KEYWORD SEARCH BY SYMBOLIC INDEXING

by

Leda Sarı

B.S, in Electrical and Electronics Engineering, Boğaziçi University, 2013

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University
2016

KEYWORD SEARCH BY SYMBOLIC INDEXING

APPROVED BY:

Assoc. Prof. Murat Saraçlar . . . .
(Thesis Supervisor)

Prof. Levent Arslan . . . .

Assoc. Prof. Taylan Cemgil . . . .

DATE OF APPROVAL: 11.01.2016

ACKNOWLEDGEMENTS

I am very grateful to my advisor Assoc. Prof. Murat Saraçlar, for his invaluable guidance, great support and close interest in this work. Without discussions with him and his help at any time, this work could not be accomplished.

I would like to thank to Prof. Levent Arslan, from whom I took my first speech processing course, and Assoc. Prof. Taylan Cemgil for their participation to my thesis committee and for their comments on this work.

I am grateful to Prof. Ayşın Ertüzün for being my undergraduate project advisor, giving the first feelings of doing research and for her support during my studies.

I owe special thanks to Prof. Bülent Sankur for providing us the BUSIM Lab as well as for sharing his intellectual point of view.

I would like to thank Speech Processing Group members especially to Erinç Dikici for being one of my lab neighbors and always kindly answering my questions about practical and technical issues as well as for his useful comments on my paperwork; to Batuhan Gündoğdu and Gözde Çetinkaya for their useful discussions and collaboration on the keyword search project. I would like to thank my other lab neighbor Burcu Tepekule for sharing the early mornings with me and also to current and previous colleagues at BUSIM including but not limited to Mehmet Yamaç, Sinem Aslan, Sezer Ulukaya and WCL members Öykü Tuncel, Can Altay, Alican Gök, Ceren Sevinç and İlhan Yıldırım for their friendship and support.

I would like to thank Bhuvana Ramabhadran and Abhinav Sethy for answering our questions about setups developed by the IBM Attila toolkit.

I would also like to thank my parents Berç and Jülyet Sarı, who have supported me with their endless love and understanding throughout my life.

This study uses the IARPA Babel Program base period language collection releases babel105b-v0.4 and babel202b-v1.0d, and is supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0012. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

ABSTRACT

KEYWORD SEARCH BY SYMBOLIC INDEXING

The aim of keyword search (KWS) is to locate written queries in large amount of audio data such as archived news broadcasts, audio/video lectures, recorded customer call-center data or conversational speech. State of the art KWS approaches are based on indexing automatic speech recognition (ASR) lattices. However, for languages having only a limited amount of transcribed audio, the ASR performance decreases which in turn reduces the KWS performance. Another problem with ASR based KWS systems is searching for out-of-vocabulary (OOV) keywords which are not covered by the ASR vocabulary. One common approach is expanding the keyword using a confusion model (CM) and searching for similar words along with the original. In this work, the KWS index is generated using symbolic representations of the data instead of ASR lattices. These symbols are obtained by encoding the search data posteriorgram which is generated using the deep neural network (DNN) output of the ASR system. In the experiments performed on the low resource language datasets of the IARPA Babel Program, we show that when combined with existing ASR lattice based KWS systems, the proposed system improves the KWS performance measured in terms of term weighted value (TWV), especially for OOV queries. In order to handle OOV queries, a discriminative approach for training the CM is also introduced which directly aims at maximizing the TWV for OOV queries. We explore the influence of discriminative training on both an existing ASR lattice based system and the symbolic index based system under low resource settings.

ÖZET

SİMGESEL DİZİNLEMEYLE ANAHTAR SÖZCÜK ARAMA

Anahtar sözcük arama (ASA) sisteminin amacı yazılı olarak verilen sorguların arşivlenmiş haber bültenleri, ses ya da video biçimindeki ders kayıtları, müşteri hizmetlerinin kayıt altına alınmış telefon görüşmeleri gibi sesli veriler içindeki yerlerinin saptanmasıdır. Mevcut en iyi ASA sistemleri otomatik konuşma tanıma (OKT) sistemi örülerini dizinlemeye dayanır. Fakat, yazılandırılmış konuşma verisi az olan dillerde, OKT sisteminin başarımı dolayısıyla da ASA başarımı düşer. OKT tabanlı sistemlerde diğer bir problem de OKT dağarcığında bulunmayan dağarcık-dışı (DD) sözcüklerin aranmasıdır. Genellikle kullanılan bir yöntem anahtar sözcüğü bir karışıklık modeliyle (KM) genişletip benzer kelimeleri de orijinal haliyle birlikte aramaktır. Bu çalışmada, ASA dizini verinin OKT tanıma örüsü gösterimi yerine verinin simgesel gösteriminden oluşturulmuştur. Bu simgeler OKT sisteminin derin yapay sinir ağı çıktısından oluşturulan arama verisi posteriorgramının kodlanmasıyla elde edilmiştir. IARPA Babel Programı'nın az kaynaklı dil verileri üzerinde yapılan deneylerde, önerilen sistemin OKT örüsü tabanlı mevcut bir ASA sistemiyle birleştirildiğinde terim ağırlıklı değer (TAD) ile ölçülen ASA başarımını özellikle DD sorgular için artırdığı gösterilmiştir. DD sözcüklerin aranmasında KM için doğrudan DD sorgularda TAD'yi enbüyüklemeyi hedefleyen bir ayırıcı eğitim yöntemi tanıtılmıştır. Ayırıcı eğitimin, kaynağı az olan dillerde, hem mevcut OKT tanıma örüsü hem de simgesel dizinlemeye dayalı ASA sistemlerine etkisi incelenmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . v
ÖZET . . . vi
LIST OF FIGURES . . . ix
LIST OF TABLES . . . xi
LIST OF SYMBOLS . . . xiv
LIST OF ACRONYMS/ABBREVIATIONS . . . xvi
1. INTRODUCTION . . . 1
2. KEYWORD SEARCH SYSTEMS . . . 4
2.1. LVCSR Output Based Search . . . 5
2.1.1. Automatic Speech Recognition . . . 6
2.1.2. Weighted Finite State Transducers . . . 8
2.1.3. ASR Output Representation . . . 8
2.1.4. Indexing . . . 10
2.1.5. Search . . . 14
2.2. Alternative Approaches for KWS . . . 18
2.2.1. Pattern Matching Approaches . . . 18
2.2.2. HMM Based Keyword-Filler Models . . . 20
2.2.3. Point Process Models . . . 20
2.2.4. Discriminative Approaches for KWS . . . 21
2.3. KWS Task Evaluation . . . 23
2.3.1. KWS Performance Evaluation . . . 23
2.3.2. Score Normalization . . . 26
2.3.3. System Combination . . . 28
2.3.4. BABEL Database . . . 29
3. SYMBOLIC INDEX BASED KWS . . . 32
3.1. Symbolic Index Based Search . . . 32
3.1.1. Posteriorgram Representation of the Data . . . 34
3.1.2. Symbolic Representation of the Search Data . . . 35

3.1.3. Symbolic Representation of the Query 36
3.1.4. Query Posteriorgram Generation 37
3.1.5. Confusion Model Generation 39
3.2. Experiments 40
3.2.1. Experiments on the Turkish Dataset 40
3.2.1.1. Individual System Performance 41
3.2.1.2. System Combination 49
3.2.2. Experiments on the Swahili Dataset 55
4. DISCRIMINATIVE TRAINING OF THE CONFUSION MODEL 63
4.1. Discriminative Training of the CM for LVSCR Lattice Based KWS . . 64
4.1.1. Experiments 67
4.2. Discriminative Training of the CM for Symbolic Index Based KWS . . 70
4.2.1. Experiments 72
5. CONCLUSIONS AND FUTURE DIRECTIONS 76
5.1. Conclusions 76
5.2. Future Directions 77
APPENDIX A: DERIVATION OF WEIGHT UPDATES 79
A.1. Deriving the Update Equation 79
A.2. Learning the Threshold (θ) of the Sigmoid Function 85
A.3. Simplifications to the Training Procedure 85
REFERENCES 87

LIST OF FIGURES

Figure 2.1. An example lattice. 9

Figure 2.2. CN of the lattice shown in Figure 2.1. 10

Figure 2.3. Lattices (a,c) and the corresponding pre-processed WFSTs (b,d) for two utterances in the database. 12

Figure 2.4. Factor generation for the first utterance. 12

Figure 2.5. Result of factor merging for the first utterance. 13

Figure 2.6. Index of the dataset, union of TFTs of the utterances. 13

Figure 2.7. A CM example. 16

Figure 2.8. Expansion of the query “mama”. 17

Figure 2.9. An example phonetic posteriorgram. 19

Figure 2.10. An example DET curve. 26

Figure 3.1. Overview of the posteriorgram based KWS setup. 33

Figure 3.2. Query FSA with the loop structures. 37

Figure 3.3. Query posteriorgrams according to vector and duration modeling. 38

Figure 3.4. Optional silence modeling for multi-word keywords 38

Figure 3.5. Overview of the unsupervised KWS setup. 40

Figure 3.6. Overview of the supervised KWS setup. 41

Figure 3.7. 3-state left-to-right HMM with begin (b), middle (m) and end (e) states. 43

Figure 3.8. TWV versus γ in score normalization. 48

Figure 3.9. TWV versus the CM pruning threshold. 50

Figure 4.1. Effect of α and θ on sigmoid approximation to TWV. 68

Figure 4.2. Increase in the approximated TWV during iterations for the LVCSR based setup. 71

Figure 4.3. Increase in the approximated TWV during iterations for the symbolic index based setup. 73

LIST OF TABLES

Table 2.1. Distribution of IV and OOV keywords according to query length given in number of words. 30

Table 3.1. MTWV for all, IV and OOV queries depending on the supervised (S) or unsupervised (U) setup 42

Table 3.2. MTWV of search results and the FER in confusion matrix depending on the posteriorgram type and codebook generation 44

Table 3.3. MTWV for all, IV and OOV queries depending on confusion matrix normalization 45

Table 3.4. Number of hits and STWV as the number of best paths changes . 45

Table 3.5. Change in MTWV depending on the score normalization 47

Table 3.6. Effect of using an optional silence in multiword queries on MTWV 49

Table 3.7. Performance of the combined system for the (un)supervised setup . 50

Table 3.8. Performance of the combined system depending on the posteriorgram and codebook 51

Table 3.9. Performance of the combined system depending on the CM normalization 52

Table 3.10. Performance of the combined system depending on n in n-best . . 52

Table 3.11. Performance of the combined system depending on the score normalization 53

Table 3.12. Performance of the combined system when γ normalization is applied 54

Table 3.13. Performance of the combined system depending on the word boundary representation 55

Table 3.14. MTWV for all, IV and OOV queries depending on γ in the supervised setup for Swahili 57

Table 3.15. Performance of the combined system for Swahili depending on γ when binary queries are used 58

Table 3.16. MTWV and STWV for Swahili depending the on the query representation 60

Table 3.17. Performance of the combined systems for Swahili depending on query representation 60

Table 3.18. MTWV for Swahili QbyE system depending on the rescoring method 62

Table 3.19. Effect of rescoring QbyE results on system combination for Swahili 62

Table 4.1. MTWV for the cases where no CM or a randomly initialized CM is used 69

Table 4.2. Relative increase in MTWV when the CM is updated at each step (Upd1 + Search) and updated after 2 steps (Upd2 + Search) . . . 69

Table 4.3. Individual results of discriminative CM training in the symbolic index based system for Turkish 74

Table 4.4. Combined results of discriminative CM training in the symbolic index based system for Turkish 75

LIST OF SYMBOLS

$c_{qr}(i,o)$	Count of the CM weight with (i,o) input-output labels on a path from q to r
$C(i,o)$	Number of frames labeled with o instead of the true symbol i in the confusion matrix
CM	Confusion model WFST
$d(.)$	Distance measure used in k-means
D	Total distortion in k-means clustering
$f_{g\mathrm{s}}$	Frequency of observing g along with s
g	Grapheme
H_q	Set of all hypotheses h for the query q
i	Input symbol of an arc in FST
$I(.)$	Indicator function
k	Index of the unit in a posteriorgram
K	Total number of classes in a posteriorgram
$l(\pi_{qr}(m))$	Input-output label pair of the arc $\pi_{qr}(m)$
L	Pronunciation lexicon
L^{-1}	Inverse of the pronunciation lexicon
n	Number of the shortest paths in FST n-best operation
n_{tps}	Number of trials per second of speech
N	Total number of utterances in a dataset
N_q	Total number of hits for a query q
$\mathrm{N}_{\mathrm{cor}}$	Number of correct detections
$\mathrm{N}_{\mathrm{spur}}$	Number of spurious detections
$\mathrm{N}_{\mathrm{true}}$	True number of occurrences in the corpus
N_{NT}	Number of non-target trials
o	Output symbol of an arc in FST
p_h^q	Unnormalized score of hypothesis h for query q
$\overline{p}_h^q$	Normalized score of hypothesis h for query q
p_1	Score of the highest scoring hit of a query

P_{FA}	Probability of false alarms
P_{miss}	Probability of miss
q	A single query or keyword
$\mathcal{Q}$	The set of queries or keywords
Q	Query WFST
Q'	Expanded query WFST
r	Proxy word confused with the original query q
s	HMM states
s_h^q	Log-domain score of hypothesis h for query q
S_k	Set of states associated with unit k
t	Time index in a posteriorgram
T	Total number of frames in an audio file
T_{speech}	Total amount of speech in the test data
w_{qr}	Total confusion weight from q to r
w_{gs}	Weight used to convert the state level posterior of s to graphemic one g
$\mathbf{x}^t$	Feature vector for the t-th frame of the audio
$\mathbf{y}^t(k)$	Posterior probability of class k at time t in a posteriorgram
$\mathbf{Y}$	Posteriorgram of an audio file
α	Shape parameter of the sigmoid function
α'	Scaling factor used in KST normalization
β	TWV parameter
$\gamma(q, h)$	The truth value of hypothesis h for query q
θ	Decision threshold
η_w	Step size in the CM weight update equation
π_{qr}	WFST path from q to r
$\pi_{qr}(m)$	mth arc on the path π_{qr}
$\sigma(p, \theta)$	Sigmoid function for a given θ

LIST OF ACRONYMS/ABBREVIATIONS

ASR	Automatic Speech Recognition
ATWV	Actual Term Weighted Value
CD	Context Dependent
CI	Context Independent
CM	Confusion Model
CN	Confusion Network
DET	Detection-Error Tradeoff Curve
DNN	Deep Neural Network
DTW	Dynamic Time Warping
FER	Frame Error Rate
FOM	Figure of Merit
FST	Finite State Transducer
HMM	Hidden Markov Model
GMM	Gaussian Mixture Model
G2P	Grapheme-to-Phoneme
IR	Information Retrieval
IV	In-Vocabulary
KST	Keyword Specific Thresholding
KWS	Keyword Search
LSTM	Long-Short Term Memory
LVCSR	Large Vocabulary Continuous Speech Recognition
MTWV	Maximum Term Weighted Value
OOV	Out-Of-Vocabulary
OTWV	Optimum Term Weighted Value
PPM	Point Process Model
ROC	Receiver Operating Characteristics
QbyE	Query-by-Example
SI	Speaker Independent

STD	Spoken Term Detection
STO	Sum-To-One Normalization
STWV	Supremum Term Weighted Value
TFT	Timed Factor Transducer
VQ	Vector Quantization
WFSA	Weighted Finite State Acceptor
WFST	Weighted Finite State Transducer

1. INTRODUCTION

As the recording capabilities of smart devices increase and become more affordable, the amount of spoken data such as archived news broadcasts, audio/video lectures, recorded customer call-center data or conversational speech increases rapidly. Therefore, retrieving the necessary information from large amounts of data becomes a challenging task.

Keyword search (KWS) is a speech processing application which aims at detecting and locating given written keywords, or queries, in large amount of audio data. The keywords to be searched can be a single-word query or they can include several words like a phrase and the search data are neither transcribed nor segmented. As the output of a KWS system the list of hits is returned for each keyword or query. These hits show the beginning and ending time of the keyword in the audio file and a detection score.

One of the most common approaches to KWS is initially transcribing the audio search data to words using a large vocabulary continuous speech recognition (LVCSR) system and then applying text retrieval methods to the word transcriptions. In these systems, the lattices obtained LVCSR system that include word hypothesis for the input speech are indexed and KWS is performed using this index. The index is a mapping from the words to the utterances that allows identifying the utterances in which the keyword is uttered during search operation. Although there are large number of words in the LVCSR lexicon, there can be words, in particular query terms, that are not covered by the lexicon which are called out-of-vocabulary (OOV) terms. Rarely used words of a language, foreign words or proper nouns are some possible candidates for OOV words. Especially in low resource languages for which only a limited amount of transcribed data is available to train a LVCSR system, the pronunciation lexicon is also limited. Therefore, in such languages the frequency of OOV words are higher. OOV rate is also higher for agglutinative languages like Turkish, Zulu or for endangered languages for which only a limited amount of data can be obtained.

The performance a LVCSR dependent KWS system for OOV queries is worse than the in-vocabulary (IV) ones since LVSCR system cannot output a result for the former which in turn prevents the search of the OOV keywords and returning an empty result. In order to deal with OOV problem in KWS, commonly used approaches are using sub-word units and applying query expansion techniques. Since OOV queries can be represented in terms of sub-word units, using a sub-word based LVCSR system instead of a word based one are preferred. The sub-word units can be phones, syllables or word-fragments. Hybrid systems that combines different types of sub-word systems are also used in KWS tasks [1–3]. Although sub-word systems increase the number of detections, they also lead to higher number of false alarms. The second approach for dealing with the OOV problem is expanding the query to a similar keyword and searching for both the original and the expanded versions of the query to increase the number of returned results by overcoming the possible transcription errors of the LVCSR system or converting the OOV words into IV ones for which results can be obtained from the recognition system. One way of expanding the queries is applying a confusion model (CM) to the query [4, 5]. The CM models substitutions, insertions and deletions. These two approaches can be combined as in [5, 6]. For example, a phonetic LVCSR system can include a phone CM that change some of the phones with alternative ones, insert or delete a few phones from the phonetic representation of the query.

There is also query-by-example (QbyE) spoken term detection (STD) task which aims to find the time spans in the utterances relevant to the keyword. The difference between these tasks is that queries are acoustic examples such as audio snippets or the spoken version of the query in QbyE-STD whereas the queries are in written form in KWS. Since the final goals are similar, techniques or audio representations used in one of these systems can be exploited in the other one. For instance, posteriorgram representation of the audio which is effectively used in QbyE-STD task [7,8] can be exploited in KWS framework. The posteriorgram shows the posterior probability of each phonetic class for each frame in an utterance and is obtained through a LVSCR system [8]. Thus, instead of indexing the LVCSR lattices, the information in posteriorgrams can be used to obtain a KWS index as will be shown in this thesis.

The are two main contributions of this thesis to the KWS systems. The first contribution is the development of an index based KWS system that uses symbolic representation of the data obtained from their posteriorgrams instead of word or sub-word unit based indices generated from the LVCSR lattices. We especially concentrate on KWS for conversational data in low resource languages for which we have large number of OOV keywords. The second contribution is development of a discriminative approach for training a CM for expanding OOV queries that directly maximizes the KWS performance criterion.

The rest of the thesis is organized as follows: Chapter 2 gives a brief overview of LVCSR systems and summarizes previous approaches for KWS. In Chapter 3, the symbolic index based KWS system will be introduced. This system makes use of the posteriorgram representation of the speech data and an index for the search data is generated as in LVCSR based approaches. In addition, query representation and its expansion using a symbolic CM will be described. After summarizing the proposed KWS system, experimental results will demonstrate the effect of parameters of the building blocks used in the system. Moreover, the KWS results will be combined with LVCSR based baseline systems to show that we can improve the KWS performance for OOV queries. In Chapter 4, a discriminative approach for training the CM will be introduced, and experimental results will be demonstrated on an existing LVSCR based KWS system. In Section 4.2, the ideas presented at the beginning of Chapter 4 and Chapter 3 will be combined and we will show that by discriminative training of the CM, we can also improve the KWS performance of the symbolic index based system. Finally, in Chapter 5, the results will be summarized and possible directions for further research will be given.

2. KEYWORD SEARCH SYSTEMS

In the KWS task, the aim is to locate written queries which are not known beforehand in spoken content. A general approach to KWS is to use an automatic speech recognition (ASR) system to obtain phonetic or textual description represented as lattices from which an index is generated and then information retrieval techniques are used for search [2, 3, 6, 9]. When the ASR system achieves low word error rate, then the text retrieval system will successfully find the query terms. However, in real world situations such as varying acoustic and channel conditions of the recordings, short utterances, spontaneous speech in telephone conversations which is full of hesitations, ungrammatical sentences, foreign words, etc., ASR performance degrades which in turn reduces the overall KWS performance. In the low-resource setting, there are not enough transcribed data to obtain reliable ASR output so the error rates are higher and the search results are not reliable. Therefore, a semi-supervised or unsupervised approach can be adopted rather than supervised methods which require large amount of annotated data. And these methods usually do not include an ASR component. Unsupervised methods in KWS are usually based on pattern matching, that is finding the best alignments between two sequences of feature vectors. For this purpose a dynamic programming technique, namely dynamic time warping (DTW) and its variants such as segmental DTW [10] are widely used. Graph based techniques exploiting the acoustic similarity of speech intervals are other methods for KWS in zero resource setting [11]. There are also posteriorgram based methods such as Gaussian posteriorgrams [12] that do not require transcriptions since they are based on modeling acoustic features with Gaussian mixture model (GMM) in an unsupervised manner. They are widely used in QbyE KWS tasks where the queries are also in spoken form.

As mentioned above, there are several approaches proposed for retrieving written or spoken keywords from spoken content. In this chapter, these approaches will be categorized into five groups:

(i) LVCSR output based search

(ii) Pattern matching approaches

(iii) HMM based keyword-filler models

(iv) Point Process Models (PPM)

(v) Discriminative approaches

Since the index-based approach introduced in this thesis is closely related to the LVCSR based KWS, these systems will be described first in detail. In the following sections, the rest of the approaches will be summarized. Later in this chapter, the evaluation of the KWS outputs and post-processing of the results such as score normalization and system combination which aims at improving the KWS performance will be given. Then the database and the limited resource conditions of the task that we have worked on under the IARPA Babel Program will be given.

2.1. LVCSR Output Based Search

These systems are cascade of an LVCSR system, that converts the speech signals into text, and a text information retrieval system. The basic approach is to obtain lattices from LVSCR system instead of a single best word sequence to overcome recognition errors. Then an index is generated from these lattices so as to compactly represent the speech data to be searched. Finally, the query is searched over the index.

Most of the KWS approaches utilize a speech recognizer at some stage of the system, for example an index to be searched is generated using the ASR lattices [2,13] or a phonetic representation of the data is obtained using an ASR component as in the posteriorgram based approaches [14,15]. Therefore, ASR systems will be summarized first. Although there are different aspects of ASR such as vocabulary size, speaker dependency and continuity of spoken data, i.e. words uttered without stops in between versus words uttered in isolation, most of the research is focused on LVSCR. In LVCSR, large vocabulary implies a vocabulary size larger than 5000 words and continuous indicates that there are no pauses in between the words. Speaker independent systems aim at recognizing speech from any speaker whereas speaker dependent (SI) systems apply normalizations per speaker to limit the acoustic variability across speakers and to optimize the recognition system.

2.1.1. Automatic Speech Recognition

The goal of ASR is to decode the word string, W, given the acoustic observation sequence, A, by maximizing its posterior probability:

$$\begin{aligned} \widehat{W} &= \arg\max_{W} P(W \mid A) && (2.1) \\ &= \arg\max_{W} P(A \mid W) P(W) && (2.2) \end{aligned}$$

where the second row follows from Bayes' rule and $\widehat{W}$ is the estimated sequence of words. The $P(A \mid W)$ term in Equation 2.2, is the likelihood of the acoustic observation given the word sequence W and is computed by an acoustic model, $P(W)$ is the prior probability of sequence W and it is given by the language model, which depends on the syntactic and semantic constraints of the language [16].

