Non-ASR based Keyword Spotting in Continuous Speech

Samik Sadhu
samiksadhu.juee@gmail.com

SPIRE LAB
Electrical Engineering,
Indian Institute of Science (IISc), Bangalore, India

19 May 2017

Section 1

Why Keyword Spotting?

- Motivation 1: Humans detect keywords in speech. Once important keywords are detected, decoding the entire speech can become trivial.

Why Keyword Spotting?

- Motivation 1: Humans detect keywords in speech. Once important keywords are detected, decoding the entire speech can become trivial.
- Motivation 2: Applications like (**indoor automation, human machine interface etc.**)

Why Keyword Spotting?

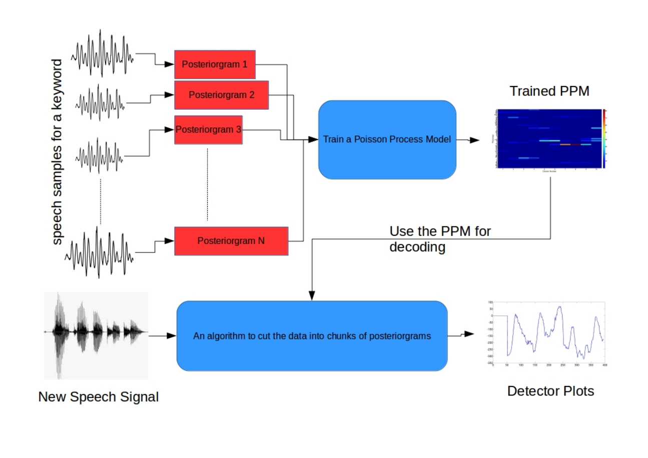
- Motivation 1: Humans detect keywords in speech. Once important keywords are detected, decoding the entire speech can become trivial.
- Motivation 2: Applications like (**indoor automation, human machine interface etc.**)
- Motivation 3: Searching for one word or a phrase over 100 hrs of data (**e.g.: from You-tube**) just like searching a text document for words!

Why Keyword Spotting?

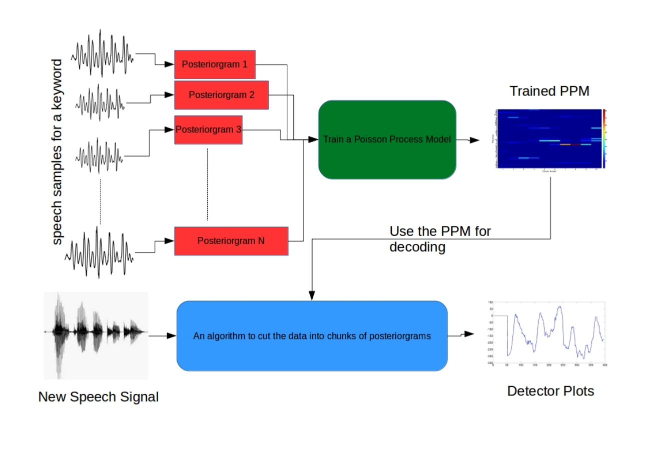
- Motivation 1: Humans detect keywords in speech. Once important keywords are detected, decoding the entire speech can become trivial.
- Motivation 2: Applications like (**indoor automation, human machine interface etc.**)
- Motivation 3: Searching for one word or a phrase over 100 hrs of data (**e.g.: from You-tube**) just like searching a text document for words!
- Motivation 4: Numerous **Cyber-Physical System** applications, **e.g. telemedicine, smart security etc.**

Section 2

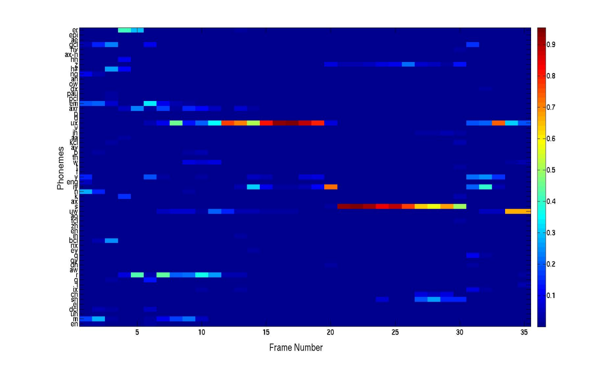
PPM Algorithm Overview



PPM Algorithm Overview

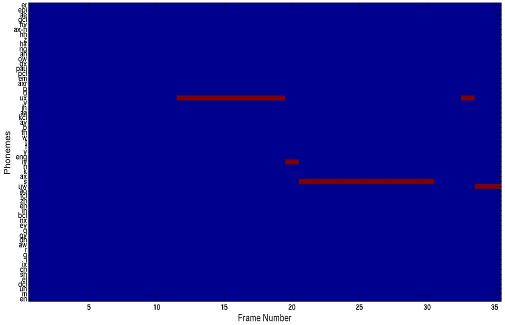


Generating phonetic events from posteriorgrams



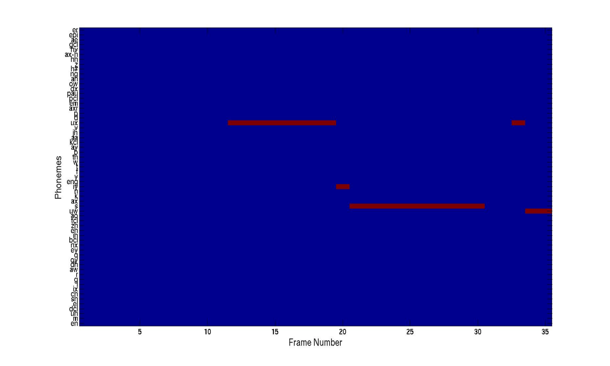
- Select a threshold δ, which is typically of the range $0.2 - 0.5$

Generating phonetic events from posteriorgrams



- Select a threshold δ, which is typically of the range $0.2 - 0.5$
- Threshold the posteriorgram to get the sparse phonetic events

Generating phonetic events from posteriorgrams



- Select a threshold δ, which is typically of the range $0.2 - 0.5$
- Threshold the posteriorgram to get the sparse phonetic events
- **But how do we model these sparse events?**

PPM with piecewise-constant λ

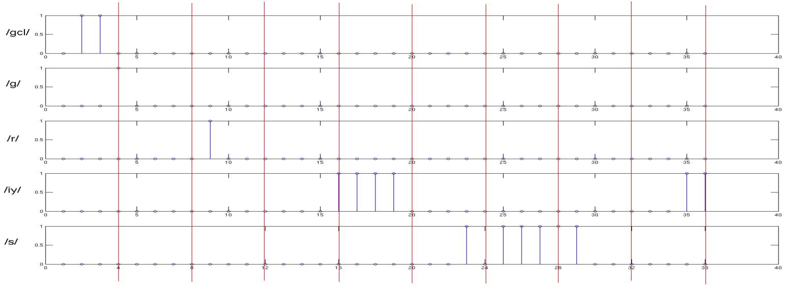


Figure: Posteriorgram and phonetic events for an instance of greasy

- Maintain 61×10 counters C counting the number of event occurrences for each section and each realization of greasy & counter T accumulating the total time of all the training instances

PPM with piecewise-constant λ

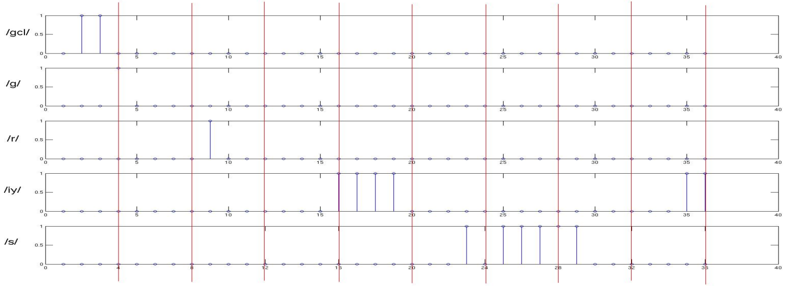


Figure: Posteriorgram and phonetic events for an instance of greasy

- Maintain 61×10 counters C counting the number of event occurrences for each section and each realization of greasy & counter T accumulating the total time of all the training instances
- The lambda matrix $\Lambda = \frac{C}{T/10}$

PPM with piecewise-constant λ

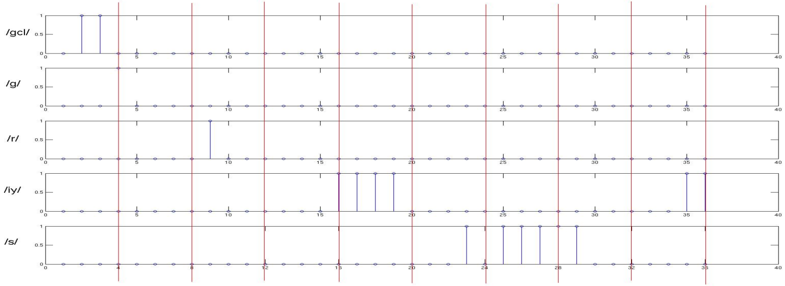
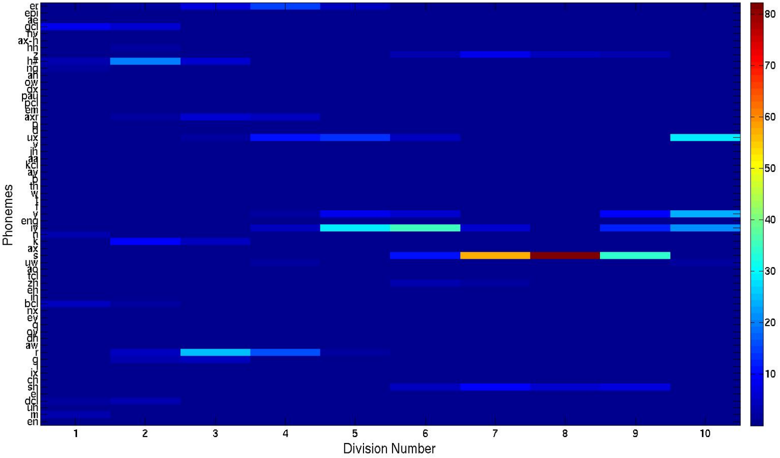


