The Impact of Arabic Dialects on the Performance of Arabic Speech Recognition Systems

تأثير بعض اللهجات العربية على التعرف الآلي على الكلام باللغة العربية

Amani Abu Gharbieh

Supervisor
Dr. Abualsoud Hannani

This Thesis was submitted in partial fulfillment of the requirements for the Master's Degree in Computing from the Faculty of Graduate Studies at Birzeit University, Palestine

October 29, 2019

Mater Thesis

The Impact of Arabic Dialects on the Performance of Arabic Speech Recognition Systems

تأثير بعض اللهجات العربية على التعرف الآلي على الكلام باللغة العربية

Amani Abu Gharbieh ID:1145147

Names of examining committee members

1. Supervisor and head of the committee: **Dr Abualsoud Hannani**
2. First Examiner: **Dr. Mohammad Hussain**
3. Second Examiner:**Dr. Radi Jarrar**

This Thesis was submitted in partial fulfillment of the requirements for the Master's Degree in Computing from the Faculty of Graduate Studies at Birzeit University, Palestine

October 29, 2019

Mater Thesis

The Impact of Arabic Dialects on the Performance of Arabic Speech Recognition Systems

This thesis is done by Amani Abu Gharbieh, with the registration number of 1145147. This thesis is approved by the thesis committee members.

Dr Abualsoud Hannani

_____

Dr. Mohammad Hussain

_____

Dr. Radi Jarrar

_____

Date approved:

_____

Dedication

I lovingly dedicate this research to

The Sake of Allah.

To Mom and Dad for their endless love.

To my husband for his support

To my precious daughters

Declaration

I, Amani J. Abu Gharbieh declare that this thesis " Impact of Arabic Dialects on Performance of Arabic Speech Recognition Systems " and all the presented work and presented results are my own and that they have not been submitted earlier elsewhere.

I confirm that this work was done under the supervision of Dr. Abualsoud Hanani from the Department of Electrical and Computer Engineering, Birzeit University, Palestine.

Amani J. Abu Gharbieh
Birzeit - Palestine
October 29, 2019

Acknowledgements

I am grateful to Allah for giving me the strength to complete this work with all surrounded obstacles. I would like also to thank my Advisor Dr. Abualsoud Hanani for his guidance and his endless support during my research.

Special thanks to my husband and daughters for being patient and giving me my space to finish my work.

Special thanks to my Mom for her endless love and prayers and for her Delicious food.

Amani J. Abu Gharbieh
October 29, 2019

Abstract

Automatic Speech Recognition (ASR) is the core of interest for most recent applications, like voice search (VS), short message dictation (SMD) and others. One of the challenges of the Speech Recognition (SR) process is human speech variations, this is due to many factors like age, gender, nationality and level of education. Generally different languages, different dialects and pronunciation have a big effect as well. Arabic language adds more challenges to SR than any other languages, this is due to the large difference between Modern Standard Arabic (MSA) and regional dialects in Arab countries.

In this research, we study the impact of Arabic dialects on the performance of Arabic ASR. This is through using different adaptation and optimization techniques in the Acoustic Model (AM). We find that using the state-of-art of Deep Neural Network (DNN) improved the performance of ASR over the traditional Hidden Markov model-Gaussian Mixture Model (HMM-GMM). And we get the best performance when we have HMM-DNN with five hidden layers, 2048 hidden dimensions, Minimum Phone Error (MPE) optimization and Mel-Frequency Cepstral Coefficients (MFCC) as feature extraction. But when we add DNN to the feature extraction stage to have Bottleneck features BNF, we get better performance than using MFCC. We should know that in feature extraction phase, we reduced the frame size to 20 millisecond (ms) and kept the time shift equal to 10 ms. This action increases the overlap between frames and reduces the data lose.

In our research, we start with Dependent Dialect ASR (DD-ASR), we used four different datasets sizes ranges between 2K to 50K utterances. We found that increasing the size of training data enhances the performance of Arabic ASR. And

the most important is adding Arabic dialect to training dataset which enhances the performance as well. We compare the 50K utterance dataset of Modern Spoken Arabic (MSA) only with the 40K utterance dataset of a mixture of Arabic dialect and MSA. We get better performance while adding the dialect to training, this means that data selection of training data is also important to improve the performance of Arabic ASR despite the size of the training dataset size. Then we thought of mixing all the dialects while training ASR to have Independent Dialect ASR (ID-ASR). The performance of ID-ASR is better than DD-ASR.

For Language Model (LM) phase, We generate the LM using the text of the training dataset for each experiment. So the size of training data affects the LM as well.

Also through our different experiments, we find that feature-space Maximum Likelihood Linear Regression (fMLLR) and Maximum A Posterior(MAP) act the same while using only one dialect trained ASR. But when we train ASR on multiple dialects MAP has better performance than fMLLR.

Our results show that employing DNN technology in any phase of ASR enhances the performance of Arabic ASR. As we used this technology to generate feature vector (Bottleneck features) and used it in the Acoustic Model phase.

ملخص

يعد التعرف التلقائي على الكلام ASR جوهر الاهتمام لمعظم التطبيقات الحديثة، مثل البحث الصوتي، وإملاء الرسائل القصيرة وغيرها. واحدة من التحديات التي تواجه عملية التعرف على الكلام هي الاختلافات في الكلام البشري، وهذا يرجع إلى العديد من العوامل مثل العمر والجنس والجنسية ومستوى التعليم. عمومًا تعدد اللغات، اللهجات وطريقة النطق لها تأثير كبير أيضًا. تضيف اللغة العربية تحديات أكبر من أي لغة أخرى، ويرجع ذلك إلى الفرق الكبير بين اللغة العربية الفصحى المعاصرة واللهجات الإقليمية في الدول العربية.

في هذا البحث، ندرس تأثير اللهجات العربية على أداء التعرف التلقائي للغة العربية. يتم ذلك من خلال استخدام أساليب مختلفة للتكيف والتحسين في النموذج الصوتي. لقد وجدنا أن استخدام أحدث تقنية للشبكة العصبية العميقة HMM-DNN أدى إلى تحسين أداء التعرف التلقائي على نموذج المعياري التقليدي HMM-GMM . ونحصل على أفضل أداء عندما يكون لدينا HMM-DNN بخمس طبقات مخفية، وأبعاد ٢٠٤٨ مخفية، وتحسين الحد الأدنى للخطأ الصوتي MPE ، واستخدام MFCC كمتجه حامل خصائص المقطع الصوتي. وعندما نضيف DNN إلى مرحلة استخراج الميزات للحصول على Bottleneck feature ، نحصل على أداء أفضل من استخدام MFCC . يجب أن نعرف أنه في مرحلة استخراج الميزات، قمنا بتقليل حجم الإطار إلى ٢٠ مللي ثانية وحافظنا على تحول التنقل الزمني مساوياً لـ ١٠ مللي ثانية. هذا الإجراء يزيد من التداخل بين الإطارات ويقلل من فقدان البيانات.

أيضاً في هذا البحث، بدأنا بتعرف تلقائي معتمد على لهجة معينة ، وقد استخدمنا أربعة

أحجام مختلفة من مجموعات البيانات تتراوح بين ٢٠٠٠ و ٥٠٠٠٠. وجدنا أن زيادة حجم بيانات التدريب يعزز أداء التعرف التلقائي. والأهم من ذلك هو إضافة اللهجة العربية إلى مجموعة بيانات التدريب مما يعزز الأداء كذلك. قارنا أداء مجموعتين من مجموعات البيانات: الأولى تحتوي على بيانات من اللغة العربية الفصحى وحجمها ٥٠٠٠٠ جملة والثانية عبارة عن مزيج من لهجة عربية واللغة العربية الفصحى وحجمها ٤٠٠٠٠ جملة. وكانت النتيجة أننا حصلنا على أداء أفضل أثناء إضافة اللهجة إلى التدريب، وهذا يعني أن اختيار البيانات ونوعها لبيانات التدريب مهم أيضًا لتحسين أداء التعرف الآلي للغة العربية وليس فقط حجم مجموعة بيانات التدريب. ثم فكرنا في مزج جميع اللهجات أثناء التدريب على التعرف الآلي للحصول على نموذج مستقل عن أي لهجة عربية. وكان أداء النموذج المستقل أفضل من النموذج المعتمد على لهجة معينة. ويعزى ذلك لحجم بيانات التدريب للهجات العربية.

في هذا البحث أيضاً حاولنا المقارنة بين أدوات المواءمة من خلال تجارب مختلفة، ووجدنا أن fMLLR وبغ باع يتصرفان بنفس الكفاءة أثناء استخدام DD-ASR أي في حالة التدريب على لهجة واحدة فقط. ولكن في حالة ID-ASR أي في حالة التدرب على أكثر من لهجة، كان أداء MAP أفضل من fMLLR .

كما وتظهر نتائجنا أن استخدام تقنية الشبكات العميقة في أي مرحلة من مراحل التعرف الآلي على الأصوات، يعزز أداء التعرف الآلي على الأصوات باللغة العربية نظرًا لأننا استخدمنا هذه التقنية لإنشاء متجه حامل خصائص الصوت واستخدمناها في مرحلة النموذج الصوتي.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Total phrase
AM	Acoustic Model
ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
ATLAS	Automatically Tuned Linear Algebra Software
ADID	Automatic Dialect Identification
BIC	Bayesian Information Criterion
BNF	Bottleneck features
CAT	Cluster Adaptive Training
CMLLR	Constrained Maximum Likelihood Linear Regression
DD-ASR	Dependent Dialect Automatic Speech Recognition
DNN	Deep Neural Network
EER	Equal Error Rate
EM-PCA	Expectation Maximization-Principal Component Analysis
Fbank	Filter Banks
FLACK	Free Lossless Audio CODEC
fMLLR	feature-space Maximum Likelihood Linear Regression
GALE	Global Autonomous Language Exploitation
GMM	Gaussian Mixture Model
GPU	Graphical Processing Unit
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
HS	Hidden State
ID-ASR	Independent Dialect Automatic Speech Recognition
IPA	International Phonetic Alphabet
LDA	Linear Discriminative Analysis
LM	Language Model
LR	Linear Regression
LVCSR	Large Vacabulary Continous Speech Recognition

MADA	Morphological Analysis and Disambiguation for Arabic
MAP	Maximum A Posterior
MFCCs	Mel-Frequency Cepstral Coefficients
MLE	Maximum Likelihood Estimation
MLLR	Maximum Likelihood Linear Regression
MLLT	Maximum Likelihood Linear Transformation
MLP	Multi-Layer Perceptron
MMI	Maximum Mutual Information
MPE	Minimum Phone Error
ms	millisecond
MSA	Modern Standard Arabic
NAP	nuisance attribute projection
NN	Neural Network
OOV	out-of-vocabulary
PDFs	Probability Density Functions
PLP	Perceptual Linear Prediction
SAT	Speaker Adaptive Training
SD	Speaker Dependent
SGMM	Subspace Gaussian Mixture Models
SLP	Single- Layer Perceptron
SI	Speaker Independent
SRILM	Stanford Research Institute Language Model
V2P	Vowelize to Phones
WCCN	within-class covariance normalization
WER	Word Error Rate
WSJ	Wall Street Journal

Chapter 1

Introduction

1.1 Overview

Automatic Speech Recognition (ASR) is one of the most popular research area in our days. ASR is a technology used to convert speech into text. This conversion creates a revolution in developing applications. The widespread of smart phones increases the necessity of these applications as well. Some of these applications are voice search like Siri in IOS systems, car navigator that allows the driver to tell the destination in speech, communicating with people with hearing problems and much more.

Because of the importance of ASR, more research is being done in this field. All the work is trying to improve the accuracy of the conversion process, they tried to work on feature extraction techniques, algorithms used in Acoustic model in order to handle the exacted features, Language Model (LM) and hardware[1]. Developing ASR is not an easy job. There are 7.5 billion human being living on this planet[2]. This huge number of people means 7.5 billion different voice print. This also means there are so many languages. Each language is facing the problem of different dialects, which is having different words for the same meaning. Geographical place of

living, different immigrants from different countries and different occupation are factors of having different dialects. For example: in Egypt, people say Arabeya (which means a car), whereas in Levantine, people say Sayyara, neither will understand the word immediately.

Arabic Language in particular is facing an added challenge since Arabic countries were occupied by different foreign countries. This affect the Arabic Language that sometimes you can't recognize the spoken language as it is a different language. Also when we talk about Arabic Language, diacritics is the first to think about. Diacritics is a special property for Arabic Language, which change the meaning of the word even if the words have the same letters. For example: جَمَل jamal means camel but جُمَلْ jomal means sentences. In this thesis, we focused on the most common and popular dialects in the Arabic countries, shown in figure1.1.1[3]; such as Gulf, Egyptian, Levantine (Jordan, Syria, Lebanon and Palestine), Maghrebi (Morocco, Algeria, Tunisia, and Libya) and MSA. See next section1.1.1 for more details about Arabic dialects.

1.1.1 Arabic language

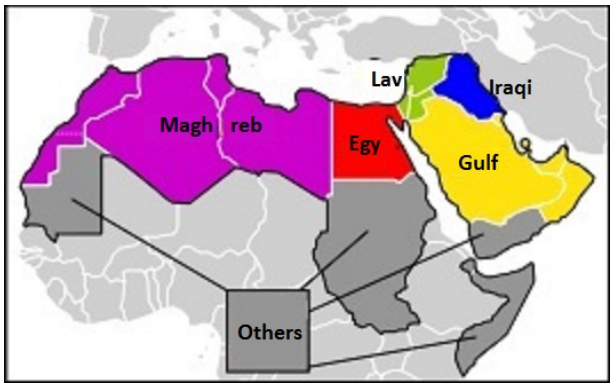


Figure 1.1.1: Different Arabic dialects used in this research

Arabic language is one of the most talked language in the world. About 420

million people speaks Arabic all around the world[4, 5]. Generally Arabic language is written from right to left, written in cursive style and have 28 letters, 3 long vowels, 3 short vowels, 3 for double diacritics and some added letter, in total 39 phoneme as shown in table 1.1[6, 7], and each phoneme has a special symbol given by International Phonetic Alphabet (IPA). In addition, we should know that Arabic Language has a special property, which is diacritics. The same letter has a different phone depending on the diacritics used for this letter وّ ، وَ، وُ، وِ، وْ which is w, wi, wo, wa and ww from left to right, this variety for each letter enlarge the data base for Arabic Language. If diacritics were not included in the training of the ASR, many phonics will not be included in the system, which will increase the error rate. Arabic Language can be classified into two main groups: *MSA*, which is the written language which follow all the rules of writing and diacritics and the *Spoken language* which is dialects in other words. Next we will explain both classes briefly.

Table 1.1: Arabic phonemes

#	Name	English Translation	Phoneme symbol in IPA	Arabic Unicode	Example of pronunciation in English
1	'alef	'/$\bar{a}$	/a:/	ا، ى	father
2	ba'	b	/$\underline{b}$/, /$\underline{p}$/	ب	bee
3	ta'	t	/$\underline{t}$/	ت	tree
4	tha'	th	/θ/	ث	three
5	jeem	j , g	/$\underline{ʒ}$/, /$\underline{g}$/	ج	jam or gorge
6	h$\bar{a}$'	ḥ	/$\underline{h}$/	ح	—-
7	kh$\bar{a}$'	kh	/$\underline{x}$/	خ	ioch(Scottish)
8	d$\bar{a}$l	d	/$\underline{d}$/	د	door
9	dh$\bar{a}$l	dh, th	/ð/	ذ	the
10	r$\bar{a}$'	r	/$\underline{r}$/	ر	rabbit

11	zain	z	/z̲/	ز	zoo
12	seen	s	/s/	س	see
13	sheen	sh	/ʃ̲/	ش	show
14	ṣad	ṣ	/sʕ̲/	ص	massage
15	ḍād	ḍ	/dʕ̲/	ض	—-
16	tā’	ṭ	/tʕ̲/	ط	star
17	zā’	ẓ	/ðʕ/	ظ	—-
18	’ayn	’	/ʕ̲/	ع	no equivalent
19	ghayn	gh	/y/	غ	mercie(French)
20	fā’	f	/f/	ف	fish
21	qāf	q	/q̲/	ق	queen
22	kāf	k	/k̲/	ك	kite
23	lām	l	/l̲/		lemon
24	meem	m	/m̲/	م	man
25	noon	n	/n/	ن	nest
26	hā’	h	/h̲/	ه	horse
27	wāw	w,ū, aw	/w̲,/u:/,/aw/, /u̲/, /o:̲/	و	window
28	yā’	y,ī, ay	/j/, /i:/, /aj/	ي	yes
29	hamza		ʔ	ء	the pause in UH-OH
30	va	v	/v/	ڤ	virus
31	LV ’lef	aa	/a:/	اا	man
32	LV wāw	oo	/oo/	وو	tool
33	LV yā’	ea , ee	/ee/	يـــي	sleep
34	SV ’a	a	/a/	اَ	
35	SV ’u	u	/u/	اُ	

36	SV ’i	i	/i/	إِ
37	’an	an	/an/	أً
38	’on	on	/on/	أٌ
39	’in	in	/in/	إٍ

Notes: LV stands for Long vowel. SV stands for Short vowel.

1.1.1.1 Modern Spoken Arabic (MSA)

The language of literature and the written media such as books, newspapers, magazines, official documents, private and business correspondence, street signs and shop signs. Also it is the language of news broadcasts on radio and television[8, 9].

1.1.1.2 Other dialects

Everything other than MSA is spoken Arabic. In other words it is dialectal speech. There are many dialects in Arabic Language and this is due to many reasons, such as different foreign occupation for Arabic countries, Arabic speakers is moving into different regions of Arabic world which introduce new words to the language, and by time it becomes part of it and sometime the pronunciation of the word changed. Research classified Arabic dialects into five main groups depending on the geographical place and number of population[4, 10].To know more about the difference between the Arabic dialects, check the examples in table 1.2 below. The major Arabic dialects are:

1. North African Arabic that includes: Morocco, Algeria, Tunisia and Libya. Their structure and words are rare, and it is a challenge to understand even to Arabic speakers. Berber and Spanish Languages influence it.

2. Egyptian Arabic. More than 80 million speak this dialect. It is widely known in the Arabic countries due to the media industry.

3. Levantine Arabic that includes: Lebanon, Syria, Jordan and Palestine. British and French languages influence this dialect.

4. Iraqi Arabic. It is geographically close to Arabic Gulf but it has its own characteristic in pronunciation.

5. Gulf Arabic that includes: Kuwait, Bahrain, Qatar, the United Arab Emirates and Oman. Gulf dialect is the closest spoken dialect to MSA with major differences.

Table 1.2: Example shows how major Arabic dialects are different

Dialect name	Example1	Example 2	Example 3
MSA Arabic letters	كثيراً	أريد	المرأة
English	a lot	want	woman
MSA IPA symbols	kaθeeran	'ureedu	'lmar'ah
Tunisian	bar$\int$a	nbvit	lu mra
Algerian	bezzaf	nbvit	lu mra
Moroccan	bezzaf	bavi	lu mra
Egyptian	kiteer	ʕayza	esittat
Lebanese	kteer	bedde	lmara
Iraqi	kulli$\int$	reddet	limrayyat
Gulf	wayed	'be	alhareem

1.1.2 Objectives

The main objective of this thesis is to investigate how close the regional Arabic dialects to Modern Spoken Arabic MSA. The problem we are trying to solve is the degradation of Arabic ASR performance because of huge dialect variation in Arabic language. In this thesis, we focus on the most common and popular dialects in the Arabic countries, such as Gulf, Egyptian, Levantine (Jordan, Syria, Lebanon and

Palestine), Maghrebi (Morocco, Algeria, Tunisia, and Libya) and MSA.
The followings are the main objectives of this thesis:

- Investigate the impact of dialect variation on state-of-the-art Arabic ASR that uses DNN for acoustic models and compare this with the traditional ASR that uses GMM for acoustic models.
- Apply the state-of-the-art techniques for Arabic dialect recognition, such as DNN, different Adaptation techniques used for AM (MAP, FMLLR and combination for both), different optimization techniques (MPE, MMI with different values of boost), and compare the results of these systems.
- Train ASR with all dialects and compare results with dialect only trained and MSA only trained ASR.
- Make the results of all experiments available and accessible to the public.

1.1.3 Research questions

The followings could be research questions to be address in this project:

- How different the impact of each major dialect on the ASR trained on MSA dataset.
- How well the Arabic ASR improves by adding dialectal data to the training dataset.
- To what extent the prior knowledge of true dialect of speaker can improve the speech recognition performance.
- How well state-of-the-art techniques that are mainly based on DNN can accommodate dialect variation for Arabic ASR.

1.2 Relative studies

According to the people desire to build an automatic computer model to emulate human verbal communication skills. Speech Recognition make this desire come true, so it is possible for computer to understand human speech in different languages[11]. Most research in ASR are trying to improve the accuracy through reducing Word Error Rate (WER). In order to achieve this improvement, each research spotted light on some aspects that might reduce WER. Some research like [12] recommends maximizing computation platforms through the adaption of parallel multi-core processors(hardware level). This adaption will reduce the sequential overhead and enhance the performance of ASR. The enhancements on ASR are through enhancing the accuracy of speech to text in all conditions of noise and non ideal circumstances while speaking, increasing recognition efficiency in order to increase throughput and reducing time consuming to real time.

Another approach to improve the efficiency of the ASR is data selection. Different techniques of data selection was used in [13, 14, 15, 16, 17, 18], but they concluded that reducing the training dataset to most reliable subset will do this improvement. For [14] data selection has improved the accuracy by 4% compared with using all the dataset.

There are research done in investigating Arabic ASR as well. There are commercial versions of ASR trained on MSA and it is running with high performance. However, when these ASRs used for spoken language, they fail and had a high WER. This proves the big difference between the spoken dialects and MSA. Researchers tried several techniques to solve this problem. Biadsy et al[9, 19] ran experiments to study the effect of using a specific ASR for each Arabic dialect, they used the average of 70 hours for training for each dialect. They found that when ASR is

trained for a specific dialect, this enhanced ASR performance for each dialect. This means that we have to detect the dialect to run its proper ASR.[9] also discussed pronunciation model and how different models affect ASR performance. Najafian worked in her PHD thesis[20] on the same idea of detecting the dialect will reduce the WER and enhance ASR performance, but she studied English dialects not Arabic, she used 10 hours for training, Najafian found that dialect mismatch between training and testing could increase the WER by seven times compared with using the same dialect in testing and training .

Another approach is through changing the used algorithms used in each phase in the ASR. For Acoustic Model (AM) phase, Hidden Markov Model (HMM) is used to model the sequential structure of speech signals, each state is using Gaussian Mixture Model (GMM) to model spectral representation. In addition, this was the state-of-the-art and each research in Speech Recognition starts from this point. Recently GMM is replaced with Deep Neural Network (DNN), which is many layers of features and great number of parameters. The results show that DNN outperformed the performance of ASR [21]. DNN requires a high computational resources but on the other hand it enhances the recognition[22]. [23] shows results of using GMM and results of using DNN, these results shows that using DNN is reducing WER by 9.79% comparing with GMM model.

Al-Haj et al[24] searched Arabic ASR, especially how to improve the Iraqi ASR, their work was at the pronunciation lexicon. Arabic language is rich enough to have same letters with different meanings when diacritics used as mentioned earlier. So they add the short vowels to their lexicon. This action has reduce the WER by 2.4% comparing of their base system of not including the short vowels.

