



People's Democratic Republic of Algeria



Ministry of Higher Education and Scientific Research

AMO Bouira University

Science and Applied Sciences Faculty

Computer Science department

LIMPAF Reasearch Laboratory

Master's Degree Thesis

in Computer Science

Major: Computer Systems Engineering

Theme

Arabic Speech Recognition Using Neural Networks

Supervised by

• OUKAS NOURREDINE

Co-Supervisor

• ZERROUKI TAHA

Realized by

• HABOUSSI SAMIA

• DJETTOU HALIMA

2021/2022

Acknowledgments

First and foremost we thank **Allah** almighty who provided us with strength, purpose, and success throughout this endeavor. Special thanks to our supervisor Dr.OUKAS Nourredine for all his patience, guidance, and support during the execution of this research. And also our co-supervisor Dr.Zerrouki Taha for his consultation, advice and guidance. Our thanks also go to Ms.Chihati Sarah for her help.

We also extend our heartfelt thanks to the members of the jury who accepted to evaluate our work. As well as all the professors who ensured our path in university courses.

Finally, we would also like to thank our families and all those who have encouraged and supported us with the ups and downs throughout this work.

Dedication

In the name of **ALLAH** the Clement and the Merciful praise to him the Almighty.
I dedicate this modest work to my dear parents who helped me understand life, to all
my dear brothers and sisters, to each member of my family, as well as to my friends
especially to my best friend djettou halima.

Haboussi Samia.

Dedication

First of all, I thank **ALLAH** who grants me his guidness and help me to achieve my aim to accomplish this thesis.

I dedicate this work to whom I owe everything, to my parents, the two candles of my life, who help and encourage me a lot to succeed in my studies from school to university.

With a great happiness and gratitude, I dedicate it to my sister and my brothers. This work is also dedicated to my dear friend and partner samia.

Djettou Halima.

Abstract

Nowadays, speech recognition is essential in designing a natural voice interface for communication between human and their modern digital life equipment. But the obvious issue in this field is the lack of wide support for several universal languages and their dialects.

This research comes to ensure the viability of designing the Automatic speech recognition model for the Modern Standard Arabic.

The automatic speech recognition model was developed by training it to recognize each character of the Modern Standard Arabic . The model's architecture followed the end-to-end speech recognition approach by using Mozilla DeepSpeech model which is a pretrained end-to-end neural network ASR in English language.

The obtained result showed a word error rate as low as 24.3% and character error rate as low as 17.6%. Therefore, we concluded that the model can reach a much better word error rate by deploying any improvement such as combining more datasets. The applications of this research are vastly available such as developing a real-time speech recognizer for Arabic audio lectures.

key Words: Arabic Automatic Speech recognition, Deep learning, Artificial neural networks, Arabic Language.

الملخص

في الوقت الحاضر ، يعد التعرف على الكلام أمراً أساسياً في تصميم واجهة صوتية طبيعية للتواصل بين الإنسان ومعدات الحياة الرقمية الحديثة. لكن المشكلة الواضحة في هذا المجال هي نقص الدعم واسع النطاق للعديد من اللغات العالمية وكذا لهجاتها المختلفة.

يأتي هذا البحث لضمان جدوى تصميم نموذج التعرف التلقائي على الكلام للغة العربية الفصحى المعاصرة. تم تطوير نموذج التعرف التلقائي على الكلام من خلال تدريبه على التعرف على كل حرف من حروف اللغة العربية الفصحى الحديثة. اتبعت بنية النموذج نهج End-to-End للتعرف على الكلام باستخدام نموذج Mozilla DeepSpeech وهو عبارة عن شبكة عصبية End-to-End مدربة مسبقاً للتعرف على الكلام في اللغة الإنجليزية .

أظهرت النتيجة التي تم الحصول عليها معدل خطأ في الكلمات (WER) منخفض يصل إلى 24.3% ومعدل خطأ في الحرف يصل إلى 17.6% وبذلك استنتجنا أن النموذج يمكن أن يصل إلى معدل خطأ الكلمة أفضل بكثير من خلال تحسينه كالجمع بين المزيد من مجموعات البيانات. تطبيقات هذا البحث كثيرة مثل تطوير أدوات للتعرف على الكلام في الوقت الفعلي للمحاضرات الصوتية العربية.