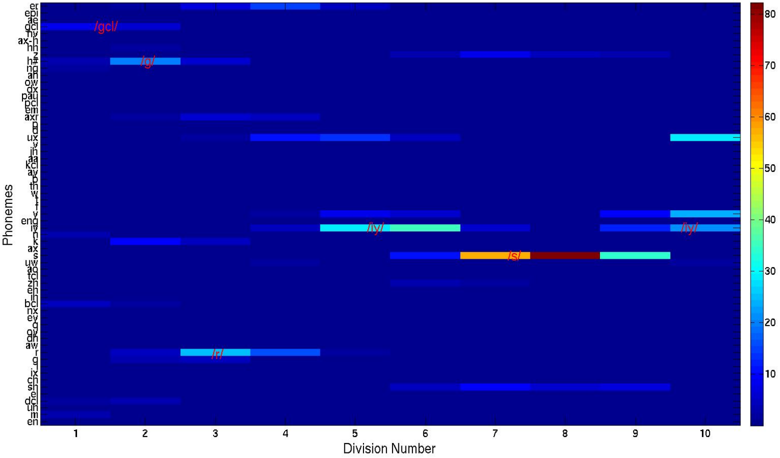
Figure: Posteriorgram and phonetic events for an instance of greasy

- Maintain 61×10 counters C counting the number of event occurrences for each section and each realization of greasy & counter T accumulating the total time of all the training instances
- The lambda matrix $\Lambda = \frac{C}{T/10}$
- **This is the solution of Λ obtained my MLE estimation**

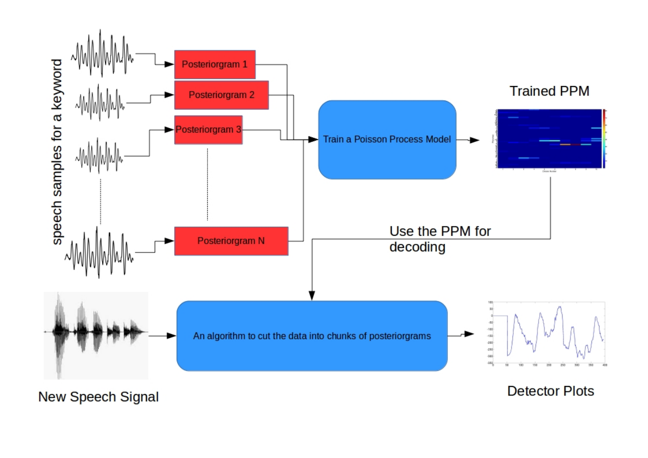
Λ for Greasy looks something like this



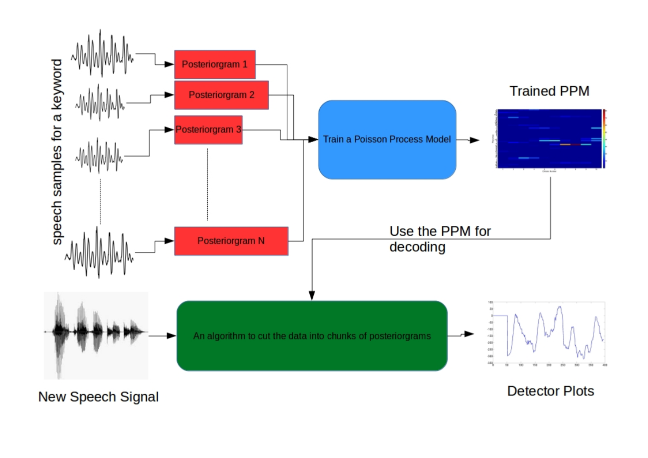
Λ for Greasy looks something like this



Overview of Poisson Process Models



Overview of Poisson Process Models



Word Duration Model $P(T|w)$

- Obtain all keyword durations from the training set

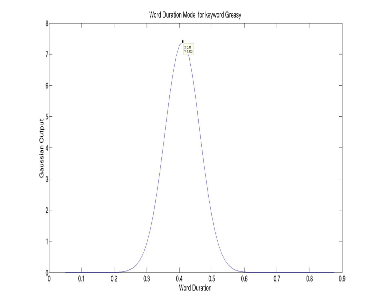


Figure: Word duration model for keyword greasy

Word Duration Model $P(T|w)$

- Obtain all keyword durations from the training set
- **Fit a Gaussian distribution** to the obtained keyword duration data

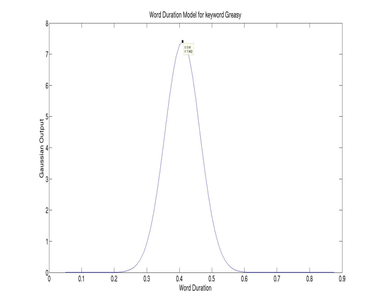


Figure: Word duration model for keyword greasy

Word Duration Model $P(T|w)$

- Obtain all keyword durations from the training set
- **Fit a Gaussian distribution** to the obtained keyword duration data
- If μ is the mean duration, we take 4 probable durations of keyword $\mu + [-1, 0, 1, 2]\sigma$ for decoding at each time instant

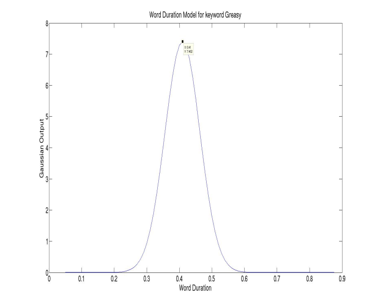


Figure: Word duration model for keyword greasy

Section 3

Change in Objective Function

Original ML optimization problem:

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

Discriminative optimization problem:

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

- $\hat{K}_x$ is the number of keyword training samples for the keyword x

Change in Objective Function

Original ML optimization problem:

$$\hat{\lambda}^{(w)}_{\substack{p,d \\ p\in\mathcal{P},1\leq d\leq D}} = \arg\max_{\lambda^{(w)}_{p,d}} \log\left(\prod_{K_w=1}^{\hat{K}_w}\prod_{p\in\mathcal{P}}\prod_{d=1}^{D}(\lambda^{(w)}_{p,d})^{n^{(K_w)}_{p,d}}\exp\left(-\lambda^{(w)}_{p,d}\frac{T^{(K_w)}}{D}\right)\right) \quad (1)$$

Discriminative optimization problem:

$$\hat{\lambda}^{(w)}_{\substack{p,d \\ p\in\mathcal{P},1\leq d\leq D}} = \arg\max_{\lambda^{(w)}_{p,d}} \log\frac{\left(\prod_{K_w=1}^{\hat{K}_w}\prod_{p\in\mathcal{P}}\prod_{d=1}^{D}(\lambda^{(w)}_{p,d})^{n^{(K_w)}_{p,d}}\exp\left(-\lambda^{(w)}_{p,d}\frac{T^{(K_w)}}{D}\right)\right)^{\frac{1}{\hat{K}_w}}}{\left(\prod_{y\in w_c}\left(\prod_{K_y=1}^{\hat{K}_y}\prod_{p\in\mathcal{P}}\prod_{d=1}^{D}(\lambda^{(w)}_{p,d})^{n^{(K_y)}_{p,d}}\exp\left(-\lambda^{(w)}_{p,d}\frac{T^{(K_y)}}{D}\right)\right)^{\frac{1}{\hat{K}_y}}\right)^{\frac{1}{|w_c|}}} \quad (2)$$

- $\hat{K}_x$ is the number of keyword training samples for the keyword x
- $n^{(K_x)}_{p,d}$ is the count of phoneme p in the d^{th} segment of the training sample number K_x for keyword x

Change in Objective Function

Original ML optimization problem:

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

Discriminative optimization problem:

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

- $\hat{K}_x$ is the number of keyword training samples for the keyword x
- $n^{(K_x)}_{p,d}$ is the count of phoneme p in the d^{th} segment of the training sample number K_x for keyword x
- w_c is the set of competing words for the keyword w

Solving the Maximization Problem

From First Order Condition:

$$\hat{\lambda}_{p,d}^{(w)} = \frac{\frac{1}{\hat{K}} \sum_{K_w=1}^{\hat{K}_w} n_{p,d}^{(K_w)} - \frac{1}{|w_c|} \sum_{y \in w_c} \left[\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} n_{p,d}^{(K_y)} \right]}{\frac{1}{D} \left[\frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} T^{(K_w)} - \frac{1}{|w_c|} \sum_{y \in w_c} \left(\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} T^{(K_y)} \right) \right]} \tag{3}$$

