotting. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 627–630, 1989.

[22] Richard C Rose and Douglas B Paul. A hidden markov model based keyword recognition system. In *International Conference on Acoustics, Speech, and Signal Processing, 1990. ICASSP-90*, pages 129–132. IEEE, 1990.

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

[24] Chao Weng and Biing-Hwang Juang. Discriminative training using non-uniform criteria for keyword spotting on spontaneous speech. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(2):300–312, 2015.

[25] Philip C Woodland and Daniel Povey. Large scale discriminative training of hidden markov models for speech recognition. *Computer Speech & Language*, 16(1):25–47, 2002.

[26] Yaodong Zhang and James R Glass. Unsupervised spoken keyword spotting via segmental dtw on gaussian posteriorgrams. In *IEEE Workshop on Automatic Speech Recognition & Understanding, 2009. ASRU*, pages 398–403. IEEE, 2009.

[27] Victor Zue, Stephanie Seneff, and James Glass. Speech database development at MIT: TIMIT and beyond. *Speech Communication*, 9(4):351–356, 1990.