An ASR system consists of the following basic components [17]: Acoustic front-end transforms the speech input into A using signal processing methods. In this step, features such as perceptual linear prediction (PLP) or mel-frequency cepstral coefficients (MFCC) are extracted. Then an acoustic model is used to compute $P(A \mid W)$ and language model is used to estimate $P(W)$. The language model estimates the probability of a word given the previous words and constrains the speech recognition search space by giving a higher probability to the most likely word sequences in the language [18]. Finally, to determine the desired word sequence $\widehat{W}$, all possible word sequences must be searched over, given the acoustic data A. Since the search space is large, a decoding algorithm is applied.

Most speech recognition systems use hidden Markov models (HMMs) to deal with the temporal variability of speech and GMMs are used to model the probability distributions of acoustic features that are associated with each state of the HMM.

That is the observation densities at each state are modeled by GMMs. There were also artificial neural network based methods to predict HMM states from windows of acoustic features. These neural networks were limited to a single hidden layer due to scarce computational resources and their performances were not significantly better than GMM-HMMs [19]. However over the last few years, advances in both machine learning algorithms and computer hardware have led to efficient methods for training deep neural networks (DNNs) that contain many layers of hidden units and a very large output layer [20]. A large output layer is important because the number of HMM states can be large. Recent studies showed that the use of DNNs, can outperform GMMs at acoustic modeling when dealing with large datasets and large vocabularies [20].

DNNs are multi-layered extensions of neural networks and has several hidden layers. Neural networks are composed of computation elements called neurons that take a weighted sum of their inputs and apply a nonlinearity to the sum to generate its output [21]. The input layer just presents the input data from the outside world, the hidden layers consist of neurons that take their input from the outputs of their previous layers. The final layer can have one or more neurons applying different nonlinearities before generating the final output of the DNN depending on the application. For instance, in multiclass classification applications such as phone classification, softmax nonlinearity is used which tries to mimic the posterior probability of the classes given the input. If there are K classes, then the output layer will have K neurons. Moreover, if the inputs to the output layer neurons are denoted by $a_k, k \in \{1, \cdots, K\}$, then the softmax output y_k of the k-th neuron can be written as

$$y_k = \frac{\exp(a_k)}{\sum_{k'=1}^{K} \exp(a_{k'})} \tag{2.3}$$

Since $\sum_{k=1}^{K} y_k = 1$ and $y_k \geq 0$, DNN outputs y_k can be treated as the posterior probability of class k given the input. In particular, if the acoustic input feature $\mathbf{x}^t$ is the DNN input at time t, then $y_k = P(k|\mathbf{x}^t)$.

2.1.2. Weighted Finite State Transducers

As the LVSCR lattice structure and the index is in weighted finite-state transducer (WFST) form, a brief overview of WFSTs will be given in this section. A detailed review of the WFST related concepts can be found in [22, 23].

A WFST is composed of finite number of states, a subset of them are initial or final states. Each arc between the states of the WFST contains an input and an output label which are either an ϵ symbol or an element of a finite input and output alphabet, respectively, and a weight is associated with the arc. Weights in a WFST may represent probabilities, durations, penalties, etc. Especially, probabilistic WFSTs are widely used in speech processing applications [22]. In addition, weight functions are defined for initial and final states. Weighted finite-state acceptors (WFSA) are defined similar to WFSTs except that the output labels of the arcs are the same as their input labels and thus the arcs of an WFSA can be represented using a single label and a weight.

A *path* in a WFST is a sequence of arcs such that the ending state of an arc is the beginning state of the next arc. A successful path is defined as a path from an initial state to a final state of the WFST [24]. The weight of a path is found by multiplying the weights of the arcs on the path. The weight of mapping an input sequence to an output sequence is calculated by accumulating the weights of the paths in the WFST that has the same input and output strings on its input and output labels. A string is said to be *accepted* by an WFST if there exists a successful path whose input label sequence is the same string [22].

2.1.3. ASR Output Representation

For a given speech utterance, an ASR system generates hypotheses based on the acoustic data. In LVCSR based KWS systems [2,3,9,13], an inverted index which maps units (words, phones, other sub-word units etc.) to the utterances is generated using these hypotheses. There are different possibilities to represent these hypotheses which will be used in indexation. These are

- One-best hypothesis
- Lattice structure
- Confusion Network (CN)

The simplest way is to take only the one-best hypothesis for the utterance which does not allow any alternative hypotheses. If the ASR system is reliable, i.e. it has a low error rate, this single hypothesis leads to a sufficient KWS performance with lower search time as the search space will be small. But for the low-resource language settings, we have limited amount of labeled data which has high error rates, alternative hypotheses should also be indexed.

One method to represent the alternatives is generating a lattice structure. These lattices are in WFST form that associate weights with hypotheses which are used to rank them [24]. Since there is a large number of hypotheses, its information content is richer. However, it includes some erroneous hypotheses along with the true ones. A lattice structure is shown in Figure 2.1.

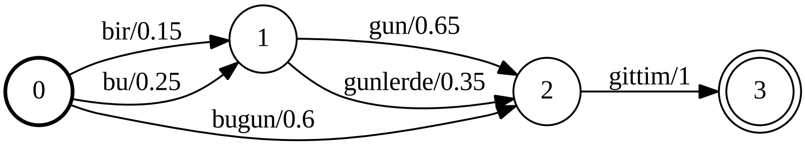


Figure 2.1. An example lattice.

CNs which are also called sausages are also used for representing multiple hypotheses in KWS applications [9, 25, 26]. CNs have a linear structure and consist of confusion bins. These bins correspond to consecutive time intervals. For each time interval, there is a set of arcs where each arc carry the word (or sub-word) hypothesis and a probability or a cost associated with it. CNs are usually derived from lattices by clustering arcs with similar timing to form a confusion bin. Thus, each confusion bin has a start and an end time which are determined from the arcs in that cluster and they are constructed such that the end time of a bin is the beginning time of the next interval bin. Figure 2.2 shows the CN structure of the lattice shown in Figure 2.1. Here 'DEL' denotes a skip symbol and '<s>' and '</s>' are used for marking utterance boundaries. In [9], CN WFSTs are constructed such that the timing information can also be encoded into the output labels of the arcs, which allows a way of generating an index to be used in KWS.

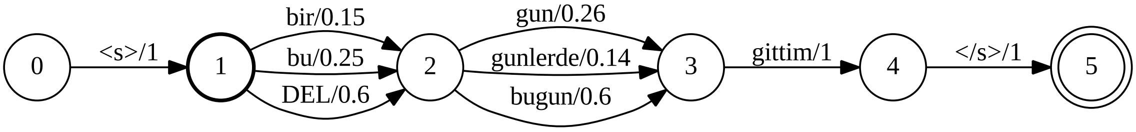


Figure 2.2. CN of the lattice shown in Figure 2.1.

Once the alternative recognition hypotheses are determined, the next step is to generate an index that will allow an efficient search mechanism.

2.1.4. Indexing

Index is a compact structure that retains the necessary information of search data and serves as a look-up block for efficient search. In KWS application, the index is a map from words to utterances along with the begin-end times of the word in that utterance. Moreover, the index can include a reliability score for each entry that indicates the level of confidence that the word is uttered in the corresponding utterance.

The indices generated for KWS purpose store all substrings encountered in the LVSCR lattices. Thus when a keyword matches to particular substring in the index, it can be retrieved.

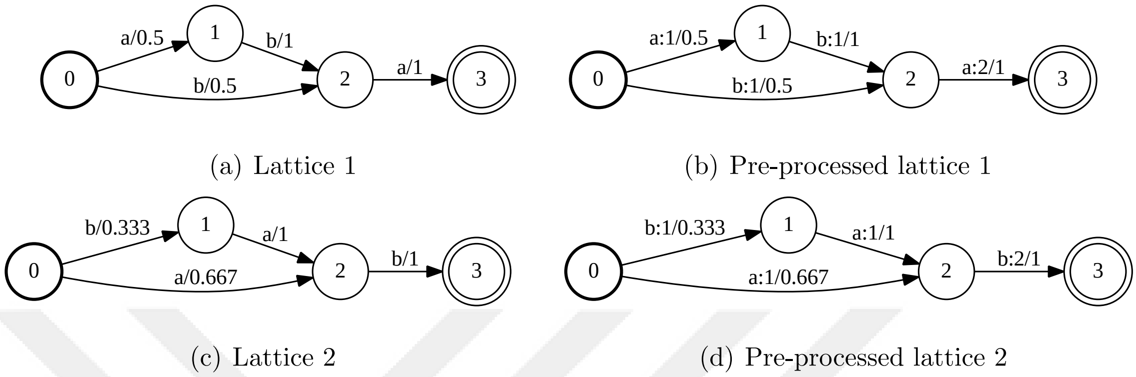
The output of a LVSCR system is a WFST with words labeling the weighted arcs. Timed factor transducer (TFT) described in [13] converts these LVSCR lattices into an index that keeps the information regarding the begin-end times of words, a score along with the word/sub-word unit and utterance information. The difference between this indexing scheme and previous approaches for indexing such as factor transducers and modified factor transducers is that TFT approach includes timing info and this information is kept on the arc weights rather than on the output labels of the arcs [13].

Initially the WFST having the same input labels and overlapping time spans are clustered and the cluster identity are kept on the output labels. For all states a forward probability and a backward probability that represents the shortest distance from that state to the final states are calculated. Then for all distinct input-output sequence of paths in the WFST the probability of observing that sequence is calculated by adding up the weights of the paths having that sequence in their labels according to the semiring used. The minimum of begin times and the maximum of the end times of these paths are used to determine the begin-end time of the current input-output label sequence. A new start and an end state is added to the transducer thus all states have an incoming arc from the new initial state having epsilon on their labels and an outgoing arc from all states to the new final state. This structure allows searching for all possible substrings of a string that is accepted by the WFST.

After generating the factor transducer described above, paths carrying the same factors or overlapping occurrences are merged by adding their probabilities. Then the result is optimized by applying determinization, meaning that each state at most one transition labeled with a input given label, and minimization, i.e. finding an equivalent transducer with the minimal number of states. At the end, the TFT for an utterance is obtained.

The index for an entire dataset is constructed by taking the union of all TFTs obtained for different utterances and then optimizing the resulting index by determinization and minimization.

Suppose that we have two utterances in our database and their lattice representations (Figures 2.3(a) and 2.3(c)) are preprocessed such that the output labels of the arcs denote the cluster identifier where each input-output pair represents an arc cluster representing overlapping start and end time pairs. Then we have structures similar to the ones shown in Figures 2.3(b) and 2.3(d). Figure 2.4 shows the factor transducer for the first utterance where there is new initial and final states. Figure 2.5 depicts the result of factor merging and optimization for the first utterance. After doing same operations to the second utterance and taking the union of both WFSTs, the index for the whole dataset shown in Figure 2.6 is generated.



(a) Lattice 1

(b) Pre-processed lattice 1

(c) Lattice 2

(d) Pre-processed lattice 2

Figure 2.3. Lattices (a,c) and the corresponding pre-processed WFSTs (b,d) for two utterances in the database.

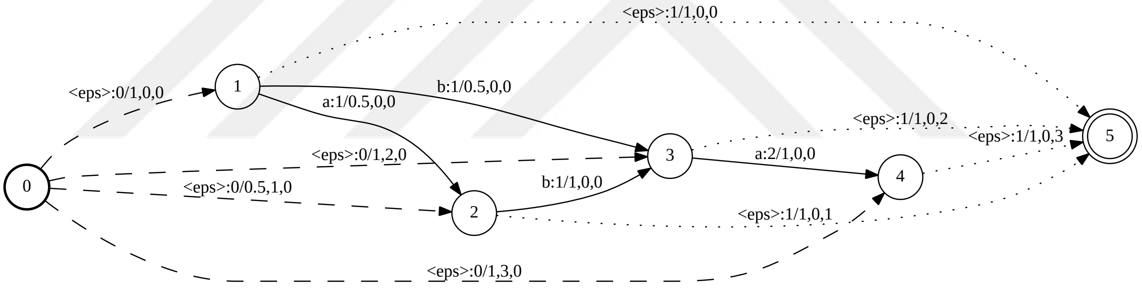


Figure 2.4. Factor generation for the first utterance.

The main advantage of the index described here is its flexibility, especially accepting various query models. Its WFST form allows searching for different queries represented with WFSTs in different forms. For example, multiword queries having an optional silence between words by adding arcs between the states at the word boundary changes the linear structure of the basic query WFSA but the index can still accept these types of queries. Therefore, the index does not need to be changed depending on the query. This results in an efficient mechanism because indexing is more time consuming than query WFST generation most of the time. The index is also computationally efficient because its search complexity is linear in the query length.

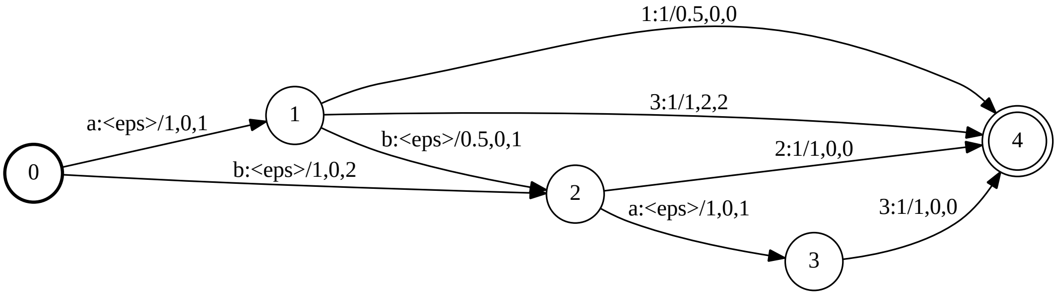


Figure 2.5. Result of factor merging for the first utterance.

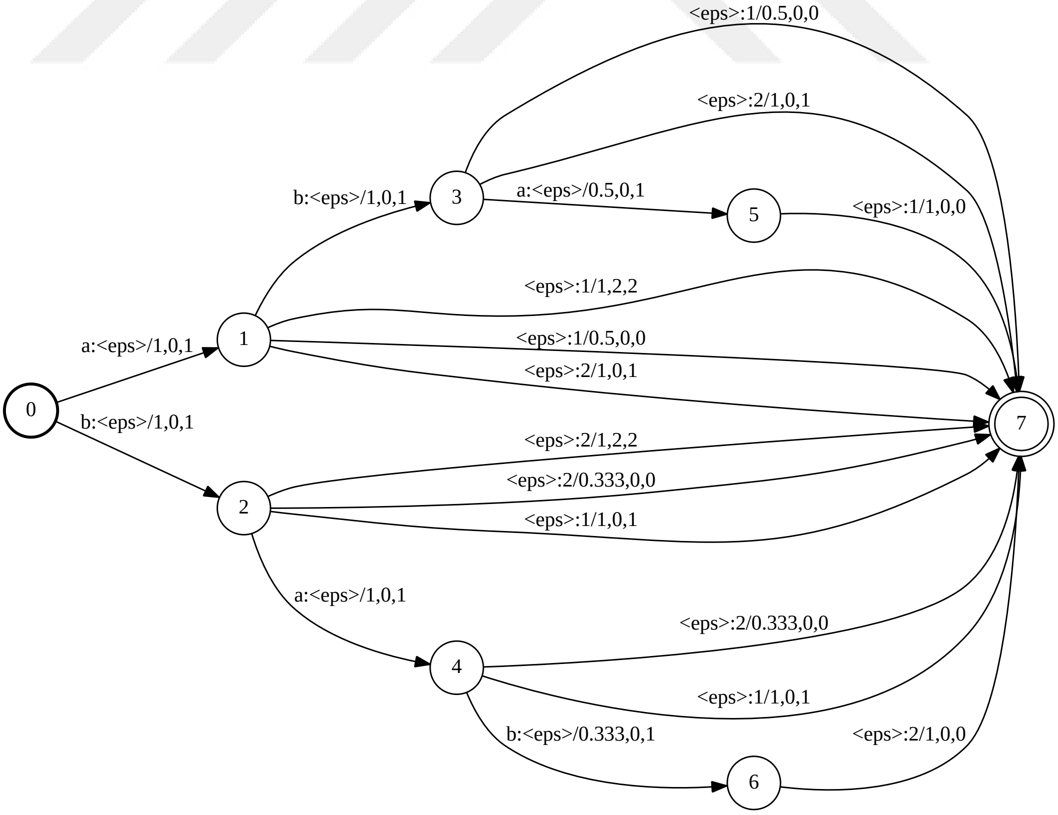


Figure 2.6. Index of the dataset, union of TFTs of the utterances.

2.1.5. Search

Search step of the index based systems consists of query representation which will produce the query WFST of the written query and searching over the index which can be in word or sub-word level. Since both the query and the index are in WFST form, composition operation gives the matching substrings in the database. In the TFT framework, begin and end time information can be extracted from the arc weights and the utterance identity can be retrieved from the output label of the arc leading to the final state. As there can possibly be multiple matching paths, a shortest path operation is also applied to keep only the paths with the lowest cost or highest probability.

If KWS is performed on a word-level index and if the words in the query are in the LVCSR vocabulary, then the query WFSA includes the words as the labels on its arcs. If we are using sub-word units like the phonemes and generating a phonetic index, the orthographic query is first converted into a sequence of phones. For IV queries, the lexicon is used to find the phonetic description of the keyword. On the other hand, if the query is OOV, i.e. it has at least one word that is not covered by the ASR vocabulary, a grapheme-to-phoneme system (G2P) is required in order to find the phonetic representation of the query. G2P conversion algorithms can be based on deterministic rules or statistics obtained from the spoken data and its orthographic transcription [27]. For example, Sequitur G2P tool described in [28] is based on statistical joint modeling.

For OOV queries a matching pattern in the index cannot be found because there will not be a matching element corresponding to the OOV word. However, since the index is in WFST form that can accept any type of WFST as query, the query can be transformed to different forms in order to get search results for OOV queries. Since the sub-word units are building blocks of the words, the OOVs can be represented in terms of these units. Therefore, one way to handle OOV keywords is using sub-word units in indexing and searching for the sub-word level representation of the query in this index. A sub-word index is generated either by generating a sub-word lattice from the ASR system or by converting a word lattice into a sub-word lattice [2]. If both the word and sub-word level indices are generated, there are different possible strategies for KWS. For example,

(i) Search the keyword in each index and then combine the search results.
(ii) Search the word index for IV words and the sub-word index for OOV words.
(iii) Search the keywords in the word index, if no result is returned, search the sub-word index.

Another option for handling OOV queries is expanding the lexicon by adding automatically generated pronunciations of a large number of words to the LVCSR lexicon [29,30]. The augmented lexicon can be used before lattice and index generation or it can be used in representing query. These systems aim at reducing the OOV rate to improve the KWS performance. However, the augmented lexicon still might not include the OOVs with respect to the original lexicon. If the OOV keywords can be anticipated before, this method can be useful, but since we do not know the OOV keywords beforehand, this approach might not be practical in real KWS applications.

The third way of tackling the OOV problem is expanding the query by generating alternative forms of it and searching for all these alternatives along with the original query in order to increase the number of hits. These alternatives will be acoustically similar to the original queries and will also help overcoming the errors in the ASR system in addition to dealing with the OOV queries. These alternatives are generated using CMs which are also represented as WFSTs [5,6,9,29]. In the most general form, CM models insertions, deletions and substitutions. Their WFST representation usually has a single state with self-loops. The input-output label pairs on the arcs of the WFST correspond to the units that are substituted with each other or if the input (output) symbol is "ϵ" it models an insertion (deletion). In a phone-based system these units correspond to phones whereas in a symbolic system, the CM models the confusions of the symbols. The weights or costs on the arcs denote the negative log-probability of insertion, deletion or substitution depending on the input-output label pair.

An example CM is shown in Figure 2.7. As in this example, weights on the arcs of a CM need not to be symmetric. For example, if the substitution of "a:e" has a weight of 2.53, the substitution “e:a” can have a weight of 3.17.

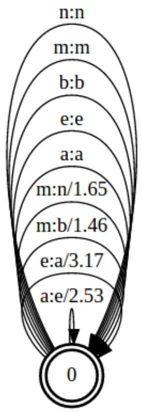


Figure 2.7. A CM example.

As in the case of indices, the CMs can be at different levels. For example, a word level CM along with a word index can be used or the WFST of a query in phonetic level can be expanded with a phonetic CM and then be searched in phonetic index. If Q and CM denote the query FSA and the CM, respectively, and $\circ$ indicates the WFST composition operation, the expanded queries Q' can be summarized as

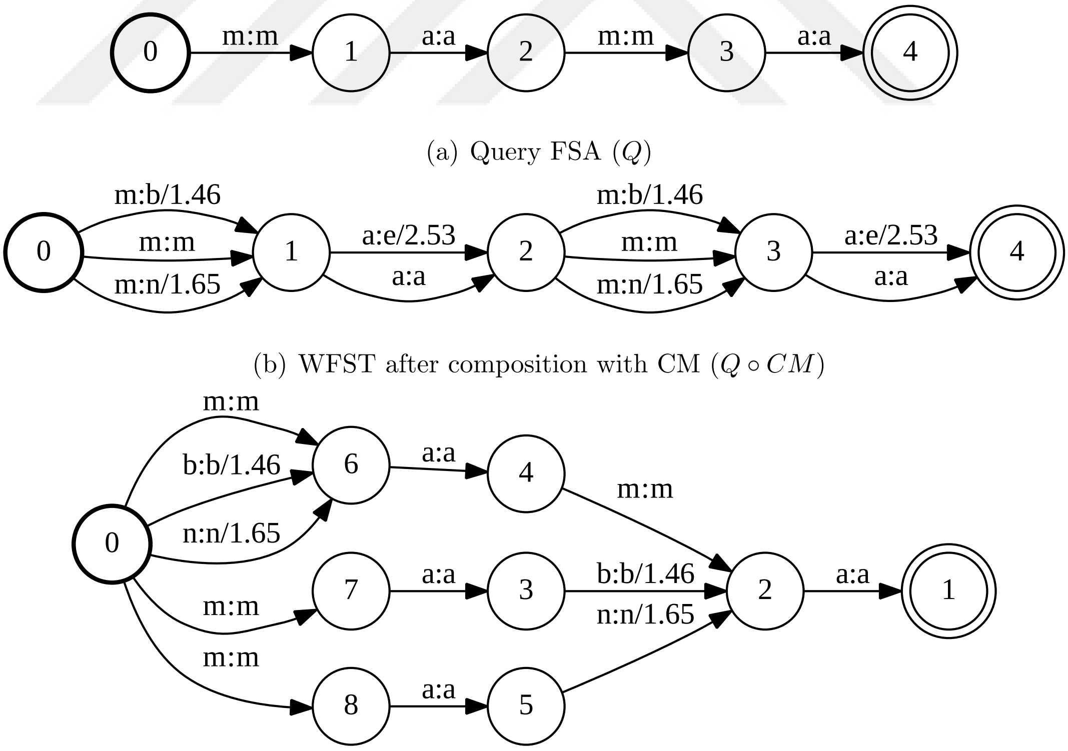
$$Q' = \text{n-best}(Q \circ CM) \tag{2.4}$$

Alternatively, we can have the index in word level and the CM in the phonetic level. In that case, the word representation of the keyword is first mapped to phonetic sequence using the lexicon/G2P, then the phonetic CM is applied to obtain the expanded query. Since our index is in word level, the expanded query should be mapped back to words. If the LVCSR lexicon is used during this stage, this method maps OOV words to IV words for which we can obtain search results. If L denotes the pronunciation lexicon that maps words to phones and if L^{-1} denotes the inverse of L, i.e. mapping phone sequences to words, the expanded queries Q'_w are generated by

$$Q'_w = \text{n-best}(Q_w \circ L \circ CM_p \circ L^{-1}) \tag{2.5}$$

In Equation 2.5, subscripts w, p are used to denote the word and phone level representations in the corresponding WFSTs. If we have a phonetic index, the last composition is not required which will become equivalent to Equation 2.4. In the case where we are using an augmented lexicon [30] as mentioned previously, L will be the augmented lexicon and the inverse lexicon will still be the inverted version of the original lexicon used in the ASR system thus mapping OOV words to IV words.

Figure 2.8 shows how the expanded query WFSTs are generated. In Figure 2.8(a) the linear FSA for the keyword "mama" is shown. Using the CM shown in Figure 2.7, the composition operation in Equation 2.5 will result in Figure 2.8(b). At this point, "baba", "mana", "bana", "nane" are among alternative queries. If we apply the n-best operation for n=5, and project the output labels, we get Figure 2.8(c) where only 5 possible confusions with the least costs are kept such as "mana".