From Second Order Condition:

$$\frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} n_{p,d}^{(K_w)} > \frac{1}{|w_c|} \sum_{y \in w_c} \left(\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} n_{p,d}^{(K_y)} \right) \tag{4}$$

The second order conditions depend strongly on the data, we have to put in more control

Modified Objective Function:

$$\underset{p\in\mathcal{P},d\in\{1,2,..D\}}{\hat{\lambda}_{p,d}^{(w)}} = \underset{\lambda_{p,d}^{(w)}}{\arg\max} \log \frac{\left(\prod_{K_w=1}^{\hat{K}_w}\prod_{p\in\mathcal{P}}\prod_{d=1}^{D}(\lambda_{p,d}^{(w)})^{(n_{p,d}^{(K_w)}+\hat{K}_w\gamma_{p,d})}\exp\left(-\lambda_{p,d}^{(w)}\frac{T^{(K_w)}}{D}\right)\right)^{\frac{1}{\hat{K}_w}}}{\left(\prod_{y\in w_c}\left(\prod_{K_y=1}^{\hat{K}_y}\prod_{p\in\mathcal{P}}\prod_{d=1}^{D}(\lambda_{p,d}^{(w)})^{n_{p,d}^{(K_y)}}\exp\left(-\lambda_{p,d}^{(w)}\frac{T^{(K_y)}}{D}\right)\right)^{\frac{1}{\hat{K}_y}}\right)^{\frac{1}{|w_c|}}} \tag{5}$$

$$\hat{\lambda}_{p,d}^{(w)} = \frac{\frac{1}{\hat{K}_w}\sum_{K_w=1}^{\hat{K}_w} n_{p,d}^{(K_w)} - \frac{1}{|w_c|}\sum_{y\in w_c}\left[\frac{1}{\hat{K}_y}\sum_{K_y=1}^{\hat{K}_y} n_{p,d}^{(K_y)}\right] + \gamma_{p,d}}{\frac{1}{D}\left[\frac{1}{\hat{K}_w}\sum_{K_w=1}^{\hat{K}_w} T^{(K_w)} - \frac{1}{|w_c|}\sum y\in w_c(\frac{1}{\hat{K}_y}\sum_{K_y=1}^{\hat{K}_y} T^{(K_y)})\right]} \tag{6}$$

- $\gamma_{p,d}$ is the stabilizing factor required for the optimal solution to satisfy the second order optimality condition

Modified Objective Function:

$$\underset{p\in\mathcal{P},d\in\{1,2,..D\}}{\hat{\lambda}^{(w)}_{p,d}} = \arg\max_{\lambda^{(w)}_{p,d}} \log \frac{\left(\prod_{K_w=1}^{\hat{K}_w}\prod_{p\in\mathcal{P}}\prod_{d=1}^{D}(\lambda^{(w)}_{p,d})^{(n^{(K_w)}_{p,d}+\hat{K}_w\gamma_{p,d})}\exp\left(-\lambda^{(w)}_{p,d}\frac{T^{(K_w)}}{D}\right)\right)^{\frac{1}{\hat{K}_w}}}{\left(\prod_{y\in w_c}\left(\prod_{K_y=1}^{\hat{K}_y}\prod_{p\in\mathcal{P}}\prod_{d=1}^{D}(\lambda^{(w)}_{p,d})^{n^{(K_y)}_{p,d}}\exp\left(-\lambda^{(w)}_{p,d}\frac{T^{(K_y)}}{D}\right)\right)^{\frac{1}{\hat{K}_y}}\right)^{\frac{1}{|w_c|}}} \tag{5}$$

$$\hat{\lambda}^{(w)}_{p,d} = \frac{\frac{1}{\hat{K}_w}\sum_{K_w=1}^{\hat{K}_w} n^{(K_w)}_{p,d} - \frac{1}{|w_c|}\sum_{y\in w_c}\left[\frac{1}{\hat{K}_y}\sum_{K_y=1}^{\hat{K}_y} n^{(K_y)}_{p,d}\right] + \gamma_{p,d}}{\frac{1}{D}\left[\frac{1}{\hat{K}_w}\sum_{K_w=1}^{\hat{K}_w} T^{(K_w)} - \frac{1}{|w_c|}\sum y \in w_c(\frac{1}{\hat{K}_y}\sum_{K_y=1}^{\hat{K}_y} T^{(K_y)})\right]} \tag{6}$$

- $\gamma_{p,d}$ is the stabilizing factor required for the optimal solution to satisfy the second order optimality condition
- A suitable value of $\gamma_{p,d}$ is selected for each phoneme p and segment d for each keyword such that the solution eq. (6) is the optimal solution

Modified Objective Function:

$$\hat{\lambda}^{(w)}_{\substack{p,d \\ p\in\mathcal{P}, d\in\{1,2,..D\}}} = \arg\max_{\lambda^{(w)}_{p,d}} \log \frac{\left(\prod_{K_w=1}^{\hat{K}_w} \prod_{p\in\mathcal{P}} \prod_{d=1}^{D} (\lambda^{(w)}_{p,d})^{(n^{(K_w)}_{p,d} + \hat{K}_w \gamma_{p,d})} \exp\left(-\lambda^{(w)}_{p,d} \frac{T^{(K_w)}}{D}\right)\right)^{\frac{1}{\hat{K}_w}}}{\left(\prod_{y\in w_c} \left(\prod_{K_y=1}^{\hat{K}_y} \prod_{p\in\mathcal{P}} \prod_{d=1}^{D} (\lambda^{(w)}_{p,d})^{n^{(K_y)}_{p,d}} \exp\left(-\lambda^{(w)}_{p,d} \frac{T^{(K_y)}}{D}\right)\right)^{\frac{1}{\hat{K}_y}}\right)^{\frac{1}{|w_c|}}} \tag{5}$$

$$\hat{\lambda}^{(w)}_{p,d} = \frac{\frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} n^{(K_w)}_{p,d} - \frac{1}{|w_c|} \sum_{y\in w_c} \left[\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} n^{(K_y)}_{p,d}\right] + \gamma_{p,d}}{\frac{1}{D}\left[\frac{1}{\hat{K}_w} \sum_{K_w=1}^{\hat{K}_w} T^{(K_w)} - \frac{1}{|w_c|} \sum y \in w_c (\frac{1}{\hat{K}_y} \sum_{K_y=1}^{\hat{K}_y} T^{(K_y)})\right]} \tag{6}$$

- $\gamma_{p,d}$ is the stabilizing factor required for the optimal solution to satisfy the second order optimality condition
- A suitable value of $\gamma_{p,d}$ is selected for each phoneme p and segment d for each keyword such that the solution eq. (6) is the optimal solution
- The stabilizing factor $\gamma_{p,d}$ can be interpreted as a boost in the number of phonetic event count $n^{(K_w)}_{p,d}$ by an extra $\gamma_{p,d}$ number of phonetic events

PPM-DPPM Detector Combination

- **DPPM suppresses false alarms** very well, however, **DPPM also misses some true keyword** locations.

PPM-DPPM Detector Combination

- **DPPM suppresses false alarms** very well, however, **DPPM also misses some true keyword** locations.
- We propose a **combination of the detector functions** obtained from PPM and DPPM ($d_w^{(PPM)}(t)$ and $d_w^{(DPPM)}(t)$ respectively) for keyword w to utilize the merits of PPM & DPPM

PPM-DPPM Detector Combination

- **DPPM suppresses false alarms** very well, however, **DPPM also misses some true keyword** locations.
- We propose a **combination of the detector functions** obtained from PPM and DPPM ($d_w^{(PPM)}(t)$ and $d_w^{(DPPM)}(t)$ respectively) for keyword w to utilize the merits of PPM & DPPM
- We obtain $d_w^{(PPM-DPPM)}$ as :

$$d_w^{(PPM-DPPM)}(t) = \begin{cases} d_w^{(PPM)}(t) \text{ for } d_w^{(DPPM)}(t) \geq \alpha_w \\ d_w^{(DPPM)}(t) \text{ for } d_w^{(DPPM)}(t) < \alpha_w \end{cases} \tag{7}$$

PPM-DPPM Detector Combination

- **DPPM suppresses false alarms** very well, however, **DPPM also misses some true keyword** locations.
- We propose a **combination of the detector functions** obtained from PPM and DPPM ($d_w^{(PPM)}(t)$ and $d_w^{(DPPM)}(t)$ respectively) for keyword w to utilize the merits of PPM & DPPM
- We obtain $d_w^{(PPM-DPPM)}$ as :

$$d_w^{(PPM-DPPM)}(t) = \begin{cases} d_w^{(PPM)}(t) \text{ for } d_w^{(DPPM)}(t) \geq \alpha_w \\ d_w^{(DPPM)}(t) \text{ for } d_w^{(DPPM)}(t) < \alpha_w \end{cases} \quad (7)$$

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

Experimental Setup

- 14 keywords chosen from TIMIT (**dark, suit, greasy, wash, water, year, carry, oily, always, about, through, enough, every, children**) and the Boston University Radio Speech corpora (**boston, city, committee, government, hundred, massachusetts, official, percent, president, program, public, thousand, year, yesterday**)

Experimental Setup

- 14 keywords chosen from TIMIT (**dark, suit, greasy, wash, water, year, carry, oily, always, about, through, enough, every, children**) and the Boston University Radio Speech corpora (**boston, city, committee, government, hundred, massachusetts, official, percent, president, program, public, thousand, year, yesterday**)
- **TIMIT:**
  1. 4620 sentences in the TIMIT training set for training PPM as well as DPPM.
  2. 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 TIMIT recipe ($50 \times 10 = 500$ sentences). Half of these sentences is used for development and the remaining half is used for testing purposes.