الكلمات المفتاحية: التعرف التلقائي على الكلام العربي، التعلم العميق، الشبكات العصبية الاصطناعية، اللغة العربية.

Résumé

De nos jours, la reconnaissance de la parole est essentielle dans la conception d'une interface vocale naturelle pour la communication entre les humains et leurs équipements de vie numériques modernes.

Cette recherche vise à assurer la viabilité de la conception du modèle de reconnaissance automatique de la parole arabe standard, et moderne. Le modèle de reconnaissance vocale automatique a été développé en l'entraînant à reconnaître chaque caractère de l'Arabe Standard Moderne. L'architecture du modèle a suivi l'approche de reconnaissance de parole End-to-End en utilisant le modèle Mozilla DeepSpeech qui est un ASR de réseau neuronal End-to-End préentraîné en langue anglaise.

Le résultat obtenu a montré un taux d'erreur de mots (WER) aussi bas que 24.3% et un taux d'erreur de caractères (CER) aussi bas que 17.6%. Par conséquent, nous avons conclu que le modèle peut atteindre un taux d'erreur plus bas de mots en déployant toute amélioration, comme la combinaison d'un plus grand nombre d'ensembles de données. Les applications de cette recherche sont largement disponibles, comme le développement d'un outil de reconnaissance de parole en temps réel pour les conférences audio arabes.

Mots Clés : Reconnaissance Automatique de la Parole Arabe, Apprentissage Profond, Réseaux de Neurones Artificiels, Langue Arabe.

Table of content

- Table of content** **i**

- List of Figures** **iv**

- List of Tables** **v**

- List of Equations** **vi**

- Abbreviations list** **vii**

- General Introduction** **1**

- 1 Arabic Language Background** **4**
 - 1.1 Introduction 4
 - 1.2 Arabic Script 5
 - 1.3 Classical Arabic 7
 - 1.4 Modern Standard Arabic 7
 - 1.5 Arabic Phonology 8
 - 1.6 Conclusion 16

- 2 Automatic Speech Recognition Overview** **17**
 - 2.1 Introduction 17
 - 2.2 ASR Architecture 17
 - 2.3 Front-end Feature Extraction 19
 - 2.4 Acoustic Modeling 21
 - 2.4.1 Neural Network Acoustic Model 21
 - 2.5 Language modeling 23
 - 2.5.1 Evaluating a Language Model 24
 - 2.6 Decoding 25

2.7	Evaluation	25
2.8	Classification of Speech Recognition System	26
2.8.1	Types of Speech Recognition System Based on Utterances	26
2.8.2	Types of Speech Recognition System Based on Speaker Model	28
2.8.3	Types of Speech Recognition Based on Vocabulary	28
2.9	ASR Tools	28
2.9.1	API Services	29
2.9.2	Frameworks	29
2.10	Conclusion	30
3	A New Neural Network for Automatic ASR	31
3.1	Introduction	31
3.2	Related works	31
3.2.1	General ASR	31
3.2.2	Automatic Arabic Speech Recognition	32
3.3	Arabic Speech Datasets for ASR	40
3.4	Proposed Approach	43
3.4.1	Basic Structure of RNN	43
3.4.2	LSTM	45
3.4.3	Connectionist Temporal Classification	47
3.4.4	Model Architecture	48
3.5	Conclusion	50
4	Implementation and Experiment	51
4.1	Introduction	51
4.2	Preliminaries	51
4.3	Data Collection and Processing	53
4.3.1	Preprocessing	54
4.3.2	Forced Aligning	54
4.4	Operation Environment	55
4.4.1	Jupyter Notebook	55
4.4.2	TensorFlow	56
4.4.3	Google Colaboratory (Colab)	56

4.5	Experiment Step	56
4.6	Results of Experiment	58
4.6.1	Comparison of Results	59
4.7	Deployment	60
4.8	Conclusion	61
	General conclusion	62
	Bibliography	64
	Appendix A	73