(a) Query FSA (Q)

(b) WFST after composition with CM ($Q \circ CM$)

(c) 5-best result of composition (Q') after projecting output labels

Figure 2.8. Expansion of the query "mama".

Once the expanded query is generated, it is searched over the index using WFST composition operation as described before.

2.2. Alternative Approaches for KWS

After reviewing the KWS methods based on LVCSR output, in this section other spoken content retrieval applications that do not necessarily depend on the ASR output will be summarized. These methods either use pattern matching, keyword-filler models or develop a discriminative method for KWS.

2.2.1. Pattern Matching Approaches

These approaches directly work on acoustic data or features extracted from them and they do not necessarily rely on an ASR system. They try to find the pattern, i.e. sequence of feature vectors that represents data, in the database that is most similar to the query pattern. Pattern matching is usually achieved by DTW which is a dynamic programming method.

In QbyE-STD, as the queries are spoken, both the query and the search data are in the same domain therefore it is suitable to use pattern matching approaches. Initial studies directly use audio features whereas in recent studies DTW based pattern matching methods are applied to different representations of audio data such as posteriorgrams. *Posteriorgram* is a time versus class representation of the input signal. In speech processing, the time axis corresponds to different frames of the audio whereas the classes can correspond to different units such as phones, HMM states or different Gaussians in a GMM depending on the application. As the vectors in a posteriorgram are posterior probabilities, each vector in the posteriorgram adds up to 1. Figure 2.9 shows an example phonetic posteriorgram of an utterance where the classes correspond to phones. In this figure, darker values correspond to higher probabilities. The advantage of using posteriorgram representation of audio is that it allows the data to be represented in a speaker-independent statistical manner as opposed to directly using spectral features which are speaker dependent [7]. Moreover, if the posteriorgrams are generated from neural networks trained in a multilingual fashion as in [31], they can also provide a language independent representation.

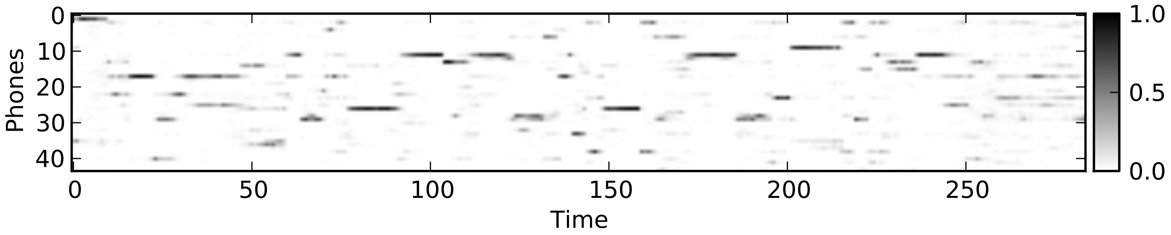


Figure 2.9. An example phonetic posteriorgram.

Variants of DTW algorithm such as segmental DTW [10] and subsequence DTW [32] are also used in QbyE-STD tasks. In [8], DTW algorithm is applied to the phonetic posteriorgrams obtained from a phonetic recognizer in order to locate queries in a QbyE-STD task. In this case, the patterns for query and test utterances are their posteriorgrams. In [12], a GMM is trained and each speech frame is represented with the posterior probabilities of the Gaussian components. Then the Gaussian posteriorgrams of the audio to be searched and the spoken queries are matched using segmental DTW. In addition to Gaussian posteriorgrams, segmental DTW is also used on posteriorgrams generated from deep belief networks in QbyE-STD systems described in [7].

The main drawbacks of DTW based pattern matching approaches to spoken KWS are their computational demand and memory requirement and their search time is significantly higher than the index look-up approaches [8]. Several methods have been proposed to deal with these problems such as lower bound estimates for distance calculation in DTW algorithm [7] or locality sensitive hashing to find an approximate similarity matrix and speed up the system [33]. For example, in [34], ideas from image processing such as Hough transform is used to perform an initial fast analysis of the similarity or distance matrix to roughly find matching regions and then results are refined with complete DTW steps. However, there are still points where further optimization is possible to improve the performance of these approximate methods for pattern matching with DTW in QbyE-STD applications [32].

2.2.2. HMM Based Keyword-Filler Models

HMM-based KWS approaches consist of training a generative model and different decoding strategies are proposed for keyword detection. These methods consist of three components: a keyword model, a background model and a filler model. The keyword and background are modeled separately and the filler model represents the non-keyword portions of the signal. Each utterance is modeled with HMMs such that the keyword model is preceded and followed by non-keyword segments [35]. Keyword detection is based on obtaining the HMM state sequence that yields the highest likelihood for the given observation using Viterbi decoding. If the resulting state sequence passes through the keyword model, then the keyword is decided to be detected. In order to overcome possible mismatches between different models, likelihood ratio approaches are also used for keyword search in HMM-based KWS systems. In these approaches, the likelihood ratio of a garbage-keyword-garbage model to the likelihood of garbage-only model is compared against a threshold in order to give a detection decision. HMM-based models do not directly aim at maximizing the KWS performance during model training which is one of the drawbacks of these approaches [36].

2.2.3. Point Process Models

Although posteriorgram representations are widely used along with DTW algorithms to find the matching patterns between queries and test utterances as described in Section 2.2.1, this representation is also used in an index based approach. However, this index is not generated from LVCSR lattices but from posteriorgram events [37]. The search is performed using likelihood ratio of these events under PPM [38]. Phonetic posteriorgrams are modeled in a sparse way as a set of phonetic or acoustic events. The arrival of phonetic events along time collectively define the temporal point pattern and PPM framework uses this representation of speech signals [38–40]. The "index" in PPM systems, is just the collection of these acoustic events [37]. Arrivals of these events are modeled by two Poisson processes one corresponding to a background model and the other one to a foreground model. Once the index is generated from the posteriorgram, KWS is performed using a detector function. This function is basically the likelihood ratio of the point pattern for the query given the word is uttered in the search data (foreground model) to the case where the query is not uttered (background model). This detector function is evaluated around the events in the PPM index within the keyword duration which is obtained from training instances. In addition to whole word search, subword unit search followed by word models is also applied for handling OOV queries. Due to their event based approach, the PPM KWS systems differ from frame-based DTW approaches applied to posteriorgrams and HMM-based methods [38]. In [41], PPM outputs are combined with LVCSR based KWS results in a low resource setting and an improvement in the KWS performance is observed for both IV and OOV queries.

2.2.4. Discriminative Approaches for KWS

These methods directly tackle the KWS problem instead of using a LVCSR system first. The main idea is to focus on KWS performance rather than the error rate of a LVSCR system which usually does not correlate with the search performance. There are KWS approaches based on large margin methods such as support vector machines and other discriminative methods that directly train a system for a pre-defined set of keywords which are usually based on neural networks. Recurrent neural networks (RNNs) such as long-short term memory (LSTM) [42] neural networks and deep neural networks are also used in discriminative approaches. These systems search for a given set of keywords and they require positive and negative examples, i.e. some utterances containing the keyword and some that do not. This is the major drawback of these systems because they are not applicable to the cases where we do not know the keywords beforehand as in our task. Moreover, in the low resource setting queries might not be encountered in the training data and can be OOV, therefore we cannot have positive labeled data even if we know the queries.

In [36], the large margin discriminative training learning approach aims at attaining a high area under the receiver operating characteristics (ROC) curve by learning a keyword spotter function. This function takes the acoustic features of the search data and a query and outputs a weighted sum of the features maximized over all possible alignments of the keyword in the data. Then, this output value represents the confidence that the keyword is uttered in that acoustic data. In this method a support vector machine like approach is followed which tries to maximize the difference between the features of utterances in which the keyword is uttered and those in which the keywords are not uttered.

In [43], a particular type of RNN, namely LSTM and an algorithm that finds the most likely labeling of unsegmented sequential data are used. The RNN structure allows the incorporation of the information over long time spans to estimate the probability of the presence of a keyword in the speech signal. System training requires a list of keywords in the order in which they appear in the input speech data along with the data itself. Search is based on decoding the input sequence.

In [44], the ideas from the above mentioned studies are combined where bidirectional LSTM network outputs are used as one of the acoustic features in the large margin approach of [36].

Based on the success of LSTMs for sequence modeling tasks, in [45], LSTM network is used as the feature extractor in a QbyE task. In this system, the activations from the last hidden layer just before the softmax layer of the LSTM network are stacked over a fixed number of frames thus embedding audio segments of different length into a fixed dimensional space. Same type of features are also extracted for the keywords to be searched and then cosine distance is used as the matching procedure. Although they used both phonetic and whole word labels as the network output, they found that the word modeling is a better approach.

Another discriminative approach for KWS is the “Deep KWS” framework described in [46] for a QbyE application. In this work, acoustic features of the data are fed into a DNN which is trained to directly predict the keywords or subword units of the keywords. Thus frame-based keyword label posteriors are obtained and then the noisy posteriors are smoothed and a confidence score for the hit is calculated using the smoothed posteriors.

Although the main KWS approach followed in this thesis is closely related to the index based approaches using LVCSR output, here a single string is used to represent each utterance instead of a lattice or CN structure. Another difference is in data representation which is based on the encoding of search data posteriorgram. The posteriogram representation is inspired from the QbyE-STD approaches that use pattern matching techniques over them. Moreover, the discriminative training of the CM that directly tries to maximize the KWS performance criterion instead of LVCSR accuracy resembles the discriminative approaches to KWS.

2.3. KWS Task Evaluation

The KWS systems described above return a lit of hits that indicate the location of the query in the search data. These lists are usually post-processed to achieve higher performance. Another way of improving the KWS performance is combining the results of different KWS systems by making use of diversity. Therefore, in this section, the definitions of the performance criteria reported in this thesis will be given and then score normalization and system combination techniques will be reviewed.

2.3.1. KWS Performance Evaluation

The output of a KWS system includes the list of hypotheses returned by the search system for each keyword. The search results contain the following information: the name of the file in which the keyword is supposed to be uttered, the begin time of the keyword in that utterance, its duration and a reliability score. These scores can then be post-processed and a threshold can be applied such that if the hypothesis has a score greater than this threshold, the hypothesis is decided to be a hit otherwise it is treated as not detected. Threshold based decision rule allows us to differentiate between the correctly detected keywords from false alarms. The KWS system performance is computed based on the actual decisions obtained after thresholding.

Once the ground truth, i.e. the exact locations of the keywords in the search data are known, the hits can be categorized into correct detections and false alarms.

In addition to the false alarms, the other type of error is missed detections. Missed hypotheses are the ones which are not returned by the KWS system although the keyword is uttered in the search file.

For evaluating the performance of a KWS system, the initial step is to align the hypotheses for keyword occurrences returned by the system with the ground truth which is the true occurrences of the keywords in the search data determined by the begin and end times. The alignment operation allows a temporal tolerance, usually taken as 0.5 seconds, and checks the overlapping system and reference occurrences. If a reference is not mapped to any system hypotheses, it signals a miss or if a system hypothesis has not a reference occurrence for the keyword, a false alarm is declared as the alignment output.

The next step is calculating the performance metrics. The Term Weighted Value (TWV) given in Equation 2.6 is defined by NIST in 2006 for Spoken Term Detection Evaluation [47] and it is a widely used KWS system performance measure in recent studies. It is a single metric evaluated at a particular operating point in the Detection error tradeoff (DET) curve which depicts the tradeoff between the probabilities of missed detections (P_{miss}) versus false alarms (P_{FA}) for different decision thresholds θ.

$$\text{TWV}(\theta) = 1 - \frac{1}{|\mathcal{Q}|} \sum_{\text{q} \in \mathcal{Q}} P_{\text{miss}}(q, \theta) + \beta P_{\text{FA}}(q, \theta) \tag{2.6}$$

In Equation 2.6, $\mathcal{Q}$ is the set of query terms, $|\mathcal{Q}|$ is the total number of queries and the weight β takes into account both the prior probability of a keyword and the relative costs of false alarms and misses and is set to 999.9 as in [47]. Miss (P_{miss}) and false alarm (P_{FA}) probabilities in Equation 2.6 are determined using

$$P_{\text{miss}} = 1 - \frac{\text{N}_{\text{cor}}(q, \theta)}{\text{N}_{\text{true}}(q)} \tag{2.7}$$

$$P_{\text{FA}} = \frac{\text{N}_{\text{spur}}(q, \theta)}{\text{N}_{\text{NT}}(q)} \tag{2.8}$$

In Equation 2.7, $\mathrm{N}_{\mathrm{true}}(q)$ is the true number of occurrences of the query term q in the corpus, $\mathrm{N}_{\mathrm{cor}}(q,\theta)$ is the number of correct (true) detections of the term q with a detection score greater than or equal to the threshold θ. Keywords that do not occur in the search data ($\mathrm{N}_{\mathrm{true}}(q) = 0$) are excluded from TWV calculation. In Equation 2.8, $\mathrm{N}_{\mathrm{spur}}(q,\theta)$ is the number of spurious (incorrect) detections of q with a detection score greater than or equal to θ. $\mathrm{N}_{\mathrm{NT}}(q)$ is the number of opportunities for incorrect detection of q in the corpus. In detection tasks, the instances of the objects are discrete items. However, KWS in continuous speech does not have discrete occurrences. Therefore, the total number of possible false alarm instances or the “Non-Target” term trials for a query $\mathrm{N}_{\mathrm{NT}}(q)$ are calculated using

$$\mathrm{N}_{\mathrm{NT}}(q) = \mathrm{T}_{\mathrm{speech}} * n_{\mathrm{tps}} - \mathrm{N}_{\mathrm{true}}(q) = \mathrm{K} - \mathrm{N}_{\mathrm{true}}(q)$$

where n_{tps} is the number of trials per second of speech (arbitrarily set to 1), and $\mathrm{T}_{\mathrm{speech}}$ is the total amount of speech in the test data (in seconds), thus K is a constant.

As seen from Equation 2.6, the benefit of correctly finding a term depends on its frequency in the search data ($\mathrm{N}_{\mathrm{true}}(q)$) but cost of a false alarm is almost independent to the term frequency since $\mathrm{T}_{\mathrm{speech}} * n_{\mathrm{tps}} >> \mathrm{N}_{\mathrm{true}}(q)$ and $\mathrm{N}_{\mathrm{NT}}(q)$ is almost constant. Therefore, TWV metric emphasizes recall of rare terms. Since OOV keywords usually have low frequency, correctly locating an OOV keyword contributes to TWV more.

The maximum value that TWV can get is 1 and it corresponds to correctly detecting all of the queries, with no false alarms. For systems returning an empty hit list, TWV is 0 and TWV can get negative values. The TWV for a given threshold is called the actual TWV (ATWV). The maximum TWV obtained from all possible threshold values is called the maximum TWV (MTWV) so it is the optimal point on the DET curve. Although, MTWV is primarily used for evaluation in this thesis, other performance measures such as oracle TWV (OTWV) and supremum TWV (STWV) are also used which are modified versions of TWV given in Equation 2.6. OTWV is calculated using separate thresholds for each query that give the MTWV for that particular query. STWV is found by taking search scores for the correct detections and false alarms as 1 and 0, respectively, thus neglecting the effects of false alarms [48]. In this thesis, Framework for Detection Evaluations (F4DE) Toolkit [49] provided by NIST is used which performs the occurrence alignment by Hungarian method for bipartite graph matching and calculates both ATWV and MTWV [48].

An example DET curve is shown in Figure 2.10 where the line on the top right shows the random performance and the curve at the lower left part shows the 0.3 iso-TWV. Filled and empty dots on the system performance curve denotes the ATWV and MTWV, respectively.

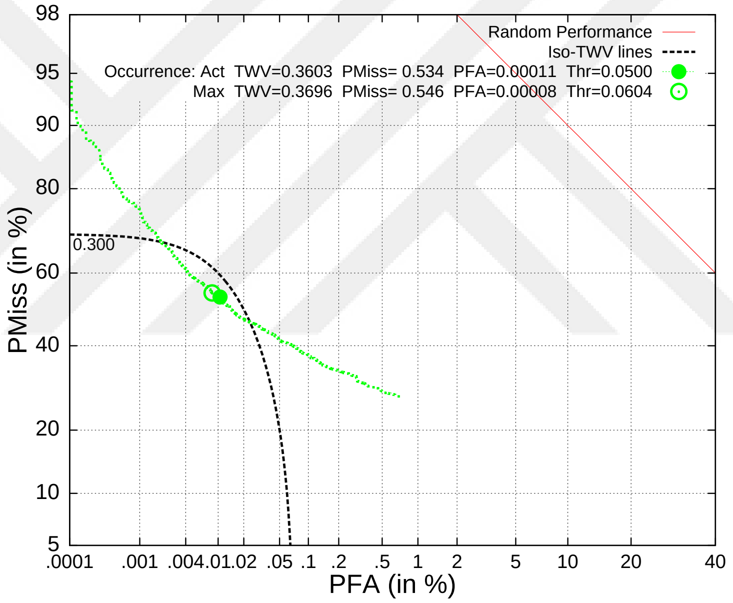


Figure 2.10. An example DET curve.

In the experiments presented in the following sections, MTWV is reported instead of the actual TWV for a certain threshold since the emphasis is not on finding a single threshold to be used in different systems. In combination experiments OTWV and STWV are also reported along with MTWV.

2.3.2. Score Normalization

Due to the characteristics of the keywords, the scores for different queries can be diverse. However, system hypotheses for different keywords should be comparable to each other since the KWS performance is evaluated based on the decisions given according to a global detection threshold. In order to achieve higher TWV, score normalization techniques such as sum-to-one (STO) normalization, keyword-specific thresholding (KST), or machine learning methods that discovers a mapping from raw scores to the normalized ones, have been applied to the raw system scores and KWS performance is measured using the normalized scores [50, 51].

Sum-to-one Normalization: STO normalization scales the scores such that the sum of all the normalized scores of query is one [50]. If h denotes a single hypothesis in the set of all hits for the query q, H_q, then the STO normalized scores $\overline{p}_h^q$ can be written in terms of unnormalized scores p_h^q using the following equation.

$$\overline{p}_h^q = \frac{p_h^q}{\sum_{h \in H_q} p_h^q} = \frac{p_h^q}{Z_q} \tag{2.9}$$

According to Equation 2.9, normalized scores of keywords with small number of detections will be boosted since the denominator will be small for those keywords. This in turn makes the scores more likely to be above the decision threshold and helps reducing the probability of miss.

Gamma Correction: When the dynamic range of scores is high, γ correction is applied with STO normalization. This normalization changes the scores and thus the global threshold results in a different TWV after score normalization. The following equation shows γ correction where p_h^q denotes the score of hit h of query q and $\overline{p}_{h,\gamma}^q$ is the normalized score.

$$\overline{p}_{h,\gamma}^q = \frac{(p_h^q)^\gamma}{\sum_{h \in H_q} (p_h^q)^\gamma} \tag{2.10}$$

Here $\gamma = 1.0$ corresponds to the case where STO is applied alone, $\gamma = 0$ leads to uniform scores for all hits of a query. In that case, the hits of a query which has many hits will have low scores for each hit.

Keyword Specific Thresholding: According to the KWS performance criterion, the TWV, if a detection of query q has a raw score of p_h^q, disregarding this hypothesis has a risk of miss proportional to $\frac{p_h^q}{N_{\text{true}}(q)}$ and a risk of false alarm of $\frac{\beta(1-p_h^q)}{T-N_{\text{true}}(q)}$. Then the keyword specific threshold should be chosen such that the latter risk is smaller and the threshold is obtained by making these risks equal as given in Equation 2.11 [52].

$$\theta(q) = \frac{N_{\text{true}}}{T/\beta + \frac{\beta-1}{\beta} N_{\text{true}}} \tag{2.11}$$

However, $N_{\text{true}}(q)$ is not known during KWS so an estimate for it is required. In [52], it is estimated from the sum of the raw scores of all the detections of the keyword q and the result is multiplied by a boosting factor α' determined from a validation set.

$$\hat{N}_{\text{true}}(q) = \alpha' \sum_h p_h^q \tag{2.12}$$

Then an exponential transformation is applied to the scores in order to be able to use a global decision threshold for all keywords. The exponent is determined as the log ratio of the global threshold θ to the KST [51]:

$$\bar{p}_h^q = (p_h^q)^{\frac{\log(\theta)}{\log(\theta(q))}} \tag{2.13}$$

2.3.3. System Combination

In general, system combination aims to obtain a combined system with better performance than the individual performances of its constituent systems by exploiting the diversity among these systems. In this study, by system combination, we imply a “late fusion” which means that we integrate the final KWS retrieval results rather than an “early fusion” in the input representation stage or a combination of different LVCSR systems that use different units (word or sub-word) or different methods (GMM-HMM, DNN) in the KWS systems. In case of KWS, diversity of the systems results from their responses to different types of queries such as query length, number of words in the query, or being an IV or OOV query. For instance, a system may be better at locating multi-word queries but having a low overall performance can be combined with another one achieving better performance for single word queries and the performance of the combined system can exceed the individual systems.

The general steps of system combination is aligning the hypotheses of different systems based on overlap of their time-spans within a tolerance, then calculating a merged score for the corresponding hypotheses and assigning a final begin-end time pair to the combined hypothesis for each keyword. The main difference between combination schemes arises from the way they calculate combined scores. Score normalization can also be incorporated into this step before combining scores in order to bring different scores to the same scale.

Some methods only select the extreme values (min or max) of the coinciding hits whereas others use a weighted sum of their scores. The weights can be 1, corresponding to summation only or the weights can be proportional to TWV performance of the individual systems. The combination scheme used in this thesis is CombMNZ which takes the sum of the scores and then scales it with the number of systems contributing to that sum which has a non-zero score for that hypothesis [50].

2.3.4. BABEL Database

In [53], the aim of the IARPA Babel project is stated as “The Babel Program will develop agile and robust speech recognition technology that can be rapidly applied to any human language in order to provide effective search capability for analysts to efficiently process massive amounts of real-world recorded speech.”

The KWS experiments in this thesis are performed on IARPA Babel Program’s databases. In the Babel project, new languages packs are released each year. These languages are low-resourced languages, i.e. only a limited amount of transcribed speech data is available. For example, Turkish, Cantonese, Pashto, Tagalog and Vietnamese constitute the first set of languages released. The data collection only contains conversational speech recorded under different channel conditions and covers multiple aspects such as different dialects which might result from recordings from different geographical regions, in recent years some scripted data become available. There is also diversity in gender, age groups, topics in the conversations, recording conditions (mobile/landline phones). Depending on the amount of transcribed data, there are different types of language packs (LP) in this database: Full LP and Limited LP. A very limited LP is also introduced in 2015 [54]. There are 80 and 10 hours of transcribed training data in full and limited LPs, respectively. In some of the packs, additional untranscribed audio data is available. There are also 10-hour transcribed development data and 10-hour evaluation data. The purpose of development data is parameter tuning for the models trained on training data. Transcriptions for evaluation data are not provided neither for Full LP nor for Limited LP. A pronunciation lexicon is also provided in both packs. In the very limited LP, only 3-hour transcribed data are available. However, language specific web data and multilingual feature extraction can be used. For KWS purposes, development and evaluation keyword sets are also provided in orthographic form but these are assumed to be not known during lattice generation and indexing. These keywords can include foreign words that are used in daily life, including words borrowed from other languages. For example, in Swahili dataset we can encounter English keywords such as ‘in fact’, ‘aunty’ etc.

Table 2.1. Distribution of IV and OOV keywords according to query length given in number of words.

	Turkish		Swahili	
Length	**IV**	**OOV**	**IV**	**OOV**
1	91	79	1310	139
2	78	9	760	46
3	50	-	172	-
4	-	-	37	1
5	-	-	9	1
6	-	-	4	1
Total	219	88	2292	188

The KWS performance is measured by different forms of TWV as given in Section 2.3.1. In the experiments, Turkish limited LP dataset (`babel105b-v0.4`) which is among the base period languages of the Babel program is used. For the posteriorgram based KWS system, some experiments are also performed on Swahili (`babel202b-v1.0d`).