Experimental Setup

- 14 keywords chosen from TIMIT (**dark, suit, greasy, wash, water, year, carry, oily, always, about, through, enough, every, children**) and the Boston University Radio Speech corpora (**boston, city, committee, government, hundred, massachusetts, official, percent, president, program, public, thousand, year, yesterday**)

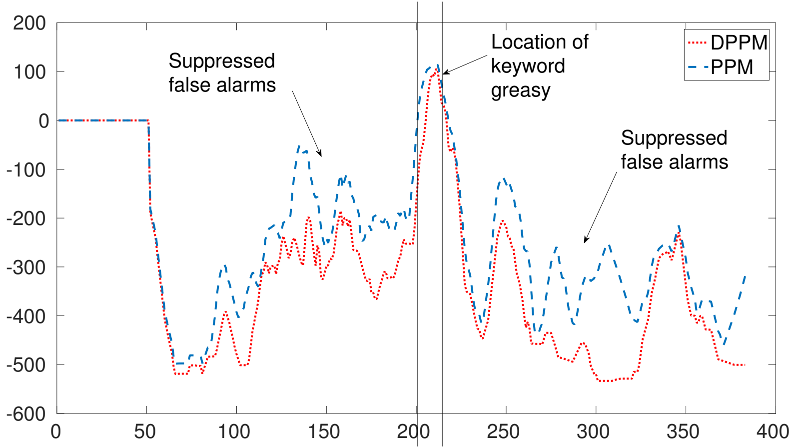
- **TIMIT:**
  1. 4620 sentences in the TIMIT training set for training PPM as well as DPPM.
  2. Test and development set of 740 sentences consisting of all the sentences of 24 speakers from TIMIT core test set ($24 \times 10 = 240$ sentences) as well as all the speakers in the development set used by Kaldi TIMIT recipe ($50 \times 10 = 500$ sentences). Half of these sentences is used for development and the remaining half is used for testing purposes.

- **BURS:**
  1. All sentences spoken by the speakers F1A, F2B, M1B, M2B, M3B for training
  2. The development set and the test set consist of the sentences spoken by the speakers F3A and M4B respectively

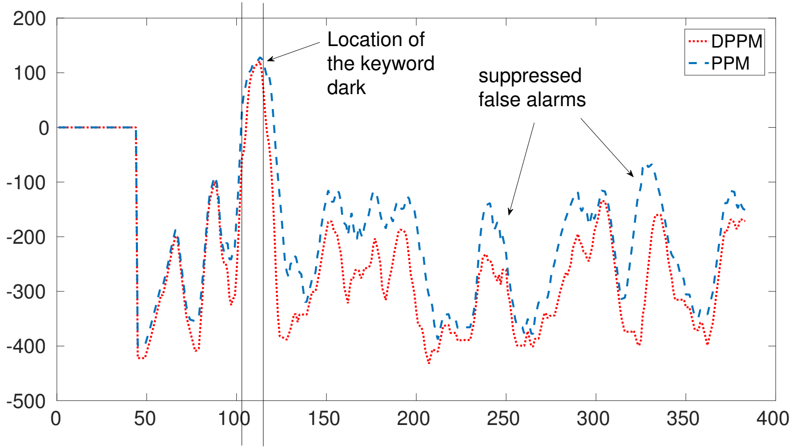
Experimental Setup

- 14 keywords chosen from TIMIT (**dark, suit, greasy, wash, water, year, carry, oily, always, about, through, enough, every, children**) and the Boston University Radio Speech corpora (**boston, city, committee, government, hundred, massachusetts, official, percent, president, program, public, thousand, year, yesterday**)
- **TIMIT:**
  1. 4620 sentences in the TIMIT training set for training PPM as well as DPPM.
  2. 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 TIMIT recipe ($50 \times 10 = 500$ sentences). Half of these sentences is used for development and the remaining half is used for testing purposes.
- **BURS:**
  1. All sentences spoken by the speakers F1A, F2B, M1B, M2B, M3B for training
  2. The development set and the test set consist of the sentences spoken by the speakers F3A and M4B respectively
- **Performance Measures:**
  1. $AROC = \frac{100 \times A}{f}$, where A is the area under the ROC curve upto a false alarm rate of f
  2. Figure of Merit (FOM) score - average of detection probabilities at $1, 2, \ldots 10$ false alarms/keyword/hour.

DPPM Supresses False Alarms



DPPM Supresses False Alarms



Comparison of PPM, DPPM and PPM-DPPM

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

Section 4

Initialization

- **Initialize a PPM with parameters** $\theta_w^{(K_{start})}$ learnt from K_{start} training samples of a keyword

Initialization

- **Initialize a PPM with parameters** $\theta_w^{(K_{start})}$ learnt from K_{start} training samples of a keyword
- Estimtae an initial **keyword detection threshold** $\gamma(K_{start})$

Initialization

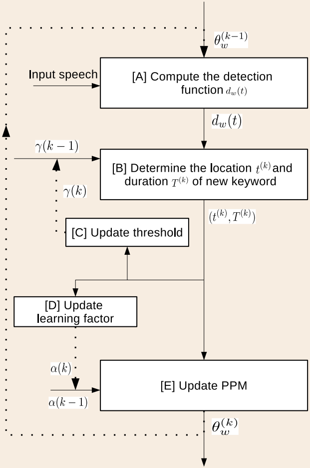
- **Initialize a PPM with parameters** $\theta_w^{(K_{start})}$ learnt from K_{start} training samples of a keyword
- Estimtae an initial **keyword detection threshold** $\gamma(K_{start})$
- Initial **learning factor** $\alpha(K_{start})$ is taken to be 1, assigning full confidence on the K_{start} initial samples

Initialization

- **Initialize a PPM with parameters** $\theta_w^{(K_{start})}$ learnt from K_{start} training samples of a keyword
- Estimtae an initial **keyword detection threshold** $\gamma(K_{start})$
- Initial **learning factor** $\alpha(K_{start})$ is taken to be 1, assigning full confidence on the K_{start} initial samples
- $K_{start} = 10$

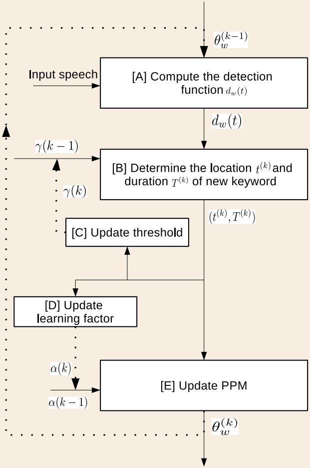
New Location and Duration Hypothesis

- $\gamma(k-1)$ be the determined threshold after the $(k-1)^{th}$ keyword detection



New Location and Duration Hypothesis

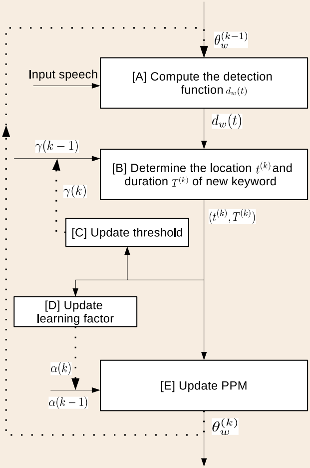
- $\gamma(k-1)$ be the determined threshold after the $(k-1)^{th}$ keyword detection
- k^{th} sample is detected in the region $[\tau_1^{(k)}, \tau_2^{(k)}]$ where $d_w(t) > \gamma(k-1)$



New Location and Duration Hypothesis

- $\gamma(k-1)$ be the determined threshold after the $(k-1)^{th}$ keyword detection
- k^{th} sample is detected in the region $[\tau_1^{(k)}, \tau_2^{(k)}]$ where $d_w(t) > \gamma(k-1)$
- End location $t^{(k)}$ of the k^{th} keyword is determined as

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



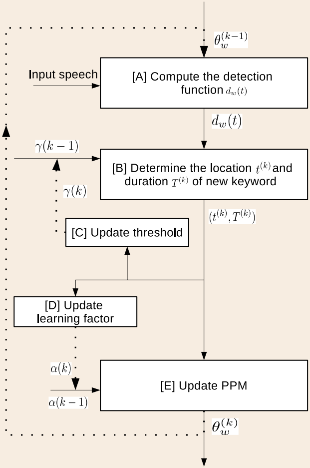
New Location and Duration Hypothesis

- $\gamma(k-1)$ be the determined threshold after the $(k-1)^{th}$ keyword detection
- k^{th} sample is detected in the region $[\tau_1^{(k)}, \tau_2^{(k)}]$ where $d_w(t) > \gamma(k-1)$
- End location $t^{(k)}$ of the k^{th} keyword is determined as

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

- Also, let the

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



New Location and Duration Hypothesis

- $\gamma(k-1)$ be the determined threshold after the $(k-1)^{th}$ keyword detection
- k^{th} sample is detected in the region $[\tau_1^{(k)}, \tau_2^{(k)}]$ where $d_w(t) > \gamma(k-1)$
- End location $t^{(k)}$ of the k^{th} keyword is determined as

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

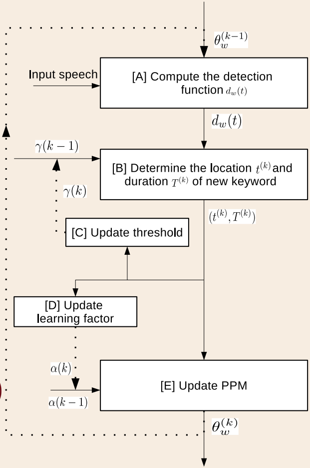
- Also, let the

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

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

$$T^{(k)} = \arg \max_{n \in \{-1,0,1,2\}} P\left(O_{\mu_w + n\sigma_w}(t^{(k)}) \,\middle|\, \mu_w + n\sigma_w, \theta_w^{(k-1)}\right) \times \beta(\mu_w + n\sigma_w | w) \tag{8}$$

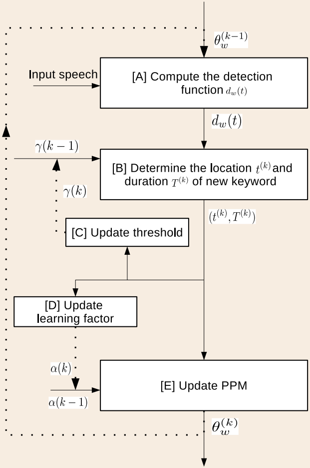
where $P(O_T(t))$ is the event/observation at time t for a duration T.