Patrick Cardinal et al[25] researched Arabic ASR and tried most used techniques

to improve the performance of Arabic ASR. They have 50 hours of recorded audio from AlJazeera news. They had achieved the best results so far for Arabic ASR, they had 17.86% in broadcast reports and 29.85% in broadcast conversation. They used Kaldi[26] toolkit for all the used models. They proposed several systems with different techniques for both GMM and DNN based Acoustic Model. Their results shows that broadcast reports has better accuracy than broadcast conversations. The best result they have got when they used DNN, ivector, fMLLR and MPE. Recent research is trying to involve DNN in all phases of ASR since DNN shows great improvement in the performance. So they used DNN in feature extraction phase with a technique called bottleneck features (BNF). BNF improves the performance of basic GMM system by 3.5% relatively for reports and conversations together[25]. Patrick Cardinal et al also applied I-vector, which used to adapt the speaker and the channel in the process of speech recognition. They applied i-vector with DNN based system. This also improved the performance of the ASR. So as a conclusion for Patrick Cardinal et al research, the best performance was achieved in the DNN based system with feature-space Maximum Likelihood Linear Regression (fMLLR), i-vector and Minimum Phone Error (MPE) techniques, and the performance was 25.78% for using conversations and reports together.

Yan et al[27] tried to improve BNF. As they used 309-hour SwitchboardI to train DNN and set its parameters in DNN based AM. They used the trained DNN as feature extraction with 2000 hour training data to train HMM-GMM, this action reduces the WER by 1.6% comparing with HMM-DNN based AM. Their system can be described as DNN-GMM-HMM. Yu et al[28] also used DNN as feature exaction. Their study shows that using unsupervised pre-training of DNN has enhanced BNF and they also find that using DNN trained to predict context dependent senone target labels produces better BNF. Using the combination of both mentioned strategies has improve the performance of ASR by 16% compared with conventional method

for training BNFs.

Elmahdy et al[29] researched Arabic dialects, especially Egyptian dialects. They trained their system on both MSA and spoken Egyptian dialect, which increase the phoneme generated in AM. They found that using some adaptation techniques like Maximum Likelihood Linear Regression (MLLR) and Maximum A-Posteriori (MAP) improve the speech recognition, as using the combination of both techniques reduced the WER by 17% comparing with the same system without using these adaptation techniques.

Ali et al[30] also researched Egyptian dialects. But they applied different approach than working on the AM. Because of the sparseness of Arabic audio dataset, they tried to overcome this problem by using the dialectical tweets on Twitter to build their LM. This action reduced the out-of-vocabulary (OOV) from 15.1% to 3.2 and the most important that this action reduced the WER from 59.6% to 44.7%. They also tried to compare using all tweets or only Egyptian tweets to check the performance of Egyptian ASR. They found that having only Egyptian tweets has lower WER by 3.4% as mentioned earlier in [14]. Depending on the last two mentioned research, we can conclude that the enhancement in the performance of Speech recognition can be done at the AM and LM.

Najafian[20] in her PHD thesis studied 14 different dialects of British Isles, she used ABI corpus for evaluation and WSJCAM0 corpus for training. She focused on the effect of these different dialects on the performance of the ASR. She concluded that to achieve better performance of ASR with the existence of many different dialects to combine different approaches. For example to have dialect identifier within the feature extraction phase which, trigger a dialect related pronunciation dictionary. This means that the best performance can be achieved by working on all ASR

component. Most of Najafian was at the Acoustic Model, tried several acoustic modeling techniques to reduce the effect of different dialects on the performance of ASR. Also discussed the model selection approach that depend on Automatic Dialect Identification (ADID). I-vector, phonotactic and ACCDIST-SVM are different systems used as ADID, each focuses on some property of speech. She also used Expectation Maximization-Principal Component Analysis (EM-PCA) and Linear Discriminative Analysis (LDA) to investigate the different dialect spaces. Then used this investigation to analyse the ADID and ASR. In GMM-HMM based AM , she found that the maximum reduction in WER was achieved by using ADID to select a specific dialect model followed by speaker adaptation. About HMM-DNN based AM, her research also showed that this model outperformed the traditional model.

In our research, we use two different computers with different hardware characteristics. The computation time for each computer to finish the same task varies. The better hardware proprieties reduce the computation time, and this proves the recommendation of [12]. Also we investigate the effect of training data size verses the variety of the training data, we find that the variety of training data is more important than the size of the training dataset, this proves the recommendations of [13, 14, 15, 16, 17, 18]. In addition, we have GMM and DNN based AM, we find that DNN based AM improves the performance of ASR as [21]. Moreover, we used Qatar Computing Research Institute (QCRI) 2014 Lexicon[31] that was released on 17 March 2014, and yet no newer version of phonetic dictionary is published. Also we discuss some adaptation and optimizing techniques. Finally, we involve DNN in feature extraction phase, and DNN proves its reliability to improve the performance of ASR, this proves the recommendation of Patrick Cardinal et al[25].

1.2.1 Thesis outline

The rest of this thesis is organized as follows. Chapter2 gives brief information about ASR, how it works and its components starting front-end until we get word transcriptions. Chapter 3 introduces the used toolkit and computer resources in our experiments, and explains the experiment names and abbreviation. Also discusses different strategies to study the effect of different dialects on Arabic ASR. Chapter 4 shows all the results we got in our experiments. Chapter 5.1 provides conclusions about the whole work and provides some recommendations for future work.

Chapter 2

Automatic Speech Recognition ASR-Background

In this chapter, we explain the component of ASR and how those component engage together in the conversion of the speech into text.

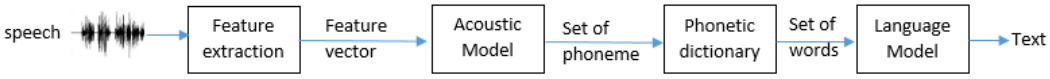


Figure 2.0.1: Automatic Speech recognition main components that handle the speech till we have the text

ASR is a technology that converts the speech into text. Figure 2.0.1 shows the main component of the ASR that converts speech into text. The procedure of conversion can be summarized in the following:

1. The speech, in our case, the speech is the dataset that will be introduced in section 3.2. This speech is being processed in the front end for feature extraction, Mel-Frequency Cepstral Coefficients (MFCC), Filter Bank (Fbank) and Perceptual Linear Prediction (PLP) are the most common techniques used for the feature extraction[32], explained more in section 2.1. The output of this stage will be a feature vector.

2. The feature vector will be the input for the AM, despite the used model HMM-GMM or HMM-DNN. The output will be sequence of phones or candidate phones. Considering that Arabic language has 39 phoneme listed in the table 1.1. Each phoneme pronounced in different way, depending on its location in the word (context), its diacritics, and many other factors such as dialect, speaker, gender, age, health situation, emotional state etc. Because of all these factors, each phoneme has several phones.

3. Those phones used as input to the phonetic dictionary, since each phoneme has several phones. Select the best match phones to make a useful word. In addition, at this step the algorithm selects the best matches for each word in order to have the best context later.

4. The generated set of words will be the input to LM. This step is to determine the context of the sentences, the best matching words was found using Bayesian rule, see section 2.5 for more details. The output will be the text.

2.1 Feature extraction

The feature extraction phase transforms the signal of speech into feature vectors. The most used feature extraction techniques are; MFCC, Fbank and PLP. Each one will be explained in the following subsections.

2.1.1 MFCC

Now we will explain how MFCC features extraction technique works. Figure 2.1.1 shows MFCC diagram and how speech signal is transformed into feature vectors. Next, we explain each step [20, 33]:

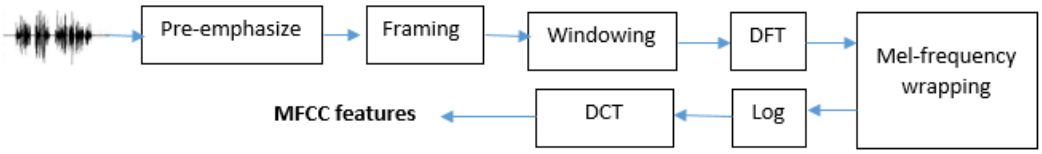


Figure 2.1.1: MFCC diagram

1. Pre-emphasize: The transformation of the signal starts with applying a high pass filter to emphasize higher frequencies as shown in equation 2.1.1, where α is a parameter and its value varies between 0.95 and 0.99. High pass filter trying to separate the speech from noise.

$$HP(z) = 1 - \alpha z^{-1} \tag{2.1.1}$$

2. Framing: The signal from the previous step is segmented into short period blocks of 20-30 ms called frames. Frame rate is the rate of capturing the signal and usually it is 10ms. There is an additional technique used which is overlapping; this technique is used to reduce the lost data by the frame-capturing rate. Figure2.1.2 explains this technique.

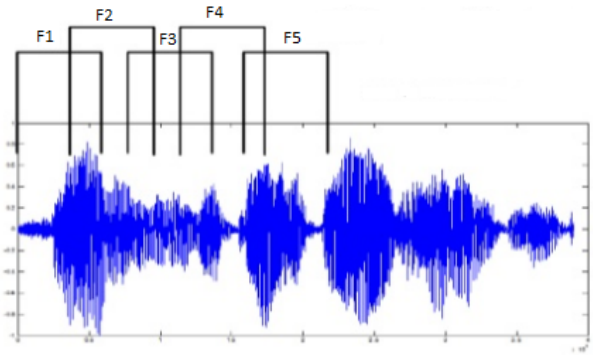


Figure 2.1.2: Overlap framing to reduce the data lose[34]

3. Windowing: Usually hamming window used to keep continuity of the resulted frames from the previous step. Therefore, each frame is multiplied with the

hamming window. So if we present the frames with f(n), where n is the number of the frame $n = \{1, ..., N\}$, so the result will be $W(\alpha) * f(n)$. The result of this multiplication is resulted in equation 2.1.2. The value of α is set to 0.46 through practice.

$$W(n, \alpha) = (1 - \alpha) - \alpha \cos(\frac{2\pi n}{N - 1}), \alpha = 0.46 \tag{2.1.2}$$

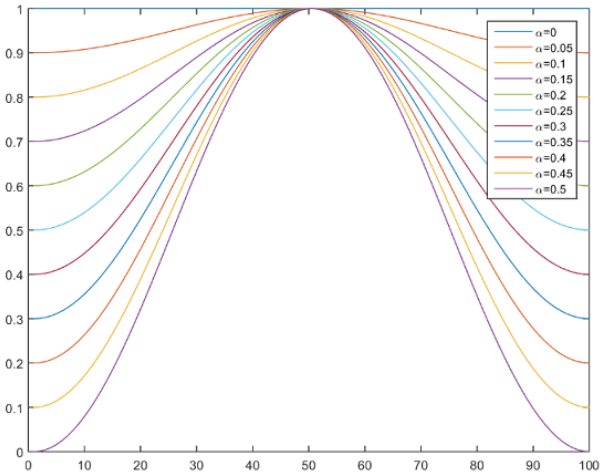


Figure 2.1.3: The hamming window presented with different value of α [35]

4. Fourier Transform: The DFT is short for Discrete Fourier Transform. The resulted frames after the window hamming is transformed from time domain into frequency domain using DFT.

5. Mel-frequency wrapping: The output spectrum from the previous step is transformed into Mel-frequency domain using Mel-scale bandpass filterbank which is shown in equation2.1.3, where $Mel(f)$ is Mel-frequency and f is frequency.

$$Mel(f) = 2595 log_{10}(1 + \frac{f}{700}) \tag{2.1.3}$$

Mel-scale bandpass filterbank consists of number of triangular bandpass filters,

those filters are used to reduce the feature size and to smooth the spectrum. Let we denote the triangular bandpass filter with $H_m(n)$ where m is the number of filter bank channels $m = \{1, 2, ..., M\}$, n is the frame number. The log spectral energy vector X_m can be calculated as shown in equation 2.1.4, where X_n is the power spectrum at frame n.

$$X_m = \sum_{n=1}^{N} ln[|X_n|^2 H_m(n)] \tag{2.1.4}$$

6. Log: Log function is applied to compress the spectral energy vector resulted from previous step.

7. DCT: which is short for Discrete Cosine Transform. Applying DCT to the compressed spectral energy vector will reduce the correlation between the different components of an acoustic vector. Equation2.1.5 showed the generated MFCC coefficients. Where c(i) is the i-th MFCC coefficient. The number of MFCC coefficients are 13.

$$c(i) = \sqrt{\frac{2}{M}} \sum_{m=1}^{M} X_m \cos(\frac{\pi i}{M}(m - 0.5)) \tag{2.1.5}$$

2.1.2 Fbank

Filter bank is another feature extraction technique. In addition, it is almost the same as MFCC but without applying DCT. This means that fbank is higher dimensional than MFCC. This technique is used with DNN based model since the correlation between the different components of an acoustic vector is not a problem for DNN based model.

Figure 2.1.4: Filter bank diagram

2.1.3 PLP

PLP is another technique used for feature extraction. Figure2.1.5 shows how PLP converts speech into feature vector. Now let's explain this transformation[36]:

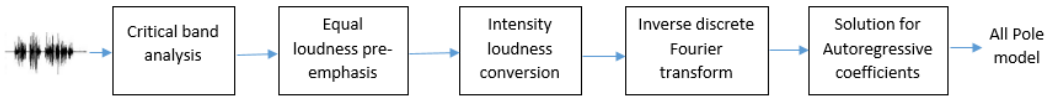


Figure 2.1.5: Perceptual Linear Prediction (PLP) diagram[36]

1. Spectral analysis: The analysis of the spectrum is similar to MFCC in this stage, so it applies equation2.1.2 for hamming window, then DFT is applied as well but it is more common to apply Fast Fourier Transform FFT. However, the power spectrum is produced in a different perspective as shown in equation2.1.6. Where $Re[S(\omega)]$ is the real component of short term speech spectrum, $IM[S(\omega)]$ is the imaginary component of the short speech spectrum and ω is the angular frequency in rad/sec, and can be calculated using equation 2.1.7.

$$P(\omega) = Re[S(\omega)]^2 + Im[S(\omega)]^2 \tag{2.1.6}$$

$$\omega = 1200\pi \sinh\left(\frac{\Omega}{6}\right) \tag{2.1.7}$$

2. Critical band analysis: The power frequency is wrapped into Bark frequency Ω as shown in equation 2.1.8

$$\Omega(\omega) = 6ln\left(\frac{\omega}{1200\pi} + \sqrt{(\frac{\omega}{1200\pi})^2 + 1}\right) \tag{2.1.8}$$

Then the Bark frequency is convolved with the power spectrum of the simulated critical-band masking curve $\Psi(\Omega)$. The critical band- curve is given by

equation 2.1.9.

$$\Psi(\Omega) = \begin{cases} 0 & \text{for } \Omega < -1.3 \\ 10^{2.5(\Omega+0.5} & \text{for } -1.3 \leq \Omega \leq -0.5 \\ 1 & \text{for } -0.5 \leq \Omega \leq 0.5 \\ 10^{-1(\Omega-0.5)} & \text{for } 0.5 \leq \Omega \leq 2.5 \\ 0 & \text{for } \Omega > 2.5 \end{cases} \tag{2.1.9}$$

The discrete critical- band masking curve $\Psi(\Omega)$ with the Power spectrum $P(\omega)$ gives critical-band power spectrum, which is described clearer in equation 2.1.10

$$\theta(\Omega_i) = \sum_{\Omega=-1.3}^{2.5} P(\Omega - \Omega_i)\Psi(\Omega) \tag{2.1.10}$$

3. Equal-loudness pre-emphasis: At this step, modeling human hearing process and taking care of higher frequency that the human can analyze. This process is denoted by $\Xi(\Omega(\omega))$, and can be found using equation 2.1.11, where $\Theta((\Omega(\omega))$ is the pre-emphasized sample by the equal loudness curve, and $E(\omega)$ is an approximation to the unequal sensitivity of human hearing at different frequencies.

$$\Xi((\Omega(\omega)) = E(\omega)\Theta((\Omega(\omega)) \tag{2.1.11a}$$

$$E(\omega) = \frac{(\omega^2 + 56.8 \times 10^6)\omega^4}{(\omega^2 + 6.3 \times 10^6)^2 \times (\omega^2 + 0.38 \times 10^9) \times (\omega^6 + 9.58 \times 10^26)} \tag{2.1.11b}$$

4. Intensity-loudness power law: Amplitude compression is taking place to simulate Stevens law of hearing[37]. This compression is also applied to reduce the spectral amplitude variation of the critical band spectrum. Equation 2.1.12

shows the compression process.

$$\Phi(\Omega) = \sqrt[3]{\Xi(\Omega)} \tag{2.1.12}$$

5. Inverse Discrete Fourier Transform IDFT and Autoregressive Coefficients: IDFT is the auto-correlation method used in PLP, and it is applied to $\Phi(\Omega)$. The first M+1 auto-correlation values are used for auto-regressive coefficients of All-pole-model

6. Note to be considered in practice: The convolution and the pre-emphasis are carried out for each sample of $\Xi(\Omega_k)$ in the $P(\omega)$ domain by one weighted spectral summation per spectral sample $\Xi(\Omega_i)$. The spectral sample can be calculated using equation 2.1.13, where W_i is the weighing function.

$$\Xi(\Omega(\omega_i)) = \sum_{\omega=\omega_{i1}}^{\omega_{ih}} W_i(\omega)P(\omega) \tag{2.1.13}$$

As mentioned earlier that MFCC and PLP are the most common used techniques for feature extraction. Choosing between them depends on the task. Honig et al[32] provide a comparison between MFCC and PLP, and tried to enhance the performance of both (MFCC and PLP) by carrying on the best features from both to provide a better technique.

2.2 Phonetic dictionary

It is the same as pronunciation dictionary. As we mentioned earlier that each phoneme has several phones. This dictionary has the possible phones for each phoneme. It is used to match the possible sequence of phonemes, which are used in composing the words from selected phones. Phonetic Dictionary is the only connection between HMMs and language model. Qatar Computing Research Institute

(QCRI)[31] released an Arabic phonetic dictionary for Modern Standard Arabic ASR on 17 March 2014, and yet no newer version of phonetic dictionary is published. Figure 2.2.1 shows a screenshot from QCRI lexicon.

```
$A$ $ A $
$A$ $ A $ a
$A$A $ A $ a n
$A'A $ A G A
$A*A $ A V A
$A*A $ A V a n
A$Ad G a $ A d
A$Ad G a $ A d a
A$Ad G u $ A d
A$Ad G u $ A d u
A$Adh G i $ A d a
A$Adh G i $ A d a h
A$Adh G i $ A d a t
A$Adh G i $ A d a t u n
A$AEAt G i $ A E A t i n
A$AEAt G i $ A E A t u n
A*AEh G a V A E a h
A*AEh G a V A E a h u
A*AEp G i V A E a
A*AEp G i V A E a h
A*AEp G i V A E a p
A*AEp G i V A E a p a
A*AEp G i V A E a p i
A*AEp G i V A E a p u
A*AEp G i V A E a t
```

Figure 2.2.1: Sample of QCRI phonetic dictionary

2.3 Language Model (LM)

As we described earlier, ASR generates list of words. But how we can select the best match of consecutive words. This is the main job of the Language Model (LM). LM doesn't rely on AM, however it is created by finding the probability of different word strings of the studied dialect (in our case). The model that assign these probabilities to sequence of words is the N-gram, where N is the number of the sequence of words,

the most used N is N=1, 2, 3. When N=1, we got 1-gram which is known as uni-gram, and when N = 2 or 3 it is known as pigram or trigram respectively, see table 2.1 to differentiate between the N-gram LM. Therefore, we need to calculate the probability of a word in given utterance. Let us have the following example: P(مدرسة|مدرسة النظامية هي مدرستي), this example calculates the probability of word "مدرسة" in the given utterance "مدرسة النظامية هي مدرستي" which is 0.25. This simple example for unigram is applied to LM with only the given small sentence. But if the question is what is the probability to have "التي" after the given utterance above. We need to check a big dataset and check for the same utterance and how many times التي comes after it[38].

In general the probability of each word w_i in N-gram LM is depending on the N-1 words prior this word as shown in equation 2.3.1, where $N < k + 1$.

Table 2.1: An example on N-gram language model

N-gram	Known as	How this N-gram LM looks like	P(I,love, my, daughter)
1-gram	Uni-gram	s, I , love , my , daughter, s	P(I) P(love) p(my) p(daughter)
2-gram	Bigram	s I, I love, love my, my daughter, daughter s	P(I|s) P(love|I) p(my|love) p(daughter|my) p(s|daughter)
3-gram	Trigram	s I love, I love my, love my daughter, my daughter s	P(I|s,s) P(love|s,I) p(my|I,love) p(daughter|love,my) p(s|my,daughter)

Note that s is a symbol added at the beginning and the end of the sentence.

$$p(w) \simeq \prod_{k=1}^{K} p(w_k|w_{k-1}, w_{k-2}, ..., w_{k-(N-1)}) \tag{2.3.1}$$

For three words w_k, w_{k-1}, w_{k-2}, let count $C(w_{k-2}, w_{k-1})$ represents the number of occurrences of the two words w_{k-2}, w_{k-1} and $C(w_{k-2}, w_{k-1}, w_k)$ represents the occurrences of the three words w_{k-2}, w_{k-1}, w_k, the probability of occurrence of w_k given w_{k-1} and w_{k-2} can be calculated using equation 2.3.2.

$$\hat{p}(w_k|w_{k-1}, w_{k-2}) = \frac{C(w_{k-2}, w_{k-1}, w_k)}{C(w_{k-2}, w_{k-1})} \tag{2.3.2}$$

There is one problem facing this estimate, which is the out-of-vocabulary OOV words. OOV is the terms or words that exist in the testing dataset but did not appear in the training dataset. A modification should be done to equ 2.3.2 to solve this problem. This modification can be done by applying backing-off and discounting parameters[39, 40]. The modified calculation is shown in equation 2.3.3, where d is the discount coefficient, α is the back-off weight and C' is the count threshold. So when there is a tri-gram sequence $\{w_{k-2}, w_{k-1}, w_k\}$ and their occurrences are less than C', then the probability based on the occurrences of the shorter context $\{w_{k-2}, w_{k-1}\}$, which is referred to backing off.

$$\hat{p}(w_k|w_{k-1}, w_{k-2}) = \begin{cases} \frac{C(w_{k-2}, w_{k-1}, w_k)}{C(w_{k-2}, w_{k-1})} & C(w_{k-2}, w_{k-1}, w_k) > C' \\ d\frac{C(w_{k-2}, w_{k-1}, w_k)}{C(w_{k-2}, w_{k-1})} & 0 < C(w_{k-2}, w_{k-1}, w_k) \leq C' \\ \alpha(w_{k-2}, w_{k-1})\, p(w_k|w_{k-1}) & otherwise \end{cases} \tag{2.3.3}$$

2.4 Acoustic Model (AM)

All proposed models in AM have Hidden Markov Model HMM as main part of them. That is because all Large Vocabulary Continuous Speech Recognition (LVCSR)

systems are based on HMM. HMM is able to characterize the observed time-varying speech data sample[20, 41].

Figure 2.4.1 shows a five state Markov chain, but S_0 and S_4 are start and end states respectively, and they are non emitting states. Each state corresponds to an observable event q at given time t. The process starts from the start state and transitions successively to the rest of the state. A probability function is assigned to every transition called transition probability. Also as shown in figure2.4.1, there is an output probabilities $f_i(O_t)$, where O is an observation sequence given as $O = \{o_1, o_2, ..., o_T\}$ at each given time t.