The lexicon for Turkish has 10110 and 41320 words in limited and full LP, respectively. Similarly, the lexicon for Swahili contains 8656 and 25289 words. Table 2.1 shows the lengths of IV and OOV queries in number of words in the development set queries for Turkish and Swahili. The OOV rate of the keywords is 28.7% for Turkish and 7.6% for Swahili. The number of letters in Turkish keywords ranges from 3 to 23 where the shortest keyword is "Ali" and the longest one is "çocukların dersleri nasıl". In Swahili, number of letters in the keyword ranges from 2 to 35, "ma" is one of the shortest keywords and "sikudhania ni huyo tunje mwingine kutoka" is the longest.

3. SYMBOLIC INDEX BASED KWS

In this chapter, a symbolic index based KWS system will be introduced. The indexing scheme here is based on lattice indexing framework described in [13,24]. The main difference here is that each utterance represented as a single string of symbols instead of alternative hypotheses. The symbols used in this system does not necessarily correspond to phonetic units depending on the encoding of the data. Moreover, we employ the posteriorgram representation which is frequently used in QbyE-STD applications instead of KWS for written queries. Thus, we combine ideas from two different directions in spoken content retrieval. In the following sections, the KWS system will be described and the experimental results will be given for Turkish and Swahili datasets in the Babel program. Some of the results presented in this chapter along with the KWS results of a DTW-based system are combined with an LVCSR based system, and in [14,15] we showed that we can improve the MTWV especially for OOV queries.

3.1. Symbolic Index Based Search

The symbolic KWS system mainly consists of two steps which are indexing and search. In the first step which is the offline step, the search data is summarized into an index generated from the symbolic representation of the search data. In the search step which is the online step, written queries are prepared for search and then FST index-based search is applied. The general KWS setup is shown in Figure 3.1.

Basically, vector quantization (VQ) is applied to the search data posteriorgram thus they are encoded into a symbolic sequence. Then the index is generated using these symbolic sequences which are in the form of strings instead of lattices or sausages used in LVCSR based KWS systems. This index contains the start, end time and utterance identity information for all symbol substrings in the dataset as described in Section 2.1.4. Once the text-based query arrives, it is converted into a phonetic sequence using a pronunciation lexicon for IV queries and a G2P system for OOV

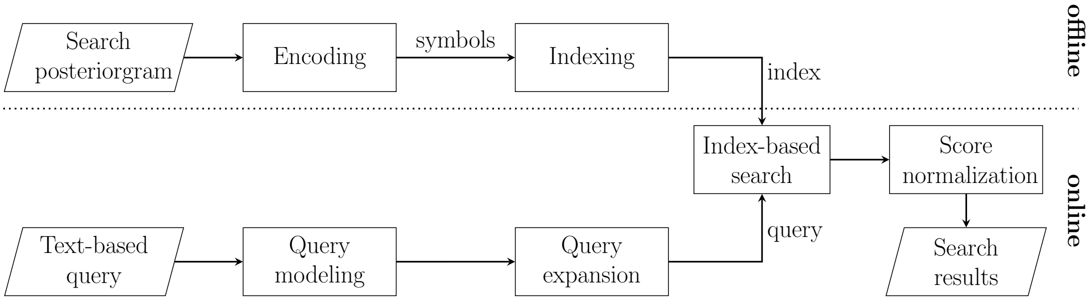


Figure 3.1. Overview of the posteriorgram based KWS setup.

queries that are not covered by the lexicon. Then query posteriorgrams are generated for the queries and they are vector quantized. Thus, symbolic representations of the queries are obtained. Then query FSAs are generated and expanded using a symbol-level CM which is in WFST form. At this point, expanded queries are in WFST form and ready for search in the WFST index. Initial search results are obtained by taking the n-best paths of the composition of the query WFST with the index. If Q, CM and I show the query WFST, CM and the index, respectively, search results are summarized as in Equation 3.1 where $\circ$ denotes the FST composition operation:

$$\text{Results} = \text{n-best}(Q \circ CM \circ I) \tag{3.1}$$

Once the initial list of hits including keyword name, utterance name, begin and end times in the utterance and a score associated with that hit are obtained, score normalization is applied to the hit list to obtain the final search results and these are used in measuring the KWS performance.

In the following subsections, posteriogram representation of the data will be reviewed, then each step in KWS will be described in detail along with various methods used in those steps. For example, an unsupervised or a supervised setup can be used in codebook generation, or there are different options for query WFST generation.

3.1.1. Posteriorgram Representation of the Data

The posteriorgram is denoted by a matrix $\mathbf{Y} = [\mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^t, \cdots, \mathbf{y}^T]$ where t denotes the time index and T is the total number of time frames in the data. In this study, we use DNN-based posteriorgrams because they can be directly obtained from the DNN output layer and they are more robust to variations in the acoustic features. For example, in Gaussian posteriorgram generation, which is based on GMMs trained on the acoustic features, the calculation of the posterior probability of each Gaussian component for each frame is required. [12]. In the DNN-based method, posteriorgram $\mathbf{Y}$ of a set of acoustic feature vectors $\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^t, \cdots, \mathbf{x}^T]$ is obtained by feeding them into the DNN of the LVCSR system. Each column of $\mathbf{Y}$, $\mathbf{y}^t$, is a posteriorgram vector which has dimension of K equal to the number of classes. When DNN applies softmax nonlinearity in its final layer with K nodes, each element of these vectors, $\mathbf{y}^t(k)$, are interpreted as the posterior for class k at time t, the equality given in Equation 3.2 holds:

$$\sum_{k=1}^{K} \mathbf{y}^t(k) = \sum_{k=1}^{K} \mathrm{P(k|\mathbf{x}^t)} = 1 \qquad \forall t \in \{1, \cdots, T\} \tag{3.2}$$

In Eq. 3.2, $\mathbf{x}^t, t = 1, 2, \cdots, T$ is the feature vector of frame t in the audio and T is the total number frames. Although K is fixed for all utterances in a dataset for a given system, T can differ for different utterances with posteriorgrams $\mathbf{Y}_n$ depending on their lengths.

From the output of the DNN, the posterior trajectories of HMM states are obtained which gives the state-level posteriorgram. These states can be context dependent (CD) or context independent (CI) depending on the output targets of the DNN. When we have the CD state-level posteriorgram, the posterior probabilities of all states in the decision tree of a phone are added to obtain the phone-level posteriorgram. Suppose that we have the posterior probability of each state s ($1 \leq s \leq S$) at each time frame t, for $1 \leq t \leq T$, that is $\mathrm{p}^t(\mathrm{s})$, then $\mathrm{p}^t(k)$, the posterior of each phonetic unit

$k, 1 \leq k \leq K$ where K is the number of phones, is computed by

$$\mathrm{p}^t(k) = \sum_{\forall \mathrm{s} \in S_k} \mathrm{p}^t(\mathrm{s}) \tag{3.3}$$

In this equation S_k denotes the set of states associated with phone k.

When the ASR is graphemic, that is based on letters, we can also generate graphemic posteriorgrams. In these systems, we can have a single decision tree, in which case there is not a one-to-one relation between the CD states and the graphemes. Therefore, first we need to find a mapping between states and graphemes to generate the graphemic posteriograms. We take both state and grapheme-level alignments of the training data, then we find the frequencies of graphemes corresponding to each state in the training data. Then we normalize these frequencies ($f_{g\mathrm{s}}$) to obtain weights of observing the grapheme g due to state s, w_{gs} and calculate the posterior of g at time t by using the weights w_{gs} and the posterior probability ($P(\mathrm{s}|t)$) of state s at time t which we get from the softmax output layer of the DNN. In Equation 3.5, we assume that g is conditionally independent of t given the state s.

$$\mathrm{w}_{gs} = \frac{f_{g\mathrm{s}}}{\sum_{g'} f_{g'\mathrm{s}}} \tag{3.4}$$

$$P(g|t) = \sum_{\mathrm{s}} P(g|\mathrm{s})P(\mathrm{s}|t) = \sum_{\mathrm{s}} \mathrm{w}_{gs} P(\mathrm{s}|t) \tag{3.5}$$

3.1.2. Symbolic Representation of the Search Data

Our KWS index is generated from strings of symbols assigned to each time frame. The basic way of obtaining a symbolic representation of the posteriorgram is to perform VQ. The aim of VQ is to obtain a set of codewords such that when the data is quantized the total distortion calculated using a distance metric is minimized. The set of codewords and an associated symbol with each of them collectively define the VQ codebook. Once the codebook is learned, new vectors can be encoded with the symbol of the closest codeword in the codebook.

In unsupervised codebook generation, k-means clustering is applied to the vectors in the search data posteriorgram to get the codewords. Then each frame is labeled with the label of the closest VQ codeword. The k-means algorithm tries to minimize the Euclidean distance between data points and their cluster centers. In this method, the codewords or the cluster centers are initialized randomly, each vector in the data is assigned to the cluster of the closest center, then the cluster centers are updated to the mean of the vectors assigned to that cluster. Although the algorithm converges to a local optimum, by repeating the algorithm using different initializations of the cluster centers and choosing the output that gives the minimum average distance, the global optimum can approximately be achieved.

In the supervised approach for codebook generation, the posteriorgram and the phonetic alignment of the training data are used. Then, using the posteriorgram vectors having the same phonetic label, a separate VQ codebook corresponding to each phone is generated. In this setup, if number of codewords in k-means is chosen as 1, it corresponds to using the average posteriorgram vector for each phonetic label. Then the codebook size will be equal to the number of phones in the system and there will be one-to-one correspondence between codewords and phones since each codeword will be the average posteriorgram of a particular phone.

3.1.3. Symbolic Representation of the Query

As opposed to the QbyE KWS systems where queries are in spoken form, in our work, we have written queries. These queries are first converted into a sequence of phonemes using the lexicon for IV queries and the output of a G2P system for OOV queries. Depending on the codebook generation setup, there are two approaches. The first one is generating the posteriorgram representation of the query and applying VQ to it as in obtaining the symbolic representation of the search data. The second approach is directly using the phonetic sequence of the query as its symbolic representation. The latter approach is applicable only when the codebook is generated in a supervised manner or when we have a transducer that maps phones to symbols directly. Once we get the symbolic sequence of the query, a linear FSA is constructed where each arc has a symbol as its input label which is the same as the output label of the arc.

3.1.4. Query Posteriorgram Generation

Since the posteriorgram is a sequence of posterior vectors, we have to model posterior vectors and the duration in order to generate a posteriorgram. The vectors are either modeled as binary vectors or average vectors and minimum or average duration is used for duration modeling.

In binary posteriorgram generation, the vector corresponding to a phone is constructed using a unit vector. These vectors contain a single 1 at the position which corresponds to the phone and other elements of the vector are 0.

In average posteriorgram generation, each symbol in the phonetic sequence of the query is represented by an average vector corresponding to that symbol. The average vector for each phone is obtained simply by averaging the posterior vectors of the frames labeled with that phone in the training posteriorgrams.

The second dimension of the posteriorgram is time. Therefore, a duration model is needed which determines how many times the vector must be repeated in the posteriorgram. For duration modeling, a minimum duration is imposed by repeating the same posteriorgram vector for a certain number of times, for instance 3 frames. Since the query WFST will be constructed after assigning symbols to the posteriorgram, minimum duration restriction is modeled in WFST construction by using loop structures in the query FSA. Figure 3.2 shows the FSA for the keyword “Ali”. This FSA will require at least 3 matching frames in the index for each symbol which is taken as the phonemes in this example. Another option for duration modeling is using the average

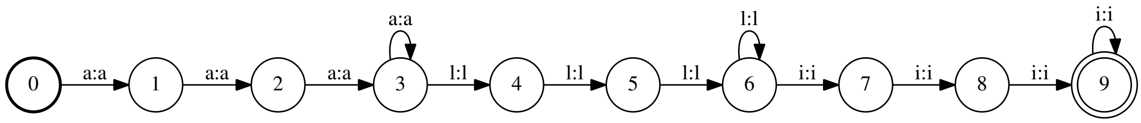
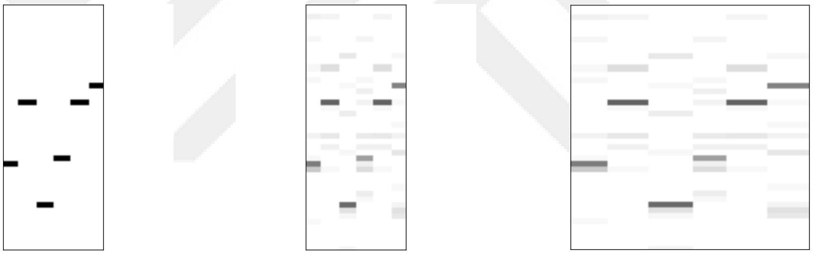


Figure 3.2. Query FSA with the loop structures.

duration information for each phone obtained from the training data. In this case, the average number of consecutive frames labeled with a particular phone is taken as its average duration.

Figure 3.3 shows the binary posteriorgram, average posteriorgram with minimum duration and average posteriorgram with average duration for the example query “meslek”. In these figures, darker colors represent posterior probabilities which are close to 1. In the binary posteriorgram, only one of the symbols have non-zero probability whereas in the rest of the figures there is a distribution over symbols. When average duration is used, each symbol can have different durations but in minimum duration modeling equal number of frames are associated with each symbol of the query.



(a) Binary vector, minimum duration

(b) Average vector, minimum duration

(c) Average vector, average duration

Figure 3.3. Query posteriorgrams according to vector and duration modeling.

For handling multi-word keywords, an optional silence is used meaning that the matching pattern can have the words one after the other without any additional symbol in between or there can be symbols corresponding to the silence element (phone or grapheme). For example, optional silence modeling for the keyword “bos ver” is shown in Figure 3.4 where we allow one frame silence in between the words.

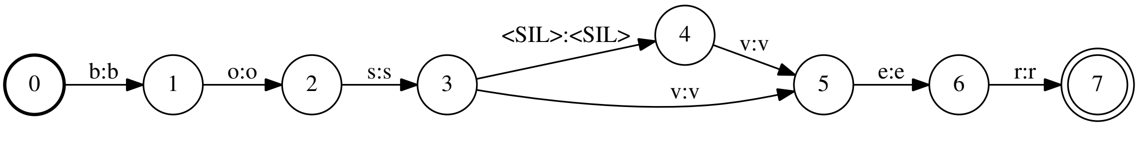


Figure 3.4. Optional silence modeling for multi-word keywords

3.1.5. Confusion Model Generation

CM generation consists of determining the list of confusion pairs (i, o) and the cost of each confusion. In the unsupervised setup, using the Euclidean distances between codebook entries, possible confusions are determined from the nearest neighbor of each codeword. The confusion cost is taken as the distance between two codewords. Thus, if two codewords are close to each other, their distance and therefore the confusion cost will be smaller.

In the supervised setup, we use phonetic alignment of the training data, then we encode the training data posteriorgram and get the symbolic sequence of the same data. Then a confusion matrix is generated by counting the number of symbols assigned instead of each phone. This matrix gives an estimate of the confusion probabilities. Then the confusions having probability less than a threshold are discarded and the remaining confusions are incorporated into the CM and their costs are calculated as the negative logarithm of the probability. Thus, if a confusion has a probability which is close to 1, it will have a lower cost and this confusion will be allowed frequently. In order to get the confusion probabilities, the confusion matrix can be normalized in different ways. Assuming that $C(i, o)$ is the number of frames labeled with o instead of the true symbol i, we can interpret the weights of the CM as the conditional probabilities $P(o|i)$, $P(i|o)$, as the joint probability $P(i, o)$ or the CM can be scaled such that the maximum element in the matrix C becomes 1.

$$P(o|i) = \frac{C(i,o)}{C(i)} = \frac{C(i,o)}{\sum_o C(i,o)} \tag{3.6}$$

$$P(i|o) = \frac{C(i,o)}{C(o)} = \frac{C(i,o)}{\sum_i C(i,o)} \tag{3.7}$$

$$P(i,o) = \frac{C(i,o)}{\sum_{i,o} C(i,o)} \tag{3.8}$$

$$C_{max}(i,o) = \frac{C(i,o)}{\max_{i,o} C(i,o)} \tag{3.9}$$

Equations 3.6-3.9 correspond to normalizing the confusion matrix C row-wise, column-wise, in a joint manner and scaling the matrix with the maximum element, respectively.

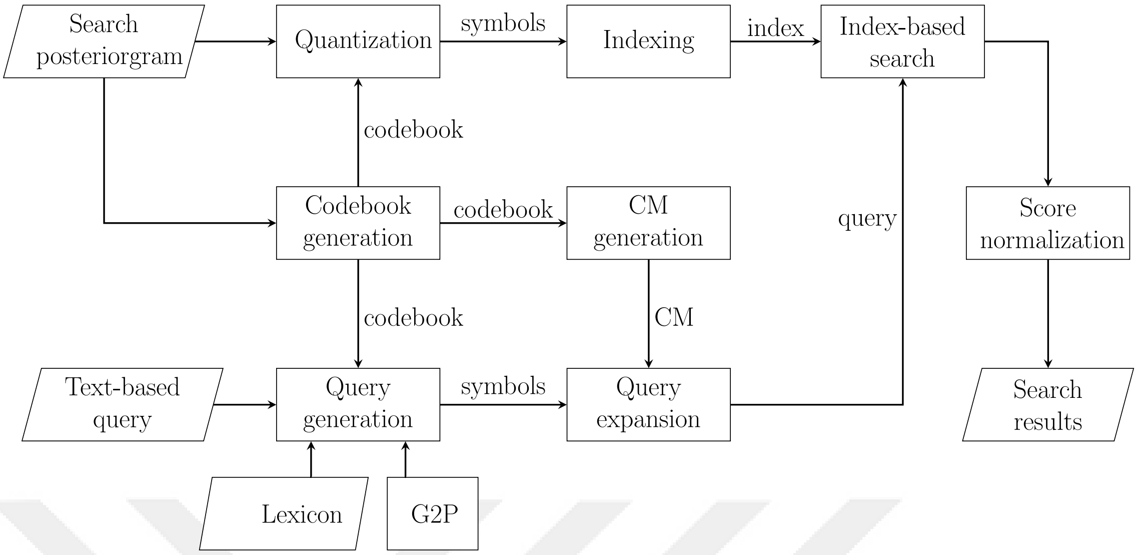


Figure 3.5. Overview of the unsupervised KWS setup.

After describing the components in the proposed KWS system, the setups are summarized in Figures 3.5 and 3.6. These figures show the setups where both the codebook and the CM are generated in an unsupervised and supervised manner, respectively.

3.2. Experiments

In this section the performance of the symbolic index based KWS system will be given and the effect of choices for the building blocks of the system will be investigated. The results will be demonstrated for Turkish (`babel105b-v0.4`) and Swahili (`babel202b-v1.0d`) datasets in the Babel program. In addition to individual performances, their combinations with an LVSCR lattice based KWS system will be given to show that there is an improvement over the baseline especially for OOV keywords.

3.2.1. Experiments on the Turkish Dataset

In this section, KWS search results are presented for Turkish Limited LP data. In the following experiments, the effect of using different setups for the basic building

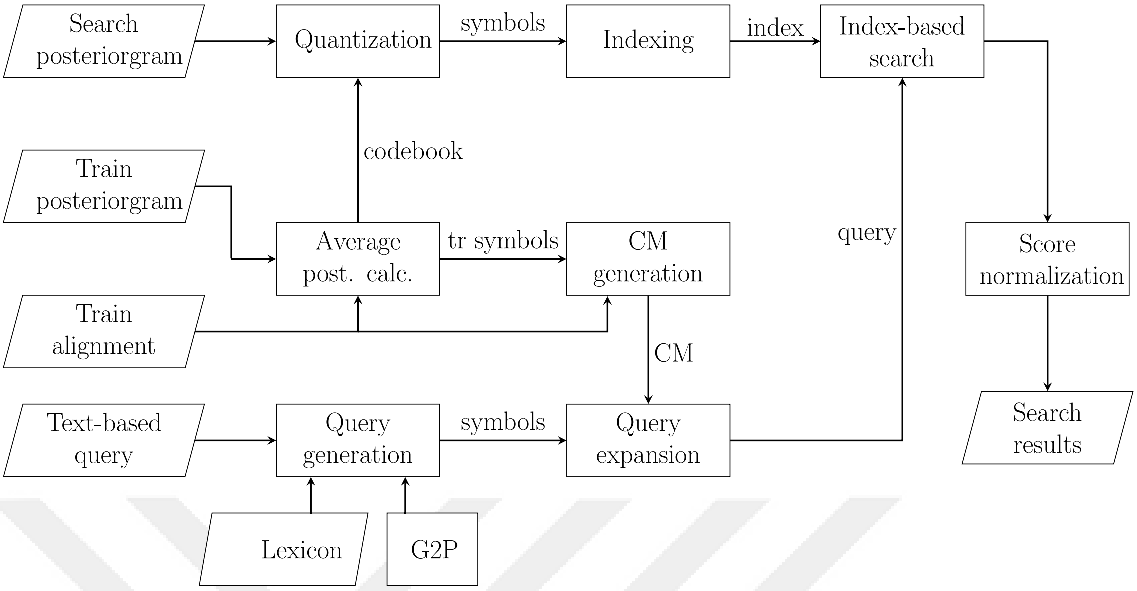


Figure 3.6. Overview of the supervised KWS setup.

blocks shown in Figure 3.1 are investigated using posteriorgrams generated for different units such as phonemes and states.

Posteriorgram generation, finding the symbolic representation of the data, indexing and search is developed based on the building blocks in the Kaldi toolkit [55]. In the following experiments, the Babel recipe of the Kaldi toolkit is used as our baseline LVCSR based system for Turkish [56]. For an overview of the pipeline, the reader is referred to [57]. The DNN from which we generate the posteriorgrams uses p-norm activations as the nonlinearity of the hidden nodes. It results in 62% WER on the 10-hour development set in Turkish limited LP dataset that we are using in the KWS experiments.

3.2.1.1. Individual System Performance. In this section, individual system performances are presented in terms of MTWV calculated over all queries, IV and OOV queries separately.

Table 3.1 shows the KWS performance obtained by using different setups for VQ codebook and CM generation when phonetic posteriorgrams are used. In this table, U

Table 3.1. MTWV for all, IV and OOV queries depending on the supervised (S) or unsupervised (U) setup.

Posteriorgram	VQ	CM	Query	All	IV	OOV
Phonetic	U	U	Bin/Avg	0.0213	0.0266	0.0098
Phonetic	S	U	Bin/Avg	0.0227	0.0237	0.0228
Phonetic	S	S	Bin/Avg	**0.0414**	**0.0394**	**0.0554**

and S denote the unsupervised and supervised setups, respectively, and the last three columns show MTWV calculated over all queries, IV and OOV queries separately. From this table, we observe that using a supervised setup in codebook generation improves the MTWV performance as compared to the fully unsupervised case. When CM is also generated in a supervised manner, the best results are achieved for both IV and OOV queries. The relative improvement in MTWV over all queries by using a supervised CM instead of an unsupervised one (82.3%) is greater than the relative improvement resulting from using a supervised codebook instead of an unsupervised one (6.6%). The same argument also holds for MTWV over OOV queries 143% versus 133%. Since, the symbolic index in this setup does not include score, only the CM weights contribute to the search scores. Therefore, the choice of CM including its weights and allowed confusions directly influence the KWS performance. That is why, the effect of using a supervised CM is more pronounced. In addition, in the supervised CM case, we get the possible confusions by learning from the errors in the training data which allows us to overcome those errors during search. Since there is a one-to-one relation between VQ labels and the phonetic alignments in the supervised setup, we can calculate a frame error rate (FER) by comparing the two strings representing the encoded data and the alignment. We define FER as the ratio of number of frames assigned to a different label by VQ to total number of frames and take the average over utterances in the training data. In the fully supervised experiment of Table 3.1, the FER is found to be 47.02%.

Regarding to the query generation method, it is observed that the symbolic representation for the phonetic sequence of the query become the same. Therefore, their

WFST representations are the same implying that the search results will be the same. In the experiments below, unless otherwise stated, binary queries are used due to their simplicity.

In the above experiments, the best results are obtained when both codebook and the CM are generated in a supervised manner. Therefore, in the following experiments the fully supervised setup is used and KWS is performed using posteriorgrams at different granularities. Moreover, in VQ generation, different number of average vectors, i.e. the codebook entries, are obtained corresponding to each unit. These units are phones or the begin (b), middle (m) and end (e) states corresponding to each phone in a 3-state left-to-right HMM representation of the phones as in Figure 3.7. Instead of CI phones or CI states, CD phones or CD states can be used.