List of Figures

1.1	Letter forms of Arabic Language.	5
1.2	Types of Arabic diacritics [11]	6
2.1	Diagram of automatic speech recognition system [36].	18
2.2	Process of extracting Mel-Frequency Cepstral Coefficient (MFCC) [82]. . .	20
2.3	Example About WER	26
3.1	Basic structure of RNN [62].	44
3.2	RNN unfold structure [73]	45
3.3	Structure of LSTM [16]	46
3.4	Architecture of DeepSpeech [42]	49
3.5	Diagram of our Proposed Approach	50
4.1	The structure of DeepSpeech directory	52
4.2	Preprocessing CommonVoice Arabic text	54
4.3	Aligned Dataset Preview	55
4.4	Cloning DeepSpeech Repository	57
4.5	Building Language Model	57
4.6	Model Training	57
4.7	Results	58
4.8	Evolution of WER	59
4.9	Example 1	60
4.10	Example 2	60
4.11	Example 3	61

List of Tables

1.1	Arabic consonantal phonemic inventory	10
1.2	Arabic vocalic phonemic inventory	11
1.3	Local variation	16
3.1	Comparison between Neural Net work-based Approaches	36
3.2	Datasets for Arabic Speech Recognition	41
4.1	Detail of the Corpus	53
4.2	Comparison between ASR systems for Modern Standard Arabic	59

List of Equations

2.1	18
2.2	18
2.3	22
2.4	23
2.5	23
2.6	23
2.7	24
2.8	25
3.1	44
3.2	44
3.3	44
3.4	46
3.5	46
3.6	46
3.7	47
3.8	47
3.9	47
3.10.	48
4.1	58

Abbreviations list

MSA	modern standard Arabic
CA	classical Arabic
DA	Dialectal Arabic
NLP	Natural Language Processing
EGY	Egyptian
LEV	Levantin
GLF	Gulf
NOR	North African Arabic
ASR	Automatic Speech Recognition
MFCC	Mel Frequency Cepstral Coefficients
FFT	Fast Fourier Transformation
LOG	Logarithm
DCT	Discrete Cosine Transformation
LDA	Linear Discriminate Analysis
AM	Acoustic Model
HMM	Hidden Markov Model
ANN	Artificial Neural Network
DNN	Deep Neural Networks
MLP	MultiLayer Perceptron (MLP)
RNN	Recurrent Neural Networks
LSTM	Long-short-term memory networks
LM	Language Model
MLE	Maximum likelihood estimation

PP	PerPlexity
WER	Word Error Rate
CER	Character Error Rate
IVRS	Interactive Voice Response Systems
AASR	Automatic Arabic Speech Recognition
GMM	Gaussian Mixture Model
CNN	Convolutional Neural Networks
E2E	End-to-End
BDRNN	Bidirectional Recurrent Neural Network
CTC	Connectionist Temporal Classification
MGB	Multi-Genre BroadCast
ReLU	Rectifier Linear Unit
CV	Common Voice
CSV	Comma Separated Values
GPUs	General Purpose Units
TPUs	Tensor Processing Units

General Introduction

Speech is the primary mode of human communication, and as a result, massive amounts of data are transmitted via speech. To carry out further processing and analysis on speech data, transcription of the speech becomes a crucial part in technology development. The technology that aids in transcribing speech is known as Automatic Speech Recognition (ASR) which is also known as speech-to-text technology. ASR has influenced the way we interact with devices in recent years, and as a result, it has changed how we live and work. Digital assistants, voice search engines, educational aids, and health aids are just a few examples of ASR applications.

Commercially, there are several commercial ASR technologies providers with reasonable reliability, availability, and performance, such as Google Cloud Speech-to-Text, Microsoft Azure Speech-to-Text, and Amazon Web Service Speech-to-Text. However, these services have some limitations, such as data privacy violations, cost, limitation of data throughput of the web services, training data availability, and domain-specific recognition.