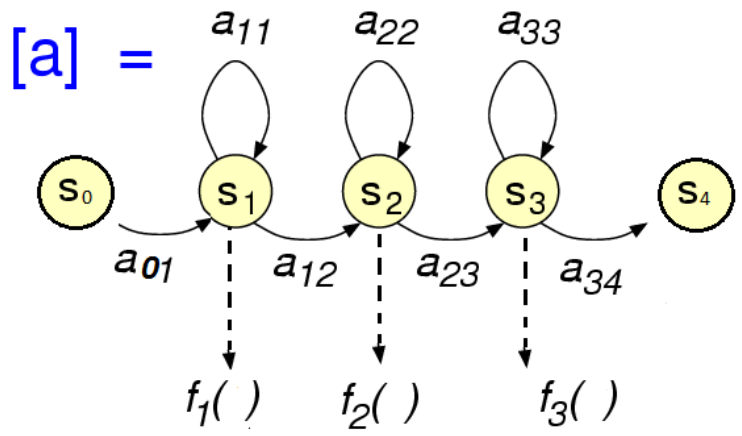


Figure 2.4.1: Hidden Markov Model (HMM)[42]

The HMM parameters for a given model are[43]:

1. Initial state occupation: $\pi = \{\pi_i\}$, is an initial state occupation probability vector. In other words it is the probability at the first state i as shown in equation 2.4.1.

$$\pi_i = p(q_0 = s_i) \quad where\ 1 \leq i \leq N\,;\ and\ N\ is\ the\ total\ number\ of\ states \tag{2.4.1}$$

2. Transition probability: All transition probabilities are defined by matrix A.

$A = a_{ij}$ can be calculated as shown in equation 2.4.2, where time t at state i is $t-1$ and t at state j. As it is probability, all probability rules are applied such as $a_{ij} \geq 0$ and $\sum_{j=1}^{N} a_{ij} = 1$, where N is the total number of states.

$$a_{ij} = p(q_t = s_j | q_{t-1} = s_i) \tag{2.4.2}$$

3. Output probability: is set of observation Probability Density Functions (PDFs). $f_i(O_t)$ is a set of PDFs output, where the observation PDF given HMM state i for D-dimensional observation vector o_t. For more details on HMM, check [43, 41]

2.4.1 HMM-GMM

As described earlier in the last section, HMM proved its ability of observing time-varying speech data sample. GMM is one of the most important probability density functions applied to continuous measurements, such as speech feature vector. The output probabilities are expressed as a weighted sum of M Gaussian component densities. So multivariate Gaussian mixture distribution is used to describe $f_i(O_t)$ as shown in equation 2.4.3, where T is a symbol means transpose operation, $c_{i,m}$ is the weight for state i, $\mu_{i,m}$ is the mean vector and $\Sigma_{i,m}$ is the covariance matrix for each mixture component, $m = 1, 2, ..., M$. $c_{i,m}$ are positive and satisfy the constraint $\sum_{m=1}^{M} c_{i,m} = 1$.

$$f_i(O_t) = \sum_{m=1}^{M} c_{i,m} \mathcal{N}(o_t | \mu_{i,m}, \Sigma_{i,m}) \tag{2.4.3a}$$

$$\mathcal{N}(o_t | \mu_{i,m}, \Sigma_{i,m}) = \frac{1}{(2\pi)^{(1/2)} |\Sigma_{i,m}|^{1/2}} exp\left[-\frac{1}{2} (o_t - \mu_{i,m})^T \Sigma_{i,m}^{-1} (o_t - \mu_{i,m}) \right] \tag{2.4.3b}$$

There are some issues should be discussed about HMMs models before putting them into practical applications[44, 45]:

1. Likelihood evaluation of HMM using the forward algorithm that lets us choose

a model which best matches the sequence of observations. Use the principles of Bayesian rule when we have a given model and a sequence of observations $O = \{o_1, o_2, ..., o_T\}$, in order to find the probability that the observed sequence was produced by model λ which is declared by $p(O|\lambda)$. The total likelihood of model can be estimated using the forward algorithm (probability) $\alpha_i(i)$ and is calculated using equation 2.4.6. The forward probability is computed through time t where $1 \leq t \leq T$ for all states of the Hmm. The backward algorithm is required for estimating the state occupancy's. The backward algorithm (probability)$\beta_i(i)$ is calculated using equation 2.4.7. The backward probability is computed through time t where $T - 1 \leq t \leq 1$ for all HMM states.

2. Estimating the HMM parameters: HMM is adjusted to become $\hat{\lambda}$ in order to maximize the probability $p(O|\lambda)$. The Baum-Welch algorithm2.4.1.1 is used for learning the HMM models to describe the training data and innovate models that fits the training data.

3. Decoding HMM state sequences: Given a model λ and a sequence of observations $O = \{o_1, o_2, ..., o_T\}$, finds the best corresponding single best state sequence $Q = \{q_1, q_2, ..., q_T\}$ that best explains the observations. The Viterbi algorithm is used for decoding HMM state sequences.

2.4.1.1 The Baum-Welch algorithm and HMM parameters

This algorithm is used to solve one of the main issues of HMM, which is estimating the parameters. EM algorithm [46] addresses the problem. The Baum-Welch algorithm is a generalized implementation of EM. For a given model λ, the Baum-Welch algorithm estimates the HMM parameters using the EM iterative process which adjust λ to become $\hat{\lambda}$ in order to maximize the probability $p(O|\lambda)$.

$$p(O|\hat{\lambda}) > p(O|\lambda) \tag{2.4.4}$$

The data of EM consists of sequence of observed data $O = O_1, O_2, ..., O_T$ and sequence of hidden state $Q = q_1, q_2, ..., q_t$. The iterations of EM algorithm leads to the maximum likelihood estimates of model parameters. Each iteration consists of Expectation stage and Maximization stage.

1. Expectation: The posterior probability given the current HMM parameters is computed during the Expectation stage. With the m-th mixture component accounting for state sequence observations $O = \{O_1, O_2, ...O_t\}$, the probability of being in state i at time t is denoted by $\gamma_t(i, m)$ which is shown in equation 2.4.5. Where $\alpha_i(i)$ is the forward probability, $\beta_i(i)$ is backward probability and q_t is a hidden state .

$$\gamma_t(i, m) = \left[\frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}\right] \left[\frac{C_{i,m}\mathcal{N}(O_t|\mu_{i,m}, \Sigma_{i,m})}{\sum_{m=1}^{M} C_{i,m}\mathcal{N}(O_t|\mu_{i,m}, \Sigma_{i,m})}\right] \tag{2.4.5}$$

$$\alpha_t(i) = P(O_1, O_2, ..., O_t, q_t = s_i|\lambda) \tag{2.4.6}$$

$$\beta_t(i) = P(O_{T-1}, O_{T-2}, ..., O_1|q_t = s_i, \lambda) \tag{2.4.7}$$

2. Maximization: At this step the mixture weights, means, and covariances are re-estimated. The re-estimated ratio between the expected number of times the system is in state i using the m-th Gaussian mixture component and the expected number of times the system is in state i is denoted by $\hat{c}_{i,m}$. The re-estimation of the mean vector is denoted by $\hat{\mu}_{i,m}$. The re-estimation of

covariance matrix is denoted by $\hat{\sigma}_{i,m}$

$$\hat{C}_{i,m} = \frac{\sum_{t=1}^{T} \gamma_t(i,m)}{\sum_{t=1}^{T} \sum_{m=1}^{M} \gamma_t(i,m)} \tag{2.4.8}$$

$$\hat{\mu}_{i,m} = \frac{\sum_{t=1}^{T} \gamma_t(i,m) O_t}{\sum_{t=1}^{T} \gamma_t(i,m)} \tag{2.4.9}$$

$$\hat{\Sigma}_{i,m} = \frac{\sum_{t=1}^{T} \gamma_t(i,m)(O_t - \hat{\mu}_{i,m})(O_t - \hat{\mu}_{i,m})'}{\sum_{t=1}^{T} \gamma_t(i,m)} \tag{2.4.10}$$

The last step is to apply all the re-estimated values of $\hat{\mu}_{i,m}$, $\hat{\sigma}_{i,m}$ and $\hat{C}_{i,m}$ in the equation 2.4.3 to find the value of the output PDF and $f_i(O_i)$.

2.4.1.2 The Viterbi algorithm and decoding HMM state sequence

This algorithm is used one issues of HMM, which is approximates the probability $p(O|\lambda)$, where O is a set of observations $O = \{o_1, o_2, ..., o_t\}$ for a given model λ by finding the most likely state sequence $Q = \{q_1, q_2, ..., q_t\}$ that maximizes $p(Q, O|\lambda)$. Equation 2.4.11 finds the maximum probability over all partial state sequences ending in state i at time t and the result is denoted by $\delta_t(i)$.

$$\delta_t(i) = max_{q_1,q_2,...,q_{t-1}} P(q_1, q_2, ..., q_t; q_t = s_i; o_1, o_2, ..., o_t|\lambda) \tag{2.4.11}$$

The following stages should be taken to find the best state sequence according to Viterbi Algorithm [20].

1. Initialization: An initial value is chosen for $\delta_t(i)$ and $\psi_1(i)$ at $t = 1$ for state i.

$$\begin{aligned} \delta_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N \\ \psi_1(i) = 0 \end{aligned} \tag{2.4.12}$$

2. Recursion: For the current state j at time t, an array $\psi_t(j)$ keeps track of the most likely previous state with highest probability.

$$\begin{aligned} \delta_t(j) &= max1 \leq i \leq N[\delta_{t1}(i)a_{ij}]b_j(o_t), 1 \leq j \leq N, 2 \leq t \leq T \\ \psi_t(j) &= argmax1 \leq i \leq N[\delta_{t1}(i)a_{ij}], 1 \leq j \leq N, 2 \leq t \leq T \end{aligned} \tag{2.4.13}$$

3. Termination: At the end of the observation sequence $O = \{o_1, ..., o_T\}$, the best score p and the probability of state q_T is computed as shown by Equation 2.4.14.

$$\begin{aligned} p &= max1 \leq i \leq N[\delta_T(i)] \\ q_T^* &= argmax1 \leq i \leq N[\delta_T(i)] \end{aligned} \tag{2.4.14}$$

4. Path backtracking: At the end of the observation sequence, after backtracking through the most likely predecessor states q_t, the optimal HMM state sequence, Q, is returned.

$$\begin{aligned} q_t &= \psi_{t+1}[q_{t+1}], t = T1, T2, ..., 1 \\ The best state sequence &: Q = q_1, q_2, ..., q_T^* \end{aligned} \tag{2.4.15}$$

2.4.2 Monophone and triphone

We should explain the two expression to understand the procedure better.
So if we use GMM based AM, the initial set of single Gaussian creates a monophone. As we discussed the 5 state HMM, remembering that the first is start and the last is end, which means they are being the silence model. Each state is represented by single Gaussian having mean, variance and mixture weights. More training is done to maximize the probability of the observation sequence. Figure 2.4.2 shows the monophone expansion for the word كتير in the Lebanese dialect which consist of four phonemes each consists of three state HMM with each state being represented by single Gaussian[47].

Figure 2.4.2: Monophone expansion for /k/t/ee/r/ (كتير)

We should know that the single Gaussian does not fit the distributions of feature vectors. So better performance is obtained by having multiple mixture components and retraining. According to Sajjan et al[47] having 8 mixture Gaussian component will represent the feature vectors distribution better than single mixture Gaussian. Since phones vary a lot, and having several phones for one phoneme, and we need to capture all these variations in order to improve the AM performance so triphone models is used.

Triphone HMMs are built by converting monophone transcriptions to triphone transcriptions referred to as word internal and a set of triphone models. Figure 2.4.3 shows the triphone expansion of the word /k/t/ee/r/. In other words , each triphone consists of 3 monophones, but the first and last triphone which has only 2 phonemes from the word and spaceor Sil as appear in figure2.4.3 to separate between words. Also we should know that we have Tied-state-Triphone which is one or more HMMs share the same set of parameters. In this model building process, similar acoustic states of triphones are tied in order to share data to ensure that all state distributions can be estimated[48, 49] . Data-driven and decision trees are the mechanisms which cluster the states.

Figure 2.4.3: Triphone expansion for /k/t/ee/r/ (كتير)

2.4.3 Adaptation techniques used for AM

Adaptation techniques are used to solve the problem of having so many speakers in different situation of recordings(noise/ clear/ outside), also to reduce the mismatch between training and testing data, such as if AM is trained on native speaker dataset, and un-native speaker used the system, this scenario shows high error rate. So the adaptation techniques try to change the model parameter to work with the new speaker parameters. This can be done by using small specific amount of adaptation data for speaker or environment. There are many adaptation techniques used at the AM such as, Maximum Likelihood Linear Regression (MLLR), Maximum A Posterior (MAP), Speaker Adaptive Training (SAT) and Speaker Space methods. Next we describe each technique briefly.

2.4.3.1 Maximum Likelihood Linear Regression (MLLR)

MLLR belongs to the Linear Regression (LR). When MLLR is firstly used, it was only updated the mean value of Gaussian Mixture, but later it was also applied on the variance as well[50]. The main reason of using MLLR is to reduce the difference between the speaker independent mean vector and used model. This can be done by obtaining transformation matrices for the model parameter to maximize the likelihood of the adapted data. $\mu_{i,m}$ in equation2.4.16a means the m-th mean vector for state i in Gaussian density function, A_c is a regression transformation matrix and b_c is a bias vector corresponding to some regression of class c. Also we can re-write equation2.4.16a by changing the transformation matrix A into the extended transformation matrix W_c as shown in equation2.4.16b, where $\xi_{i,m}$ is the extended mean vector.

$$\hat{\mu}_{i,m} = A_c \mu_{i,m} + b_c \tag{2.4.16a}$$

$$\hat{\mu}_{i,m} = W_c \xi_{i,m} \tag{2.4.16b}$$

$$W_c = [b_c, A_c] \tag{2.4.16c}$$

$$\xi_{i,m} = [1, \mu_{i,m}^T]^T \tag{2.4.16d}$$

The adaptation data $O_R = \{o_1, o_2, ..., o_R\}$ with D-dimensional feature vectors is used for training W_c. And then use the new speaker adaptive mean instead of the Gaussian PDFs in equation2.4.3[51, 50]

2.4.3.2 Maximum A Posterior (MAP)

It is another technique to adapt the new dialect with the model by modifying the model parameter. This technique takes advantage of prior knowledge to add limitation on the parameter deviation. This adaptation technique uses Bayesian rules. When you have set of observation $O_R = \{o_1, o_2, ..., o_R\}$, and the MAP estimate of HMM parameter λ, we can find $p(O|\lambda)$. So in case we are given the posterior distribution $p(\lambda|O)$, and need to find the value of λ that maximize the posterior distribution $p(\lambda|O)$, we use the Bayesian rule as shown in equation 2.4.17.

$$\lambda = \hat{argmax_\lambda} p(O|\lambda)p(\lambda) \tag{2.4.17}$$

This approach as said depends on the prior knowledge. The question is what happen in case of there is no prior data available to estimate the model parameter. To solve this issue, the prior knowledge combined with the MAP adaptation. The Expectation Maximization (EM) algorithm is used as maximum likelihood (ML) to estimate λ. Equation 2.4.18 defines the MAP update formula for state i and mixture component m and and observation sequence $O = \{o_1, o_@, ..., O_R\}$, where $\mu_{i,m}$ is the prior mean, τ is the balancing factor between ML mean estimate and the prior mean, $\bar{\mu}_{i,m}$ is the mean of the adaptation data and $N_{i,m}$ is the probability of occupying the

mth Gaussian mixture component of state i at time t for R observations.

$$\hat{\mu}_{i,m} = \frac{N_{i,m}}{\tau + N_{i,m}} \bar{\mu}_{i,m} + \frac{\tau}{\tau + N_{i,m}} \mu_{i,m} \tag{2.4.18a}$$

$$N_{i,m} = \sum_{r=1}^{R} \sum_{t=1}^{T_r} \gamma_t^r(i,m) \tag{2.4.18b}$$

Goronzy et al[52] talked about new approach which combined MLLR and MAP. They found that this combination reduced the error rate by 38% compared to the speaker Independent (SI) system.

2.4.3.3 CMLLR

CMLLR is short for Constrained Maximum Likelihood Linear Regression. Its main approach is to apply adaptation transform to observation data instead of model parameters. They find that this method reduced the runtime compared with the full variance transform. CMLLR estimate the mean and variance using the same transform. Equations 2.4.19 and 2.4.20 show this transformation, where A_c is the constrained transformation, Σ_{ik} is the co-variance matrix and μ_{ik} is the mean vector.

$$\hat{\mu}_{ik} = A_c \mu_{ik} + b_c \tag{2.4.19}$$

$$\hat{\Sigma}_{ik} = A_c \Sigma_{ik} A_c^T \tag{2.4.20}$$

Digalakis et al[53] discussed the performance of CMMLR and MLLR, the performance of both would be similar when the same form of transformation matrix is used. Ferras et al[54] discussed how CMLLR and fMLLR can act the same, and this can happen when single transformation is used the model-space transform.

2.4.3.4 fMLLR

fMLLR is short for feature-space Maximum Likelihood Linear Regression. fMLLR is used to discover a linear transform of an acoustic space to maximize the probability of test data given the speaker independent model. Also we should know that fMLLR is applied directly to the acoustic feature vector without extra computation. fMLLR can be calculated using the same equations of MLLR as shown in equation 2.4.16.

2.4.3.5 Speaker Adaptive Training (SAT)

If we have a training dataset with R speakers, initially MLLR is applied such that the SI Gaussian mean vector μ is mapped to an estimate of SD model for each speaker $\hat{\mu}_r$ in the training set as shown in equation 2.4.21, where A_r is a full matrix and b_r is a bias vector that compromise the speaker specific transformation W_r

$$\hat{\mu}_r = A_r\mu + b_r \tag{2.4.21}$$

From a set of observation sequence for each speaker where x_r is the observation sequence from speaker r, the optimum set of HMM parameters, $\hat{\lambda}$, and the set of speaker transformations $\hat{W} = \{\hat{W}_1, \hat{W}_2, ..., \hat{W}_R\}$ are jointly estimated to maximize the likelihood of the training data as shown in Equation2.4.22. Then the Gaussian means, variance and mixture weights of these models are updated using EM algorithm.

$$(\hat{\lambda}, \hat{W}) = argmax_{\lambda,W} \prod_{r=1}^{R} p(x_r\,; W_r, \lambda) \tag{2.4.22}$$

2.4.4 Neural network background & HMM-DNN

In this section, we introduce DNN. Then we show how DNN got involved in the AM which is known by HMM-DNN. And at the end, we show how DNN is used in different parts of ASR.

2.4.4.1 Deep Neural Network (DNN)

Deep neural networks are set of algorithms, modeled to simulate the human brain neural. It is designed to recognize patterns. It is used for clustering and classifying. Classifying such as: detect voices, identify speakers, transcribe speech to text and recognize sentiment in voices. Clustering such as: search and comparing documents. The capacity of a neural network is decided by its architecture, which are: the number of layers, number of nodes in each layer and the ability of information to travel backward. These nodes are known as perceptron as shown in figure 2.4.4, where $x_i = \{x_1, x_2, ..., x_n\}$ are the input data, $w_i = \{w_1, w_2, ..., w_n\}$ are the weights which allow input to contribute lesser or greater amounts to the sum of input data,$\sum$ is used to indicate that there is addition operation, $f(x)$ is the activation function as shown in equation2.4.23, $f(x)$ is also known as sigmoid function and $y(x)$ is the sum of weighed input data as shown in equation2.4.24[55, 56].

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.4.23}$$

$$y(x) = f\left(\sum_{i=1}^{n} w_i x_i\right) \tag{2.4.24}$$

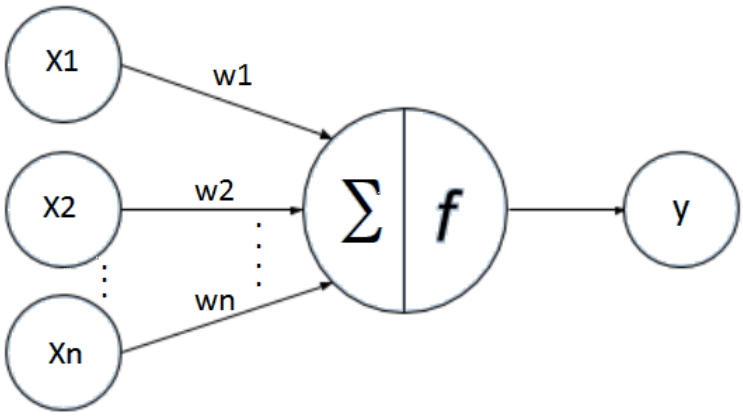


Figure 2.4.4: Basic element of DNN "perceptron"

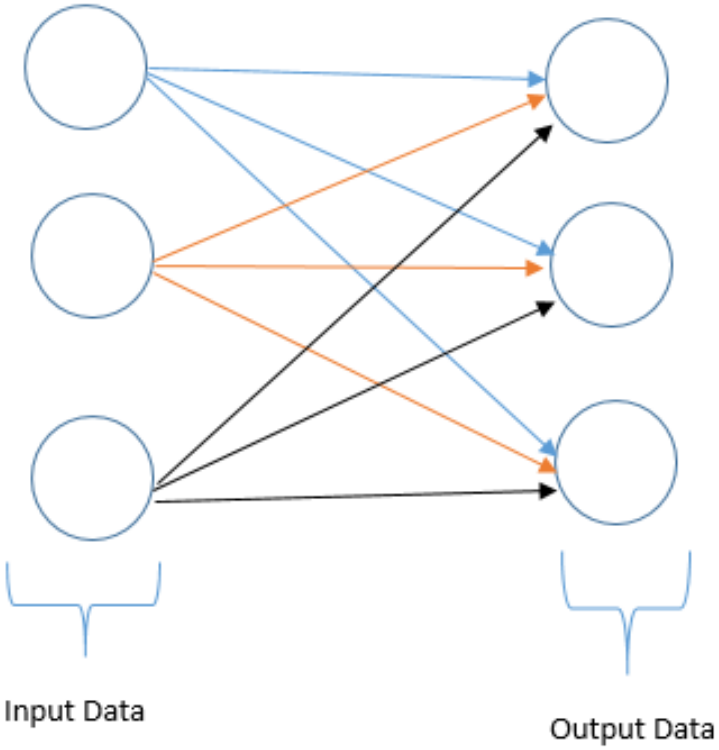
The structure of NN is mainly have two forms:

1. Single- Layer Perceptron (SLP): as shown in figure2.4.5a. SLP consists of input layer that receives the input data, and an output layer that send the output to the user. In SLP which has n inputs, the weights define a hyper-plane with n-1 dimensional space, This only can classify the data into 2 groups only. So SLP is limited with linearly separable data[56].