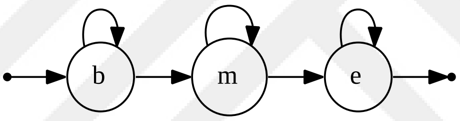


Figure 3.7. 3-state left-to-right HMM with begin (b), middle (m) and end (e) states.

In Table 3.2, the first column shows the posteriorgram type used in the experiment, the codebooks are generated such that there is an average vector corresponding to each unit written in the second column. The third column shows the VQ codebook size, the fourth column shows the FER (in %) for the confusion matrices used in CM generation. Finally, MTWV for all, IV and OOV queries are given in the last three columns. For example, in the third experiment given in Table 3.2, average CD state-level posteriorgram vectors corresponding to each phone are taken as the codewords and the codebook size is 43 because there are 43 phones for Turkish in this dataset. Using this codebook, the FER for training data is 49.7%.

According to Table 3.2, the best result is obtained when the phonetic posteriorgrams are used with a codebook consisting of codewords associated with each phone. In the experiments, it is observed that when the codebook size is larger than the number of units in the posteriorgram, KWS performance is lower than the case where the

Table 3.2. MTWV of search results and the FER in confusion matrix depending on the posteriorgram type and codebook generation.

Postgram	VQ	Codebook Size	FER (%)	MTWV		
				All	IV	OOV
Phonetic	Phone	43	47.02	**0.0414**	**0.0394**	**0.0554**
Phonetic	3-state	131	81.88	0.0072	0.0086	0.0052
CD State	Phone	43	49.70	0.0227	0.0255	0.0209
CD State	3-state	131	53.72	0.0249	0.0315	0.0094

codebook size is smaller. This is observed both in the supervised setup and in the unsupervised setup where different values of k in k-means is used. For example, when using phonetic posteriograms if we have a codebook at the 3-state-level which leads to a codebook of size 131 instead of a phonetic one with size 43, KWS performance is lower when we use the larger codebook. When we have a large codebook and working with a lower dimensional posteriorgram, we have less number of basis vectors spanning that space but we have a larger codebook which has more than necessary vectors. In that case, some codewords will be a combination of the basis vectors and will increase the error rate after quantization because the probability of assigning a vector to a different vector will be higher. This effect is also reflected by the FERs given in Table 3.2, as the codebook size increases while working on a certain type of posteriorgram, the FER also increases.

As mentioned in the supervised CM generation, there are different possible normalization options for the confusion matrix while generating the CM. In Table 3.3, phonetic posteriorgrams are used in the supervised setup but in each case different normalizations given in Equations 3.6-3.9 are applied.

Since the matrices are pruned after normalization, the allowed confusions change depending on the normalization type. In column normalization case, types of confusions differ as compared to row normalization, and the allowed confusions result in better

Table 3.3. MTWV for all, IV and OOV queries depending on confusion matrix normalization.

CM	All	IV	OOV
$P(o\|i)$	0.0414	0.0394	0.0554
$P(i\|o)$	0.0305	0.0224	0.0641
$P(i,o)$	0.0205	0.0176	0.0415
$C_{max}(i,o)$	0.0222	0.0180	0.0449

matches and KWS improves especially for OOV queries. As shown in this table, joint and maximum normalization lead to similar results, since after normalization both matrices become a scaled version of the other. Therefore, the possible confusions will be the same but weights of the CM will be different. However, in MTWV calculation except the threshold leading to MTWV, their performances become similar.

As shown in Equation 3.1, the search is done using FST composition and the search results are obtained by taking the n-best (shortest, least cost) path of the resulting WFST. Unless otherwise stated, 100-best is taken in the experiments. In the results shown in Table 3.4, the effect of the number of best paths taken is investigated. The n-best operations are applied to the results obtained from the fully supervised system shown in Table 3.1. In Table 3.4, number of hits and STWV are included.

Table 3.4. Number of hits and STWV as the number of best paths changes.

n-best	Number of hits	STWV
100	16833	0.1501
250	26598	0.1604
500	40041	0.1818
750	51958	0.1886
1000	62612	0.1946

As the number of paths increases, the number of results also increases. Therefore, additional correct hits are detected which leads to increase in STWV. However, they are accompanied by additional false alarms. That is why the increase in STWV is not reflected to MTWV which is almost constant at 0.0414, 0.0394, 0.0554 for all, IV and OOV queries in all cases. As shown in the second column of Table 3.4, the total number of hits does not increase linearly with n. Another point to note is that the number of hits per query does not necessarily increase linearly with n because the overlapping hits in time are represented with a single hit and additional paths might only contribute to those hits. Since our index is frame-based, this effect is frequently observed because if we find a match spanning certain number of frames, and a longer segment consisting of that hit and a few more leading and/or trailing frames with zero cost will also be among the best paths but in system evaluation, they will be counted as a single hit. In addition, we might not have sufficient (more than n) number of paths in which case increasing n will not change the number of hits.

Table 3.5 shows the effect of score normalization on MTWV. As the search scores are completely determined from the weights in the CM, the frame level matching gives lower scores for longer hits. To normalize against this effect, the negative log of scores are divided to the number of frames for the match. This normalization is shown in the first column. Another normalization is KST which is described in Section 2.3.2. Second column in Table 3.5 shows the effect of the boosting parameter α' in KST. Only in the second row of the table, both types of normalizations are used. In all cases, STO is applied after the given normalization. In Table 3.5, ’+’ and ’-’ in the first column shows whether query length normalization is applied or not. Although the MTWVs are less sensitive to the parameter α' used in the estimation of the $\mathrm{N}_{\mathrm{true}}$, using query length normalization led to lower MTWV. Our frame-based CM and the query WFST with a loop structure result in matches where different confusions are used different number of times consecutively. Simply dividing the score with the total length treats their repetition counts as if they are same. This situation might distort the scores and the threshold leading to MTWV which in turn might result in a lower performance.

Another type of normalization is applying γ correction along with STO normal-

Table 3.5. Change in MTWV depending on the score normalization.

Query length	KST (α')	All	IV	OOV
+	-	0.0248	0.0260	0.0313
+	1.5	0.0228	0.0185	0.0369
-	1.5	0.0410	0.0387	0.0530
-	2.0	0.0410	0.0386	0.0536
-	0.2	0.0396	0.0390	0.0558
-	0.02	0.0391	0.0406	0.0486

ization. Figure 3.8 shows the change in MTWV as γ changes for the supervised setup given in Table 3.1. When γ is close to 0, scores are uniform and scores of queries with large number of hits tend to become lower than threshold which reduces correct decisions. When γ becomes larger than 1, small scores in the unnormalized hits become almost zero which again reduces the number of correct hits above the threshold and therefore the MTWV. In this setup, we get the highest MTWV (0.0431) over all queries at 0.095. After that point MTWV for OOV queries increase for a while but the overall performance does not improve. Since STWV calculation only considers the truth value of the hit, STWV remains the same at $0.1501, 0.1510, 0.1477$ for all, IV and OOV queries for all values of γ. Although for each setup we can find an optimal γ, this value is not generalizable and since the optimal MTWV is usually close the STO normalized version ($\gamma = 1$), STO is preferred.

In constructing our query FST, we use minimum duration. Another possibility is using average duration of each phone where the durations are determined using the phonetic alignment of the training data by counting the consecutive frames labeled with each phone. Thus, we determine the number of repetitions of each vector in the query posteriorgram. Such queries necessitate longer matches with possibly the same symbol. However, a slightly noisy frame in the search data might be mapped into another symbol which prevents a possible match of the query. Therefore, average

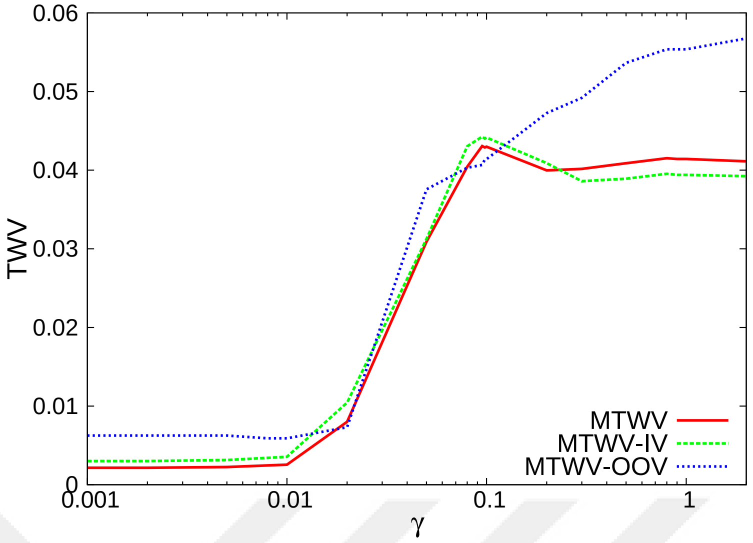


Figure 3.8. TWV versus γ in score normalization.

duration modeling reduces the detection rate. When we experiment with the duration, we get MTWV of 0.0 when we use average durations. That is why, minimum duration modeling is preferred in the experiments where we allow longer matches due to the loop structure in the query FST.

In the FSA representation of the query, multiword queries are handled by either forcing an arc with the silence symbol in between the words or the FSA is constructed such that the silence symbol is optional. In Table 3.6, 'Required' and 'Optional' correspond to these two methods. In this table, their effects on MTWV is shown for two setups which differ only in their CM normalization. Both of them use the supervised codebook and supervised CM and binary posteriorgrams are used in query representation. According to the table, gain comes from IV queries and there is not a significant change in MTWV over OOV queries. As there are higher number IV queries than OOVs, this improvement also reflected to the MTWV over all queries. Forcing silence requires a match index which has either silence or some symbol that is confusable with the silence symbol according to the CM. This situation restricts the search space and leads to less number of detections. If we make the word boundary information optional, we also allow direct transition from one word to the other which is a quite possible situation because consecutive words are rarely spoken in isolation. The advantage of optional silence is clearly seen in IV queries because most of the multiword queries are IV (128 out of 137 multiword queries) as given in Table 2.1.

Table 3.6. Effect of using an optional silence in multiword queries on MTWV.

Word boundary	CM	All	IV	OOV
Required	$P(o\|i)$	0.0414	0.0394	0.0554
Optional	$P(o\|i)$	0.0478	0.0524	0.0502
Required	$P(i\|o)$	0.0305	0.0224	0.0641
Optional	$P(i\|o)$	0.0472	0.0447	0.0638

In supervised CM generation, a threshold is applied in order to prune the CM and limit the number of possible confusions. If the threshold is high, there are few confusions which reduces the number of hits. On the other hand, if the threshold is low, there are many confusions which lead to higher number of false alarms along with many correct detections and since there are higher number of matches search time increases as we decrease the CM pruning threshold. Figure 3.9 shows how MTWV and STWV change as the CM threshold increases. Increase in STWV with the decreasing threshold shows that the correct detection rate increases. This also affects the MTWV which also increases with lower thresholds.

3.2.1.2. System Combination. In this subsection, KWS results of combined systems will be presented. In the combination, the baseline system is Kaldi Babel recipe [56] and in combination either all results of the proposed system or only the results for OOV keywords are combined with the baseline which is denoted by subscript OOV notation. For combination, we use CombMNZ method described in Section 2.3.3. The baseline system uses a word-level index generated from ASR lattices. OOV keywords are first converted to phonetic sequence using Sequitur G2P software [28] then a phonetic CM is applied which is generated by comparing the phonetic alignment and the

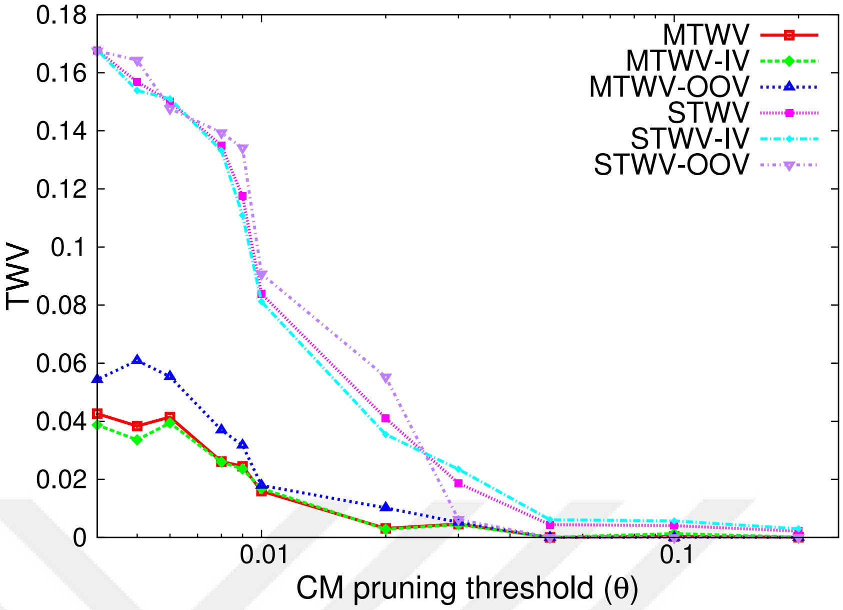


Figure 3.9. TWV versus the CM pruning threshold.

best path on the phonetic lattice. By converting OOV words to acoustically similar IV words as in Equation 2.5, KWS is performed on the word index. Finally, KST normalization is applied to the raw system output. Although the overall MTWV does not increase for most of the setups in Table 3.7, increase in STWV suggests that the symbolic index-based system introduces additional hits to the baseline but due to the mismatch between scores, this is not reflected to an increase in MTWV. As in the

Table 3.7. Performance of the combined system for the (un)supervised setup.

System	MTWV			OTWV			STWV		
	All	IV	OOV	All	IV	OOV	All	IV	OOV
B	0.3745	0.4500	0.1887	0.4157	0.4896	0.2240	0.6624	0.7567	0.4220
B + UU	0.3452	0.4260	0.1843	0.4654	0.5410	0.2727	0.6729	0.7606	0.4493
B + UUoov	0.3640	0.4500	0.1843	0.4285	0.4896	0.2727	0.6701	0.7567	0.4493
B + SU	0.3514	0.4222	0.1916	0.4649	0.5367	0.2815	0.6824	0.7595	0.4855
B + SUoov	0.3723	0.4500	0.1916	0.4310	0.4896	0.2815	0.6803	0.7567	0.4855
B + SS	0.3650	0.4321	**0.2184**	0.4796	0.5424	0.3195	0.6924	0.7704	0.4935
B + SSoov	**0.3786**	0.4500	**0.2184**	0.4417	0.4896	0.3195	0.6826	0.7567	0.4935

individual systems, the supervised setup (SS) that uses the phonetic posteriorgrams and a codebook with a codeword per phone achieves the best combined result. In this setup, there is 15.7% relative improvement in MTWV over OOV queries. Table 3.8 shows the combination of the supervised setups in which different VQ codebooks and posteriorgrams are used. The pairwise notation in the table denotes the posteriogram type and VQ type where $\mathrm{Ph}, \mathrm{St}, 3\mathrm{St}$ are used for phonetic units, CD states and states in the 3-state HMM of each phone. As in the individual results, the best performance is achieved in the Ph-Ph case.

Table 3.8. Performance of the combined system depending on the posteriorgram and codebook.

System	MTWV			STWV		
	All	IV	OOV	All	IV	OOV
B	0.3745	0.4500	0.1887	0.6624	0.7567	0.4220
B + Ph-Ph	0.3650	0.4321	0.2184	0.6924	0.7704	0.4935
B + Ph-Ph$_{\text{oov}}$	0.3786	0.4500	0.2184	0.6826	0.7567	0.4935
B + St-Ph	0.3417	0.4216	0.1622	0.6738	0.7582	0.4586
B + St-Ph$_{\text{oov}}$	0.3613	0.4500	0.1622	0.6727	0.7567	0.4586
B + St-3St	0.3646	0.4396	0.1820	0.6736	0.7588	0.4563
B + St-3St$_{\text{oov}}$	0.3704	0.4500	0.1820	0.6721	0.7567	0.4563

Table 3.9 shows the effect of CM generation method on combination. As in Table 3.3, columnwise normalization $P(i|o)$ leads to the best result by increasing the number of correct hits in the combination. In this case, 21.6% relative improvement is observed in MTWV over OOV queries as compared to the baseline.

Table 3.10 shows the effect of the number of paths taken after searching over the index. Although the individual MTWV performances are flat, combination of $n = 750$ improves the MTWV over OOV queries by 0.7% as compared to $n = 100$ case and 16.5% as compared to the baseline. Increasing n increases the STWV as in Table 3.4.

Table 3.9. Performance of the combined system depending on the CM normalization.

System	MTWV			STWV		
	All	IV	OOV	All	IV	OOV
B	0.3745	0.4500	0.1887	0.6624	0.7567	0.4220
B + $P(o\|i)$	0.3650	0.4321	0.2184	0.6924	0.7704	0.4935
B + $P(o\|i)_{\text{oov}}$	0.3786	0.4500	0.2184	0.6826	0.7567	0.4935
B + $P(i\|o)$	0.3697	0.4349	0.2295	0.6980	0.7724	0.5081
B + $P(i\|o)_{\text{oov}}$	**0.3823**	0.4500	**0.2295**	**0.6867**	0.7567	**0.5081**
B + $P(i,o)$	0.3744	0.4437	0.2075	0.6846	0.7688	0.4699
B + $P(i,o)_{\text{oov}}$	0.3799	0.4500	0.2075	0.6759	0.7567	0.4699
B + $C_{max}(i,o)$	0.3723	0.4409	0.2055	0.6853	0.7679	0.4748
B + $C_{max}(i,o)_{\text{oov}}$	0.3793	0.4500	0.2055	0.6773	0.7567	0.4748

Table 3.10. Performance of the combined system depending on n in n-best.

System	MTWV			STWV		
	All	IV	OOV	All	IV	OOV
B + 250	0.3635	0.4305	0.2180	0.6964	0.7747	0.4967
B + 250_{oov}	0.3794	0.4500	0.2180	0.6834	0.7567	0.4967
B + 500	0.3618	0.4272	0.2187	0.7025	0.7783	0.5092
B + 500_{oov}	0.3801	0.4500	0.2187	0.6870	0.7567	0.5092
B + 750	0.3615	0.4256	0.2198	0.7035	0.7787	0.5119
B + 750_{oov}	0.3799	0.4500	**0.2198**	0.6877	0.7567	0.5119
B + 1000	0.3611	0.4244	0.2195	0.7048	0.7787	0.5161
B + 1000_{oov}	0.3798	0.4500	0.2195	**0.6889**	0.7567	**0.5161**

Table 3.11 shows the effect of score normalization on combination. The notation in this table is same as Table 3.5 and a '+' in the third column denotes OOV-only combination. As in the individual performances shown in Table 3.5, changing α' can improve MTWV over OOV queries but in combination there is a decrease in MTWV for IV queries. Although the MTWV over OOV queries are always better than the baseline system, query length normalization results in the smallest improvement among the systems given in the table. In Table 3.11, STWV results are not included because in all cases the hit lists are the same and the difference is in score normalization. Since STWV is not based on the score, STWV are the same in each case. When all results from the symbolic index based KWS are combined with the baseline STWV over all, IV and OOV queries are 0.6922, 0.7702 and 0.4935, respectively. When only the results for OOV queries are combined, they are 0.6826, 0.7567 and 0.4935.

Table 3.11. Performance of the combined system depending on the score normalization.

System			MTWV		
Query length	KST (α')	Combine OOV	All	IV	OOV
+	-	-	0.3594	0.4291	0.2037
+	-	+	0.3637	0.4500	0.2037
+	1.5	-	0.3619	0.4302	0.1975
+	1.5	+	0.3656	0.4500	0.1975
-	1.5	-	0.3587	0.4260	0.2094
-	1.5	+	0.3769	0.4500	0.2094
-	2.0	-	0.3583	0.4265	0.2101
-	2.0	+	0.3748	0.4500	0.2101
-	0.2	-	0.3580	0.4229	0.2143
-	0.2	+	0.3773	0.4500	0.2143
-	0.02	-	0.3534	0.4200	0.2084
-	0.02	+	0.3719	0.4500	0.2084

As discussed previously, γ correction is applied to scores while doing STO normalization. As shown in Figure 3.8, for the supervised setup $\gamma = 0.095$ leads to the highest MTWV. When we combine those results with the baseline, we get the values shown in the last two lines of Table 3.12. Baseline results, and the combination of the supervised setup are also included in table which were reported in Table 3.7, previously. Although individual system performance in the $\gamma = 0.095$ case is better than the SS case, combination is not improved because γ value changes the range of the scores which might not in line with the range of scores in the baseline system. The score mismatch affects the combined scores and therefore the MTWV.

Table 3.12. Performance of the combined system when γ normalization is applied.

System	MTWV			STWV		
	All	IV	OOV	All	IV	OOV
B	0.3745	0.4500	0.1887	0.6624	0.7567	0.4220
B + SS	0.3650	0.4321	**0.2184**	0.6924	0.7704	0.4935
B + SS$_{\text{oov}}$	**0.3786**	0.4500	**0.2184**	0.6826	0.7567	0.4935
B + $\gamma = 0.095$	0.3556	0.4251	0.2041	0.6924	0.7704	0.4935
B + $\gamma = 0.095_{\text{oov}}$	0.3687	0.4500	0.2041	0.6826	0.7567	0.4935

Table 3.13 shows the KWS performance of the combination of individual results given in Table 3.6 and the baseline. As discussed previously, using an optional word boundary (O) improves the results for IV queries as compared to required word boundary (R) which is also observed from STWV-IV column of the table. If we compare the setups that use CM with $P(o|i)$ normalization denoted by ‘oi’ in the table, we can see that using optional silence leads to higher TWVs for all queries and IV queries. However, the same conclusion does not hold in the setups where CM with $P(i|o)$ normalization denoted by ‘io’ is used. In this case, the main improvement is in OOV queries as a result of the CM. Even in the individual results given in Table 3.6, the last two experiments have higher MTWV over OOV queries. As we are mainly interested in improving the OOV performance, these results supports the claim that CM is an

important block of the KWS system.

Table 3.13. Performance of the combined system depending on the word boundary representation.

System	MTWV			STWV		
	All	IV	OOV	All	IV	OOV
B + R-oi	0.3650	0.4321	0.2184	0.6924	0.7704	0.4935
B + R-oi$_{\text{oov}}$	0.3786	0.4500	0.2184	0.6826	0.7567	0.4935
B + O-oi	0.3657	0.4328	0.2194	0.6959	0.7753	0.4935
B + O-oi$_{\text{oov}}$	0.3788	0.4500	0.2194	0.6826	0.7567	0.4935
B + R-io	0.3697	0.4349	0.2295	0.6980	0.7724	0.5081
B + R-io$_{\text{oov}}$	0.3823	0.4500	0.2295	0.6867	0.7567	0.5081
B + O-io	0.3664	0.4305	0.2292	0.6980	0.7724	0.5081
B + O-io$_{\text{oov}}$	0.3822	0.4500	0.2292	0.6867	0.7567	0.5081

3.2.2. Experiments on the Swahili Dataset

In the base period of the Babel program, the systems were phonetic but in the following years, the amount of transcribed speech data decreased. Therefore, the systems became graphemic since obtaining phonetic pronunciation lexicons for low resource languages can be difficult.

The DNN from which our graphemic posteriorgrams are calculated is trained by the Babel team of IBM Research. The setup is developed using IBM Attila toolkit [58]. The network is trained using multilingual features as only 3 hours of transcribed data is available for the target language Swahili. In the very limited LP condition, text data from web is also exploited for lexicon expansion and language modeling to improve the ASR performance which also improves the KWS performance. Still, the ASR performance has WER of around 60%. For KWS, word-level lattices are used in indexing. IV queries are represented at word level and the grapheme level whereas OOV queries are only represented at the grapheme level. Query expansion by a CM is applied only to the graphemic FSTs of queries longer than three letters to prevent false alarms since the CM also models deletions. The queries are searched for in the index via WFST composition. Results are obtained in cascaded fashion, that is if any token results for a query are present, they are used; otherwise, the grapheme results are used. This system is used as our baseline in system combination experiments.

For our experiments, we get state-level posteriorgrams using the DNN described above and we calculated graphemic posteriorgrams using Equations 3.4 and 3.5.