Updating $\gamma(k)$ After k^{th} Detection

- After k^{th} keyword detection, obtain the set

$$M^{(k)}(w) = \{\beta_{\hat{k}} | 1 \leq \hat{k} \leq k\} \tag{9}$$



Updating $\gamma(k)$ After k^{th} Detection

- After k^{th} keyword detection, obtain the set

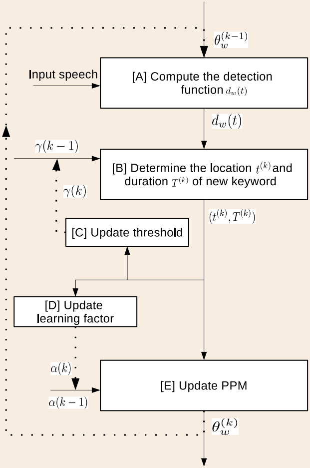
$$M^{(k)}(w) = \{\beta_{\hat{k}} | 1 \leq \hat{k} \leq k\} \tag{9}$$

- Update $\gamma(k)$ after k^{th} keyword detection

$$\gamma(k) = \begin{cases} 0.1 \times median(M^{(k)}(w)) & \text{for } k = K_{start} \\ 0.5 \times median(M^{(k)}(w)) & \text{for } k > K_{start} \end{cases} \tag{10}$$

$$\theta_w^{(k)} = \{\lambda_{p,d}^{(k)} = (\alpha(k))\lambda_{p,d}^{(k-1)} + (1-\alpha(k))\hat{\lambda}_{p,d} \;|\; p \in \mathcal{P}, 1 \leq d \leq D\} \tag{11}$$

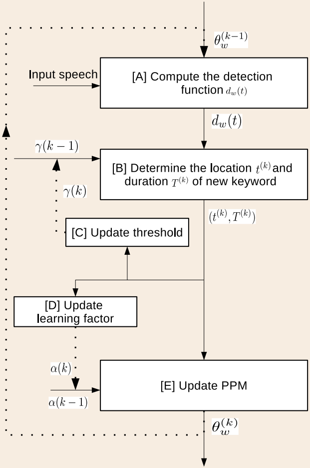
$$\alpha(k) = \frac{\sum_{\hat{k}=1}^{k-1} T^{(\hat{k})}}{\sum_{\hat{k}=1}^{k} T^{(\hat{k})}}. \tag{12}$$



Updating the Model and Learning Factor $\alpha(k)$

- Parameter set **before** k^{th} **detection**

$$\theta_w^{(k-1)} = \left\{ \lambda_{p,d}^{(k-1)} \middle| p \in \mathcal{P}, 1 \leq d \leq D \right\} \qquad (13)$$



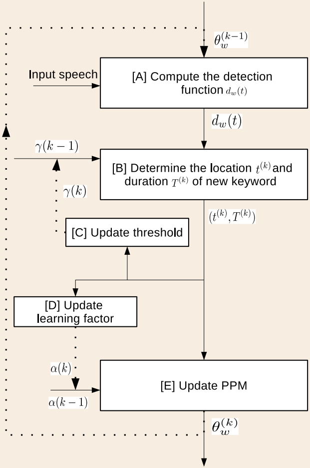
Updating the Model and Learning Factor $\alpha(k)$

- Parameter set **before** k^{th} **detection**

$$\theta_w^{(k-1)} = \left\{ \lambda_{p,d}^{(k-1)} \middle| p \in \mathcal{P}, 1 \leq d \leq D \right\} \tag{13}$$

- Obtain the **parameters of the** k^{th} **sample**

$$\hat{\theta}_w = \left\{ \hat{\lambda}_{p,d} = \frac{n_{p,d} D}{T^{(k)}} \middle| p \in \mathcal{P}, 1 \leq d \leq D\} \right\} \tag{14}$$



Updating the Model and Learning Factor $\alpha(k)$

- Parameter set **before** k^{th} **detection**

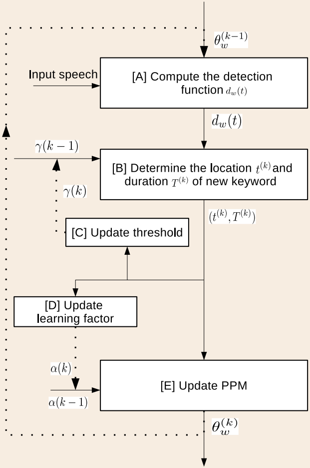
$$\theta_w^{(k-1)} = \left\{\lambda_{p,d}^{(k-1)} \middle| p \in \mathcal{P}, 1 \le d \le D\right\} \qquad (13)$$

- Obtain the **parameters of the** k^{th} **sample**

$$\hat{\theta}_w = \left\{\hat{\lambda}_{p,d} = \frac{n_{p,d}D}{T^{(k)}} \middle| p \in \mathcal{P}, 1 \le d \le D\}\right\} \qquad (14)$$

- Update the **model** as

$$\theta_w^{(k)} = \{\lambda_{p,d}^{(k)} = (\alpha(k))\lambda_{p,d}^{(k-1)} + (1-\alpha(k))\hat{\lambda}_{p,d} \; | \; p \in \mathcal{P}, 1 \le d \le D\} \qquad (15)$$



Updating the Model and Learning Factor $\alpha(k)$

- Parameter set **before** k^{th} **detection**

$$\theta_w^{(k-1)} = \left\{\lambda_{p,d}^{(k-1)} \middle| p \in \mathcal{P}, 1 \leq d \leq D\right\} \quad (13)$$

- Obtain the **parameters of the** k^{th} **sample**

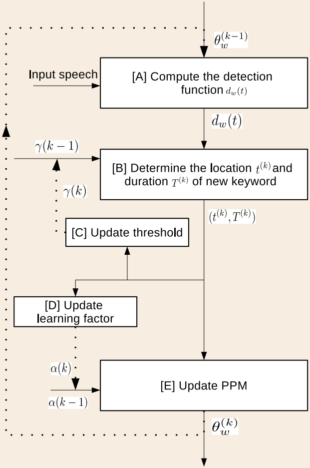
$$\hat{\theta}_w = \left\{\hat{\lambda}_{p,d} = \frac{n_{p,d}D}{T^{(k)}} \middle| p \in \mathcal{P}, 1 \leq d \leq D\}\right\} \quad (14)$$

- Update the **model** as

$$\theta_w^{(k)} = \{\lambda_{p,d}^{(k)} = (\alpha(k))\lambda_{p,d}^{(k-1)} + (1-\alpha(k))\hat{\lambda}_{p,d} \\ | \, p \in \mathcal{P}, 1 \leq d \leq D\} \quad (15)$$

- Update **learning factor** $\alpha(k)$ as

$$\alpha(k) = \frac{\sum_{\hat{k}=1}^{k-1} T^{(\hat{k})}}{\sum_{\hat{k}=1}^{k} T^{(\hat{k})}}. \quad (16)$$



Experimental Setup

- Use eight keywords obtained from the TIMIT [?] SA1 and SA2 sentences, namely **greasy, water, dark, wash, carry, oily, suit, year**

Experimental Setup

- Use eight keywords obtained from the TIMIT [?] SA1 and SA2 sentences, namely **greasy, water, dark, wash, carry, oily, suit, year**
- TIMIT training set consisting of 4620 sentences is used for training as well as the **online learning corpus**

Experimental Setup

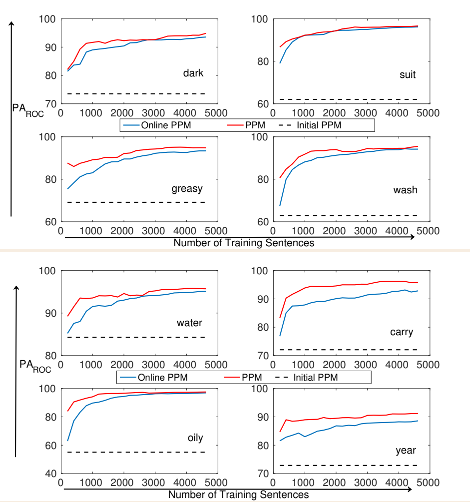
- Use eight keywords obtained from the TIMIT [?] SA1 and SA2 sentences, namely **greasy, water, dark, wash, carry, oily, suit, year**
- TIMIT training set consisting of 4620 sentences is used for training as well as the **online learning corpus**
- Test set has 740 sentences comprising of all the sentences 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)