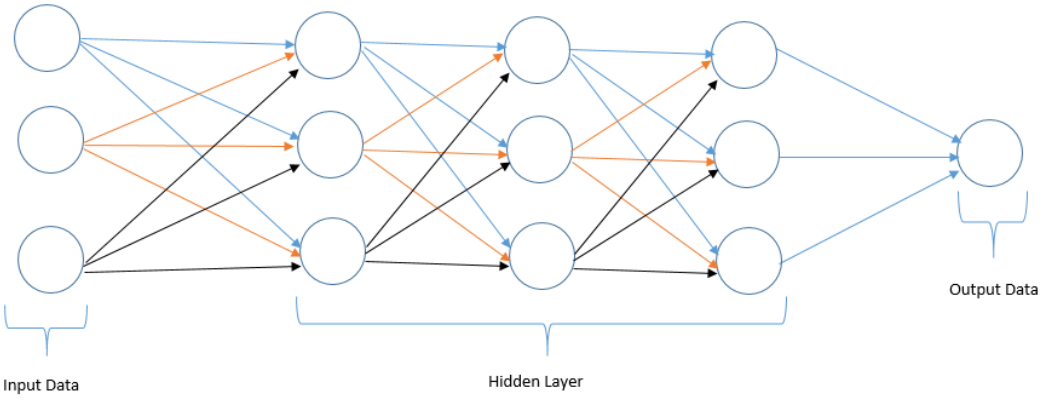
2. Multi-Layer Perceptron (MLP): as shown in figure 2.4.5b. MLP consists of input layer, at least one hidden layer and output layer[56]. During the training process, the input data is a forward direction one layer at a time, then the output of this layer will be input to the next layer. The hidden layer main job is to model non-linear patterns, and we should know that the complexity of the system is increased by increasing the hidden layers.

2.4.4.2 HMM-DNN

The combination of HMM-DNN represent the powerful static classification and modeling sequential patterns. Many years ago NN were introduced to estimate the HMM state-posterior probabilities given the acoustic observations. But when NN is applied to model context-dependent states which uses large amount of training data



(a) Neural network with single hidden layer



(b) Neural network with multiple hidden layer

Figure 2.4.5: Neural network structure

to fine-tune a deep structure with multiple linear layer[57, 58, 59]. Figure 2.4.6 shows the structure of HMM-DNN based acoustic model, where HMM is applied to model the speech dynamics, and DNN's output estimate the posterior probabilities of HMM's tied-triphone. The Viterbi algorithm2.4.1.2 can be applied to train HMM-DNN.

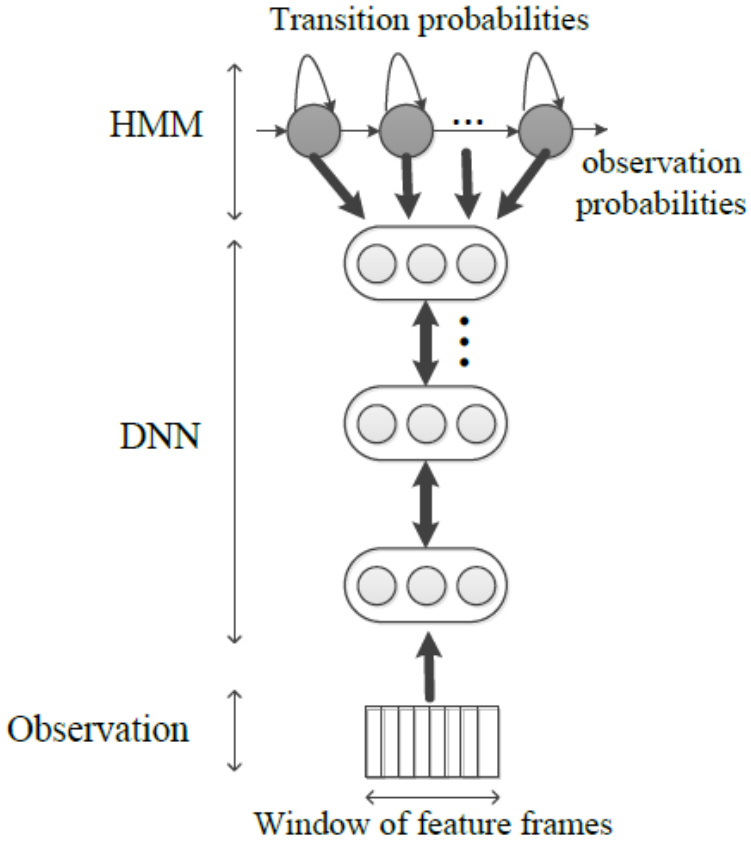


Figure 2.4.6: HMM-DNN structure [55]

2.4.4.3 DNN use in the ASR

After the great throughput of DNN at AM, researchers invest this improvements and popularize the use of DNN technology in other parts of ASR such as, feature extraction and LM. **Bottleneck Features**

Bottleneck feature (BNF) is created using DNN technology. BNF is obtained by training DNN-HMM based AM, in order to set the parameter of DNN, and then used this trained DNN as **feature extraction**. This algorithm improves the performance of speech recognition, enhance the extracted feature and reduce feature dimension. This means that BNF are generated from multi-layer perceptron with hidden layers, usually it consists of five layers and three of them are hidden. At one layer of these hidden layers the number of perceptron is less than other hidden layers, this layer gives the BNF as shown in figure2.4.7. These feature are used as an input to GMM-HMM based AM[28]. Using MFCC combined with BNF enhance the performance of ASR [60]

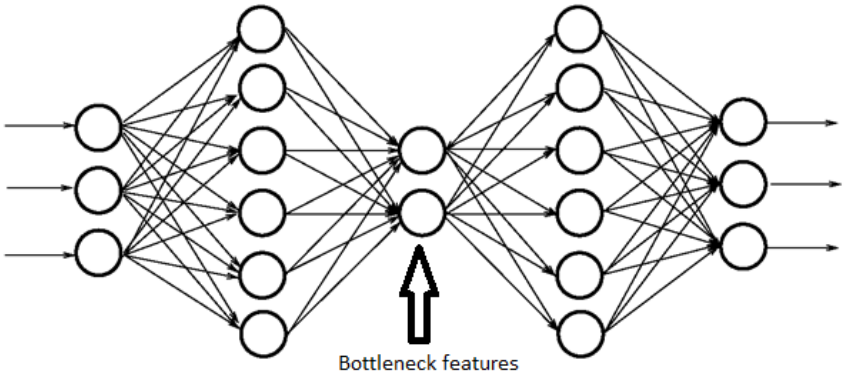


Figure 2.4.7: Bottleneck feature extraction[61]

DNN at the language model

Also After the great improvement at the feature extraction phase, DNN is used in LM as well. How does DNN improve LM. This question well be answered after discussing word embedding matrix. Word embedding matrix is the dictionary of all words used in the given language, each word is mapped to a set of numbers in high-dimensional space as vector representation. We should know that these numbers changed over time. Word embeddings are used as input to Neural Networks NN and with time, NN train itself by encoding rare characteristics such as semantics and contextual information for each word. NN is not only predicting the next

word regardless of the context, it has the ability to give itself feedback from prior experiences. Therefore, it maintains the hidden states as they are changes over different input data. The mentioned hidden states HS are analogous to short-term memory. So HS remembers past experiences, so when it comes to the decision, past experience and current input influence the current answer[62, 63].

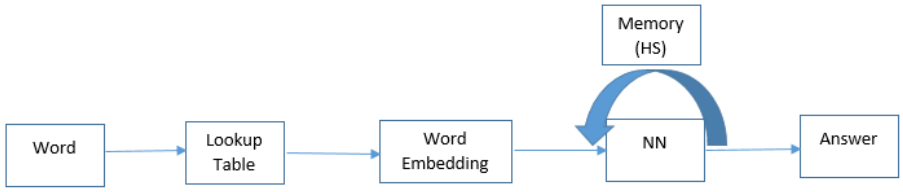


Figure 2.4.8: How NN helps in LM and improves it[63]

2.5 Speech recognition and Bayesian rule

As described in section 2, the speech is converted into acoustics feature vector $O = \{O_1, O_2, O_3, ..., O_n\}$. The ASR will transform this vector into the most likely sequence of words W; $W = \{W_1, W_2, W_3, ..., W_m\}$. In other words ASR will try to find the words given a sequence of feature vector $p(W|O)$. And here Bayesian rule is involved to change $p(W|O)$ into easier form that we can calculate as shown in equation 2.5.1.

$$P(W|O) = P(O|W) * P(W)/P(O) \tag{2.5.1a}$$

$$P(O) = \sum_i P(O|W_i) \tag{2.5.1b}$$

$P(O|W)$ is calculated by Acoustic models (HMM-GMM or HMM-DNN), $P(W)$ is calculated by using LM.

2.6 ASR toolkits

There are many tools used for ASR, such as[64]:

1. Kaldi: Kaldi[65] toolkit is one of powerful tools for speech recognition. It was release on May 14, 2011. It is writen in C++ and had Apache license v2.0. It is open source. According to[66] kaldi is the first recommended toolkit. It is available for windows Operating system and Linux as well. Kaldi can be downloaded from http://kaldi-asr.org/doc/install.html.
   When it is downloaded, we should run make command in the terminal to make sure that all the the commands works correctly. Most of the kaldi projects exists in egs folder, which is one of sub-folders of Kaldi.

2. CMUSphinx: it is a group of speech recognition systems developed by Carengie University. It is written in java code. CMUSphinx can be downloaded from the following link http://cmusphinx.sourceforge.net/wiki/download. It is only available for Linux operating system.

3. HTK: It is short for Hidden Markov Model Toolkit. It is written in C language. HTK can be downloaded from the following link http://htk.eng.cam.ac.uk/download.shtml. It is available for windows Operating system and Linux as well. It is a portable toolkit for building and manipulating hidden Markov models. It is not totally open source.

4. Julius: Is a two-pass large vocabulary continuous speech recognition (LVCSR) engine. First created in 1997 by the Interactive Technology Consortium. The only full model is available in Japanese language, and only a simple AM for English language. It is written in C language. It can be downloaded from the following link $http://julius.osdn.jp/en_index.php?q=index-en.html\#download_julius$. It is available for windows Operating system and Linux as well. It is an open source toolkit.

5. Simon: Is a speech recognition toolkit with graphical user interface. It uses CMUSphinx, HTK and Julius as the basis of the toolkit. It is available for Linux Operating system and there is a running version for windows as well. Simon can be downloaded from the following link https://simon.kde.org/download. It is an open source toolkit.

6. iATROS speech recognizer. It is similar to Julius. It is written in C language. It is only available for Linux operating system. It can be downloaded from the following link https://www.prhlt.upv.es/page/projects/multimodal/idoc/iatros. It uses HMM models.

7. RWTH ASR: It is not an open source toolkit. It is available for Linux platform. RWTH ASR can be downloaded from the following link https://www-i6.informatik.rwth-aachen.de/rwth-asr/. It is written in C++ language. It is available for Linux and Mac operating system.

8. Jasper project: It has a graphical user interface as well as Simon. It is an open source toolkit. It is written in Python Language. It is available for Linux operating system. It can be downloaded from the following link https://jasperproject.github.io/#about

Chapter 3

Methodology and Implementation

In this chapter, we introduce the used toolkit 3.1, datasets 3.2 used in our experiments, and how we prepare our data 3.3. We explain different scenarios 3.4 in order to answer our thesis questions. Also we add more information about our experiments 3.5 and to clarify our naming convention. At the end of this chapter, we list the computer resources used in this experiment 3.6.

3.1 Kaldi toolkit

Kaldi is an open source software (free license), written in C++ and the user used shell commands and scripts in command line prompt. Kaldi provides libraries to support HMM,GMM and DNN systems. Kaldi has a recipe for Arabic ASR with a good WER, which was developed by Ali et al[23], which is considered as baseline for our work with some modifications to serve our need. Kaldi support GPU, which is needed to run all HMM-DNN experiments. In order to run the GPU correctly, we prepared kaldi to work with the GPU through Cuda development toolkit[67], which is integrated with NVIDIA GTX980M graphic card.

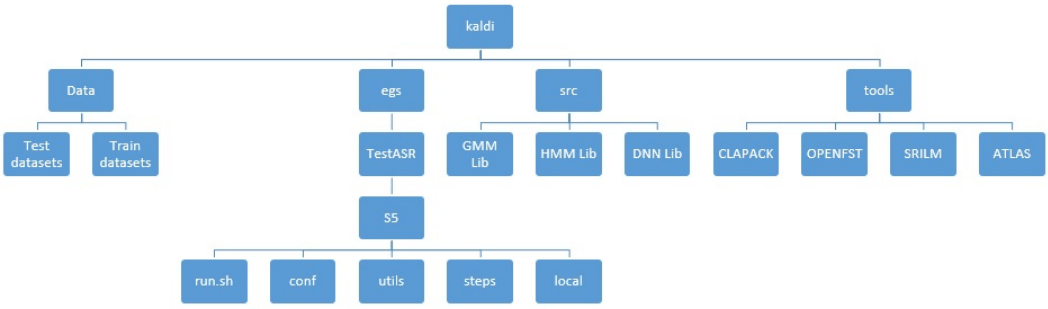


Figure 3.1.1: Kaldi hierarchy and main important folders

To know more about kaldi toolkit. Here is the main main important folders of Kaldi, which you should have when you download it to your computer. As shown in figure 3.1.1 kaldi has four main sub-folders, which are:

1. Tools: This folder contains all tools needed to run an exact recipe, such as SRILM to create Language Model from training dataset. OPENFST for finite state transducers (FSTs) which is used to construct, combine, optimize, and search weighted FSTs. ATLAS which is a big library for matrix calculations. And other tools.

2. src: This folder contains the source code of all libraries that support HMM, GMM and DNN systems.

3. Data: The storage of all dataset used to build ASR.

4. egs: This folder contains all the examples created using kaldi and can be reused for similar problems. In this sub-folder we have our recipe of TestASR. Inside TestASR we should have

    - s5: This sub-folder contains all the executable files for this recipe.

    - conf: short for configurations. This sub-folder contains all the required configurations of MFCC, Fbank, DNN decoding and more.

- steps: This sub-folder is a soft link, refers to Wall Street Journal WSJ recipe, which contains the main steps of creating ASR.
- utils: This sub-folder is a soft link, refers to WSJ recipe, which contains all the computational scripts like make graphs.
- local: This sub-folder contains the executable scripts for the TestASR (in our case).

3.2 Dataset

In this research, we use The following datasets:

1. Gale phase 2 Arabic broadcast news speech part 2 (LDC2015S01)[68]: This part of dataset contains approximately 170 hours of Arabic broadcast news speech. LDC, MediaNet, Tunis, Tunisia and MTC, Rabat, Morocco collected this data in 2006 and 2007. The sample rate for LDC2015S01 is 16000 and it's sample type is PCM.
2. Arabic dialectal dataset (Egyptian): VarDial 2018 program[69] collected and transcribed this dataset on March 12, 2018. Egyptian training dataset is about 12 hours and 23 minutes, almost 3177 utterances. The Egyptian testing data set is about 2 hours, almost 315 utterances. The sample rate for this dataset is 16000 and its sample encoding is 16-bit Signed Integer PCM.
3. Arabic dialectal dataset (Gulf): VarDial 2018 program[69] collected and transcribed this dataset on March 12, 2018. Gulf training dataset is about 10 hours and 9 minutes, almost 2873 utterances. The Gulf testing data set is about 2 hours, almost 265 utterances. The sample rate for this dataset is 16000 and its sample encoding is 16-bit Signed Integer PCM.

4. Arabic dialectal dataset (Levantine): VarDial 2018 program[69] collected and transcribed this dataset on March 12, 2018. Levantine training dataset is about 10 hours and 16 minutes. The Levantine testing data set is about 2 hours. The sample rate for this dataset is 16000 and its sample encoding is 16-bit Signed Integer PCM.

5. Arabic dialectal dataset (North African Arabic): VarDial 2018 program[69] collected and transcribed this dataset on March 12, 2018. North African Arabic training dataset is about 10 hours and 28 minutes. The North African Arabic testing data set is about 2 hours. The sample rate for this dataset is 16000 and its sample encoding is 16-bit Signed Integer PCM.

6. Arabic dialectal dataset (Modern spoken Arabic MSA): VarDial 2018 program[69] collected and transcribed this dataset on March 12, 2018. MSA training dataset is about 12 hours and 24 minutes. The sample rate for this dataset is 16000 and its sample encoding is 16-bit Signed Integer PCM.

7. Language Model: We built 3-gram language model using Stanford Research Institute Language Model (SRILM)[70].

8. Lexicon: we used the lexicon provided by Qatar Computing Research Institute (QCRI)[31].

3.3 Data preparation

Maybe data preparation is one of the hardest stage at all. Our data consists of AUDIO and TEXT files. All the voices in the audio is transcribed, so we can get multiple phones for each phoneme. My job is to uniform all the datasets mentioned earlier3.2. The audio files is converted to wav format with 16000 sample rate.

About text preparations, the used dataset is created to serve different job. So all the dialects are mixed. Therefore, I split each dialect into an independent file using java. In addition, at this step we split the given data into training and testing dataset. This means that we have two raw files for each dialect. Then, we use those files to create the main files of running Kaldi toolkit. The main files are text, utt2spk, spk2utt and wav.scp. We create those files for each training dataset and testing dataset. Java code is used to create those files.

3.4 Methodology

In this section, we list the main scenarios to run the experiments of this research.

3.4.1 Measure how close each dialect to MSA and apply several adaptation techniques

We create an ASR model trained on MSA dataset, and in our case it is Gale dataset, then we use this model to test different dialectal speech one at a time. The tested dialects are Egyptian, Lavantine, Gulf and North African dialects. For each dialect, We apply number of feature exaction, adaptation techniques and optimization techniques to improve the performance and reduce error rate. We consider the results of this model to be our reference point, in order to measure the changes done to improve the accuracy. We name the experiment results after the tested dialect. For example, we call the experiment results Egypt when we tested Egyptian dialect. Therefore, by the end of this scenario we have four different results, which are Egypt, Gluf, NOR and LAV. We have the state-of-art GMM-HMM ASR, DNN-HMM ASR and run each dialect on both system. We add Delta, Delta-Delta, LDA and MLLT to MFCC. Then we apply several adaptation and optimization techniques to study its effect on the performance of our ASR model. All results are shown in chapter 4.

3.4.2 Compare fMLLR, MAP and the combination of both

Some studies show a great improvement of using MAP adaptation on the performance of ASR. And some studies recommend using the combination of MAP and fMLLR. So We decide to study the effect of fMLLR and MAP while using Arabic dataset. We apply fMLLr adaptation on top of LDA, MLLT and SAT, and we apply MAP adaptation using the same settings of fMLLR. Then we combine both fMLLR and MAP to check whether this action improves the performance of ASR or not. MAP and fMLLR are applied to HMM-GMM AM. And we apply these adaptation techniques on all dialects in this research. All results are shown in chapter 4.

3.4.3 Independent model for all dialects

In this scenario, We train our ASR on all dialects (Egypt, NOR, LAV and Gulf) and MSA. And as mentioned earlier 3.5, we call this experiment ALL_tt. As other experiments, we apply all adaptation and optimization techniques, but this time We have a pool of different dialects; which means that we mix the testing files of all dialects in this research, to be tested at the same time on this dialect free ASR.
In order to add more improvements, we add bottleneck feature BNF to this model. Then we compare the results of this model with the best results we got from the earlier models, all results are shown in chapter 4.

3.5 Experiment descriptions

Here we introduce our experiments and declare naming convention as shown in table3.1. In this table we see terms that need explanation as well, which are explained in 3.5.1.

We plan Our experiments to answer our main questions. So we start by measuring how close each Arabic dialect to MSA as shown in our first scenario 3.4.1. Also we compare the performance of HMM-GMM and HMM-DNN. In addition, we compare the performance of ASR while using MAP and fMLLR adaptation techniques, and combination of both as shown in 3.4.2. Then we build Independent Dialect ASR (ID-ASR) as shown in 3.4.3.

We start by selecting the training dataset for each experiment run, table 3.2 shows the utterance number for training and testing for each experiment. Therefore, we train the ASR on one dialect of Egypt, LAV, Gulf and NOR at a time. We have four different training datasets for each tested dialect, If we denote for the dialect by X, we can see X, X1, X2 and X3 in our results. Here is the explanation for each one:

1. X : This model is trained on MSA only and tested X dialect. The size of the training data is 50K utterances.

2. X1: This model is trained on X only and tested X. Our dataset for X is very small (2K to 3K utterances).

3. X2: This model is trained on X + MSA, the size of the training data is between 4K to 5K utterances.

4. X3:This model is trained on X and MSA, the size of the training data is 40K utterances, mostly MSA because of the shortage of dialectal dataset.

Also we train ASR on all dialects and MSA, and we denote this experiment by ALL. ALL_tt means that this ASR is trained on all dialects and had pool testing for all dialects. But ALL_X where X is a dialect, means that this ASR is trained on ALL dialects but had X dialect for testing. We add this step to explain the results and have better look at the each dialect. So we can compare different models that

testing the same dataset.

Table 3.1: The explanation of experiments names

Name	Used techniques	Additional information
combined	——	combine fmllr and MAP
tri1	MFCC+ Delta	• Number of leaves of HMM is 2500. • Total Gaussian number is 30000. • Used the alignment of mono-phone training • tri is short for tri-phone training.
tri2a	MfCC+Delta +Delta - Delta	• Number of leaves of HMM is 3000. • Total Gaussian number is 40000. • Used the alignment of tri1 training.
tri2b	LDA +MLLT	• Number of leaves of HMM is 4000. • Total Gaussian number is 50000. • Used the alignment of tri1 training.
tri2b_mmi	LDA+ MLLT	• make dentals from tri2b • Used the alignment of tri2b training. • The default value of boost is 0.0 . • Decoding iteration 3 and 4
tri2b_mmi_b0.1	LDA+ MLLT	• used the alignment of tri2b training. • The Boost value is 0.1 . • Decoding iteration 3 and 4
tri2b_mpe	LDA+ MLLT	• used the alignment of tri2b training. • Decoding iteration 3 and 4
tri3b	LDA +MLLT + SAT	• Number of leaves 5000. • Total Gaussian number is 100000.

		• Used the alignment of tri2b training.
tri3b_dnn_2048x5	build on tri3b	• DNN with 5 hidden layers • The number of hidden dimension is 2048 • Used MFCC as feature extraction • Learning rate is 0.008
tri3b_dnn_2048x5 _fbank	build on tri3b	• Used Fbank as feature extraction
tri3b_dnn_2048x5 _smb	build on tri3b	• Used MFCC as feature extraction • Apply MPE optimization on DNN model.
tri3b_dnn_2048x5 _smb_fbank	build on tri3b	• Used Fbank as feature extraction • Apply MPE optimization on DNN model.
tri4b	LDA +MLLT+ SAT	used the alignment of tri2b training.

In addition, we need to know that many researchers worked on Arabic Language. Ali et al[23] used kaldi toolkit, and run many experiments to tune DNN. Therefore, they decide the optimal number of hidden layers and hidden dimensions by practice, which is set to five hidden layers and 2048 hidden dimensions. Therefore, we set the values of these parameters as other researchers did.

3.5.1 Important information we have to know about the experiments

Here we explain techniques used in running the experiments within this research, to check how it improves the performance of ASR. We have techniques to improve the feature vector, different adaptation techniques and different optimization techniques.

1. MFCC: Feature extraction method which is explained in section 2.1.1

2. Fbank: Feature extraction method which is explained in section 2.1.2

3. Delta Δ: Delta is the difference between two consecutive raw features. In order to capture time evolution of the spectral content of the signal, we have to add dynamic feature to MFCC feature vector. Delta is calculated from each frame static information. This is used to increase the degree of the feature vector.

4. Delta-Delta: Which is acceleration coefficients that are calculated from Delta. And Delta-Delta is used to increase the degree of the feature vector as well.