The best results for Turkish are obtained by using the supervised setup with phonetic codebooks. In the graphemic Swahili system, a similar setup is used with graphemic codebook, i.e. using the graphemic alignment of the training data and its graphemic posteriorgram, an average vector is calculated. Due to words borrowed from English and French, the grapheme inventory of Swahili (`babel202b-v1.0d`) is larger than the alphabet of the language itself. For example, letters ‘x’ and ‘q’ are not used in the language but some words originating from English can contain them. Similarly, á and é are borrowed from French words. Therefore, we cannot encounter some of the graphemes in the training data and have codebook entries consisting of zeros. These vectors are taken to be binary where the 1 is located at the position corresponding to the grapheme. Using this codebook, the symbolic equivalent of the data and the index is generated. In the search phase of the graphemic system, we express the query as a sequence graphemes and generate the query WFST using the symbolic sequence of the query. Table 3.14 shows the MTWV calculated over all, IV and OOV queries in the supervised graphemic system depending on the score normalization parameter γ, STWV is 0.1968 in all cases. The best MTWV performances for all queries and OOV queries are obtained when $\gamma = 0.1$ and $\gamma = 0.3$, respectively. The CM in this experiment is also generated in a supervised manner with $P(o|i)$ normalization with a pruning threshold of 0.02.

These individual results of the symbolic index based KWS system are also combined a LVCSR lattice based baseline system (‘B’) described above which is developed

Table 3.14. MTWV for all, IV and OOV queries depending on γ in the supervised setup for Swahili.

γ	All	IV	OOV
0.1	0.0826	0.0847	0.0582
0.3	0.0802	0.0811	0.0746
1.0	0.0692	0.0698	0.0680

using the IBM Attila toolkit. As in the combinations for Turkish, two types of combinations are used. In the first case, both IV and OOV results are merged. In the second case denoted by $_{\text{oov}}$, only the OOV results of the proposed system is combined with the baseline.

Table 3.15 shows the combination results for Swahili. Here the baseline system performs well and improving its performance is a challenge. However, we do not make use of any additional information like web or multilingual data as in the baseline so the systems are not comparable. Here we show whether we can improve the baseline by introducing hits. When the supervised symbolic index based KWS setup is used to search for binary queries, and if we investigate the effect of score normalization parameter for $\gamma \in 0.1, 0.3, 1.0$, the increase in STWV for IV queries show that we can locate additional hits for IV queries but we cannot introduce new OOV hits to the baseline. Therefore, in the combination number of FAs increase and our hits may distort the scores of the overlapping hits in the baseline which hinders an improvement in the combined results. Although $\gamma = 0.3$ led to the best OOV performance for the symbolic KWS system alone as shown in Table 3.14, in combination it does not lead to the best performance. If only STO is applied ($\gamma = 1.0$), no improvement is observed even if we only merge our OOV results with the baseline.

Although the individual performance of the KWS system is close to the Turkish phonetic system, no improvement is observed in combination as the baseline is quite successful. Therefore, a change in query representation is made by using example

Table 3.15. Performance of the combined system for Swahili depending on γ when binary queries are used.

System	MTWV			OTWV			STWV		
	All	IV	OOV	All	IV	OOV	All	IV	OOV
B	**0.5701**	**0.5747**	**0.5210**	**0.6908**	**0.6932**	**0.6617**	0.8182	0.8174	**0.8270**
B + 0.1	0.5181	0.5226	0.4749	0.6328	0.6344	0.6124	0.8202	0.8197	0.8270
B + 0.1_{oov}	0.5627	0.5740	0.4749	0.6864	0.6924	0.6124	0.8172	0.8165	0.8270
B + 0.3	0.4908	0.4958	0.4476	0.6400	0.6418	0.6184	0.8202	0.8197	0.8270
B + 0.3_{oov}	0.5599	0.5740	0.4476	0.6869	0.6924	0.6184	0.8172	0.8165	0.8270
B + 1.0	0.5062	0.5121	0.4509	0.6614	0.6626	0.6464	0.8202	0.8197	0.8270
B + 1.0_{oov}	0.5641	0.5740	0.4509	0.6890	0.6924	0.6464	0.8172	0.8165	0.8270

posteriogram segments as our query posteriograms as if we are doing a QbyE search. Instead of using the graphemic sequence directly as our symbolic representation, we get example segments either from the training or the search data. In order to get the symbolic representation of the query, we quantize the example posteriorgram segment and find a WFST which is closest to our graphemic query WFST used before. Then we search for the expanded version of this query WFST in the index. The example posteriorgrams can be extracted either from training or search data. In the first case, we use the word level alignment of the training data to get the segments for IV queries. In the second case, we take the highest scoring hit of the baseline, which performs well, for each query and take the posteriorgram segment corresponding to the hypothesized file and the time interval. Then we generate the query WFST as described below.

A linear FSA that contains the symbols assigned to the frames of the example posteriorgram can be searched in the index but directly using this FSA would be too restrictive. Instead, a query WFST which is closer to our original query structure is obtained and then its expanded version is searched over the index. If Q denotes the query FSA that includes loops and imposes minimum duration, CM is the CM and O is the linear FSA generated using the observation symbols, i.e. the symbol string assigned to the example posteriorgram segment, an WFST ($\hat{Q}$) that is closer to Q and

O is calculated as follows

$$\hat{Q} = 1\text{-best}(Q \circ CM \circ O) \tag{3.10}$$

One point to note here is that, if we take our examples from the training data, O cannot always be obtained only using the confusions in the original CM. In that case, a CM which is generated from the same confusion matrix but with a lower pruning is used because the lower threshold allows higher number confusions and a WFST $\hat{Q}$ can be found. However, in expanding $\hat{Q}$, we still use the original CM.

Table 3.16 shows the KWS results for Swahili. In addition to the binary query case ('Bin'), QbyE results where examples are taken from training ('Train') or development ('Dev') data are also shown. In the QbyE systems, for the queries without any examples, we used hits from the binary representation which is denoted by 'Concatenation' in Table 3.16. In addition, we combined all results from QbyE and binary queries which is denoted by 'Merge' in the table. In the 'Merge' case, we take the overlapping hits into account and a query with example can have hits resulting from both QbyE and binary search results. We observe that when we get the examples from development data, QbyE performs better than the binary query case and when we combine the two, we achieve the highest performance. However, we do not see any improvement when our examples come from the training data.

When we combine the results given Table 3.16 with the baseline, we get TWVs shown in Table 3.17. Although QbyE works better in the individual system, we still do not observe an improvement when we combine our search results with the LVCSR lattice based baseline. As we observe from STWV results, we introduce additional correct hits for IV queries but the results are not reflected to the MTWV. We also observed that the performance with query examples taken from the training data is lower and if we look at the symbolic representation of the example posteriorgrams, the assigned symbols are frequently mapped to the noisy codebook entries which might result from the mismatch between the acoustic conditions between training and search data.

Table 3.16. MTWV and STWV for Swahili depending the on the query representation.

System	Combination	Source	MTWV			STWV
			All	IV	OOV	All
Bin	-	-	0.0692	0.0698	0.0680	0.1968
Bin + QbyE	Concatenation	Train	0.0507	0.0494	0.0680	0.1712
Bin + QbyE	Merge	Train	0.0636	0.0637	0.0680	0.2334
QbyE	-	Dev	0.2631	0.2619	0.2786	0.2730
Bin + QbyE	Concatenation	Dev	0.2669	0.2652	**0.2887**	0.2918
Bin + QbyE	Merge	Dev	**0.2733**	**0.2731**	0.2864	**0.3708**

Table 3.17. Performance of the combined systems for Swahili depending on query representation.

System	Comb	Source	MTWV			STWV		
			All	IV	OOV	All	IV	OOV
B	-	-	**0.5701**	**0.5747**	**0.5210**	0.8182	0.8174	**0.8270**
B + Bin	-	-	0.5062	0.5121	0.4509	0.8202	0.8197	0.8270
B + Bin$_{\text{oov}}$	-	-	0.5641	0.5740	0.4509	0.8172	0.8165	0.8270
B + QbyE	-	Dev	0.5653	0.5703	0.5147	0.8181	0.8173	0.8270
B + QbyE$_{\text{oov}}$	-	Dev	0.5692	0.5740	0.5147	0.8172	0.8165	0.8270
B + (QbyE + Bin)	Concat	Train	0.5027	0.5079	0.4509	0.8280	0.8281	0.8270
B + (QbyE + Bin)$_{\text{oov}}$	Concat	Train	0.5641	0.5747	0.4509	0.8172	0.8174	0.8270
B + (QbyE + Bin)	Merge	Train	0.4966	0.5012	0.4509	**0.8289**	**0.8291**	0.8270
B + (QbyE + Bin)$_{\text{oov}}$	Merge	Train	0.5641	0.5747	0.4509	0.8172	0.8174	0.8270
B + (QbyE + Bin)	Concat	Dev	0.5433	0.5481	0.4975	0.8186	0.8179	0.8270
B + (QbyE + Bin)$_{\text{oov}}$	Concat	Dev	0.5671	0.5747	0.4975	0.8172	0.8174	0.8270
B + (QbyE + Bin)	Merge	Dev	0.5197	0.5248	0.4634	0.8211	0.8206	0.8270
B + (QbyE + Bin)$_{\text{oov}}$	Merge	Dev	0.5652	0.5747	0.4634	0.8172	0.8174	0.8270

When we determine our query examples from the baseline system, the hits returned by the symbolic index based KWS will contain the example itself. If we look at the highest scoring hits of the proposed KWS system, they point to the locations where examples are taken except for five keywords. These queries are very short ones like “eh”, “ma”, “ana”, and they tend to occur frequently even within other words in the language. For the rest, the system returns the example as a hit with a different score than the baseline since the scoring mechanisms are different. In order to deal with score mismatch problem in combination, the hits are rescored using the score of the best hit in the baseline. If the best hit of the proposed method overlaps with that of the baseline, score of the baseline is taken as our score, otherwise our score remained as is. The rest of the hits of each keyword are normalized so that the total score of the hits of each query add up to 1. For normalization, either uniform scores are assigned to the rest, or the rest of the scores of a keyword are scaled to add up to 1 minus the best score for that query. Assuming that the scores of a query are ordered as $p_1 > p_2 > ... > p_{N_q}$ where N_q is the total number of hits, and let p'_1 be the new score of the highest scoring hit, then the rest of the hits, p'_m, will be determined according to Equation 3.12.

$$p'_m = \frac{1 - p'_1}{N_q - 1} \qquad \text{(Uniform rescoring)} \qquad (3.11)$$

$$p'_m = \frac{p_m(1 - p'_1)}{\sum_{m'=2}^{N_q} p_{m'}} \qquad \text{(Rescaling)} \qquad (3.12)$$

Tables 3.18 and 3.19 show the individual and combined results, respectively. In these tables, ‘Uniform’ and ‘Scaled’ denote uniform rescoring and rescaling of scores, respectively. In both tables, uniform scaling performs slightly better than rescaling but no improvement is observed in the combined results. Although the individual TWVs are lower than the STO-normalized scores for QbyE given in Table 3.16, there is about 1% relative improvement in the combined results.

Table 3.18. MTWV for Swahili QbyE system depending on the rescoring method.

Rescoring	All	IV	OOV
Uniform	0.2564	0.2551	0.2780
Scaled	0.2526	0.2509	0.2750

Table 3.19. Effect of rescoring QbyE results on system combination for Swahili.

System	MTWV			OTWV			STWV		
	All	IV	OOV	All	IV	OOV	All	IV	OOV
B + QbyE	0.5653	0.5703	0.5147	0.6897	0.6919	0.6617	0.8181	0.8173	0.8270
B + QbyE_{oov}	0.5692	0.5740	0.5147	0.6901	0.6924	0.6617	0.8172	0.8165	0.8270
B + Uniform	0.5541	0.5579	0.5198	0.6829	0.6847	0.6612	0.8181	0.8173	0.8270
B + $\text{Uniform}_{\text{oov}}$	0.5695	0.5740	0.5198	0.6901	0.6924	0.6612	0.8172	0.8165	0.8270
B + Scaled	0.5540	0.5579	0.5188	0.6812	0.6829	0.6603	0.8181	0.8173	0.8270
B + $\text{Scaled}_{\text{oov}}$	0.5694	0.5740	0.5188	0.6900	0.6924	0.6603	0.8172	0.8165	0.8270

4. DISCRIMINATIVE TRAINING OF THE CONFUSION MODEL

In this chapter, a discriminative training procedure for CM which aims at maximizing the KWS performance criterion, MTWV is introduced. The CM generated is then used in an LVSCR lattice based system where the CM is used to map OOV queries to IV words in order to overcome the OOV problem in KWS. Some of the results of these experiments can also be found in [59]. Then, the discriminative training approach is applied to the CM of the symbolic index based system introduced in Chapter 3. As the symbolic index has a linear string form instead of lattices, simplifications will occur in the update equations of the training procedure. The experiments for the simplified version are performed on the posteriorgram based KWS system that uses the supervised setup on the Turkish dataset. The results are also combined with the LVCSR lattice based baseline system to show that we can improve the results by changing the weights of the CM.

If the keyword is represented as WFSA, its expanded version is obtained by composing the query WFSA with CM from right as in Equation 2.4. Then, the search results obtained for the expanded query are taken as the results for the original query.

In the phonetic system, the query q is first converted into a sequence of phonemes, the query WFSA is generated, the CM is applied and then the confused sequence is converted back to words so that we can search for them in word-based index as given in Equation 2.5. Let r denote the confused version (proxy) of q after applying the CM and converting back to word by following path π_{qr} in the composition of query WFSA and the CM. Then the scores s_h^q for hypothesis h resulting from searching for r, which is in log-domain, is the combination of the confusion weight w_{qr} from the path π_{qr} and the weight coming from the index n_h^q for the hypothesis h, which is equivalent to the log-probability of detecting the word r in the index, as given in Equation 4.1

$$s_h^q = -\log(p_h^q) = w_{qr} + n_h^q \tag{4.1}$$

However, if q has multiple pronunciations, then the r term can be obtained in multiple ways by using different confusions for different pronunciations and following different paths $\pi_{qr'}$ for the same query. In that case, w_{qr} takes all the contributions from $\pi_{qr'}$ as explained in Appendix A.1.

4.1. Discriminative Training of the CM for LVSCR Lattice Based KWS

CM does not only allow searching for alternatives but also affects the KWS system performance because the weight on the CM contributes to the overall search scores along with the weights coming from the index WFST as described above. Therefore, the aim of training a CM is to learn the weights on the arcs of the CM so that KWS performance can increase by changing the search scores.

In KWS task at hand, our performance criterion is TWV as described in Section 2.3.1. Therefore, in discriminative training of the CM we try to maximize the TWV, directly. In [5], similar ideas are followed to maximize the figure-of-merit (FOM), which is the detection rate averaged over the range of 0 to 10 false alarms per query per hour, instead of the TWV.

Let $\overline{p}_h^q$ denote the STO normalized score for hypothesis h of the query q, then TWV can be written as

$$\mathrm{TWV} = \frac{1}{|\mathcal{Q}|}\sum_{\mathrm{q}\in\mathcal{Q}}\sum_{h\in H_q} I(\overline{p}_h^q \geq \theta)\left\{\frac{I(h\in H_q^+)}{\mathrm{N}_{\mathrm{true}}(q)} - \beta\frac{I(h\in H_q^-)}{\mathrm{K}-\mathrm{N}_{\mathrm{true}}(q)}\right\} \tag{4.2}$$

where $I(.)$ denotes the indicator function which becomes 1 when its argument evaluates to true and it is 0 otherwise, H_q denotes the set of hypotheses for query q and finally H_q^+

and H_q^- respectively indicate the subsets of H_q that are located correctly and incorrectly with respect to the ground truth and only consider the time span disregarding the decision score.

If gradient ascent method is chosen for the weight learning during the discriminative training, the CM weight corresponding to the substitution of input label i with output label o, namely $w(i,o)$ can be updated using a step size η_w as follows:

$$w(i,o) \longleftarrow w(i,o) + \eta_w \nabla w(i,o) \text{ where } \nabla w(i,o) = \frac{\partial \mathrm{TWV}}{\partial w(i,o)} \tag{4.3}$$

According to Equation 4.3, the amount of update requires the calculation of the derivative of the objective function, TWV, with respect to $w(i,o)$. Since only $\overline{p}_h^q$ terms depend on the weights, in order to calculate $\frac{\partial \mathrm{TWV}}{\partial w(i,o)}$, only the derivative of the $I(\overline{p}_h^q \geq \theta)$ in Equation 4.2 is required. However, the indicator function is not differentiable. One possible solution is to use an approximation to the $I(.)$ function as in [5] where the indicator functions are approximated by a sigmoid function. The sigmoid function $\sigma()$ given in Equation 4.4 has two parameters α and θ. The steepness of the sigmoid function is determined by α, the larger it becomes, the better $\sigma()$ approximates the indicator. The θ parameter acts as the threshold in the indicator function $I(\overline{p}_h^q \geq \theta)$.

$$I(\overline{p}_h^q \geq \theta) \approx \sigma(\overline{p}_h^q; \alpha, \theta) = \frac{1}{1 + \exp\left(-\alpha(\overline{p}_h^q - \theta)\right)} \tag{4.4}$$

Using the sigmoid approximation, the approximate TWV can be written as

$$\mathrm{TWV} \approx \frac{1}{|\mathcal{Q}|} \sum_{\mathrm{q} \in \mathcal{Q}} \sum_{h \in H_q} \sigma(\overline{p}_h^q; \alpha, \theta) \left\{ \frac{I(h \in H_q^+)}{\mathrm{N_{true}}(q)} - \beta \frac{I(h \in H_q^-)}{\mathrm{K} - \mathrm{N_{true}}(q)} \right\} \tag{4.5}$$

Then instead of calculating $\frac{\partial \mathrm{TWV}}{\partial w(i,o)}$, we can take the partial derivative of smooth approximation of the TWV given in the above equation. For notational simplicity let

$$\gamma(q,h) = \left\{ \frac{I(h \in H_q^+)}{\mathrm{N_{true}}(q)} - \beta \frac{I(h \in H_q^-)}{\mathrm{K} - \mathrm{N_{true}}(q)} \right\} \tag{4.6}$$

After substituting the expressions into the definitions and following steps given in Appendix A.1, the derivative of the approximate TWV can be found as

$$\frac{\partial \text{TWV}}{\partial w(i,o)} \approx \frac{1}{|\mathcal{Q}|} \sum_{\mathrm{q} \in \mathcal{Q}} \sum_{h} \alpha \sigma(\overline{p}_h^q; \alpha, \theta)(1 - \sigma(\overline{p}_h^q; \alpha, \theta)) \cdot \left[\sum_{(q,r)} \frac{Z_r}{Z_q} \big(\overline{p}_h^q - \overline{p}_h^r I(r \in R_h) \big) \sum_{r' \in r} e^{-w_{qr'}} c_{qr'}(i,o) \right] \gamma(q,h) \quad (4.7)$$

In Eq. 4.7, $c_{qr'}(i,o)$ denotes the number of occurrences of weight $w(i,o)$ in the path $\pi_{qr'}$, Z_q and Z_r denote the normalization factors during STO for q and r terms, respectively, r' denotes r terms that are obtained via different paths due to multiple pronunciations and $w_{qr'}$ term is the weight of this path.

In order to learn the weights of the CM, we have to know the paths from a query term q to its confused version r. To get these paths, which are sequences of input-output phone labels along with the associated weight, the arcs of the composition of query WFSA and the CM are traversed and the information of visited paths are stored during query processing. The list of r terms corresponding to each q are obtained using the $i : o$ labels of the phones while traversing a path starting from a query q. The crucial aim of this step is to obtain the count of each weight $w(i,o)$ on each path from q to r, i.e. finding $c_{qr}(i,o)$ terms in Equation 4.7. Then the r terms are searched over the index without applying the CM to obtain the contribution of the index to the search score of an h resulting from r. The search scores of these terms constitute the part of the scores for the original queries that come from the index only. Since STO normalization is applied to scores, in the original pipeline we have the normalized scores. However, the normalization factors should be known for the optimization problem as shown in Equation 4.7. Therefore, we keep these normalization factors while obtaining the normalized scores. Once the normalized scores, their normalization factors, counts of the weights on paths from q to r are calculated and the ground truth for the hypotheses are obtained from the alignment file of the scoring system, one can proceed to the optimization step (Equations 4.3 and 4.7). In addition, the optimization step requires properly chosen parameters α and θ for the sigmoid approximation and a step size η_w.

After updating the CM, the new CM should be used in the KWS pipeline and new search scores should be obtained to continue to CM learning with the updated scores. However, re-searching the index will be time consuming and it will increase the time complexity of discriminative training. Therefore, assuming that the r terms corresponding to a query q, and therefore paths $\pi_{q,r}$ and weight counts (c_{qr}) leading to r terms remain the same, we first update $w(i, o)$ using Equation 4.3, then search scores and their normalization factors are updated according to the weights obtained in the previous step and we continue to optimization using the updated scores obtained from Equations. 4.8-4.10:

$$p_h^q \longleftarrow \sum_r \exp\left(-\sum_{m=1}^{\pi_{qr}} w_{qr}(i_m, o_m) - n_{hr}\right) \tag{4.8}$$

$$Z_q \longleftarrow \sum_{h \in H_q} p_h^q \tag{4.9}$$

$$\overline{p}_h^q \longleftarrow \frac{p_h^q}{Z_q} \tag{4.10}$$

After proceeding certain number of iterations, we get the final CM weights and apply it to the original KWS system. In the following section, results will presented.

4.1.1. Experiments

In the experiments, Turkish limited LP database is used where we searched over 10-hour development data. The LVSCR system, lattice generation and indexation is based on IBM Attila speech recognition toolkit [58]. The details of the baseline system can be found in [4, 60]. The baseline LVSCR system is based on DNN with CD HMM state targets and the DNN has five layers. The word-level index is generated from the lattices and the IV and OOV queries are handled separately in the KWS setup. IV queries are directly searched within the word-level index whereas OOV queries are first mapped into IV words by applying G2P conversion, applying a phonetic CM and then mapping back to words. Then, this query WFST is searched within the word level index. Then the IV and OOV results are concatenated.

For WFST based operations like obtaining the list of r terms corresponding to query q and extracting the (i, o) label pair counts by traversing the WFST paths from q to r, OpenFST toolkit [23] is used. Then these components are embedded into the baseline setup described above to extract the sufficient information for discriminative training of the CM.

In the experiments, the effects of the steepness parameter α and threshold θ of the sigmoid approximation is observed. Since we try to maximize the MTWV, the threshold giving MTWV has to be calculated. However, depending on α, the maximum of approximated TWV differs. Moreover, if STO normalized scores are used during TWV calculation all scores will be between 0 and 1, then optimal threshold above 1 will not be meaningful since all scores will be below the threshold. Therefore, proper choice of α is required. Figure 4.1 shows how the sigmoid approximation of TWV changes as α and θ changes. According to this figure, $\alpha > 10$ so that we can have an optimal θ for MTWV calculation.

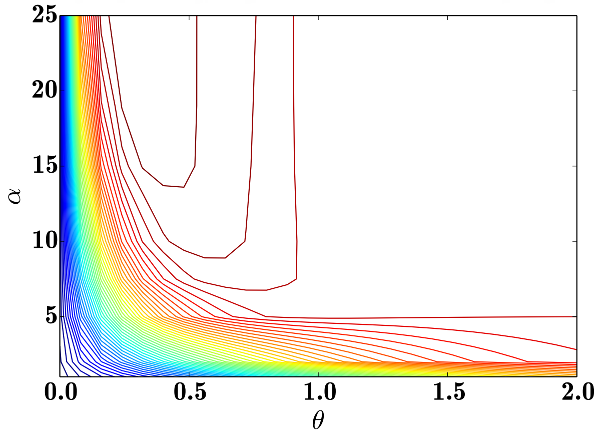


Figure 4.1. Effect of α and θ on sigmoid approximation to TWV.

Since our aim is to improve the results for OOV queries, MTWV is not only reported for all queries but also for OOV queries only. As shown in Table 4.1, if we do not use any CM, we get 0.0 MTWV for OOVs because we cannot find any hit for them. However, even if we use a random CM, we can achieve positive MTWV for OOV queries. In this random CM, the cost of confusing a symbol with itself is set to 0 as this is not a confusion indeed, and the costs, which are negative log-probabilities, of the rest of the confusions of that symbol are normalized such that their total probability is fixed to a certain value, for example 0.2 in our case.