Experimental Setup

- Use eight keywords obtained from the TIMIT [?] SA1 and SA2 sentences, namely **greasy, water, dark, wash, carry, oily, suit, year**
- TIMIT training set consisting of 4620 sentences is used for training as well as the **online learning corpus**
- Test set has 740 sentences comprising 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)
- The performance measure is the percentage area under the ROC curve given by $PA_{ROC} = \frac{100 \times A}{f}$ where A is the are under the ROC curve upto a false alarm rate of f, then

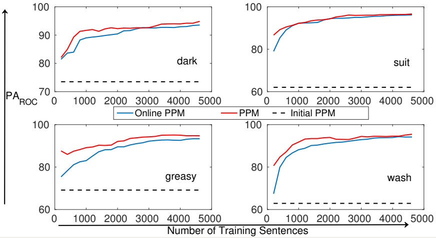
Comparison of PPM and Online PPM (1/2)

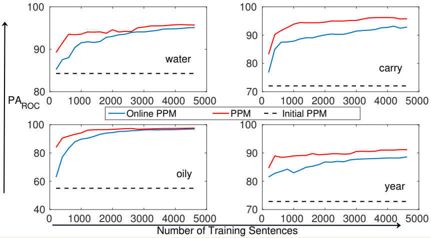
- **PPM** is trained with ground truth occurrences of the keyword



Comparison of PPM and Online PPM (1/2)

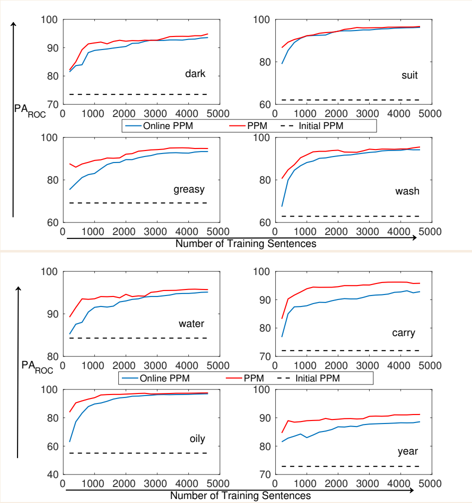
- **PPM** is trained with ground truth occurrences of the keyword
- **Online PPM** is trained using our algorithm





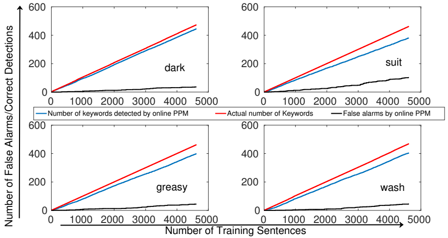
Comparison of PPM and Online PPM (1/2)

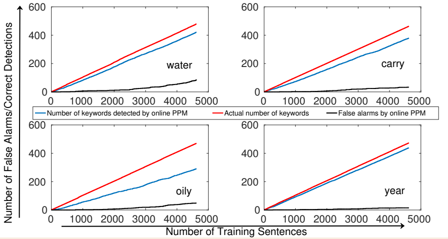
- **PPM** is trained with ground truth occurrences of the keyword
- **Online PPM** is trained using our algorithm
- Even with 1% of the training samples, **online PPM** eventually catches up with PPM trained with ground truth training samples. Average final drop in PA_{ROC} performance is 2%



Comparison of PPM and Online PPM (2/2)

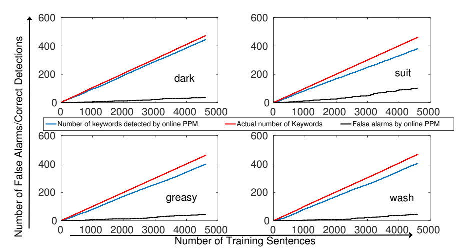
- Online PPM correct detections and false alarm counts wrt. number of training sentences from the **online learning corpus**

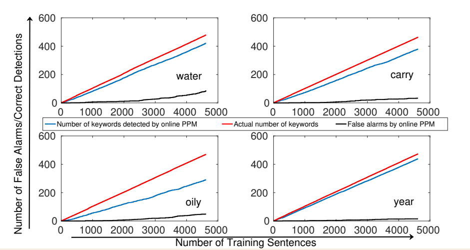




Comparison of PPM and Online PPM (2/2)

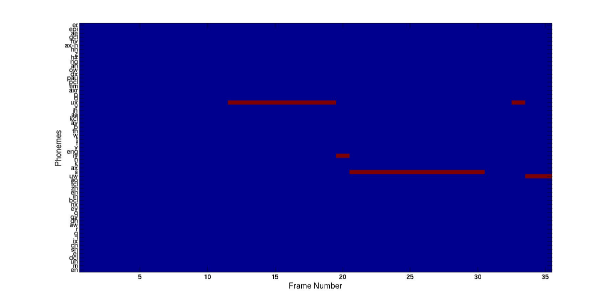
- Online PPM correct detections and false alarm counts wrt. number of training sentences from the **online learning corpus**
- **Conclusion:** Provided enough keywords occur in test utterances, our algorithm is capable of training a PPM as good as a PPM trained with a large training set





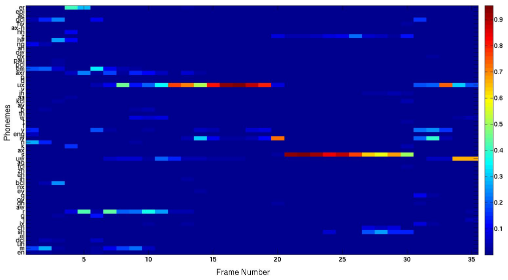
Section 5

Motivation



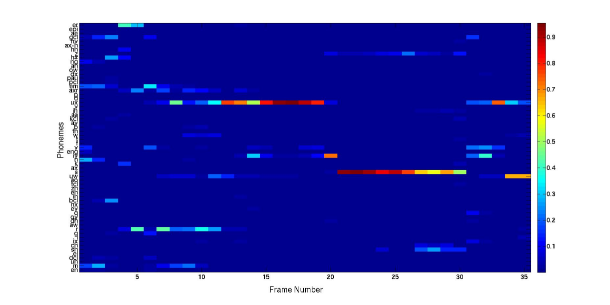
- Aim at using the excess information in the posteriorgram for better keyword spotting

Motivation



- Aim at using the excess information in the posteriorgram for better keyword spotting
- **2-D matched filters** can be used to detect keyword locations which matches with keyword locations

Motivation



- Aim at using the excess information in the posteriorgram for better keyword spotting
- **2-D matched filters** can be used to detect keyword locations which matches with keyword locations
- Infinitely many possible **posteriorgram filters**
  1. Matched Filter
  2. Level Discriminative Optimal Filter

Training

- From N training examples of keyword w obtain a set of posteriorgrams $\mathcal{X}^{(w)} = \{X_i^{(w)} | i = 1, 2, \ldots N\}$

Training

- From N training examples of keyword w obtain a set of posteriorgrams $\mathcal{X}^{(w)} = \{X_i^{(w)} | i = 1, 2, \ldots N\}$
- The posteriogram $X_i^{(w)}$ has a dimension $p \times K_i^{(w)}$ ($p =$ number of phonemes [typically 48 or 61] and $K^{(w)} =$ number of frames)

Training

- From N training examples of keyword w obtain a set of posteriorgrams $\mathcal{X}^{(w)} = \{X_i^{(w)} | i = 1, 2, \ldots N\}$
- The posteriogram $X_i^{(w)}$ has a dimension $p \times K_i^{(w)}$ ($p =$ number of phonemes [typically 48 or 61] and $K^{(w)} =$ number of frames)
- Normalize the posteriorgrams to a common dimension $p \times \hat{K}^{(w)}$ where $\hat{K}^{(w)} \geq max\{K_1^{(w)}, K_2^{(w)}, \ldots K_N^{(w)}\}$ and hence obtain a normalized set of posteriorgrams $\mathcal{X}_{norm}^{(w)} = \{\hat{X}_i^{(w)} | i = 1, 2, \ldots N\}$.

Training

- From N training examples of keyword w obtain a set of posteriorgrams $\mathcal{X}^{(w)} = \{X_i^{(w)} | i = 1, 2, \dots N\}$
- The posteriogram $X_i^{(w)}$ has a dimension $p \times K_i^{(w)}$ ($p =$ number of phonemes [typically 48 or 61] and $K^{(w)} =$ number of frames)
- 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 $\mathcal{X}_{norm}^{(w)} = \{\hat{X}_i^{(w)} | i = 1, 2, \dots N\}$.
- Average matched Filter $M^{(w)}$ for w is obtained as

$$M^{(w)} = \frac{1}{|\mathcal{X}_{norm}^{(w)}|} \sum_{X \in \mathcal{X}_{norm}^{(w)}} X \tag{17}$$

Training (1/2)

- Maximize 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.