5. LDA: Stands for Linear Discriminat Analysis, is a known method used to estimate linear subspace with discriminating properties in order to analyze multiple classes, LDA focus on dimensions that separates classes and orders dimensions according to class separabilty. In other words, LDA is uses to reduce dimensionalty of input features so that samples belonging to the same class are close together but samples from different classes are far apart from each other[71, 72, 73].

6. MLLT: stands for Model-space transforms build on LDA. MLLT takes the output reduced feature space from LDA and derives unique transformation for each speaker which is considered as a step for speaker normalization to minimize the differences among speakers. MLLT Estimates the parameters of

a linear transform in order to maximize the likelihood of the training data given a diagonal-covariance Gaussian mixture models; the transformed features are better represented by the model than the original features[74, 75].

7. MAP (Maximum A Posterior). Explained in section 2.4.3.2.

8. SAT (Speaker Adaptive Training). Explained in section 2.4.3.5.

9. fMLLR (feature-space Maximum Likelihood Linear Regression). Explained in 2.4.3.4.

10. MMI: Stands for Maximum Mutual Information. The mutual information between the acoustic data and its dialect class (in our case). MMI Gives a high discriminating ability to the system. AS discriminative models learn the parameters of a joint probability model so that classification errors are minimised. MMI is used to maximize the conditional certainty of a class of utterances given an observation sequence. MMI is an optimization technique works on sentence level[76].

11. MPE: Stands for Minimum Phone Error. MPE gives a high discriminating ability to the system as MMI, but MPE works on phone level. MPE is an objective function designed for continuous speech recognition[77].

Table 3.2: More details about running experiments; #utterances for training and testing for each experiment

exp name	#Train utterances	#Test utterances
Egypt	49462	3177
Egypt1	3177	297
Egypt2	5665	297
Egypt3	40008	297
LAV	49462	2939
LAV1	2939	327
LAV2	5427	327
LAV3	39889	327
Gulf	49462	2707
Gulf1	2707	259
Gulf2	5195	259
Gulf3	39773	259
NOR	49462	2866
NOR1	2866	346
NOR2	5354	346
NOR3	39853	346
ALL_tt	44264	1229
ALL_Egypt	44264	297
ALL_Gulf	44264	259
ALL_LAV	44264	327
ALL_NOR	44264	346

3.5.2 Word Error Rate - WER

WER is a common metric to measure the performance of the speech recognition system. Equation 3.5.1 shows how WER is calculated, where S is the number of the substitutions, D is the number of deletion, I is the number of insertions, C is the number of corrects and N is the total number of words in the reference where $N = S + D + C$

$$WER = \frac{S + D + I}{N} \tag{3.5.1}$$

$$WER = \frac{S + D + I}{S + D + C} \tag{3.5.2}$$

Once we know the error rate, we can calculate the accuracy as shown in equation 3.5.3, where W_{Acc} is the word accuracy.

$$W_{Acc} = 1 - WER \tag{3.5.3}$$

3.6 Computer resources

The computer resources available for this research are:

1. MSI GT72S 6QE Dominator Pro G: NVIDIA GTX980M G-SYNC 1536 cores, intel core i7; 4 cores 8 threads (8x3600 MHz), 32 GB Ram, 700 GB storage. We name this computer MSI to use later in our discussion.

2. Dell OptiPlex 3010 I5, GPU 1650 cores, 8 GB Ram, 500 GB Storage. We name this computer DELL to use later in our discussion.

Chapter 4

Experiments and Results

In this chapter, we list all our results using different adaptations and different dataset sizes for all dialects, all adaptations and details are mentioned in table 3.1. The results are divided into two parts; Time consuming depending on the computer resources and WER.

4.1 Time consuming depending on the computer resources

In this section, we list more details about the experiments. The details of used computer, start time, end time and consumed timed for each experiment are listed in table4.1. Each dialect have four main experiments by controlling the training data, review 3.4 for details.

The main experiments Egypt, LAV, NOR and Gulf, shown in table4.1 are all run on DELL computer. The ASR is trained on MSA only using Gale phase 2. Egypt experiment is the first experiment to run; start by training the ASR and then decode the Egyptian dialect. This explain why the consumed time for this experiment is

more than LAV, NOR and Gulf. After we get the trained ASR on MSA, we use the same model to decode LAV, NOR and Gulf. The rest of our main experiments mentioned in table4.1 run on MSI computer.

We want to take closer look to Egypt, Egypt3 and ALL_tt, $50K$, $40K$ and $45K$ are the training size of each of them respectively. The consumed time for these experiments in table4.1 is the time for both training the ASR and testing. The consumed time for Egypt is 31.66 days compared with the consumed time for Egypt3 and ALL_tt which are 15.69 and 15.32 days respectively. Consumed time using DELL is almost twice the time used to run the same experiment conditions on MSI. Another example showing the used time on different computers. Experiments run only for decoding, on DELL we have { LAV, Gulf and NOR} and on MSI we have { ALL_Egypt, ALL_Gulf and ALL_LAV}. The consumed time for the three experiments ran on DELL is 38.88 days. And the consumed time for the three experiments ran on MSI is 9.97 days. Here we can tell that the computer qualification affect the computation time.

Table 4.1: More details about running experiments; computer name, start and end time for each experiment

exp name	Computer name	Start time	End Time	Time (minutes)	Time (days)
Egypt	DELL	11 May 7:30am	11 Jun 11:12pm	45,582	31.66
Egypt1	MSI	21 May 9:00am	30 May 10:30pm	13770	9.57
Egypt2	MSI	25 May 5:00am	31 May 10:30pm	9690	6.73
Egypt3	MSI	26 May 7:00am	10 Jun 11:30pm	22590	15.69
LAV	DELL	12 Jun 5:00am	24 Jun 11:12pm	18372	12.77
LAV1	MSI	01 Jun 2:00am	10 Jun 11:45pm	14265	9.91
LAV2	MSI	01 Jun 12:15am	10 Jun 11:45pm	14363	9.98
LAV3	MSI	01 Jun 7:30am	17 Jun 9:22pm	23872	16.58
Gulf	DELL	26 Jun 9:39am	08 July 10:48am	17349	12.05
Gulf1	MSI	19 Jun 8:00am	24 Jun 2:15pm	7575	5.26
Gulf2	MSI	19 Jun 6:00am	24 Jun 12:15pm	7575	5.26
Gulf3	MSI	21 Jun 1:00am	30 Jun 10:44am	13544	9.41
NOR	DELL	09 Jul 7:44am	23 Jul 9:15am	20251	14.06
NOR1	MSI	30 Jun 3:30pm	07 Jul 11:45pm	10575	7.35
NOR2	MSI	30 Jun 4:00pm	08 Jul 1:00am	10620	7.375
NOR3	MSI	01 Jul 11:00am	12 Jul 1:25pm	15985	11.1
ALL_tt	MSI	17 Jul 7:30am	01 Aug 3:00pm	22050	15.32
ALL_TT_Add Bottleneck		11 Sep 11:34pm	19 Sep 8:42pm	11348	7.88
ALL_Egypt	MSI	04 Aug 9:33pm	08 Aug 3:15pm	5382	3.74
ALL_Gulf	MSI	01 Aug 8:40pm	04 Aug 4:55am	3345	2.33
ALL_LAV	MSI	04 Aug 9:51pm	08 Aug 7:30pm	5619	3.9
ALL_NOR	MSI	09 Aug 1:49am	11 Aug 10:16am	3387	2.35
Total Time				317109	220.215

Notes: The following experiments ran only for testing datasets, using exist ASR model. LAV, Gulf, NOR, ALL_Egypt, ALL_Gulf, ALL_LAV and ALL_NOR

4.2 WER for different dialects

In this section, we discuss Word Error Rate for all Arabic dialects that is part of our research.

4.2.1 Egyptian Arabic dialect

We List below the results we get while experimenting Egyptian dialect. We get the best results using HMM-DNN with five hidden layers and 2048 hidden dimensions, then apply MPE optimization. The best results for the different datasets are for the same configurations. From the table 4.2, we can see that when we train the ASR on Gale phase 2 Arabic only and test Egyptian dialect, the WER is 82.62%. However, when we add the Egyptian dialect to the training set, the results get better; the best WER is reduced by 19.62% which become 62.99% compared with the model trained on MSA only.

Why we add the MSA through all the steps of experimenting the dialect? That is because the available dialectal speech of Egyptian dialect is 12.5 hours. But if we get more dialectal speech, the results will enhance more.

Table 4.2: WER of Egyptian dialect with different dataset sizes, different adaptation and optimization

Run type	EYGPT	EYGPT1	EYGPT2	EYGPT3
combined	80.12	86.01	72.45	75.46
tri1	85.6	89.13	78.7	79.83
tri2a	85.7	88.54	77.38	78.85
tri2b	81.14	86.65	74.49	74.5
tri2b_mmi	76.93	89.41	73.42	72.12
tri2b_mmi_b0.1	76.11	88.82	72.73	71.95
tri2b_mpe	77.83	85.97	72.26	71.98
tri3b	79.43	85	72.32	75.42
tri3b_dnn_2048x5	74.09	85.53	70.32	71.29
tri3b_dnn_2048x5_fbank	100.08	90.95	75.39	69.08
tri3b_dnn_2048x5_smb	70.73	82.62	66.24	62.99
tri3b_dnn_2048x5_smb_fbank	100	89.24	73.09	64.85
tri4b	81.17	86.62	74.52	74.59

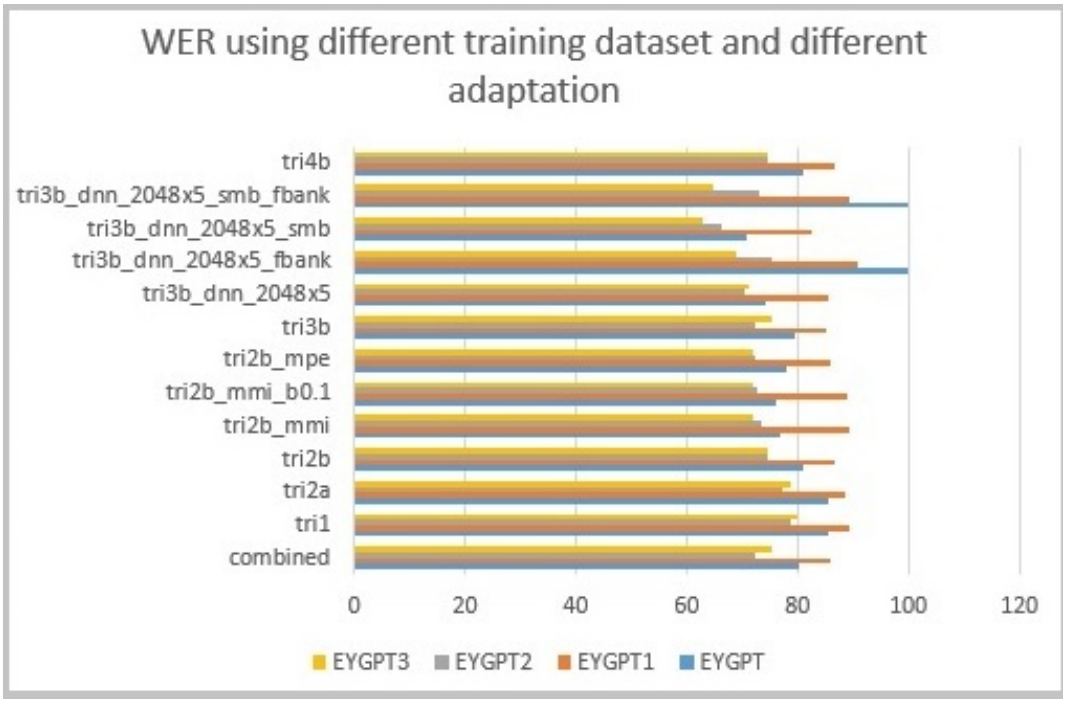


Figure 4.2.1: WER of Egyptian dialect with different dataset sizes, different adaptation and optimization

From the previous table4.2, we want to compare the results of HMM-GMM vs HMM-DNN, the experiment tri2b_mpe which is HMM-GMM with MPE optimization and tri3b_dnn_ 2048x5_smb which is HMM-DNN with MPE optimization. Figure4.2.2 shows that HMM-DNN has better WER than HMM-GMM. Our best results was for EYGPT3 experiment overall our experiments. And HMM-DNN reduced the WER by 8.99% compared with HMM-GMM. We need to mention that the numbers appear on HMM-DNN column in this graph is an absolute difference between HMM-GMM and HMM-DNN results.

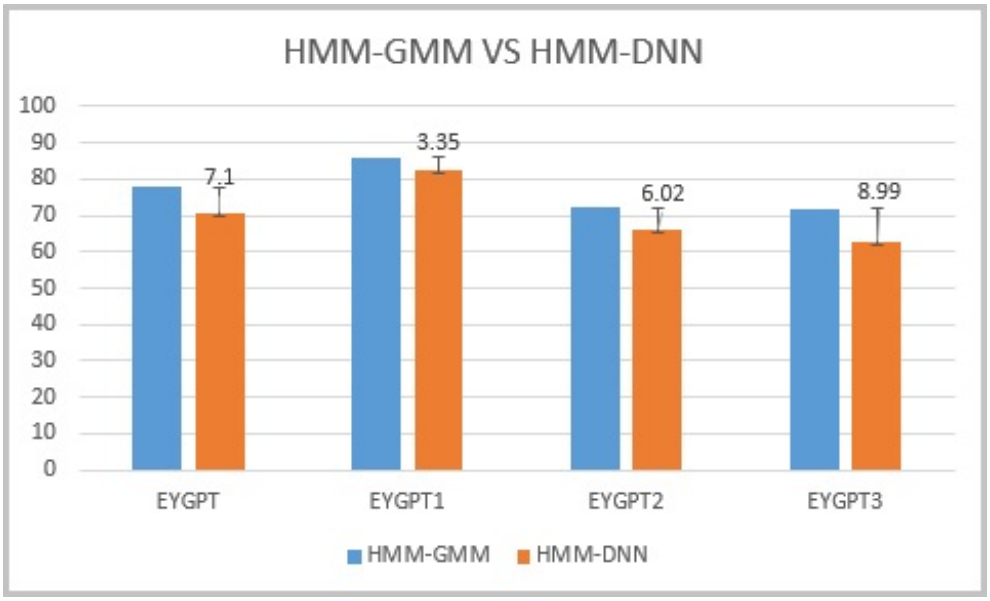


Figure 4.2.2: WER of comparing HMM-GMM and HMM-DNN with MPE optimization for Egyptian dialect

In addition, we need to discuss the difference between training ASR using MSA only and involve the dialect in the training. Figure4.2.3 shows that involving the dialect in training enhance the performance of ASR. The numbers on X-axis are the Run type as shown in table4.2 in the same order. On the other hand we can see that the size of training data of Egypt is greater than Egypt3, this confirms that selected data is much important than the size of the training data.

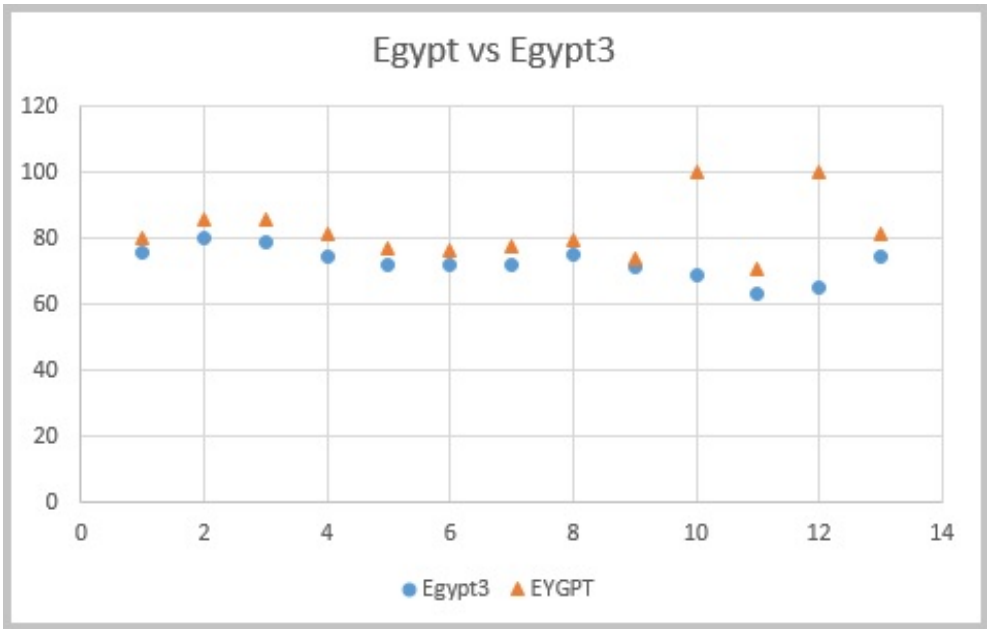


Figure 4.2.3: Compare MSA trained ASR (Egypt) verses dependent dialect ASR (Egypt3)

According to [29] there is a great enhancement of combining adaptation techniques, which are fMLLR and MAP. So we test fMLLR, MAP and combine them to check if this enhance the performance of our ASR. Figure4.2.4 shows the results of testing fMLLR, MAP and combining them. The results show that there is no difference between using either adaptation on its own and there is no enhancement of combining them as well.

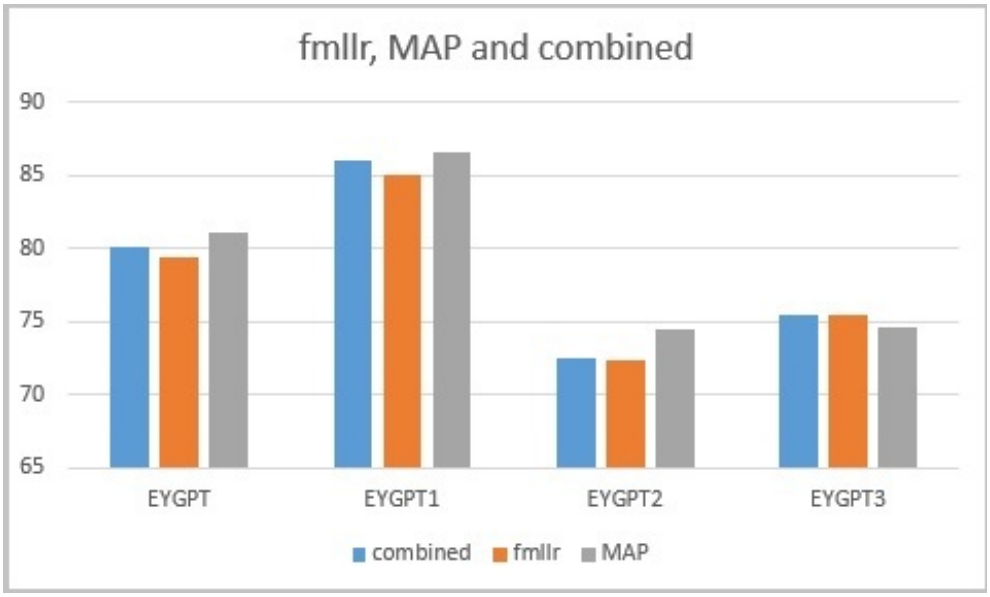


Figure 4.2.4: WER of comparing fmllr, Map and combination of both for Egyptian dialect

4.2.2 Gulf Arabic dialect

We List below the results we get while experimenting Gulf dialect. We get the best results using HMM-DNN with five hidden layers and 2048 hidden dimensions, then apply MPE optimization. The best results for the different datasets are for the same configurations. From the table 4.3, we can see that when we train the ASR on Gale phase 2 Arabic only and test Gulf dialect, the WER was 71.01%. But when we add the Gulf dialect to the training set, the results get better, the best WER is reduced by 16.89% which become 54.12% compared with the model trained on MSA only.

Table 4.3: WER of Gulf dialect with different dataset sizes, different adaptation and optimization

Run type	Gulf	Gulf1	Gulf2	Gulf3
combined	81.12	92.52	68.72	69.53
tri1	87.04	95.74	76.57	75.63
tri2a	86.39	95.27	75.66	74.98
tri2b	81.05	93.96	72.81	69.19
tri2b_mmi	75.61	96.82	71.68	65.75
tri2b_mmi_b0.1	74.52	96.63	70.9	65.74
tri2b_mpe	77.22	94.66	70.95	65.63
tri3b	80.42	92.59	68.58	69.38
tri3b_dnn_2048x5	74.91	91.7	66.2	61.3
tri3b_dnn_2048x5_fbank	99.96	97.62	76.64	62.03
tri3b_dnn_2048x5_smb	71.01	91.06	62.73	54.12
tri3b_dnn_2048x5_smb_fbank	99.95	97.05	71.78	57.69
tri4b	81.07	93.98	72.73	69.09

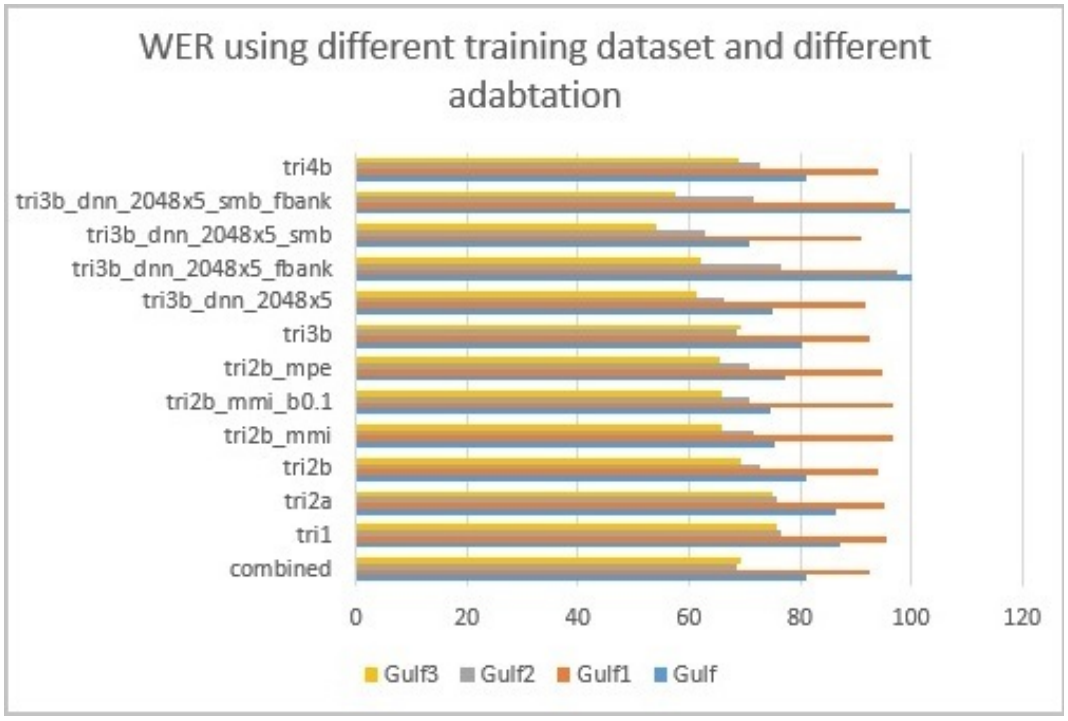


Figure 4.2.5: WER of Gulf dialect with different dataset sizes, different adaptation and optimization

From the previous table4.3, we want to compare the results of HMM-GMM vs HMM-DNN, the experiment tri2b_mpe which is HMM-GMM with MPE optimization and tri3b_dnn_ 2048x5_smb which is HMM-DNN with MPE optimization. Figure4.2.6 shows that HMM-DNN has better WER than HMM-GMM. Our best results was for Gulf3 experiment overall our experiments. And HMM-DNN reduced the WER by 11.51% compared with HMM-GMM. We need to mention that the numbers appear on HMM-DNN column in this graph is an absolute difference between HMM-GMM and HMM-DNN results.