Table 4.1. MTWV for the cases where no CM or a randomly initialized CM is used.

	All	**OOV**
No CM	0.3285	0.0000
Random CM	0.3311	0.0266

During training, we either use the CM updated after each pass over the hit list in the original KWS setup in order to get the updated scores $\overline{p}_h^q$ and the normalization factors Z_q, or we use Equations 4.8-4.10 to update the scores a few iterations without searching over the index again. Table 4.2 shows these two cases and the relative improvement in MTWV as compared to using the initial random CM. According to this table, the improvement is more pronounced for OOV queries ($\sim 85\%$) than for all queries ($\sim 1.7\%$). Although the gain for OOVs is smaller in “Upd2 + Search” case, the overall MTWV is higher. Moreover, we do not search over the index again at each step and thus the training takes shorter than “Upd1 + Search”. Since the performances are comparable, “Upd2 + Search” can be preferred over the first case. In general, these results show that query expansion by a CM is an effective way of handling OOV queries which is also a conclusion of [29].

Table 4.2. Relative increase in MTWV when the CM is updated at each step and updated after 2 steps.

	All	**Increase (%)**	**OOV**	**Increase (%)**
Random CM	0.3311	-	0.0266	-
Upd1 + Search	0.3341	1.7	0.0493	85.3
Upd2 + Search	0.3350	2.0	0.0482	81.2

One point to note in these experiments is that the stopping point for the algorithm. In the above experiments, we observed that when we have a large number of iterations, TWV starts to decrease. Therefore, we stop the iterations at the point where we encounter a decrease in the approximated TWV.

A wide variety of confusions are allowed in this random initial CM. However, reducing the number of confusions by pruning the CM WFST can help increasing the CM because as long as we do not lose any correct hits, we can reduce the number of false alarms resulting from too much confusion which in turn increases TWV given in Equation 2.6.

When there is large amount of training data for which transcriptions and phonetic alignments are available, we can generate a CM by counting the number of substitutions between the reference transcriptions and the hypotheses generated from phonetic recognizers. Which can be called as maximum-likelihood CM (ML-CM). Then the ML-CM can be used as the initial model for CM training instead of the random one. When we use such a CM as our initial CM and take $\alpha = 10$, a step size η_w changing with time, i.e. $\eta_w \sim \eta_0 t^{-1}$ where t corresponds to the iteration count, the change in the approximated TWV is shown in Figure 4.2. Here KWS over the index is done once at the beginning and the weights are updated according to Equations 4.8-4.10 during iterations.

Although the training procedure with ML-CM as its initial model shows an improvement here, we do not observe a significant improvement if we use the trained CM in the baseline KWS setup. However, according to results discussed previously, even if we do not have ML-CM or if we cannot find a reliable estimate for it due to low-resource conditions, we can learn the weights of the CM starting from a randomly initialized CM in which case we can get approximately 2% relative increase in MTWV over all queries.

4.2. Discriminative Training of the CM for Symbolic Index Based KWS

In this section, discriminative training of the CM will be applied to the CM used in the symbolic index based KWS system. Due to the index structure, the weight

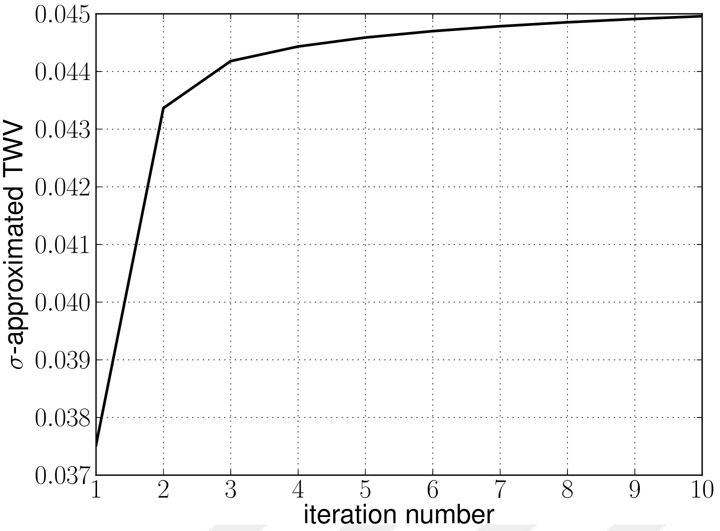


Figure 4.2. Increase in the approximated TWV during iterations for the LVCSR based setup.

update equations will become simpler leading to more interpretable equations.

For discriminative training of the CM, one of the statistics that we have to extract during search is the count of weights on the FST paths from a query q to its proxy version r. Since there is a loop structure in the query WFST in the symbolic index based system, we lose duration information during search. In order to get the path counts $c_{qr}(i, o)$ for each input-output label pair (i, o), the symbolic sequence corresponding to the hits should be investigated after search. As in the QbyE experiments of Section 3.2.2, a sequence that is close to both the query FST and the symbol sequence of the hit is found by Equation 3.10. Then, the number of (i, o) pairs used to get the hit of a query, and the score of this hit is found. In this case, there will be a single r term corresponding to each hit and the update equations will become simpler. In addition, there are not any weights in the index so searching for a proxy r term over the index will have no cost ($n_{hr} = 0$) and the score of a hit due to r will be $p^q_{hr} = \exp(-w_{qr})$. By making simplifications as shown in Appendix A.3, we get the derivatives of the scores and the sigmoid approximation of TWV as in the following equations.

$$\frac{\partial \overline{p}_h^q}{\partial w(i,o)} = \overline{p}_h^q \Big(- c_{qr}(i,o) + \sum_{h' \in H_q} \overline{p}_{h'}^q c_{qr'}(i,o) \Big) \tag{4.11}$$

By substituting $\frac{\partial \overline{p}_h^q}{\partial w(i,o)}$ into the expression for $\frac{\partial \mathrm{TWV}}{\partial w(i,o)}$ we get $\nabla w(i,o) = \frac{\partial \mathrm{TWV}}{\partial w(i,o)}$ which is then used in the weight update equation 4.3.

$$\begin{aligned} \frac{\partial \mathrm{TWV}}{\partial w(i,o)} =& \frac{1}{|\mathcal{Q}|} \sum_{\mathrm{q} \in \mathcal{Q}} \sum_{h \in H_q} \alpha \sigma(\overline{p}_h^q, \theta)(1 - \sigma(\overline{p}_h^q, \theta)). \\ & \overline{p}_h^q \Big(- c_{qr}(i,o) + \sum_{h' \in H_q} \overline{p}_{h'}^q c_{qr'}(i,o) \Big) \gamma(q,h) \end{aligned} \tag{4.12}$$

The second term in Equation 4.11 can be interpreted as the expected count of the (i,o) pair, $\mathrm{E}[c_{qr}(i,o)]$, over the hits of query q. Thus, the amount of update will be proportional to the difference between the expected count and the count on the path leading to the particular hit. Then the difference will be scaled by $\gamma(q,h)$ which is positive if the hit corresponds to a correct location according to the ground truth. Therefore, if a hit is correct and an (i,o) pair is encountered more than the expected count for that query, its cost $w(i,o)$ will be reduced which is desirable since this confusion has a positive effect on the search results for q.

4.2.1. Experiments

In the symbolic index based KWS system, the CM is applied at the frame level, so the counts are higher than the case where we apply the CM at the phonetic level and convert it bask to words as described in Chapter 4. In addition, we have a limited number of queries to train and test the queries. Therefore, using the same set of keywords for both training and testing the CM can lead to overfitting. Therefore, the experiments are cross-validated. 5x5-cross validation is applied where the OOV keywords are split into 5-folds one of which is reserved for testing and the rest for training. This is repeated for 5 times. After training the CMs, they are used in the symbolic index based KWS setup. Initially, a CM is trained with all of the OOV keywords without cross-validation and the final CM is used for expanding both the IV and OOV queries. In the cross-validation experiments, the CM trained with 4 splits and is tested on the 5th set of keywords. By resampling the set of keywords, this process is repeated 5 times.

Figure 4.3 shows how the sigmoid approximation of the TWV changes during training. Here each iteration correspond to one pass over the whole training keyword list.

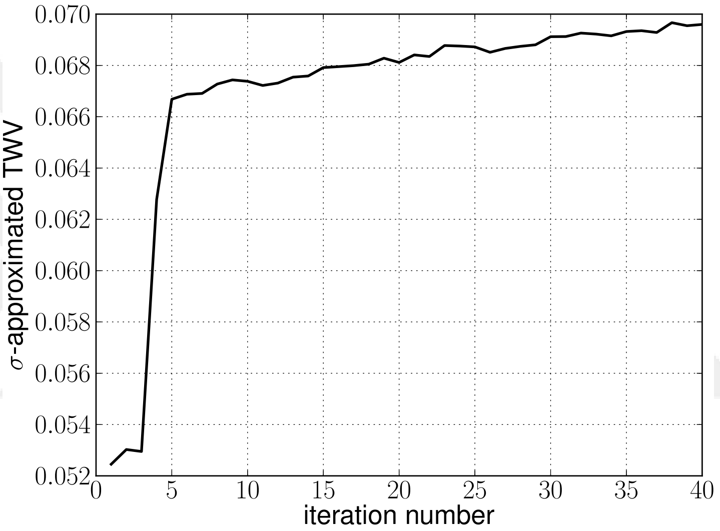


Figure 4.3. Increase in the approximated TWV during iterations for the symbolic index based setup.

Tables 4.3 and 4.4 show the individual and merged system performances, respectively. The experiments are performed on the Babel Turkish Limited LP dataset. Initial CM is taken from the supervised setup where $P(o|i)$ normalization is applied to the confusion matrix. For discriminative training of the CM, α is chosen as 20, step size is taken as 0.9 and stochastic gradient descent (SGD) [61] is used during the updates, i.e. the weights are updated after processing each query. In these tables, 'CM_0' denotes the performance when the initial CM is used, 'CV-Avg' denotes the average performance of different trials in cross-validation and 'CV-AvgCM' shows the case where the CM which is calculated by taking the average weights of the CMs for each (i, o) pair learned during cross-validation. In addition, 'NoCV' show the results when OOV queries are used in training and the CM tested on both IV and OOV queries. In the 'CV-Avg' experiment, CM_0 is applied to IV queries as we have five different CMs that are tested on different subsets of OOV queries. In the last three rows of Table 4.3, increase in overall MTWV arises from the improvement in MTWV for OOVs. In the table, cross validated result CV-Avg leads to 12.6% relative improvement in MTWV for OOV queries as compared to the performance of CM_0. Moreover, 17.3 and 18.6% relative improvement and up to 1% absolute improvement in MTWV over OOV queries is observed in the last two rows. In all cases, STWV over all, IV and OOV queries are 0.1501, 0.1510 and 0.1477, except MTWV over all queries for NoCV case which is 0.1494. NoCV row of the table can be treated as an upper bound for MTWV over OOV queries since the CM is both trained and tested on the same OOV queries which gives 18.6% relative improvement for OOV queries. When we apply an average model ('CV-AvgCM'), we also observe an increase in MTWV for IV queries which suggests a positive effect of regularization on learning.

Table 4.3. Individual results of discriminative CM training in the symbolic index based system for Turkish.

	MTWV		
System	All	IV	OOV
CM_0	0.0414	0.0394	0.0554
CV-Avg	0.0436	0.0394	0.0624
CV-AvgCM	0.0456	0.0401	0.0640
NoCV	0.0453	0.0399	0.0657

In combination experiments, Kaldi Babel recipe is used as the baseline system. The subscript 'oov' notation is used for system combinations where only the OOV results are combined with the baseline. By training the initial CM, we improved the MTWV performance over OOV queries. Although different hits are not introduced as seen from the STWV columns in Table 4.4 which are almost constant, changing the CM weights affects the scores of the hits and thus the combined scores result in a higher MTWV performance. In the cross validation experiments if we combine the result of each trial and take the average performance as shown in 'B + CV-Avg$_{\mathrm{oov}}$' where we achieve 1.1% and 16.3% relative improvement for all and OOV queries, respectively, as compared to the baseline. The main improvement as compared to CM$_0$ comes from OOV queries where we get 0.5% relative improvement. As in the case of individual results, using an average CM gave the best combination performance where we achieved 16.5% and 0.6% relative improvement in MTWV over OOV queries as compared to the baseline and the initial CM, respectively. This result suggests that adding a regularization term in our update equation can lead to a better performance.

Table 4.4. Combined results of discriminative CM training in the symbolic index based system for Turkish.

System	MTWV			STWV		
	All	IV	OOV	All	IV	OOV
B	0.3745	0.4500	0.1887	0.6624	0.7567	0.4220
B + CM$_0$	0.3650	0.4321	0.2184	0.6924	0.7704	0.4935
B + CM$_{0,\mathrm{oov}}$	0.3786	0.4500	0.2184	0.6826	0.7567	0.4935
B + CV-Avg	0.3648	0.4321	0.2195	0.6924	0.7704	0.4935
B + CV-Avg$_{\mathrm{oov}}$	0.3787	0.4500	0.2195	0.6826	0.7567	0.4935
B + CV-AvgCM	0.3613	0.4265	0.2198	0.6919	0.7697	0.4935
B + CV-AvgCM$_{\mathrm{oov}}$	0.3792	0.4500	0.2198	0.6826	0.7567	0.4935
B + NoCV	0.3600	0.4260	0.2198	0.6919	0.7697	0.4935
B + NoCV$_{\mathrm{oov}}$	0.3781	0.4500	0.2198	0.6826	0.7567	0.4935

5. CONCLUSIONS AND FUTURE DIRECTIONS

In this thesis, first a symbolic index based KWS system which uses posteriorgam representation of the audio is introduced. Secondly, for improving the KWS performance for OOV queries, a discriminative approach is developed that directly tries to maximize the KWS performance measure. We showed the effectiveness of our proposed methods on two different languages, Turkish and Swahili, from the IARPA Babel program. For the discriminative training approach, experiments are performed both on an existing LVSCR based KWS system and on the symbolic index based KWS system introduced in this thesis. The conclusions and future directions for research will be summarized in the following sections.

5.1. Conclusions

In Chapter 3, an index based KWS systems based on a symbolic representation of the data is introduced. In this system where the index has no scores, the way we obtain the CM affects the performance more than the way we get the symbolic representation of the data since the CM will also account for possible encoding errors. We found that both the confusion pairs and their weights affect the performance. The former allows us to locate additional hits whereas the latter changes the scores of the hits, thus they contribute to the increase in MTWV. We concluded that the size of quantization codebook should be comparable to the number of classes in the posteriorgram. A minor improvement is observed when we model multi-word queries with an optional silence. We also applied different score normalization techniques. Depending on the normalization type we could improve the KWS performance but the gains as compared to the STO normalization is limited. It might be concluded that STO normalization is preferable both in the individual results and for the systems that will be used in combination as STO does not require an additional parameter to be tuned.

Although our individual KWS performance is not comparable to the baseline systems especially for IV words, when we combine our results with the baseline sytems, we observed improvement in MTWV for OOV queries, especially. Since one of the problems in LVCSR based systems in the low resource language setting is handling OOV queries, the system introduced here contributes to the KWS systems. This improvement also corroborates the observation that developing diverse and complementary systems is a way of improving the performance of a KWS system.

As mentioned before, expanding keywords by a CM is an important aspect of the KWS system. The experimental results also supported the fact that we can improve the KWS performance especially for OOV keywords. We observed that by changing the weights of the CM via a discriminative approach as shown in Chapter 4, we can improve the MTWV performance of the system by changing the scores of the hits.

5.2. Future Directions

The symbolic index generated in this study assigns a single string to each utterance, therefore it might resemble the LVSCR based KWS systems which use the one-best hypothesis. One way to extend this study is generating a lattice-like structure from symbols assigned to the posteriorgram frames. Another direction can be applying a variable length encoding scheme to the vectors, thus temporal properties can be incorporated into the symbols and instead of frame-level labeling of the data we can use segment-level labeling. By assigning alternative symbols to each segment, a CN like structure can be obtained to represent alternative hypotheses.

The CM in the symbolic index based system is applied at frame level which leads to low raw scores in the symbolic index based KWS system. Therefore, alternative query representations can be found by omitting the cost of the repetition of same confusion one after the other. Another possibility is to use this idea in rescoring the hits by backtracking the matched query and finding the path from the original query to the matched one similar to the way we obtain symbolic query in the QbyE-like approach applied for the Swahili dataset. Then using the cost of sequentially repeated confusions once, we can estimate a new score for each hit. However, the observed sequence can itself be noisy and can necessitate the use of a smoothing method for the observed sequence.

A possible direction for the discriminative training of a CM is to automatically learn the confusion pairs along with their confusion weights by adding or removing confusion pairs. In addition, insertions and deletions can be incorporated into the model, this might be helpful especially when the CM is applied at frame level, however this might also result in an increase in search time.

As summarized in the experiments, different normalizations are applied to the hit lists. Most of the time, improvement in the individual system did not lead to an improvement in system combinations at the same rate. Therefore, we might find a system combination scheme that allows efficient combination of systems with different score ranges and/or distributions.

APPENDIX A: DERIVATION OF WEIGHT UPDATES

In this Appendix, the details of the steps in deriving the update equations for the CM weights will presented. The focus will be on deriving the expression given in Equation 4.7 which determines the change in weights as shown in Equation 4.3.

A.1. Deriving the Update Equation

TWV is defined as [47]

$$\mathrm{TWV}(\theta) = 1 - \frac{1}{|\mathcal{Q}|}\sum_{\mathrm{q}\in\mathcal{Q}} P_{\mathrm{miss}}(q,\theta) + \beta P_{\mathrm{FA}}(q,\theta) \tag{A.1}$$

$$\mathrm{TWV}(\theta) = 1 - \frac{1}{|\mathcal{Q}|}\sum_{\mathrm{q}\in\mathcal{Q}} \left\{ 1 - \frac{\mathrm{N}_{\mathrm{cor}}(q,\theta)}{\mathrm{N}_{\mathrm{true}}(q)} + \beta\frac{\mathrm{N}_{\mathrm{spur}}(q,\theta)}{\mathrm{N}_{\mathrm{NT}}(q)} \right\} \tag{A.2}$$

where

$$\mathrm{N}_{\mathrm{cor}}(q,\theta) = \sum_{h\in H_q^+} I(p_h^q \geq \theta) \qquad\qquad \mathrm{N}_{\mathrm{spur}}(q,\theta) = \sum_{h\in H_q^-} I(p_h^q \geq \theta)$$

In the system, each query term is composed with the CM which leads to a set of new terms, r, and some of the scores resulting from searching these new terms r are merged (summed). Therefore, a merged score, p_h^q, for h-th occurrence of a query term q can be written as

$$p_h^q = \sum_{r\in R_h} p_{hr}^q = \sum_{r\in R_h} \exp(-\mathrm{log} s_{hr}^q) = \sum_{r\in R_h} \exp(-w_{qr} - n_{hr}) \tag{A.3}$$

where w_{qr}, Equation (A.4), is the sum of the log-weights in the confusion path from term q to r, n_{hr} is the log-probability of detecting r in the index and R_h is the set of

r terms that lead to the search hypothesis h.

$$w_{qr} = \sum_{m=1}^{|\pi_{qr}|} w_{qr}(i_m, o_m) \tag{A.4}$$

In Equation (A.4), m and (i_m, o_m), respectively, denote the arc indices and the input-output label pairs of the arcs of the confusion path π_{qr} from term q to r. Then the merged score can be rewritten as

$$p_h^q = \sum_{r \in R_h} \exp\left(- \sum_{m=1}^{|\pi_{qr}|} w_{qr}(i_m, o_m) - n_{hr} \right) \tag{A.5}$$

Moreover, normalized scores $\overline{p}_h^q$ are used instead of the raw scores p_h^q in detection. For STO normalization, $\overline{p}_h^q$ can be written as follows:

$$\overline{p}_h^q = \frac{p_h^q}{Z_q} = \frac{p_h^q}{\sum_{h' \in H_q} p_{h'}^q} = \frac{\sum_{r \in R_h} p_{hr}^q}{\sum_{h' \in H_q} \sum_{r'} p_{h'r'}^q} \tag{A.6}$$

where

$$p_{hr}^q = \exp\big(- w_{qr} - n_{hr} \big) = \exp\big(- \sum_{m=1}^{|\pi_{qr}|} w_{qr}(i_m, o_m) - n_{hr} \big)$$

By using these normalized scores, TWV can be written as

$$\begin{aligned}
\text{TWV} &= 1 - \frac{1}{|\mathcal{Q}|} \sum_{\mathrm{q} \in \mathcal{Q}} \left\{ 1 - \frac{\sum_{h \in H_q^+} I(\overline{p}_h^q \geq \theta)}{\mathrm{N}_{\text{true}}(q)} + \beta \frac{\sum_{h \in H_q^-} I(\overline{p}_h^q \geq \theta)}{\mathrm{K} - \mathrm{N}_{\text{true}}(q)} \right\} \\
&= \frac{1}{|\mathcal{Q}|} \sum_{\mathrm{q} \in \mathcal{Q}} \left\{ \frac{\sum_{h \in H_q^+} I(\overline{p}_h^q \geq \theta)}{\mathrm{N}_{\text{true}}(q)} - \beta \frac{\sum_{h \in H_q^-} I(\overline{p}_h^q \geq \theta)}{\mathrm{K} - \mathrm{N}_{\text{true}}(q)} \right\} \\
&= \frac{1}{|\mathcal{Q}|} \sum_{\mathrm{q} \in \mathcal{Q}} \sum_{h \in H_q} \left\{ I(\overline{p}_h^q \geq \theta) \left\{ \frac{I(h \in H_q^+)}{\mathrm{N}_{\text{true}}(q)} - \beta \frac{I(h \in H_q^-)}{\mathrm{K} - \mathrm{N}_{\text{true}}(q)} \right\} \right\}
\end{aligned} \tag{A.7}$$

Our aim is to maximize Equation (A.7) with respect to each weight $w(i, o)$ with input and output labels i and o, respectively.