Training (1/2)

- Maximize 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.
- Objective function to train a $p \times \hat{K}$ dimension filter $M^{(w)}$

$$\begin{aligned}\mathcal{O}(M^{(w)}) &= \left[\frac{1}{|L^{(w)}|} \sum_{w_L \in L^{(w)}} \left(\sum_{x=1}^{p} \sum_{y=1}^{\hat{K}} X^{(w_L)}_{[p \times \hat{K}]}[x,y] M^{(w)}[x,y] - V \right)^2 \right. \\ &+ \left. \frac{1}{|L_c^{(w)}|} \sum_{\hat{w}_L \in L_c^{(w)}} \left(\sum_{x=1}^{p} \sum_{y=1}^{\hat{K}} X^{(\hat{w}_L)}_{[p \times \hat{K}]}[x,y] M^{(w)}[x,y] \right)^2 + \sum_x \sum_y (M^{(w)}[x,y])^2 \right]\end{aligned} \tag{18}$$

Training (1/2)

- Maximize 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.
- Objective function to train a $p \times \hat{K}$ dimension filter $M^{(w)}$

$$\begin{aligned}\mathcal{O}(M^{(w)}) \quad &= \quad \left[\frac{1}{|L^{(w)}|} \sum_{w_L \in L^{(w)}} \left(\sum_{x=1}^{p} \sum_{y=1}^{\hat{K}} X_{[p \times \hat{K}]}^{(w_L)}[x,y] M^{(w)}[x,y] - V \right)^2 \right. \\ &+ \quad \left. \frac{1}{|L_c^{(w)}|} \sum_{\hat{w}_L \in L_c^{(w)}} \left(\sum_{x=1}^{p} \sum_{y=1}^{\hat{K}} X_{[p \times \hat{K}]}^{(\hat{w}_L)}[x,y] M^{(w)}[x,y] \right)^2 + \sum_x \sum_y (M^{(w)}[x,y])^2 \right]\end{aligned} \tag{18}$$

- Convex Optimization Problem :

$$\hat{M}^{(w)} = \arg \min_{M^{(w)}} \mathcal{O}(M^{(w)}) \tag{19}$$

Training (1/2)

- Maximize 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.
- Objective function to train a $p \times \hat{K}$ dimension filter $M^{(w)}$

$$\begin{aligned}\mathcal{O}(M^{(w)}) &= \Bigg[\frac{1}{|L^{(w)}|}\sum_{w_L \in L^{(w)}}\left(\sum_{x=1}^{p}\sum_{y=1}^{\hat{K}} X^{(w_L)}_{[p\times\hat{K}]}[x,y]M^{(w)}[x,y] - V\right)^2 \\ &+ \frac{1}{|L_c^{(w)}|}\sum_{\hat{w}_L \in L_c^{(w)}}\left(\sum_{x=1}^{p}\sum_{y=1}^{\hat{K}} X^{(\hat{w}_L)}_{[p\times\hat{K}]}[x,y]M^{(w)}[x,y]\right)^2 + \sum_x\sum_y (M^{(w)}[x,y])^2\Bigg] \end{aligned} \tag{18}$$

- Convex Optimization Problem :

$$\hat{M}^{(w)} = \arg\min_{M^{(w)}} \mathcal{O}(M^{(w)}) \tag{19}$$

- $X^{w_L}_{[p\times\hat{K}]}$ is the posteriorgram of the w_L-th keyword example from a set of $L^{(w)}$ keyword examples, p is the number of phonemes. Similarly, $X^{\hat{w}_L}_{[p\times\hat{K}]}$ is the posteriorgram of the $\hat{w}_L$-th competing keyword example from the set $L_c^{(w)}$. $\hat{K}$ is the normalized filter dimension and V is the parameter which sets to a high level

Training (2/2)

- 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.

$$\sum_{a=1}^{p} \sum_{b=1}^{\hat{K}} \zeta(a, b, \alpha, \beta) M^{(w)}[a, b] = \gamma(\alpha, \beta) \text{ for } 1 \leq \alpha \leq p \text{ and } 1 \leq \beta \leq \hat{K} \tag{20}$$

Training (2/2)

- 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.

$$\sum_{a=1}^{p} \sum_{b=1}^{\hat{K}} \zeta(a, b, \alpha, \beta) M^{(w)}[a, b] = \gamma(\alpha, \beta) \text{ for } 1 \leq \alpha \leq p \text{ and } 1 \leq \beta \leq \hat{K} \quad (20)$$

-

$$\begin{aligned} \zeta(a, b, \alpha, \beta) &= \frac{1}{|L^{(w)}|} \sum_{w_L \in L^{(w)}} X^{(w_L)}_{[p \times \hat{K}]}[a, b] X^{(w_L)}_{[p \times \hat{K}]}[\alpha, \beta] \\ &+ \frac{1}{|L_c^{(w)}|} \sum_{\hat{w}_L \in L_c^{(w)}} X^{(\hat{w}_L)}_{[p \times \hat{K}]}[a, b] X^{(\hat{w}_L)}_{[p \times \hat{K}]}[\alpha, \beta] + \delta(a - \alpha, b - \beta) \end{aligned} \quad (21)$$

Training (2/2)

- 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.

$$\sum_{a=1}^{p} \sum_{b=1}^{\hat{K}} \zeta(a, b, \alpha, \beta) M^{(w)}[a, b] = \gamma(\alpha, \beta) \text{ for } 1 \leq \alpha \leq p \text{ and } 1 \leq \beta \leq \hat{K} \tag{20}$$

-

$$\begin{aligned} \zeta(a, b, \alpha, \beta) &= \frac{1}{|L^{(w)}|} \sum_{w_L \in L^{(w)}} X^{(w_L)}_{[p \times \hat{K}]}[a, b] X^{(w_L)}_{[p \times \hat{K}]}[\alpha, \beta] \\ &+ \frac{1}{|L_c^{(w)}|} \sum_{\hat{w}_L \in L_c^{(w)}} X^{(\hat{w}_L)}_{[p \times \hat{K}]}[a, b] X^{(\hat{w}_L)}_{[p \times \hat{K}]}[\alpha, \beta] + \delta(a - \alpha, b - \beta) \end{aligned} \tag{21}$$

-

$$\gamma(\alpha, \beta) = V \sum_{w_L \in L^{(w)}} X^{(w_L)}_{[p \times \hat{K}]}[\alpha, \beta] \tag{22}$$

Training (2/2)

- 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.

$$\sum_{a=1}^{p} \sum_{b=1}^{\hat{K}} \zeta(a, b, \alpha, \beta) M^{(w)}[a, b] = \gamma(\alpha, \beta) \text{ for } 1 \leq \alpha \leq p \text{ and } 1 \leq \beta \leq \hat{K} \quad (20)$$

-

$$\begin{aligned} \zeta(a, b, \alpha, \beta) &= \frac{1}{|L^{(w)}|} \sum_{w_L \in L^{(w)}} X_{[p \times \hat{K}]}^{(w_L)}[a, b] X_{[p \times \hat{K}]}^{(w_L)}[\alpha, \beta] \\ &+ \frac{1}{|L_c^{(w)}|} \sum_{\hat{w}_L \in L_c^{(w)}} X_{[p \times \hat{K}]}^{(\hat{w}_L)}[a, b] X_{[p \times \hat{K}]}^{(\hat{w}_L)}[\alpha, \beta] + \delta(a - \alpha, b - \beta) \end{aligned} \quad (21)$$

-

$$\gamma(\alpha, \beta) = V \sum_{w_L \in L^{(w)}} X_{[p \times \hat{K}]}^{(w_L)}[\alpha, \beta] \quad (22)$$

-

$$\delta(a - \alpha, b - \beta) = \begin{cases} 1, & \text{if } a = \alpha \text{ and } b = \beta \\ 0 & \text{else} \end{cases} \quad (23)$$

Obtaining $L_c^{(w)}$

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

$$\begin{aligned} \hat{M}^{(w)}_{no-D} &= \arg\min_{M^{(w)}} \left[\frac{1}{|L^{(w)}|} \sum_{w_L \in L^{(w)}} \left(\sum_{x=1}^{p} \sum_{y=1}^{\hat{K}} X^{(w_L)}_{[p \times \hat{K}]}[x,y] M^{(w)}[x,y] - V \right)^2 \right. \\ &+ \left. \sum_x \sum_y (M^{(w)}[x,y])^2 \right] \end{aligned} \quad (24)$$

Obtaining $L_c^{(w)}$

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

$$\hat{M}_{no-D}^{(w)} = \arg\min_{M^{(w)}} \left[\frac{1}{|L^{(w)}|} \sum_{w_L \in L^{(w)}} \left(\sum_{x=1}^{p} \sum_{y=1}^{\hat{K}} X_{[p \times \hat{K}]}^{(w_L)}[x,y] M^{(w)}[x,y] - V \right)^2 + \sum_x \sum_y (M^{(w)}[x,y])^2 \right] \quad (24)$$

- $M_{no-D}^{(w)}$ is used to search for the keywords w in a **development set** dev to obtain **a set of keywords and false alarm locations**

Obtaining $L_c^{(w)}$

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

$$\hat{M}^{(w)}_{no-D} = \arg\min_{M^{(w)}} \left[\frac{1}{|L^{(w)}|} \sum_{w_L \in L^{(w)}} \left(\sum_{x=1}^{p} \sum_{y=1}^{\hat{K}} X^{(w_L)}_{[p \times \hat{K}]}[x,y] M^{(w)}[x,y] - V \right)^2 + \sum_x \sum_y (M^{(w)}[x,y])^2 \right] \quad (24)$$

- $M^{(w)}_{no-D}$ is used to search for the keywords w in a **development set** dev to obtain **a set of keywords and false alarm locations**
- The threshold is varied as a percentage of the maximum value of the correct detector plots to generate **lists of competing words** $\hat{L}_c^{(w)}$