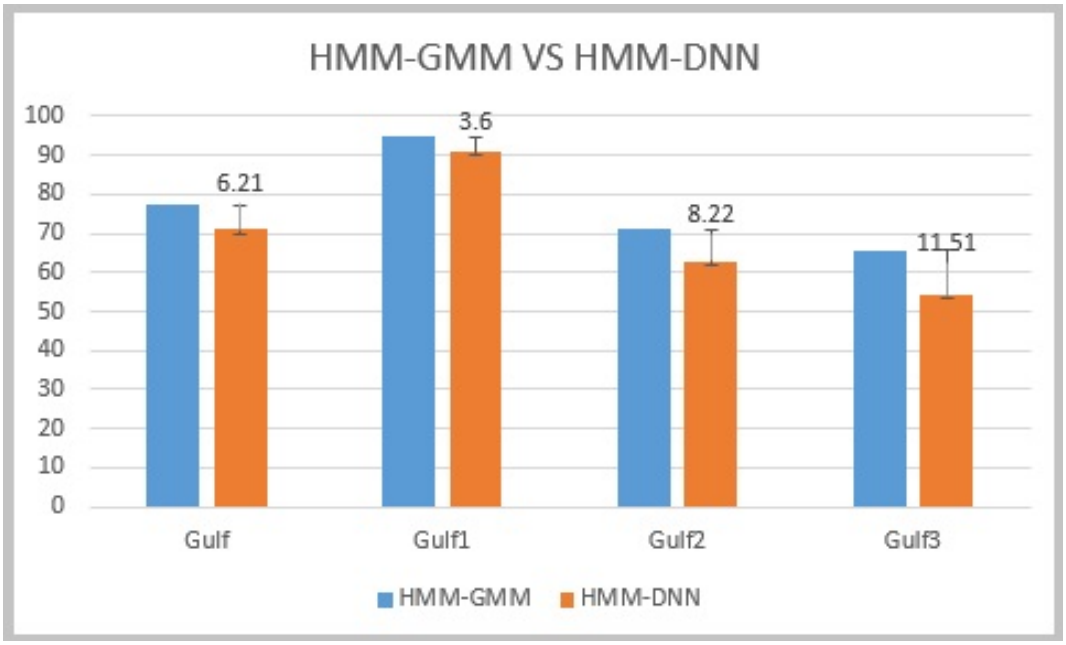


Figure 4.2.6: WER of comparing HMM-GMM and HMM-DNN with MPE optimization for Gulf dialect

In addition, we need to discuss the difference between training ASR using MSA only and involve the dialect in the training. Figure4.2.7 shows that involving the dialect in training enhance the performance of ASR. The numbers on X-axis are the Run type as shown in table4.3 in the same order. Also we can see that the size of training data of Gulf is greater than Gulf3, this confirms that selected data is much important than the size of the training data.

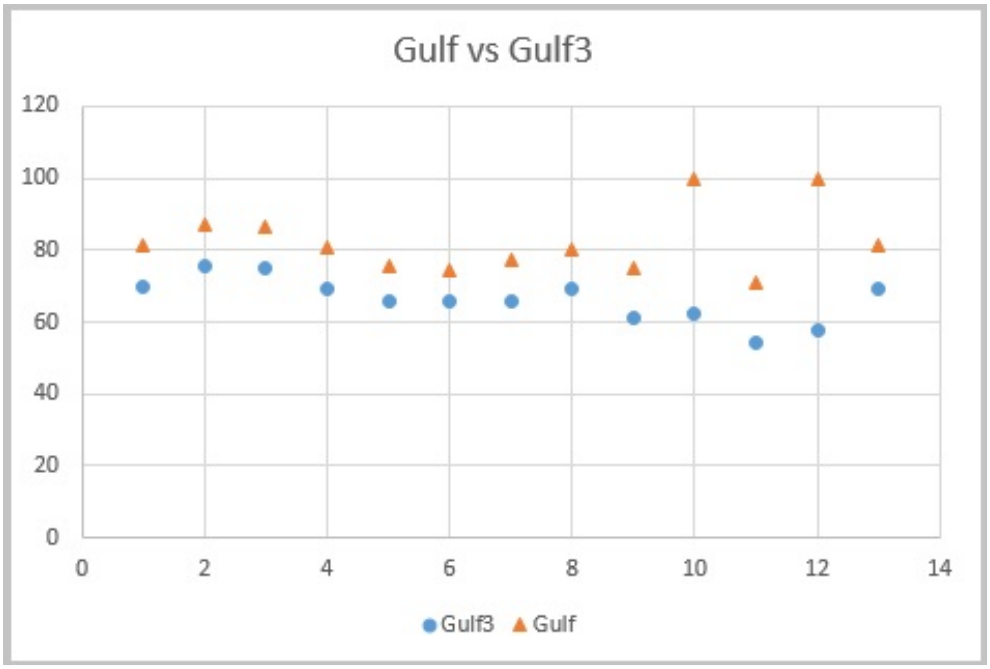


Figure 4.2.7: Compare MSA trained ASR (Gulf) verses dependent dialect ASR (Gulf3)

We test different adaptation techniques when we experiment Gulf dialect. We test fMLLR, MAP and combine both of them to check if these adaptation techniques affect the performance of our ASR. Figure4.2.8 shows that both adaptation has the same affect on our ASR and combining fMLLR and MAP do not enhance the performance as well.

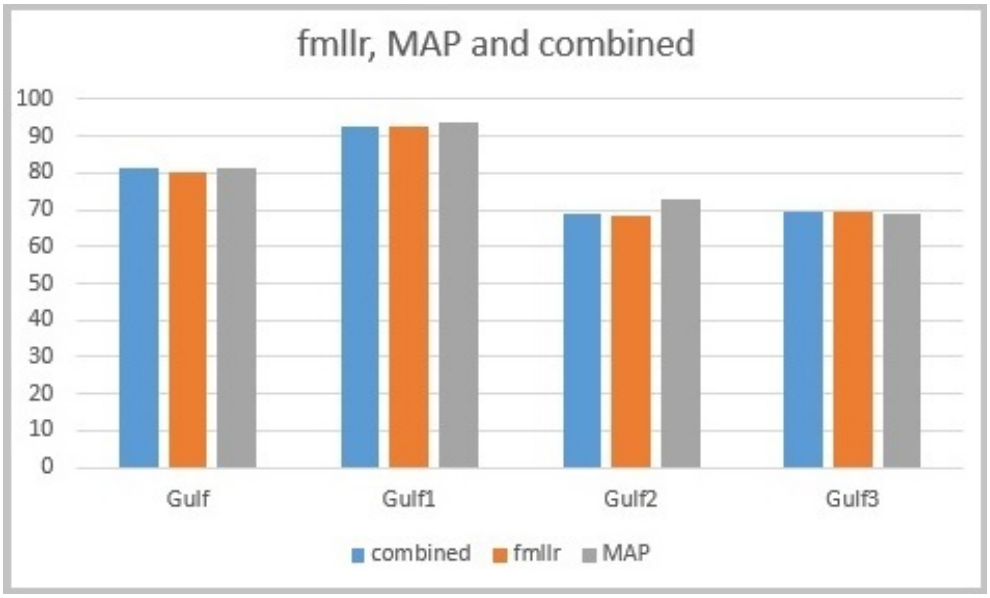


Figure 4.2.8: WER of comparing fmllr, Map and combination of both for Gulf dialect

4.2.3 Laventine Arabic dialect

We List below the results we get while experimenting Laventine dialect. Laventine dialect is complicated and if you try to listen to 15 min speech, you can check the situation here. In our early start of working with Laventine dialect, we change so many parameters trying to reduce WER. We have used different values for Acoustic Scale; used for lattice generation, we used 0.1 , 0.08 and 1.0. We get the best results using 0.1 for Acoustic Scale. In addition, we change the lattice_beam which is by default is 6, we have tested 8 and 10. We use these numbers because of the warning messages we get in the log files. The values of the discussed parameters are the same values for the other tested dialects in this research.

For our different datasets, we get different configuration for the best WER. For LAV, we get the best results while using MMI with 0.1 boost. For LAV1 and LAV2, we get the best results while using HMM-DNN with five hidden layers, 2048 hidden

dimensions, MPE optimization and MFCC as feature extraction. And for LAV3, we get the best results while using HMM-DNN with five hidden layers, 2048 hidden dimensions and MPE optimization but using fbank as feature extraction. From the table 4.4, we can see that when we train the ASR on Gale phase 2 Arabic only and test Laventine dialect, the WER was 90.6%. But when we add the Laventine dialect to the training set, the results get better, the best WER is reduced by 21.48% which become 69.12% compared with the model trained on MSA only.

Table 4.4: WER of Laventine dialect with different dataset sizes, different adaptation and optimization

Run type	LAV	LAV1	LAV2	LAV3
combined	105.42	94.7	81.53	83.61
tri1	97.49	95.43	85.74	83.61
tri2a	97.26	94.58	85.29	82.95
tri2b	99.15	94.12	83.26	78.82
tri2b_mmi	91.91	99.7	82.79	76.63
tri2b_mmi_b0.1	90.6	99.67	82.03	76.72
tri2b_mpe	95.08	98.22	80.36	76.96
tri3b	100.29	94.08	81.24	81.12
tri3b_dnn_2048x5	101.18	95.18	82.67	79.07
tri3b_dnn_2048x5_fbank	99.93	98.27	90.04	74.17
tri3b_dnn_2048x5_smb	97.26	93	76.1	70.71
tri3b_dnn_2048x5_smb_fbank	99.94	97.61	82.84	69.12
tri4b	99.27	94.13	83.28	78.79

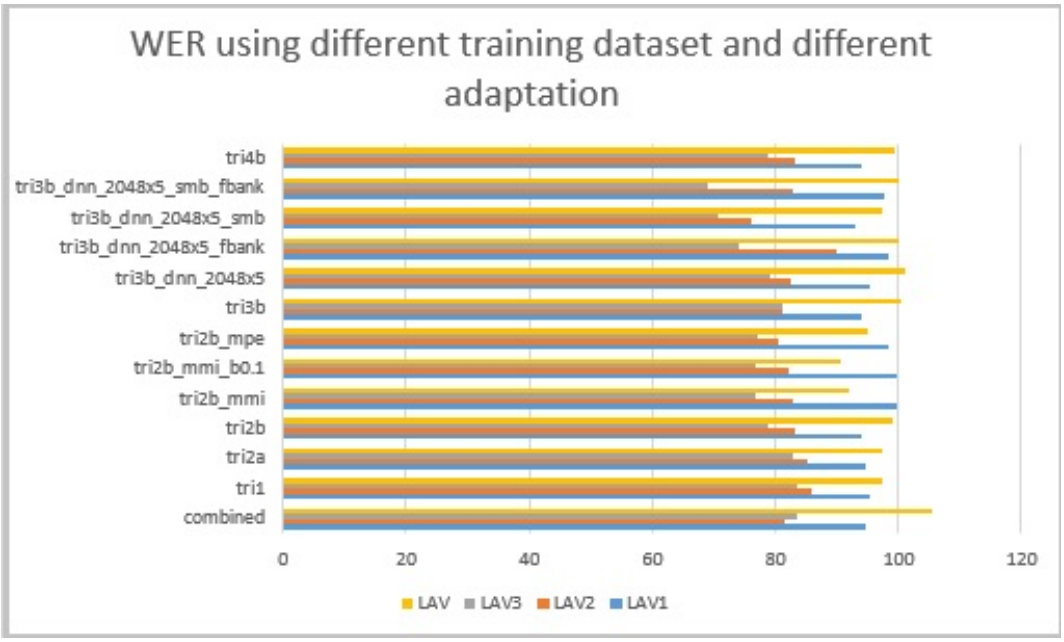


Figure 4.2.9: WER of Laventine dialect with different dataset sizes, different adaptation and optimization

From the previous table4.4, we want to compare the results of HMM-GMM vs HMM-DNN, the experiment tri2b_mpe which is HMM-GMM with MPE optimization and tri3b_dnn_ 2048x5_smb which is HMM-DNN with MPE optimization. Figure4.2.2 shows that HMM-DNN has better WER than HMM-GMM. Our best results was for LAV3 experiment overall our experiments. And HMM-DNN reduced the WER by 6.25% compared with HMM-GMM. About LAV which is MSA only trained ASR, the HMM-DNN do not act as expected because of the complication of the dialect itself, but when we add the Laventine dialect to the training dataset, the performance of the ASR has enhanced. We need to mention that the numbers appear on HMM-DNN column in this graph is an absolute difference between HMM-GMM and HMM-DNN results.

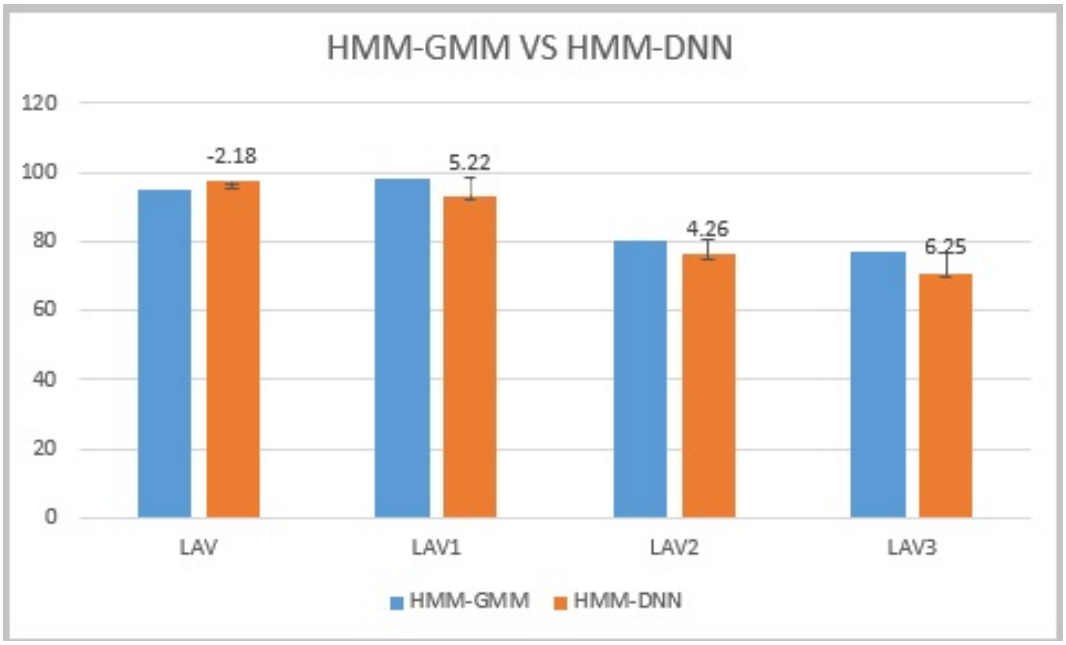


Figure 4.2.10: WER of comparing HMM-GMM and HMM-DNN with MPE optimization for Laventine dialect

Also we need to discuss the difference between training ASR using MSA only and involve the dialect in the training. Figure4.2.11 shows that involving the dialect in training enhances the performance of ASR. The numbers on X-axis are the Run type as shown in table4.4 in the same order. Also we can see that the size of training data of LAV is greater than LAV3, this confirms that selected data is much important than the size of the training data.

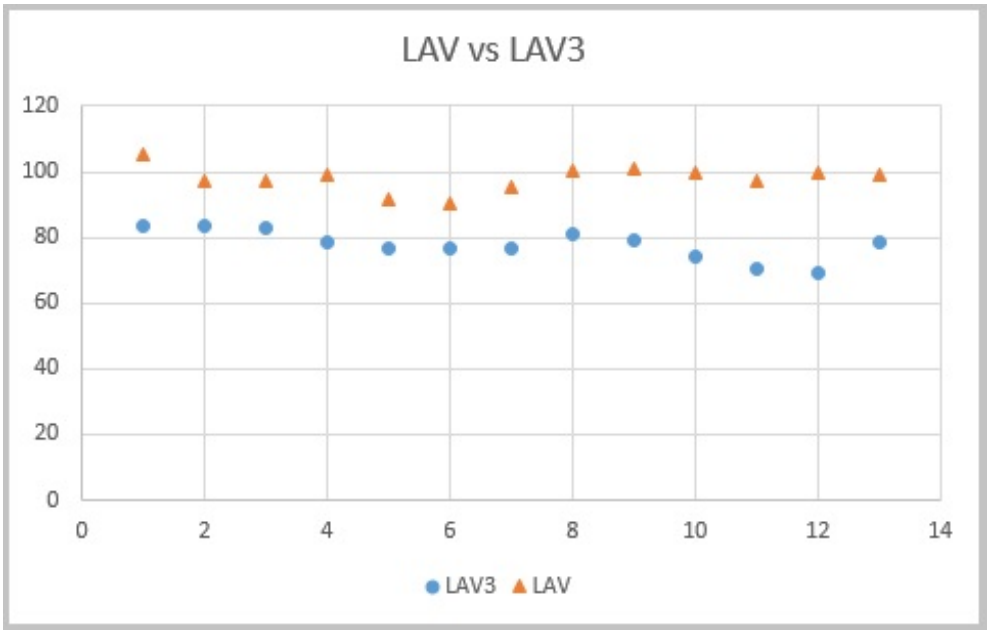


Figure 4.2.11: Compare MSA trained ASR (LAV) verses dependent dialect ASR (LAV3)

We also test different adaptation techniques when we are experimenting Laventine dialect. We have applied fMLLR, MAP and combine both of them to check if these adaptation techniques affect the performance of our ASR. Figure4.2.12 shows that both adaptation has the same affect on our ASR and combining fMLLR and MAP do not enhance the performance as well.

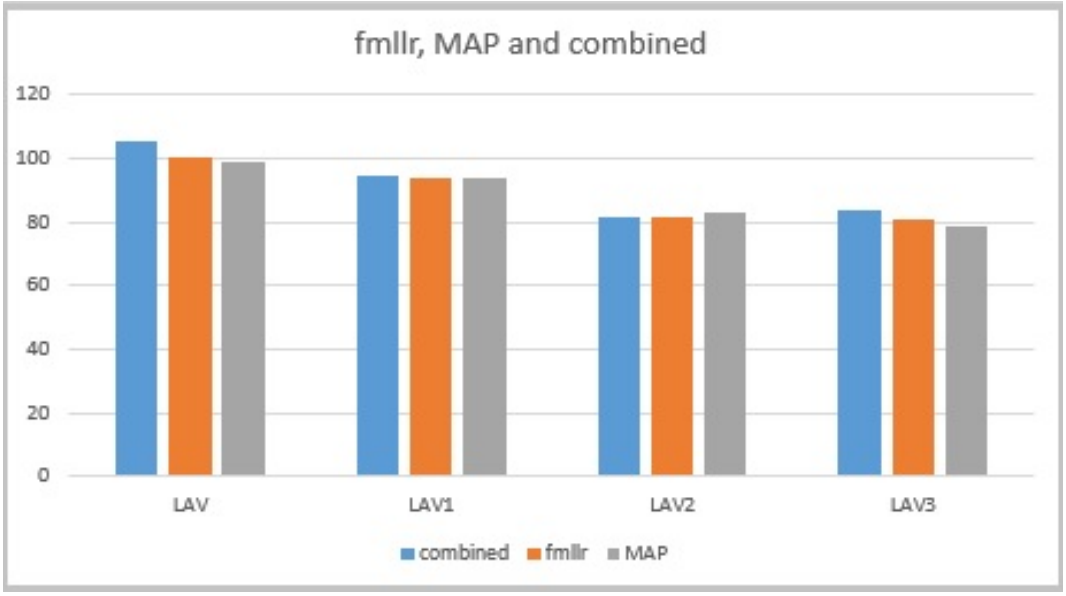


Figure 4.2.12: WER of comparing fmllr, Map and combination of both for Laventine dialect

4.2.4 North Africa Arabic dialect

We List below the results we get while experimenting North Africa NOR dialect. NOR dialect is complicated as well as Laventine dialect. For our different datasets, we get different configuration for the best WER. For NOR, we get the best results while using MMI with 0.1 boost. For NOR1, we get the best results while using MAP adaptation. For NOR2 and NOR3, we get the best results while using HMM-DNN with five hidden layers, 2048 hidden dimensions, MPE optimization and MFCC as feature extraction. From the table 4.5, we can see that when we train the ASR on Gale phase 2 Arabic only and test NOR dialect, the WER is 95.95%. But when we add the NOR dialect to the training set, the results get better, the best WER is reduced by 29.37% which become 66.58% compared with the model trained on MSA only.

Table 4.5: WER of North Africa dialect with different dataset sizes, different adaptation and optimization

Run type	NOR	NOR1	NOR2	NOR3
combined	129.51	88.4	81.4	83.54
tri1	97.78	91.86	84	86.39
tri2a	97.55	89.9	82.79	86.02
tri2b	110.84	87.86	81.4	82.39
tri2b_mmi	98.14	99.27	81.12	76.81
tri2b_mmi_b0.1	95.95	99.66	80.88	77.38
tri2b_mpe	104.4	96.67	78.59	78.69
tri3b	113.49	88.2	80.53	82.81
tri3b_dnn_2048x5	122.61	91.41	81.46	82.29
tri3b_dnn_2048x5_fbank	99.97	97.75	90.37	73.51
tri3b_dnn_2048x5_smb	112.71	88.17	73.76	66.58
tri3b_dnn_2048x5_smb_fbank	99.95	95.54	80.67	67.11
tri4b	111.06	87.83	81.45	82.21

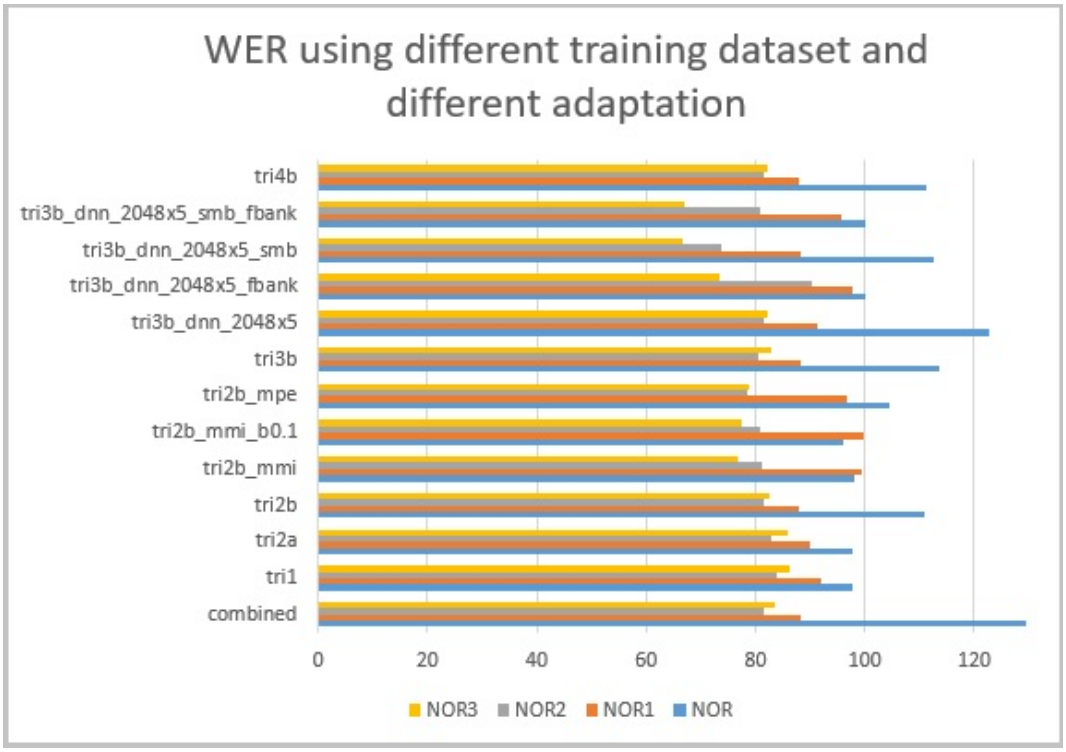


Figure 4.2.13: WER of North Africa dialect with different dataset sizes, different adaptation and optimization

From the previous table4.5, we want to compare the results of HMM-GMM vs HMM-DNN, the experiment tri2b_mpe which is HMM-GMM with MPE optimization and tri3b_dnn_ 2048x5_smb which is HMM-DNN with MPE optimization. Figure4.2.14 shows that HMM-DNN has better WER than HMM-GMM. Our best results is for NOR3 experiment overall our experiments. And HMM-DNN reduce the WER by 12.11% compared with HMM-GMM. About NOR which is MSA only trained ASR, the HMM-DNN do not act as expected because of the complication of the dialect itself, but when we add the North African dialect to the training dataset, the performance of the ASR has enhanced. We need to mention that the numbers appear on HMM-DNN column in this graph is an absolute difference between HMM-GMM and HMM-DNN results.