For the optimization of the weights $w(i, o)$, it is required to differentiate Eq. (A.7). Since the indicator function $I(.)$ is not a differentiable function, we can approximate it with a sigmoid as in [5]

$$I(\overline{p}_h^q \geq \theta) \approx \sigma(\overline{p}_h^q, \theta, \alpha) = \frac{1}{1 + \exp\left(-\alpha(\overline{p}_h^q - \theta)\right)} \tag{A.8}$$

For notational simplicity, the shape parameter α of the sigmoid function will be omitted in the sequel and we will let

$$\gamma(q, h) = \frac{I(h \in H_q^+)}{\mathrm{N}_{\mathrm{true}}(q)} - \beta \frac{I(h \in H_q^-)}{\mathrm{K} - \mathrm{N}_{\mathrm{true}}(q)} \tag{A.9}$$

Then Equation (A.7) and $\frac{\partial \mathrm{TWV}}{\partial w(i,o)}$ becomes

$$\mathrm{TWV} = \frac{1}{|\mathcal{Q}|} \sum_{\mathrm{q} \in \mathcal{Q}} \sum_{h \in H_q} \sigma(\overline{p}_h^q, \theta) \gamma(q, h) \tag{A.10}$$

$$\frac{\partial \mathrm{TWV}}{\partial w(i, o)} = \frac{1}{|\mathcal{Q}|} \sum_{\mathrm{q} \in \mathcal{Q}} \sum_{h \in H_q} \frac{\partial \sigma(\overline{p}_h^q, \theta)}{\partial w(i, o)} \gamma(q, h) \tag{A.11}$$

where

$$\begin{aligned} \frac{\partial \sigma(\overline{p}_h^q, \theta)}{\partial w(i, o)} &= \frac{\partial \sigma(\overline{p}_h^q, \theta)}{\partial \overline{p}_h^q} \frac{\partial \overline{p}_h^q}{\partial w(i, o)} \\ &= \frac{\alpha \exp\left(-\alpha(\overline{p}_h^q - \theta)\right)}{\left(1 + \exp\left(-\alpha(\overline{p}_h^q - \theta)\right)\right)^2} \frac{\partial \overline{p}_h^q}{\partial w(i, o)} \\ &= \alpha \sigma(\overline{p}_h^q, \theta)(1 - \sigma(\overline{p}_h^q, \theta)) \frac{\partial \overline{p}_h^q}{\partial w(i, o)} \end{aligned} \tag{A.12}$$

The last term in the above equality is written as

$$
\begin{aligned}
\frac{\partial \overline{p}_h^q}{\partial w(i,o)} &= \frac{\partial}{\partial w(i,o)} \frac{p_h^q}{Z_q} = \frac{1}{Z_q^2} \left\{ \frac{\partial p_h^q}{\partial w(i,o)} Z_q - p_h^q \frac{\partial Z_q}{\partial w(i,o)} \right\} \\
&= \frac{1}{Z_q^2} \left\{ Z_q \sum_{r \in R_h} \frac{\partial p_{hr}^q}{\partial w(i,o)} - p_h^q \sum_{h' \in H_q} \sum_{r' \in R_{h'}} \frac{\partial p_{h'r'}^q}{\partial w(i,o)} \right\} \qquad \text{(A.13)}
\end{aligned}
$$

where

$$
\begin{aligned}
\frac{\partial p_{hr}^q}{\partial w(i,o)} &= \frac{\partial}{\partial w(i,o)} \exp(-w_{qr} - n_{hr}) = \exp(-w_{qr} - n_{hr}) \frac{-\partial w_{qr}}{\partial w(i,o)} \\
&= \exp(-w_{qr} - n_{hr}) \frac{\partial}{\partial w(i,o)} \left(-\sum_{m=1}^{|\pi_{qr}|} w_{qr}(i_m, o_m) \right) \\
&= -p_{hr}^q \sum_{m=1}^{|\pi_{qr}|} I(l(\pi_{qr}(m)) = (i,o)) \\
&= -p_{hr}^q c(w(i,o), \pi_{qr}) = -p_{hr}^q c_{qr}(i,o) \qquad \text{(A.14)}
\end{aligned}
$$

Here $c_{qr}(i,o) = c(w(i,o), \pi_{qr})$ denotes the number of occurrences of weight $w(i,o)$ in the confused (proxy) path π_{qr}:

$$
c(w(i,o), \pi_{qr}) = c_{qr}(i,o) = \sum_{m=1}^{|\pi_{qr}|} I(l(\pi_{qr}(m)) = w(i,o)) \qquad \text{(A.15)}
$$

In Equation (A.15), $I(.)$ is the indicator and $l(\pi_{qr}(m)) = w(i,o)$ is the input-output label pair of the mth arc on the confusion path, π_{qr}, from q to r. Now, $\frac{\partial \overline{p}_h^q}{\partial w(i,o)}$ becomes

$$\begin{aligned}\frac{\partial \overline{p}_h^q}{\partial w(i,o)} = \frac{1}{Z_q^2} \Bigg\{ & Z_q \sum_{r\in R_h} -p_{hr}^q c_{qr}(i,o) \\ & - p_h^q \Bigg(\sum_{h'\in H_q} \sum_{r'\in R_{h'}} - \exp\big(- w_{qr'} - n_{h'r'}\big) c_{qr'}(i,o) \Bigg) \Bigg\} \end{aligned} \tag{A.16}$$

$$\begin{aligned} = \frac{1}{Z_q^2} \Bigg\{ & -Z_q \sum_{r\in R_h} p_{hr}^q c_{qr}(i,o) \\ & + p_h^q \Bigg(\sum_{r'\in \bigcup R_{h'}} \exp(-w_{qr'}) c_{qr'}(i,o) \sum_{h':r'\in R_{h'}} \exp\big(- n_{h'r'}\big) \Bigg) \Bigg\} \end{aligned} \tag{A.17}$$

In order to simplify the notation, assume that a query term r is composed with an identity CM where $w_{rr} = 0$. Then the following can be written:

$$p_h^r = \exp(-n_{hr}) \tag{A.18}$$

$$Z_r = \sum_{h:r\in R_h} p_h^r = \sum_h \exp(-n_{hr}) \tag{A.19}$$

$$\overline{p}_h^r = \frac{p_h^r}{Z_r} \tag{A.20}$$

By making use of these and changing the index of summation of the last term in

Equation A.17, it becomes

$$\begin{aligned}
\frac{\partial \overline{p}^q_h}{\partial w(i,o)} &= \frac{1}{Z_q^2}\left\{-\sum_{r\in R_h} p^q_{hr} c_{qr}(i,o) Z_q + p^q_h\left(\sum_{r\in\bigcup R_{h'}} \exp(-w_{qr}) c_{qr}(i,o) Z_r\right)\right\} &\text{(A.21)}\\
&= \frac{1}{Z_q}\left\{-\sum_{r\in R_h} p^q_{hr} c_{qr}(i,o) + \overline{p}^q_h\left(\sum_{r\in\bigcup R_{h'}} \exp(-w_{qr}) c_{qr}(i,o) Z_r\right)\right\} &\text{(A.22)}\\
&= \frac{1}{Z_q}\sum_{r\in\bigcup R_{h'}} (-\exp(-w_{qr} - n_{hr}) c_{qr}(i,o) I(r\in R_h) \\
&\qquad\qquad + \overline{p}^q_h \exp(-w_{qr}) c_{qr}(i,o) Z_r) &\text{(A.23)}\\
&= \frac{1}{Z_q}\sum_r c_{qr}(i,o)\exp(-w_{qr})\big(\overline{p}^q_h Z_r - \exp(-n_{hr}) I(r\in R_h)\big) &\text{(A.24)}\\
&= \frac{1}{Z_q}\sum_r c_{qr}(i,o)\exp(-w_{qr})\big(\overline{p}^q_h Z_r - p^r_h I(r\in R_h)\big) &\text{(A.25)}\\
&= \frac{1}{Z_q}\sum_r c_{qr}(i,o)\exp(-w_{qr}) Z_r\big(\overline{p}^q_h - \overline{p}^r_h I(r\in R_h)\big) &\text{(A.26)}
\end{aligned}$$

In Equation A.23 we combined two summation terms into a single one over all r terms for q, i.e. for $r \in R^q$, where $R^q = \bigcup_{h\in H_q} R_h$. Then, the derivative of the TWV becomes

$$\begin{aligned}
\frac{\partial \mathrm{TWV}}{\partial w(i,o)} =& \frac{1}{|\mathcal{Q}|}\sum_{\mathrm{q}\in\mathcal{Q}}\sum_h \alpha\sigma(\overline{p}^q_h,\theta)(1-\sigma(\overline{p}^q_h,\theta))\cdot \\
& \left[\sum_{r\in R^q}\frac{Z_r}{Z_q}\big(\overline{p}^q_h - \overline{p}^r_h I(r\in R_h)\big)e^{-w_{qr}} c_{qr}(i,o)\right]\gamma(q,h) &\text{(A.27)}\\
\frac{\partial \mathrm{TWV}}{\partial w(i,o)} =& \frac{1}{|\mathcal{Q}|}\sum_{\mathrm{q}\in\mathcal{Q}}\sum_h \alpha\sigma(\overline{p}^q_h,\theta)(1-\sigma(\overline{p}^q_h,\theta))\cdot \\
& \left[\sum_{(q,r)}\frac{Z_r}{Z_q}\big(\overline{p}^q_h - \overline{p}^r_h I(r\in R_h)\big)\sum_{r'\in r} e^{-w_{qr'}} c_{qr'}(i,o)\right]\gamma(q,h) &\text{(A.28)}
\end{aligned}$$

where the summation $\sum_{r'\in r} e^{-w_{qr'}} c_{qr'}(i,o)$ corresponds to the sum over FST paths from query term q to confused term r arising from multiple pronunciation modeling of q in which case multiple maths can lead to the same r term. This expression can be interpreted as the expected count of the weight because $w_{qr'}$ is the negative log-probability so $e^{-w_{qr'}}$ becomes the probability itself. If we replace $\gamma(q,h)$ in Equation A.28 with the expression given in Equation A.9, we get the equation given in Equation 4.7.

Finally, we can update the weights of the CM by using a step size η as

$$w \longleftarrow w + \eta \nabla w \text{ where } \nabla w = \frac{\partial \text{TWV}}{\partial w(i,o)} \tag{A.29}$$

A.2. Learning the Threshold (θ) of the Sigmoid Function

When we maximize MTWV, we also need to find the threshold θ as we update the weights and therefore the search scores. In order the get the new values of θ, gradient based update is used as in determining the weights of the CM (Equation A.29). Therefore, $\frac{\partial \text{TWV}}{\partial \theta}$ should be calculated as shown in the following equations:

$$\frac{\partial \text{TWV}}{\partial \theta} = \frac{1}{|\mathcal{Q}|} \sum_{\text{q} \in \mathcal{Q}} \sum_{h \in H_q} \frac{\partial \sigma(\overline{p}_h^q, \theta)}{\partial \theta} \gamma(q,h) \tag{A.30}$$

$$\frac{\partial \sigma(\overline{p}_h^q, \theta)}{\partial \theta} = -\alpha \sigma(\overline{p}_h^q, \theta)(1 - \sigma(\overline{p}_h^q, \theta)) \tag{A.31}$$

$$\frac{\partial \text{TWV}}{\partial \theta} = \frac{-1}{|\mathcal{Q}|} \sum_{\text{q} \in \mathcal{Q}} \sum_{h \in H_q} \alpha \sigma(\overline{p}_h^q, \theta)(1 - \sigma(\overline{p}_h^q, \theta)) \gamma(q,h) \tag{A.32}$$

A.3. Simplifications to the Training Procedure

The update equations involve summation over different units corresponding to multiple paths in the WFST strucure. When the relations become one-to-one, for example if there is a single path from q to r or R_h consists of a single r term, derivative terms become simpler. When we apply our CM training procedure to the CM of the symbolic index based system as discussed in Section 4.2, such simplifications occur. In this section, they will be summarized and the simpler versions of the update terms will be derived. Since the index does not contain scores, $n_{hr} = 0, p_h^r = 1, p_{hr}^q = \exp(-w_{qr})$. There is a single r term corresponding to each hit h ($r \leftrightarrow h$) so $Z_r = p_h^r = 1$ then

$\overline{p}_h^r = 1$. And there is a single path from q to r. Then Equation A.22 becomes,

$$
\begin{aligned}
\frac{\partial \overline{p}_h^q}{\partial w(i,o)} &= \frac{1}{Z_q}\left\{ -p_{hr}^q c_{qr}(i,o) + \overline{p}_h^q \big(\sum_{r' \in \bigcup R_{h'}} \exp(-w_{qr'}) c_{qr'}(i,o) \big) \right\} \\
&= \frac{1}{Z_q}\left\{ -p_h^q c_{qr}(i,o) + \overline{p}_h^q \big(\sum_{h' \in H_q} p_{h'}^q c_{qr'}(i,o) \big) \right\} \\
&= -\overline{p}_h^q c_{qr}(i,o) + \overline{p}_h^q \left(\sum_{h' \in H_q} \overline{p}_{h'}^q c_{qr'}(i,o) \right) \\
&= \overline{p}_h^q \left(-c_{qr}(i,o) + \sum_{h' \in H_q} \overline{p}_{h'}^q c_{qr'}(i,o) \right) \qquad \text{(A.33)}
\end{aligned}
$$

Thus we get the Equation 4.11 in Section 4.2. The expression for $\frac{\partial \overline{p}_h^q}{\partial w(i,o)}$ can be substituted into the expression for $\frac{\partial \text{TWV}}{\partial w(i,o)}$ by using Equations A.11 and A.12.

$$
\begin{aligned}
\frac{\partial \text{TWV}}{\partial w(i,o)} &= \frac{1}{|\mathcal{Q}|} \sum_{\text{q} \in \mathcal{Q}} \sum_{h \in H_q} \alpha \sigma(\overline{p}_h^q, \theta)(1 - \sigma(\overline{p}_h^q, \theta)) \frac{\partial \overline{p}_h^q}{\partial w(i,o)} \gamma(q,h) \\
&= \frac{1}{|\mathcal{Q}|} \sum_{\text{q} \in \mathcal{Q}} \sum_{h \in H_q} \alpha \sigma(\overline{p}_h^q, \theta)(1 - \sigma(\overline{p}_h^q, \theta)) \overline{p}_h^q \Big(-c_{qr}(i,o) + \sum_{h' \in H_q} \overline{p}_{h'}^q c_{qr'}(i,o) \Big) \gamma(q,h)
\end{aligned}
$$

(A.34)

Thus we obtain the $\nabla w(i,o)$ term which is used in the weight update equation given in Equation A.29.

REFERENCES

1. Mamou, J., B. Ramabhadran and O. Siohan, “Vocabulary independent spoken term detection”, *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 615–622, ACM, 2007.

2. Saraclar, M. and R. Sproat, “Lattice-Based Search for Spoken Utterance Retrieval”, *HLT-NAACL*, pp. 129–136, 2004.

3. Parada, C., A. Sethy and B. Ramabhadran, “Query-by-example spoken term detection for OOV terms”, *IEEE Workshop on Automatic Speech Recognition & Understanding, ASRU*, pp. 404–409, IEEE, 2009.

4. Saraclar, M., A. Sethy, B. Ramabhadran, L. Mangu, J. Cui, X. Cui, B. Kingsbury and J. Mamou, “An empirical study of confusion modeling in keyword search for low resource languages”, *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 464–469, 2013.

5. Karanasou, P., L. Burget, D. Vergyri, M. Akbacak and A. Mandal, “Discriminatively trained phoneme confusion model for keyword spotting”, *Interspeech*, pp. 2434–2437, 2012.

6. Can, D., E. Cooper, A. Sethy, C. White, B. Ramabhadran and M. Saraclar, “Effect of pronounciations on OOV queries in spoken term detection”, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3957–3960, IEEE, 2009.

7. Zhang, Y., *Unsupervised speech processing with applications to query-by-example spoken term detection*, Ph.D. Thesis, Massachusetts Institute of Technology, 2013.

8. Hazen, T. J., W. Shen and C. White, “Query-by-example spoken term detection

using phonetic posteriorgram templates", *IEEE Workshop on Automatic Speech Recognition & Understanding, ASRU*, pp. 421–426, IEEE, 2009.

9. Mangu, L., B. Kingsbury, H. Soltau, H.-K. Kuo and M. Picheny, "Efficient spoken term detection using confusion networks", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7844–7848, IEEE, 2014.

10. Park, A. S. and J. R. Glass, "Unsupervised pattern discovery in speech", *IEEE Transactions on Audio, Speech, and Language Processing,*, Vol. 16, No. 1, pp. 186–197, 2008.

11. Norouzian, A., R. Rose, S. H. Ghalehjegh and A. Jansen, "Zero resource graph-based confidence estimation for open vocabulary spoken term detection", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8292–8296, IEEE, 2013.

12. Zhang, Y. and J. R. Glass, "Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams", *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pp. 398–403, IEEE, 2009.

13. Can, D. and M. Saraclar, "Lattice indexing for spoken term detection", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 19, No. 8, pp. 2338–2347, 2011.

14. Sarı, L., B. Gündoğdu and M. Saraçlar, "Fusion of LVCSR and Posteriorgram Based Keyword Search", *Sixteenth Annual Conference of the International Speech Communication Association (Interspeech)*, pp. 824–828, 2015.

15. Sarı, L., B. Gündoğdu and M. Saraçlar, "Posteriorgram based approaches in keyword search", *23th Signal Processing and Communications Applications Conference (SIU)*, pp. 1183–1186, May 2015.

16. Rabiner, L. and B.-H. Juang, *Fundamentals of speech recognition*, Prentice Hall,

Inc., Upper Saddle River, NJ, USA, 1993.

17. Jelinek, F., *Statistical methods for speech recognition*, MIT press, Cambridge, MA, USA, 1997.

18. Huang, X., A. Acero, H.-W. Hon and R. Foreword By-Reddy, *Spoken language processing: A guide to theory, algorithm, and system development*, Prentice Hall, 2001.

19. Bourlard, H. A. and N. Morgan, *Connectionist speech recognition: a hybrid approach*, Vol. 247, Kluwer Academic Publishers, Norwell, MA, USA, 1993.

20. Hinton, G., L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups", *IEEE Signal Processing Magazine*, Vol. 29, No. 6, pp. 82–97, 2012.

21. Rumelhart, D. E., G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors", *Nature*, Vol. 323, No. 6088, pp. 533–536, 1986.

22. Mohri, M., F. Pereira and M. Riley, "Weighted finite-state transducers in speech recognition", *Computer Speech & Language*, Vol. 16, No. 1, pp. 69–88, 2002.

23. Allauzen, C., M. Riley, J. Schalkwyk, W. Skut and M. Mohri, "OpenFst: A General and Efficient Weighted Finite-State Transducer Library", *CIAA*, Vol. 4783, pp. 11–23, 2007, `http://www.openfst.org`.

24. Allauzen, C., M. Mohri and M. Saraclar, "General indexation of weighted automata: application to spoken utterance retrieval", *Proceedings of the Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval at HLT-NAACL*, pp. 33–40, Association for Computational Linguistics, 2004.

25. Mangu, L., E. Brill and A. Stolcke, "Finding consensus in speech recognition: word

error minimization and other applications of confusion networks", *Computer Speech & Language*, Vol. 14, No. 4, pp. 373–400, 2000.

26. Chiu, J., Y. Wang, J. Trmal, D. Povey, G. Chen and A. Rudnicky, "Combination of FST and CN search in spoken term detection", *Proceedings of Interspeech*, pp. 2784–2788, 2014.

27. Kanthak, S. and H. Ney, "Context-dependent acoustic modeling using graphemes for large vocabulary speech recognition", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vol. 2, pp. 845–848, Citeseer, 2002.

28. Bisani, M. and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion", *Speech Communication*, Vol. 50, No. 5, pp. 434 – 451, 2008.

29. Chen, G., O. Yilmaz, J. Trmal, D. Povey and S. Khudanpur, "Using proxies for OOV keywords in the keyword search task", *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 416–421, 2013.

30. Chen, G., S. Khudanpur, D. Povey, J. Trmal, D. Yarowsky and O. Yilmaz, "Quantifying the value of pronunciation lexicons for keyword search in low resource languages", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8560–8564, IEEE, 2013.

31. Rodriguez-Fuentes, L. J., A. Varona, M. Penagarikano, G. Bordel and M. Diez, "High-performance Query-by-Example Spoken Term Detection on the SWS 2013 evaluation", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7819–7823, IEEE, May 2014.

32. Anguera, X., "Information retrieval-based dynamic time warping", *Interspeech*, pp. 1–5, 2013.

33. Jansen, A. and B. V. Durme, "Indexing Raw Acoustic Features for Scalable Zero Resource Search", *Interspeech*, pp. 2466–2469, 2012.

34. Jansen, A., K. Church and H. Hermansky, “Towards spoken term discovery at scale with zero resources”, *Interspeech*, pp. 1676–1679, 2010.

35. Silaghi, M.-C., “Spotting subsequences matching an HMM using the average observation probability criteria with application to keyword spotting”, *Proceedings of the National Conference on Artificial Intelligence*, Vol. 20, p. 1118, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

36. Grangier, D., J. Keshet and S. Bengio, “Discriminative keyword spotting”, *Automatic speech and speaker recognition: large margin and kernel methods*, pp. 175–194, 2009.

37. Kintzley, K., A. Jansen and H. Hermansky, “Featherweight phonetic keyword search for conversational speech”, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7859–7863, IEEE, 2014.

38. Jansen, A. and P. Niyogi, “Point process models for spotting keywords in continuous speech”, *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 17, No. 8, pp. 1457–1470, 2009.

39. Jansen, A., “Whole word discriminative point process models”, *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5180–5183, 2011.

40. Kintzley, K., A. Jansen and H. Hermansky, “MAP Estimation of Whole-Word Acoustic Models with Dictionary Priors”, *Interspeech*, pp. 787–790, 2012.

41. Liu, C., A. Jansen, G. Chen, K. Kintzley, J. Trmal and S. Khudanpur, “Low-resource open vocabulary keyword search using point process models”, *Interspeech*, pp. 2789–2793, 2014.

42. Hochreiter, S. and J. Schmidhuber, “Long Short-Term Memory”, *Neural Computation*, pp. 1735–1780, 1997.

43. Fernández, S., A. Graves and J. Schmidhuber, “An application of recurrent neural networks to discriminative keyword spotting”, *Artificial Neural Networks–ICANN 2007*, pp. 220–229, Springer, 2007.

44. Wollmer, M., F. Eyben, J. Keshet, A. Graves, B. Schuller and G. Rigoll, “Robust discriminative keyword spotting for emotionally colored spontaneous speech using bidirectional LSTM networks”, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3949–3952, IEEE, 2009.

45. Chen, G., C. Parada and T. N. Sainath, “Query-by-example keyword spotting using long short-term memory networks”, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5236–5240, 2015.

46. Chen, G., C. Parada and G. Heigold, “Small-footprint keyword spotting using deep neural networks”, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4087–4091, IEEE, 2014.

47. Fiscus, J. G., J. Ajot, J. S. Garofolo and G. Doddingtion, “Results of the 2006 spoken term detection evaluation”, *Proceedings of ACM SIGIR Workshop on Searching Spontaneous Conversational Speech*, pp. 51–55, Citeseer, 2007.

48. *KWS14 Keyword Search Evaluation Plan*, 2014, `http://www.nist.gov/itl/iad/mig/upload/KWS14-evalplan-v11.pdf`, July 2015.

49. *NIST Framework For Detection Evaluations (F4DE)*, `http://www.nist.gov/itl/iad/mig/tools.cfm`, July 2015.

50. Mamou, J., J. Cui, X. Cui, M. J. Gales, B. Kingsbury, K. Knill, L. Mangu, D. Nolden, M. Picheny, B. Ramabhadran *et al.*, “System combination and score normalization for spoken term detection”, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8272–8276, 2013.

51. Karakos, D., R. Schwartz, S. Tsakalidis, L. Zhang, S. Ranjan, T. Tim Ng, R.-C.

Hsiao, G. Saikumar, I. Bulyko, L. Nguyen, J. Makhoul, F. Grezl, M. Hannemann, M. Karafiat, I. Szoke, K. Vesely, L. Lamel and V.-B. Le, "Score normalization and system combination for improved keyword spotting", *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 210–215, IEEE, 2013.

52. Wang, Y. and F. Metze, "An in-depth comparison of keyword specific thresholding and sum-to-one score normalization", *Proceedings of Interspeech, Singapore*, pp. 2474–2478, 2014.

53. Harper, M., *Babel – Addressing the Language Deluge*, 2011, `http://www.iarpa.gov/images/files/programs/babel/Babel_Overview_UNCLASSIFIED-2011-05-31.pdf`, December 2015.

54. *KWS15 Keyword Search Evaluation Plan*, 2015, `http://www.nist.gov/itl/iad/mig/upload/KWS15-evalplan-v05.pdf`, December 2015.

55. Povey, D., A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann *et al.*, "The Kaldi Speech Recognition Toolkit", *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, IEEE Signal Processing Society, Dec. 2011.

56. *Kaldi Babel recipe*, 2015, `https://github.com/kaldi-asr/kaldi/tree/master/egs/babel/s5c`, December 2015.

57. Trmal, J., G. Chen, D. Povey, S. Khudanpur, P. Ghahremani, X. Zhang, V. Manohar, C. Liu, A. Jansen, D. Klakow *et al.*, "A keyword search system using open source software", *IEEE Spoken Language Technology Workshop (SLT)*, pp. 530–535, IEEE, 2014.

58. Soltau, H., G. Saon and B. Kingsbury, "The IBM Attila speech recognition toolkit", *IEEE Spoken Language Technology Workshop (SLT)*, pp. 97–102, IEEE, 2010.

59. Sarı, L. and M. Saraçlar, "Discriminative training of the keyword search confu-

sion model", *23th Signal Processing and Communications Applications Conference (SIU)*, pp. 1175–1178, May 2015.

60. Cui, J., X. Cui, B. Ramabhadran, J. Kim, B. Kingsbury, J. Mamou, L. Mangu, M. Picheny, T. N. Sainath and A. Sethy, "Developing speech recognition systems for corpus indexing under the IARPA Babel program", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6753–6757, 2013.

61. Bottou, L., "Large-scale machine learning with stochastic gradient descent", *Proceedings of COMPSTAT'2010*, pp. 177–186, Springer, 2010.