Obtaining $L_c^{(w)}$

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

$$\hat{M}^{(w)}_{no-D} = \arg\min_{M^{(w)}} \left[\frac{1}{|L^{(w)}|} \sum_{w_L \in L^{(w)}} \left(\sum_{x=1}^{p} \sum_{y=1}^{\hat{K}} X^{(w_L)}_{[p \times \hat{K}]}[x,y] M^{(w)}[x,y] - V \right)^2 + \sum_x \sum_y (M^{(w)}[x,y])^2 \right] \tag{24}$$

- $M^{(w)}_{no-D}$ is used to search for the keywords w in a **development set** dev to obtain **a set of keywords and false alarm locations**
- The threshold is varied as a percentage of the maximum value of the correct detector plots to generate **lists of competing words** $\hat{L}_c^{(w)}$
- The list $\hat{L}_c^{(w)}$ giving **the best performance on** dev is defined to be the list of competing words $L_c^{(w)}$

Decoding

- Decoding using any posteriorgram filter is performed in the same manner

Decoding

- Decoding using any posteriorgram filter is performed in the same manner
- 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 $\mathcal{L} = \{\mathcal{N}(\mu + n\sigma|\mu,\sigma) | n = -1, 0, 1, 2\}$

Decoding

- Decoding using any posteriorgram filter is performed in the same manner
- 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:

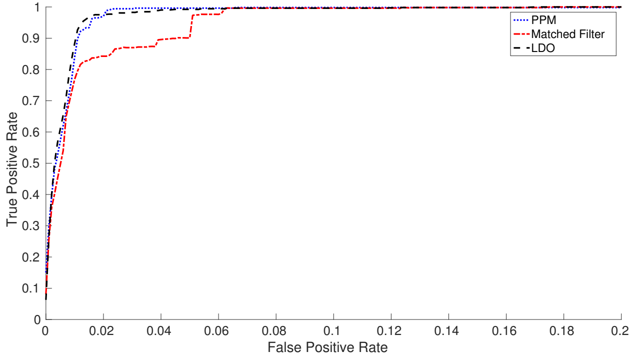
$$d_w(t) = \max_{n \in \{-1,0,1,2\}} \left(\sum_{i=1}^{p} \sum_{j=t-\mu-n\sigma+1}^{t} X_{test}[i][j] M^{(w)}_{[p\times(\mu+n\sigma)]}[i][j-t+\mu+n\sigma] \mathcal{N}(\mu+n\sigma|\mu,\sigma) \right) \tag{25}$$

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 and Results

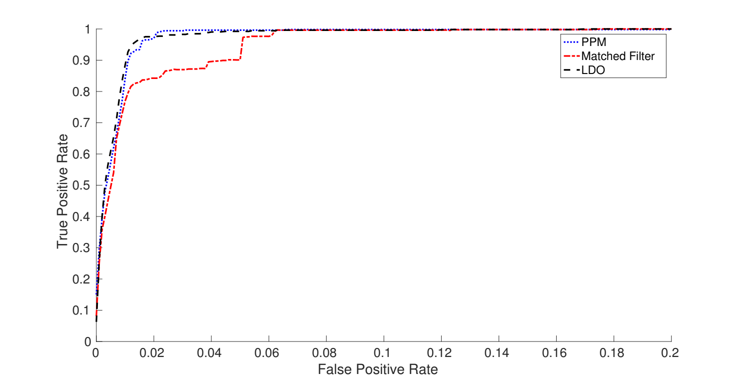
- The algorithm is tested using 14 keyword from the TIMIT database

Experimental Setup and Results



- The algorithm is tested using 14 keyword from the TIMIT database
- Figure shows the **average Receiver Operating Curve (ROC) for 14 keywords.** for PPM, LDO and matched filtering.

Experimental Setup and Results



- The algorithm is tested using 14 keyword from the TIMIT database
- Figure shows the **average Receiver Operating Curve (ROC) for 14 keywords.** for PPM, LDO and matched filtering.
-

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

Section 6

The Problem Statement

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

Baseline System[1]

- **Segmental Dynamic Time Warping** (sDTW) based KWS using **Gaussian Posteriorgram** features in **a completely unsupervised paradigm** of KWS exactly in the same setting as described in the posed problem statement
- Local sDTW Conditions:
  1. **Adjustment window condition:** The DTW path 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., $|i_x - i_y| \leq R$
  2. **Step length of start co-ordinates:** A R frame step jump based DTW computation.

[1] Zhang, Yaodong, and James R. Glass. "Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams."

IEEE Workshop on Automatic Speech Recognition & Understanding, 2009.

The Proposed Adaptive Matched Filtering Algorithm

- **Initialization:** Initial matched filter $M_{init}^{(w)}$ for w obtained with 5 samples of w and Gaussian posteriorgram (GP)

The Proposed Adaptive Matched Filtering Algorithm

SPIRE LAB

- **Initialization:** Initial matched filter $M_{init}^{(w)}$ for w obtained with 5 samples of w and Gaussian posteriorgram (GP)
- Consider matched filter $M_k^{(w)}$ after detection of k keywords and $peak(k)$ be the set of maximum values of the detector output at the keyword locations

The Proposed Adaptive Matched Filtering Algorithm

- **Initialization:** Initial matched filter $M_{init}^{(w)}$ for w obtained with 5 samples of w and Gaussian posteriorgram (GP)
- Consider matched filter $M_k^{(w)}$ after detection of k keywords and $peak(k)$ be the set of maximum values of the detector output at the keyword locations
- **First level verification:** Decode through test sentences using the filter $M_k^{(w)}$ and use a very low threshold $0.2 \times \max(peak(k))$ to get a preliminary location of keyword. If a probable location of the keyword is obtained, the end and duration of the keyword is hypothesized as for **Online Learning of PPM**

The Proposed Adaptive Matched Filtering Algorithm

- **Initialization:** Initial matched filter $M_{init}^{(w)}$ for w obtained with 5 samples of w and Gaussian posteriorgram (GP)
- Consider matched filter $M_k^{(w)}$ after detection of k keywords and $peak(k)$ be the set of maximum values of the detector output at the keyword locations
- **First level verification:** Decode through test sentences using the filter $M_k^{(w)}$ and use a very low threshold $0.2 \times \max(peak(k))$ to get a preliminary location of keyword. If a probable location of the keyword is obtained, the end and duration of the keyword is hypothesized as for **Online Learning of PPM**
- **Second level verification:** 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.

The Proposed Adaptive Matched Filtering Algorithm

- **Initialization:** Initial matched filter $M_{init}^{(w)}$ for w obtained with 5 samples of w and Gaussian posteriorgram (GP)
- Consider matched filter $M_k^{(w)}$ after detection of k keywords and $peak(k)$ be the set of maximum values of the detector output at the keyword locations
- **First level verification:** Decode through test sentences using the filter $M_k^{(w)}$ and use a very low threshold $0.2 \times \max(peak(k))$ to get a preliminary location of keyword. If a probable location of the keyword is obtained, the end and duration of the keyword is hypothesized as for **Online Learning of PPM**
- **Second level verification:** 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.
- GP_{new} be new keyword GP feature, then matched filter $M_{k+1}^{(w)}$ is obtained as

$$M_{k+1}^{(w)} = \frac{k}{k+1} M_k^{(w)} + \frac{1}{k+1} GP_{new} \tag{26}$$

Experimental Setup

- 8 keywords are chosen from the TIMIT database SA1 and SA2 sentences, namely **greasy, water, dark, wash, carry, oily, suit, year**

Experimental Setup

- 8 keywords are chosen from the TIMIT database SA1 and SA2 sentences, namely **greasy, water, dark, wash, carry, oily, suit, year**
- Out of 4620 training sentences, 5 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

Experimental Setup

- 8 keywords are chosen from the TIMIT database SA1 and SA2 sentences, namely **greasy, water, dark, wash, carry, oily, suit, year**
- Out of 4620 training sentences, 5 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
- Performance 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**

Experimental Setup

- 8 keywords are chosen from the TIMIT database SA1 and SA2 sentences, namely **greasy, water, dark, wash, carry, oily, suit, year**
- Out of 4620 training sentences, 5 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
- Performance 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**
- The GMM for generating the Gaussian posteriorgram features is trained using 462 random sentences from the TIMIT train corpus with 5×61 mixture components.

Results

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**

- The average $P@N$ performance for the 8 keywords improved from 0.5356 to 0.7545

Results

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**

- The average $P@N$ performance for the 8 keywords improved from 0.5356 to 0.7545
- 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

Results

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**

- The average $P@N$ performance for the 8 keywords improved from 0.5356 to 0.7545
- 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
- However, the $P@N$ performance of MF_{adapt} improves over the baseline GP_{base} system to 0.7545

Section 7

Future Work

- **Multi-modal KWS:** Fusion of auditory, visual and articulatory data for KWS

Future Work

- **Multi-modal KWS:** Fusion of auditory, visual and articulatory data for KWS
- **Robust KWS:** KWS systems robust to noisy environment and other external disturbances

Future Work

SPIRE LAB

- **Multi-modal KWS:** Fusion of auditory, visual and articulatory data for KWS
- **Robust KWS:** KWS systems robust to noisy environment and other external disturbances
- **KWS System Combination:** Combination of ASR and non-ASR based KWS techniques

Thank You!