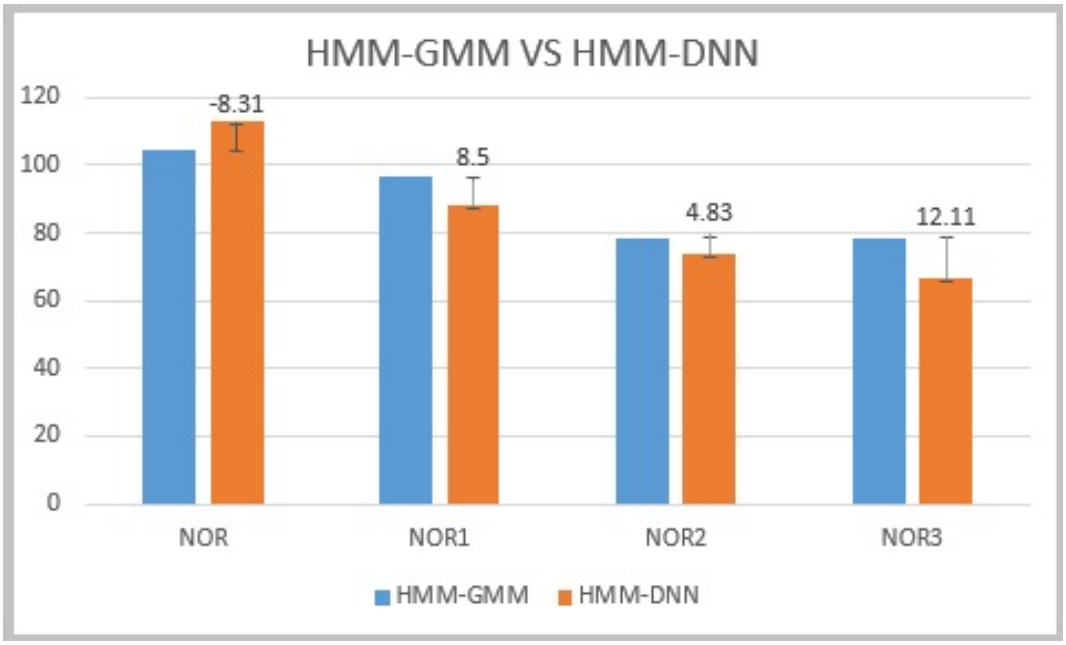


Figure 4.2.14: WER of comparing HMM-GMM and HMM-DNN with MPE optimization for North African dialect

In addition, we need to discuss the difference between training ASR using MSA only and involve the dialect in the training. Figure4.2.15 shows that involving the dialect in training enhance the performance of ASR. The numbers on X-axis are the Run type as shown in table4.5 in the same order. Also we can see that the size of training data of NOR is greater than NOR3, this confirms that selected data is much important than the size of the training data.

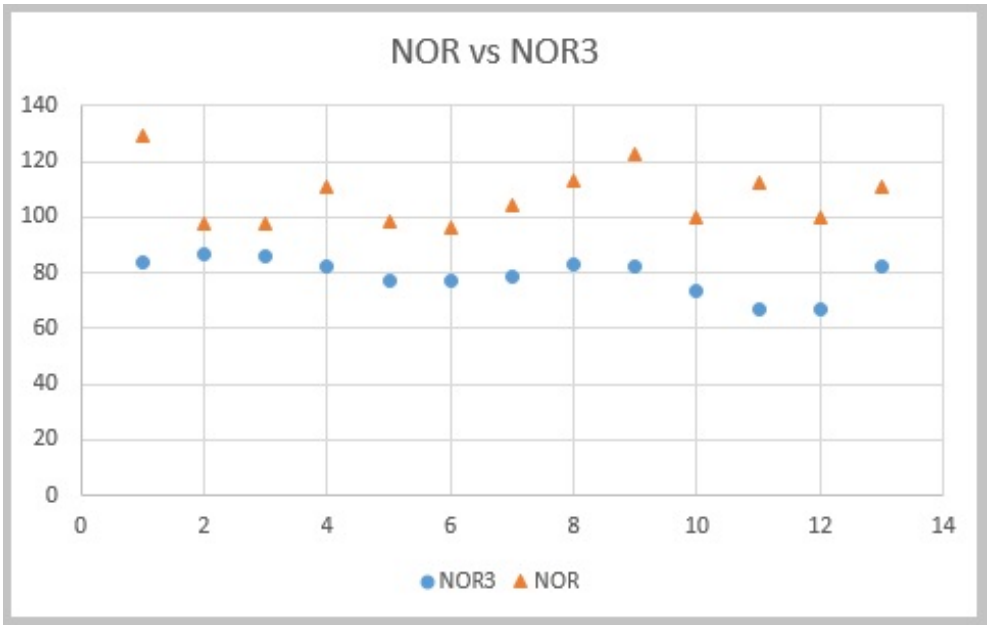


Figure 4.2.15: Compare MSA trained ASR (NOR) verses dependent dialect ASR (NOR3)

We also test different adaptation techniques when we experiment North African dialect. We apply fMLLR, MAP and combine both of them to check if these adaptation techniques affect the performance of our ASR. Figure4.2.16 shows that both adaptation has the same affect on our ASR and combining fMLLR and MAP do not enhance the performance of our ASR as well.

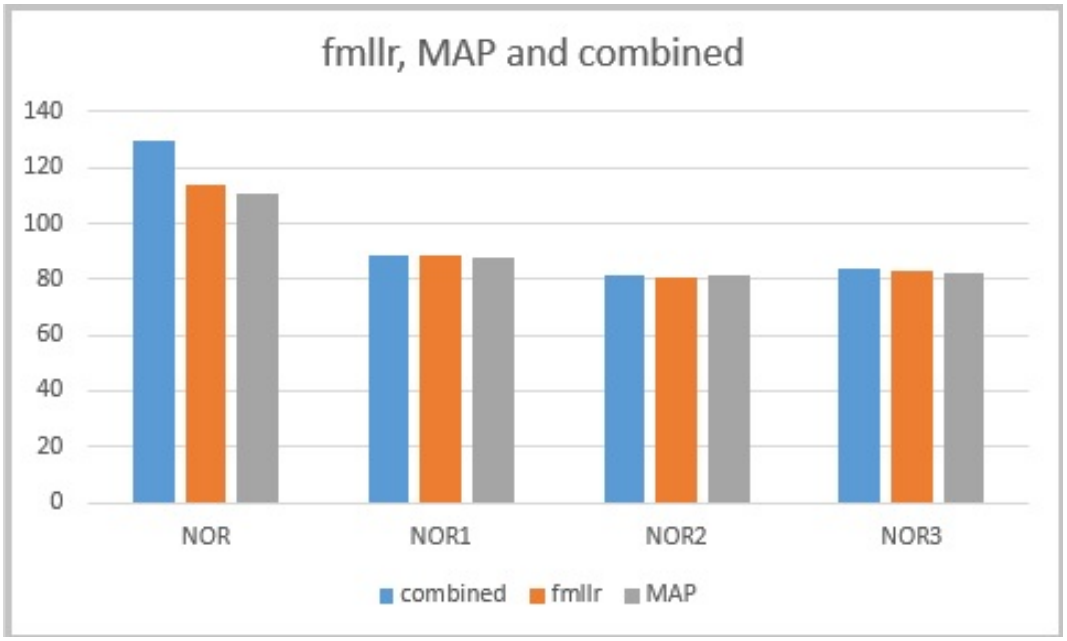


Figure 4.2.16: WER of comparing fmllr, Map and combination of both for North Africa dialect

4.2.5 All dialect trained ASR

Our main idea is to create a special model for each dialect, and to create a front-end system to detect the dialect, and then to run an appropriate dialect model for a given speech. But according to our results, and small datasets for each dialect, we decide to build an Independent Dialect ASR (ID-ASR); which means that we want to create an ASR to be suitable for all dialects, in order to have only one ASR for all available dialects without adding a dialect detection at the front-end. This can be done by training the ASR on all available dialects then testing this ASR on all dialects. Table 4.6 shows the results of training our ASR on all dialects and the best results of dependent dialect ASR (DD-ASR); NOR3, LAV3, Gulf3 and Egypt3. We combine the results in one table to ease our comparison between DD-ASR and ID-ASR. We apply the same adaptation and optimization techniques on each model. By comparing the results we can see that ID-ASR has better results than DD-ASR.

We get the best results while we use HMM-DNN with five hidden layers, 2048 hidden dimensions, MPE optimization and MFCC as feature extraction.

Table 4.6: WER of ID-ASR and DD-ASR with different adaptation and optimization

Run type	ALL	NOR3	LAV3	Gulf3	Egypt3
combined	74.47	83.54	83.61	69.53	75.46
tri1	78.31	86.39	83.61	75.63	79.83
tri2a	77.59	86.02	82.95	74.98	78.85
tri2b	72.86	82.39	78.82	69.19	74.5
tri2b_mmi	71.54	76.81	76.63	65.75	72.12
tri2b_mmi_b0.1	72.11	77.38	76.72	65.74	71.95
tri2b_mpe	70.56	78.69	76.96	65.63	71.98
tri3b	74.61	82.81	81.12	69.38	75.42
tri3b_dnn_2048x5	68.62	82.29	79.07	61.3	71.29
tri3b_dnn_2048x5_fbank	69.01	73.51	74.17	62.03	69.08
tri3b_dnn_2048x5_smb	59.21	66.58	70.71	54.12	62.99
tri3b_dnn_2048x5_smb_fbank	64.04	67.11	69.12	57.69	64.85
tri4b	72.83	82.21	78.79	69.09	74.59

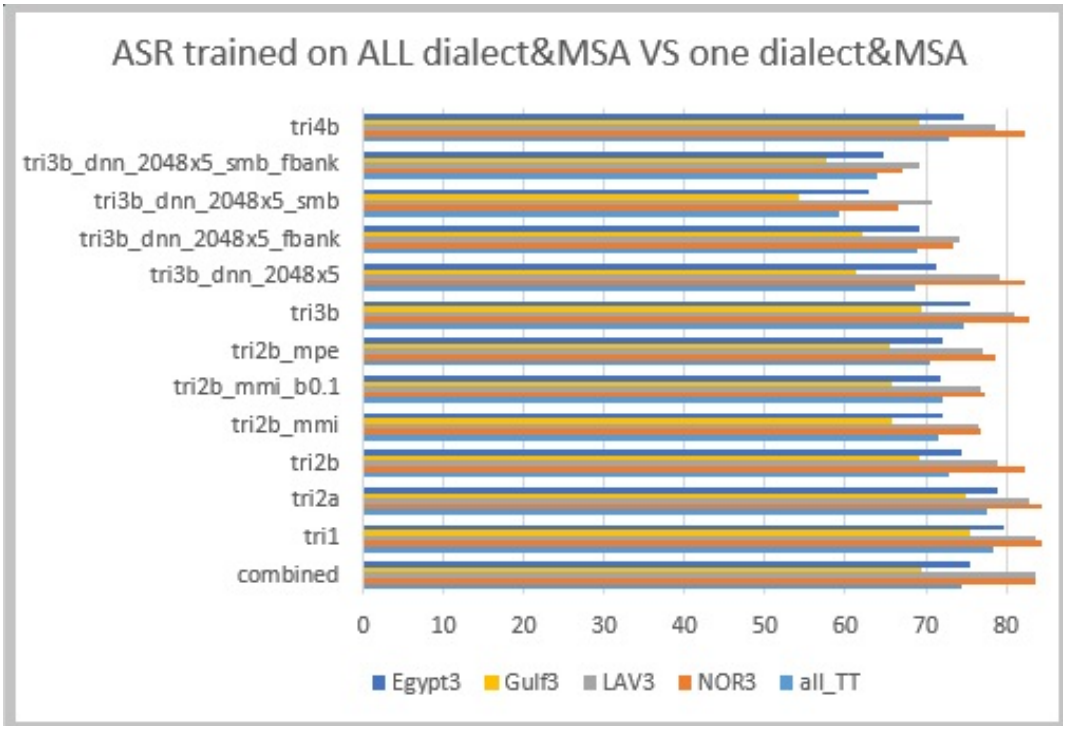


Figure 4.2.17: WER of aLL-dialect trained ASR with pool testing and One-dialect trained ASR with different adaptation and optimization

If we have closer look to table 4.6, we can see that ID-ASR has better results than NOR3, Egypt3 and LAV3, but it has higher WER than Gulf3. Figure 4.2.18 shows the difference between ID-ASR and DD-ASR, we found that the ID-ASR has better WER than DD-ASR. But Gulf dialect in DD-ASR scores better WER. Gulf dialect is the closer to MSA within the dialects in our research, and this might be the reason. We can not generalize any results from this step. So we decided to use the same ID-ASR but testing each dialect separately; one at a time.

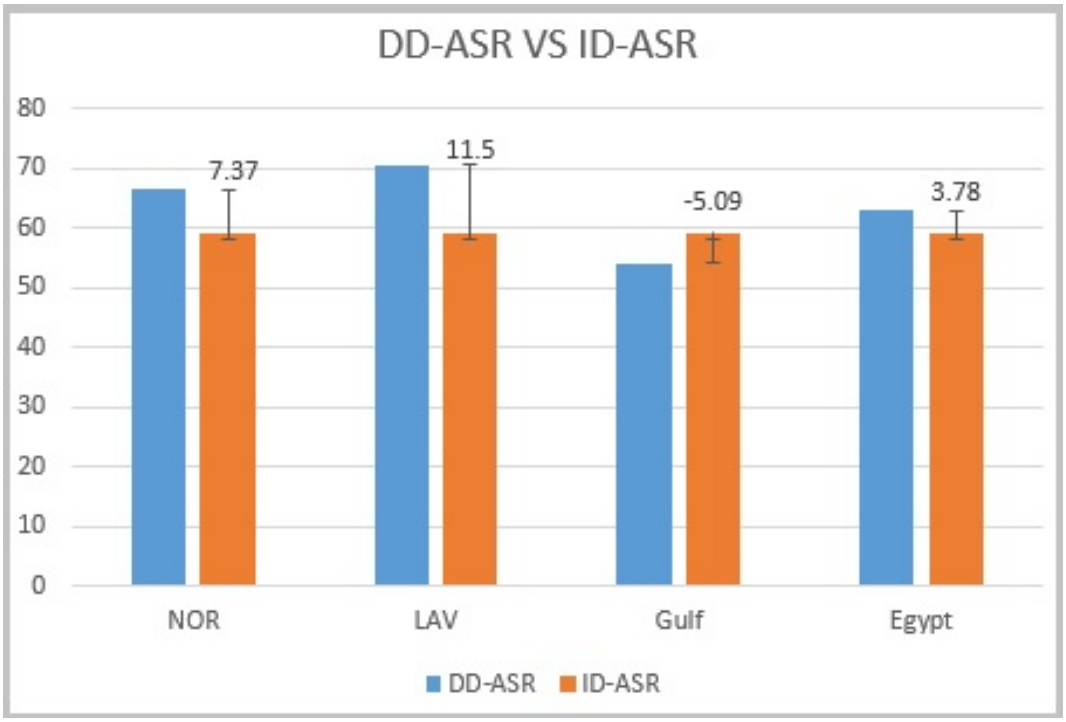


Figure 4.2.18: WER of DD-ASR verses ID-ASR where DD-ASR tests only one dialect while ID-ASR tests a pool of dialects

We use ID-ASR model and test this ASR for specific dialect. We can see that the higher WER for NOR and LAV. And this what we have discussed earlier about the difficulty of both of those dialects. The average of specific dialects is almost equal to pool testing. We need more datasets for both dialects in order to enhance the performance of our ASR. The available speech for each dialect in our research is almost 13 hours of speech. So our total is 52 hours for all dialects in this research. Compare this to high performance ASR that at least have 170 hours of speech; as most accurate ASR results in different researches. So we tried to reduce the problem to involve MSA in the whole experiment (170 hours). But as we discuss in our literature review, the data selection improve the performance of ASR. So if we get more dialect dataset the results will be better. Table 4.7 lists all the results of testing ID-ASR for a pool of dialects and a specific dialect at a time. We get the best results for all datasets while we have HMM-DNN with five hidden layers, 2048

hidden dimensions, MPE optimization and MFCC as feature extraction.

Table 4.7: WER of ID-ASR with known and pool testing with different adaptation and optimization

Run type	ALL_TT	ALL_Gulf	ALL_Egypt	ALL_LAV	ALL_NOR
combined	74.47	67.17	72.51	78.62	78.74
tri1	78.31	73.39	77.39	81.24	81.5
tri2a	77.59	72.9	76.37	80.5	80.15
tri2b	72.86	67.85	71.66	76.26	75.61
tri2b_mmi	71.54	66.18	71.43	74.31	74.98
tri2b_mmi_b0.1	72.11	67.33	71.81	75.13	75.41
tri2b_mpe	70.56	65.05	70.26	73.61	74.16
tri3b	74.61	67.21	72.61	78.96	78.62
tri3b_dnn_2048x5	68.62	58.96	66.22	75.59	75.52
tri3b_dnn_2048x5_fbank	69.01	62.99	67.31	73.37	73.48
tri3b_dnn_2048x5_smb	59.21	51.81	58.53	64.37	63.08
tri3b_dnn_2048x5_smb _fbank	64.04	57.53	63.75	68.66	66.93
tri4b	72.83	67.83	71.5	76.15	75.7

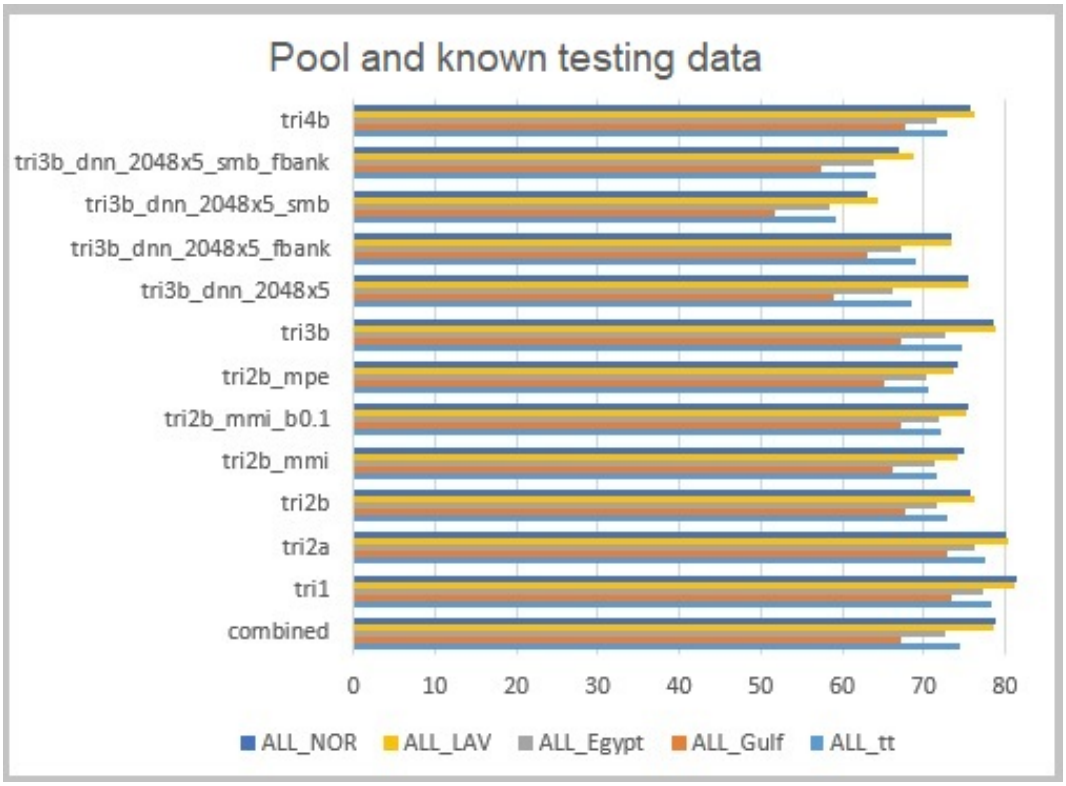


Figure 4.2.19: WER of ID-ASR with known and pool testing with different adaptation and optimization

Here, we want to compare the best results we get for each dialect using ID-ASR and DD-ASR. We get the best results for both models while having HMM-DNN with five hidden layers, 2048 hidden dimensions, MPE optimization and MFCC as feature extraction. Table 4.8 and figure 4.2.20 show that ID-ASR has better performance than DD-ASR for all dialects in this research.

Table 4.8: WER of ID-ASR and DD-ASR testing the same dialect

Dialect	DD-ASR	ID-ASR	Difference
Egyptian	62.99	58.53	4.46
Gulf	54.12	51.81	2.31
LAV	70.71	64.37	6.34
NOR	66.56	63.08	3.48

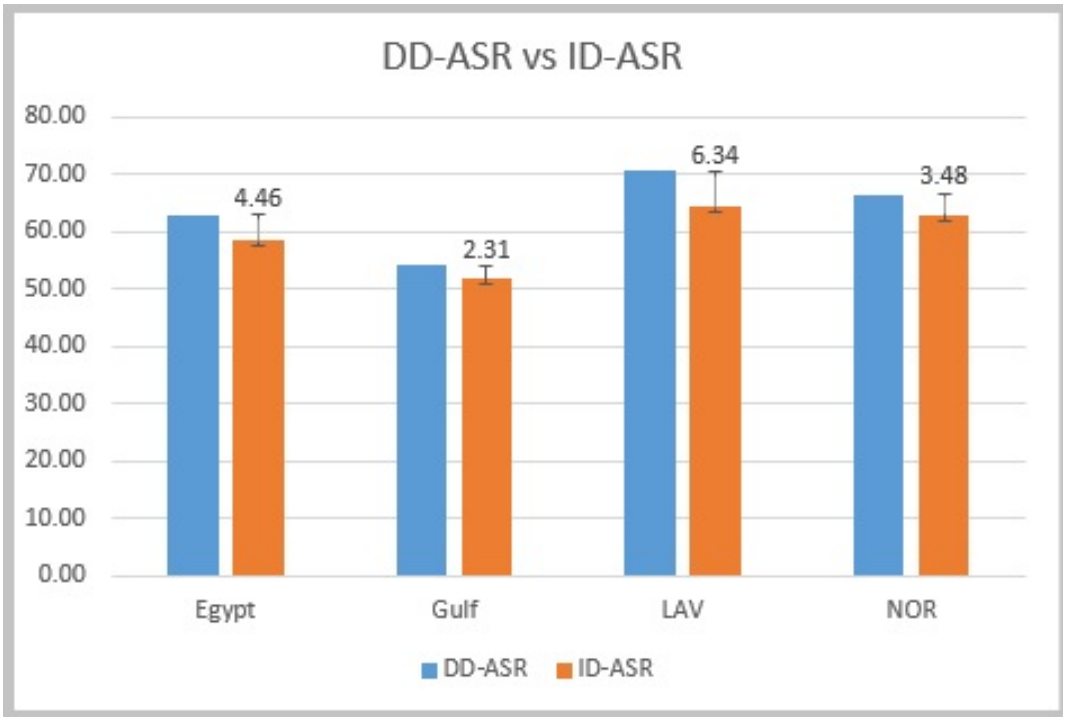


Figure 4.2.20: WER of DD-ASR vs ID-ASR when both models test one dialect only

In addition for ID-ASR and testing a pool of dialects, we discuss some adaptation techniques; MAP, fMLLR and combination of both as earlier models in this research. Figure 4.2.21 shows that using MAP has 72.83% WER. On the other hand, using fMLLR has 74.61% WER. This means that using MAP in ID-ASR is better than using fMLLR. So in this case, we should build the rest of the adaptation on MAP. And this might happen in some future work.

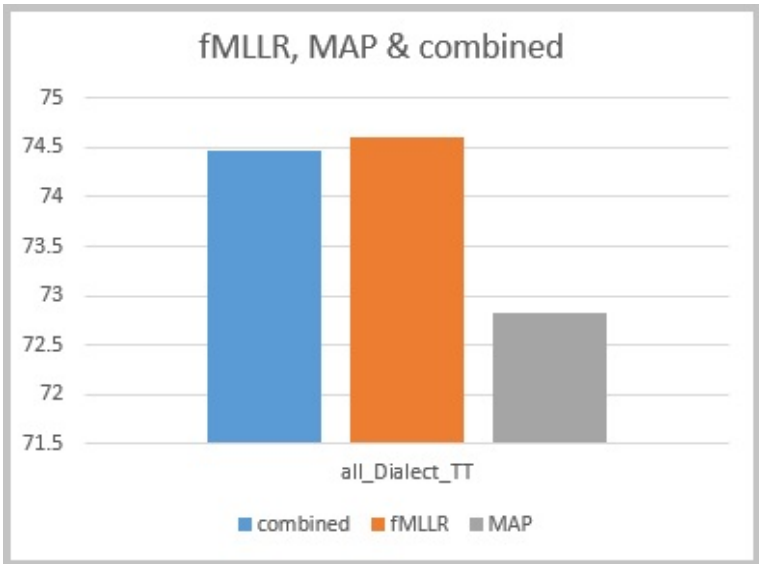


Figure 4.2.21: WER of comparing fmllr, Map and combination of both for ID-ASR with pool testing

4.2.6 Adding bottleneck feature BNF extraction to ID-ASR

The higher performance of ASR is attached with DNN technology. For more information about BNF review 2.4.4.3. So we employ DNN in the feature extraction stage, then try some adaptation and optimization techniques. Table 4.9 shows that using Bottleneck features (BNF), as described in [25], as feature extraction has enhance the performance of our ASR. Figure 4.2.22 reflects the data in table 4.9, and it shows the enhancement of BNF over MFCC.

This is a sample experiment to show how much DNN enhance the performance of ASR.

Table 4.9: WER of ID-ASR with bottleneck verses MFCC feature extraction with different adaptation and optimization

Run type	MFCC	BNF
tri1	78.31	67.38
tri2a	77.59	67.18
tri3b	74.61	66.45
tri3b_dnn_2048x5	68.62	63.51
tri3b_dnn_2048x5_smb	59.21	58.31

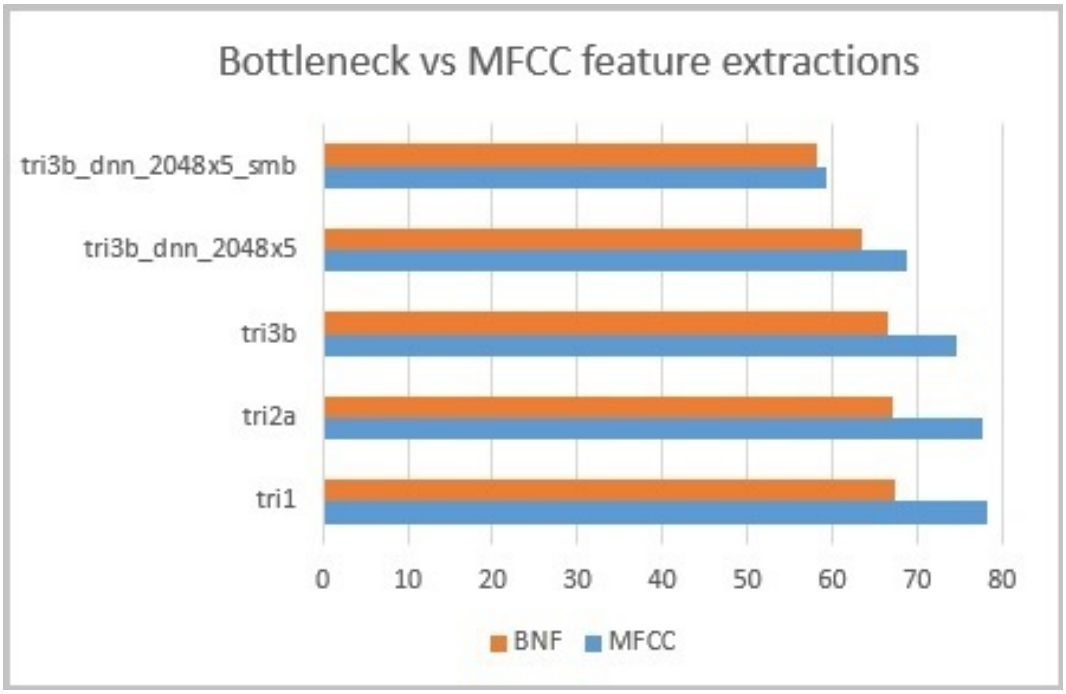


Figure 4.2.22: WER of ID-ASR with bottleneck verses MFCC feature extraction with different adaptation and optimization

Chapter 5

Conclusions and Future Work

5.1 Conclusion

Working in this research we have seen that the greater the size of the training dataset the better the results will be. Also when we look closer into our experiments of Dialect and Dialect3, these two different training datasets which are almost has the same size of training data, but the selection of the data shows that having the dialect involved in the training enhance the performance of our ASR by 29% for NOR, 21% for LAV, 16.89% for Gulf and 19.62% for Egyptian dialects.

Also we can see that the performance of ASR with HMM-DNN acoustic model is better than HMM-GMM acoustic model, as shown in figure 4.2.2 that for the best dataset HMM-DNN has improved the performance of ASR for 8.99% for Egyptian dialect, figure 4.2.6 shows that for the best dataset HMM-DNN has improved the performance of ASR for 11.51% for Gulf dialect, figure 4.2.10 shows that for the best dataset HMM-DNN has improved the performance of ASR for 6.25% for LAV dialect and figure 4.2.14 shows that for the best dataset HMM-DNN has improved the performance of ASR for 12.11% for NOR dialect.

Most of our experiments has the best results when we had HMM-DNN with five hidden layers, 2048 hidden dimensions, MPE optimization and MFCC as feature extraction. For DD-ASR the best results for Egyptian dialect is 62.99%, for Gulf dialect is 54.12%, for NOR is 66.58% and for ID-ASR the best result is 59.21%. But for DD-ASR testing LAV dialect, we get the best result while using fbank feature extraction and it is 69.12%

.

When we start our research, we wonder in case of only MSA trained ASR which dialect will have the least WER. The results are Egyptian dialect for 70.73%, the Gulf dialect for 71.01%, LAV dialect for 90.6% and NOR dialect for 96.96%. So the closest dialect to MSA are Egyptian dialect and Gulf dialect.

In our research, we also compare between MAP and fMLLR adaptation techniques. All the experiment of DD-ASR show that there is no difference between using MAP or fMLLR or combining them. But in case of ID-ASR, MAP shows better results than fMLLR as shown in figure 4.2.21. So we will try to build the rest of the adaptation on MAP as future work and compare the results with what we have at this phase of our research.

Using the technology of DNN enhance the performance of ASR. We try a short experience to involve DNN in feature extraction, and only apply it on ID-ASR. We can see through the results that we get better performance using Bottleneck feature extraction, which is clear in figure4.2.22. So we can say that employing DNN in feature extractions and Acoustic Model enhance the performance of Arabic ASR.

5.2 Future work

To get more speech for each dialect, and try building the dependent dialect ASR and check the performance. We need at least 170 -as most reliable ASRs in other researches- hours for each dialect to get most of the dialect. And if we can get Iraqi dialect as well to add to our ASR, since we can not find a good quality source to experiment Iraqi dialect.

The used lexicon was provided by Qatar Computing Research Institute QCRI, which had released 2014. The lexicon is the pronunciation dictionary for Modern Standard Arabic ASR. We need to add to this lexicon all the words pronunciation of the dialects in order to enlarge the pronunciation library.

We need to work on the Language Model LM as well, in our research we depend on the training data to create LM. We need to get more dialectical text to build a good source for LM. Also we can include the DNN technology to improve the performance of our LM. In our work, we build Trigram LM. The recent work, they are building 4-gram LM. We might build 4-gram LM and check the performance of ASR.

We need to do more experiments on Bottleneck feature extraction and try to change some parameters to improve the performance of ASR. Also we might try i-vector feature extraction, as described in [18, 78], as well and check if this works with dialectical speech. And evaluate each method for our dialects.

Bibliography

[1] Sadaoki Furui. "Automatic speech recognition and its application to information extraction". In: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics. 1999, pp. 11–20.

[2] *Current World Population*. URL: `http://www.worldometers.info/world-population/` (visited on 11/24/2016).

[3] Omar F Zaidan and Chris Callison-Burch. "Arabic dialect identification". In: *Computational Linguistics* 40.1 (2014), pp. 171–202.

[4] *Complete List of Arabic Speaking Countries 2014*. URL: `http://istizada.com/complete-list-of-arabic-speaking-countries-2014/` (visited on 11/28/2016).

[5] Uri Horesh and William M Cotter. "Current Research on Linguistic Variation in the Arabic-Speaking World". In: *Language and Linguistics Compass* 10.8 (2016), pp. 370–381.

[6] *IPA for Arabic Language*. URL: `https://en.wikipedia.org/wiki/Help:IPA_for_Arabic` (visited on 01/13/2017).

[7] Farheen Javed. "Arabic and English phonetics: A comparative study". In: *The Criterion: An International Journal in English* 4.4 (2013).

[8] Karin C Ryding. *A reference grammar of modern standard Arabic*. Cambridge university press, 2005.

[9] Fadi Biadsy. "Automatic dialect and accent recognition and its application to speech recognition". PhD thesis. Columbia University, 2011.

[10] *Learn Arabic the Easy Way*. URL: `http://www.myeasyarabic.com/index.html` (visited on 11/22/2016).

[11] MA Anusuya and Shriniwas K Katti. "Speech recognition by machine, a review". In: *arXiv preprint arXiv:1001.2267* (2010).

[12] Rashmi Makhijani, Urmila Shrawankar, and Vilas M Thakare. "Opportunities & Challenges In Automatic Speech Recognition". In: *arXiv preprint arXiv:1305.2846* (2013).

[13] Kai Wei et al. "Submodular subset selection for large-scale speech training data". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pp. 3311–3315.

[14] M Doulaty Bashkand, Oscar Saz, and Thomas Hain. "Data-Selective Transfer Learning for Multi-Domain Speech Recognition". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. ISCA (International Speech Communication Association). 2015, pp. 2897–2901.

[15] Yi Wu, Rong Zhang, and Alexander Rudnicky. "Data selection for speech recognition". In: *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*. IEEE. 2007, pp. 562–565.

[16] A Nagroski, Lou Boves, and Herman Steeneken. "In search of optimal data selection for training of automatic speech recognition systems". In: *Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on*. IEEE. 2003, pp. 67–72.

[17] Evandro Gouvea and Marelie H Davel. "Kullback-Leibler Divergence-Based ASR Training Data Selection." In: *INTERSPEECH*. 2011, pp. 2297–2300.

[18] Olivier Siohan and Michiel Bacchiani. "ivector-based acoustic data selection." In: *INTERSPEECH*. 2013, pp. 657–661.

[19] Fadi Biadsy, Pedro J Moreno, and Martin Jansche. "Google's cross-dialect Arabic voice search". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2012, pp. 4441–4444.

[20] Maryam Najafian. "Acoustic model selection for recognition of regional accented speech". PhD thesis. Ph. D. dissertation, University of Birmingham, 2016.

[21] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. "Acoustic modeling using deep belief networks". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.1 (2012), pp. 14–22.

[22] Jon Barker et al. "The Third 'CHiME' Speech Separation and Recognition Challenge: Dataset, Task and Baselines". In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE. 2015, pp. 504–511.

[23] Ahmed Ali et al. "A complete KALDI recipe for building Arabic speech recognition systems". In: *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE. 2014, pp. 525–529.

[24] Hassan Al-haj et al. "Pronunciation Modeling for Dialectal Arabic Speech Recognition". In: *ASRU*. 2009, pp. 525–528.

[25] Patrick Cardinal et al. "Recent advances in ASR applied to an Arabic transcription system for Al-Jazeera". In: *in Interspeech*. 2014.

[26] *Speech recognition research toolkit*. URL: `https://sourceforge.net/projects/kaldi/` (visited on 12/22/2016).

[27] Zhi-Jie Yan, Qiang Huo, and Jian Xu. "A scalable approach to using DNN-derived features in GMM-HMM based acoustic modeling for LVCSR." In: *INTERSPEECH*. 2013, pp. 104–108.

[28] Dong Yu and Michael L Seltzer. "Improved Bottleneck Features Using Pretrained Deep Neural Networks." In: *Interspeech*. Vol. 237. 2011, p. 240.

[29] Mohamed Elmahdy et al. "Cross-lingual acoustic modeling for dialectal Arabic speech recognition". In: *In Proceedings of INTERSPEECH*. 2010, pp. 873–876.

[30] Ahmed Ali, Hamdy Mubarak, and Stephan Vogel. "Advances in Dialectal Arabic Speech Recognition: A Study Using Twitter to Improve Egyptian ASR". In: *Improve Egyptian ASR. International Workshop on Spoken Language Translation (IWSLT*. 2014.

[31] *Modern Standard Arabic Pronunciation Dictionary*. URL: `http://alt.qcri.org/resources/msa-dictionary/` (visited on 07/18/2019).

[32] Florian Honig et al. "Revising Perceptual Linear Prediction (PLP)." In: *INTERSPEECH*. 2005, pp. 2997–3000.

[33] Parwinder Pal Singh and Pushpa Rani. "An Approach to Extract Feature using MFCC". In: *International organization of Scientific Research, IOSR Journal of Engineering (IOSRJEN)* 4.08 (2014), pp. 21–25.

[34] *Text independent speaker recognition system*. URL: `http://www.slideshare.net/deepeshlekhak/text-independent-speaker-recognition-system` (visited on 12/22/2016).

[35] *Audio Signal Processing and Recognition*. URL: `http://mirlab.org/jang/books/audioSignalProcessing/` (visited on 12/22/2016).

[36] Hynek Hermansky. "Perceptual linear predictive (PLP) analysis of speech". In: *the Journal of the Acoustical Society of America* 87.4 (1990), pp. 1738–1752.

[37] Stanley S Stevens. "On the psychophysical law." In: *Psychological review* 64.3 (1957), p. 153.

[38] Dan Jurafsky and James H Martin. *Speech and language processing*. Vol. 3. Pearson, 2014.

[39] Slava Katz. "Estimation of probabilities from sparse data for the language model component of a speech recognizer". In: *IEEE transactions on acoustics, speech, and signal processing* 35.3 (1987), pp. 400–401.

[40] Hermann Ney, Ute Essen, and Reinhard Kneser. "On structuring probabilistic dependences in stochastic language modelling". In: *Computer Speech & Language* 8.1 (1994), pp. 1–38.

[41] Mark Gales and Steve Young. "The application of hidden Markov models in speech recognition". In: *Foundations and trends in signal processing* 1.3 (2008), pp. 195–304.

[42] *Hidden Markov Model*. URL: `http://www.esat.kuleuven.be/psi/spraak/demo/Recog/page4.html#HMM` (visited on 01/10/2017).

[43] Lawrence R Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

[44] Jeff A Bilmes et al. "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models". In: *International Computer Science Institute* 4.510 (1998), p. 126.

[45] Poonam Bansal et al. "Improved hybrid model of HMM/GMM for speech recognition". In: *International Book Series "Information Science and Computing"* (2008).

[46] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B* 39.1 (1977), pp. 1–38.

[47] Sharada C. Sajjan and Vijaya C. "Speech Recognition Using Monophone and Triphone Based Continuous Density Hidden Markov Models". In: *International Journal of Research and Scientific Innovation(IJRSI)* 2.11 (Nov. 2015). ISSN 2321 - 2705, pp. 30–35.

[48] Young Steve, Evermann Gunnar, A MARK, et al. "The HTK book (for HTK Version 3.4)". In: *Steve Young, Gunnar Evermann, Dan Kershaw, Gareth Moore, Julian Odell, Dave Ollason, Valtcho Valtchev, Phil Woodland* (2009).

[49] Philip C Woodland et al. "Large vocabulary continuous speech recognition using HTK". In: *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on.* Vol. 2. Ieee. 1994, pp. II–125.

[50] Mark JF Gales and Philip C Woodland. "Mean and variance adaptation within the MLLR framework". In: *Computer Speech & Language* 10.4 (1996), pp. 249–264.

[51] Juri Ganitkevitch. "Speaker adaptation using maximum likelihood linear regression". In: *Rheinish-Westflesche Technische Hochschule Aachen, the course of Automatic Speech Recognition, www-i6. informatik. rwthaachen. de/web/Teaching/Seminars/SS05/ASR/Juri Ganitkevitch Ausarbeitung. pdf.* Citeseer. 2005.

[52] Silke Goronzy and Ralf Kompe. "A combined MAP+ MLLR approach for speaker adaptation". In: *Proceedings of the Sony Research Forum.* Vol. 99. 1999.

[53] Vassilios V Digalakis, Dimitry Rtischev, and Leonardo G Neumeyer. "Speaker adaptation using constrained estimation of Gaussian mixtures". In: *IEEE Transactions on speech and Audio Processing* 3.5 (1995), pp. 357–366.

[54] Marc Ferras et al. "Constrained MLLR for speaker recognition". In: *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*. Vol. 4. IEEE. 2007, pp. IV–53.

[55] Dong Yu and Li Deng. *Automatic speech recognition: A deep learning approach*. Springer, 2014.

[56] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining-Book, Chapter 5-Classification: Alternative Techniques*. 2005.

[57] Edmondo Trentin and Marco Gori. "A survey of hybrid ANN/HMM models for automatic speech recognition". In: *Neurocomputing* 37.1 (2001), pp. 91–126.

[58] Nelson Morgan and Herve Bourlard. "Continuous speech recognition using multilayer perceptrons with hidden Markov models". In: *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*. IEEE. 1990, pp. 413–416.

[59] Horacio Franco et al. "Context-dependent connectionist probability estimation in a hybrid hidden Markov model-neural net speech recognition system". In: *Computer Speech & Language* 8.3 (1994), pp. 211–222.

[60] Jun Qi and Javier Tejedor. "Robust submodular data partitioning for distributed speech recognition". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 2254–2258.

[61] *Face Recognition in Subspaces (Face Image Modeling and Representation) Part 2*. URL: `http://what-when-how.com/face-recognition/face-recognition-in-subspaces-face-image-modeling-and-representation-part-2/` (visited on 01/06/2017).

[62] Richard Socher et al. “Parsing natural scenes and natural language with recursive neural networks”. In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 129–136.

[63] *Understanding Natural Language with Deep Neural Networks Using Torch*. Mar. 2015. URL: https://devblogs.nvidia.com/parallelforall/understanding-natural-language-deep-neural-networks-using-torch/ (visited on 01/20/2017).

[64] *List of speech recognition software*. July 2014. URL: https://en.wikipedia.org/wiki/List_of_speech_recognition_software (visited on 01/18/2017).

[65] Daniel Povey et al. “The Kaldi Speech Recognition Toolkit”. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Catalog No.: CFP11SRW-USB. Hilton Waikoloa Village, Big Island, Hawaii, US: IEEE Signal Processing Society, Dec. 2011.

[66] *TOP 5 OPEN SOURCE SPEECH RECOGNITION TOOLKITS*. July 2016. URL: http://blog.neospeech.com/2016/07/08/top-5-open-source-speech-recognition-toolkits/ (visited on 01/16/2017).

[67] *GPU Managment and Development*. URL: https://docs.nvidia.com/deploy/cuda-compatibility/ (visited on 08/05/2019).

[68] *Linguistic Data Consortium*. URL: https://www.ldc.upenn.edu/ (visited on 12/22/2016).

[69] *VarDial 2018*. URL: http://alt.qcri.org/vardial2018/ (visited on 07/18/2019).

[70] *SRILM - The SRI Language Modeling Toolkit*. URL: http://www.speech.sri.com/projects/srilm/ (visited on 08/05/2019).

[71] Xuedong Huang et al. *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR, 2001.

[72] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[73] Pylkkonen Janne. *LDA Based Feature Estimation Methods for LVCSR*. Adaptive Informatics Research Centre Helsinki University of Technology, Finland.

[74] M.J.F. Gales. "Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition". In: *COMPUTER SPEECH AND LANGUAGE* 12 (1998), pp. 75–98.

[75] Josef Psutka. "Benefit of Maximum Likelihood Linear Transform (MLLT) Used at Different Levels of Covariance Matrices Clustering in ASR Systems". In: *Multiple-Taxonomy Question Classification for Category Search on Faceted Information*. Vol. 4629. Sept. 2007, pp. 431–438. DOI: 10.1007/978-3-540-74628-7_56.

[76] Narada Dilp Warakagoda. *A Hybrid ANN-HMM ASR system with NN based adaptive preprocessing 1996, This is a master thesis from Norges Tekniske Hogskole Institutt for Teleteknikk Transmisjonsteknikk*. URL: http://jedlik.phy.bme.hu/~gerjanos/HMM/hoved.html (visited on 11/28/2016).

[77] D. Povey and P. C. Woodl. "P.C.: Minimum phone error and I-smoothing for improved discriminative training". In: *In: Proc. ICASSP*. 2002, pp. 105–108.

[78] Ahilan Kanagasundaram et al. "i-vector Based Speaker Recognition on Short Utterances." In: *INTERSPEECH 2011*. Aug. 2011, pp. 2341–2344.