ASR continues to be an active and exciting research field in general and in the Arabic language in particular. Arabic is a Semitic language and one of the world's oldest languages. It is now the sixth most widely spoken language in the world. The Arabic language is divided into three types: Classical Arabic, Modern standard Arabic (MSA), Dialectal Arabic.

- **Problem Statement**

Due to the high variability and complex morphology of the Arabic language, current research on it is still limited. The first works on Arabic ASR have concentrated on developing recognizers for Modern Standard Arabic (MSA). The most difficult problems in developing highly accurate ASR systems for Arabic are the morpholog-

ical complexity, predominance of non diacritized text material, and the enormous dialectal variety. Traditional ASR systems used a modular design. Different models are trained for pronunciation lexicon, acoustic modeling, and language modeling separately in these systems, each of which can be complex. For example, training of an acoustic model is a multi-stage process of model training and time alignment between the speech acoustic feature sequence and output label sequence.

- **Objectives**

The main objective of this work is to develop an ASR system using Neural network "end to end approach" for Modern Standard Arabic Language. In end-to-end (E2E) approach, all the traditional ASR's models are trained to convert the features of acoustic to text transcriptions directly. End-to-end ASR's purpose is to make it easier to train the above module-based components into a single deep neural network, in order to fix these issues. End-to-end ASR approach depends only on acoustic and language data with no need of linguistic knowledge, and train the model with a single algorithm. So, the goal of this research will be achieved by:

- Collecting dataset that reflect and represent the Modern Standard Arabic language.
- Align the transcriptions of the audio file and save it as (comma-separated values) file to feed it to the model.
- Develop a model to recognize the speech and map it into a textual format, to achieve the goal of this research.

- **Methodology**

The methodology of this research to develop the End to End ASR model is to find a way to collect and pre-processing data and preparing them for the next stage.

Second, taking the data from the last step and pass it the DeepSpeech model to train and improve using it.

This reseach is organized as follows:

Chapter 1: Arabic Language Background

In this chapter, we will give basic background about Arabic where we will discuss its two major classes, are Arabic dialectal and MSA and its script and phonology

Chapter 2: Automatic Speech Recognition Overview

In this chapter, we will talk about the important concepts related to Automatic Speech Recognition, we will give a brief overview of the various aspects of modern speech recognition systems, also we will describe the frameworks and services that aid in the development of the ASR system

Chapter 3: A New Neural Network for Automatic ASR

In this chapter, we will review surveys and journal papers about Automatic Speech Recognition and we will mention most of the techniques related to it. Therefore we will present our proposed approach to develop a speech recognition model for Modern Standard Arabic.

Chapter 4: Implementation and Experiment

In this chapter, we will cover the implementation phase of our proposed approach for this research and discuss the obtained results.

Finally, we close our work with a general conclusion.

Arabic Language Background

1.1 Introduction

Arabic is one of the most widely spoken languages in the world, with an estimated 274 million native speakers [74]. Arabic is the sixth most spoken language in the world, After English, Mandarin, Hindi, Spanish and French [74]. The modern standard Arabic (MSA) language is based on classical Arabic. It's mostly found in contemporary publications, education, and journalism. MSA is syntactically, morphologically and phonologically derived from classical Arabic (CA), the language of the Quran (Islam's Holy Book). However, it is now the most widely used language since, in addition to being formal, it is easier to learn than classical Arabic. While most Arabic-speaking residents can communicate with one another despite dialectal differences, communication can be difficult for some other countries because they speak different languages. In general, Arabic can be classified into three groups:

- Modern Standard Arabic **اللغة العربية المعاصرة** : Modern Standard Arabic and it is also called Al-Arabiyya Al-Fusha, and MSA is also known as the main language of all Arabs. MSA is used more often in writing than in spoken form.
- Classical Arabic (CA) **اللغة العربية العتيقة** : Muslims regard classical Arabic as sacred since it is the language used in the Quran (Koran). CA also known as alturath, used to be the main language in the pre-Islamic time, more than 1400 years ago. CA is commonly used today in studying Arabic poetry and in Friday prayers' speeches in mosques.

- Dialectal Arabic (DA) *العامية العربية*: Arabic dialects differ depending on the social environment and how concerned and impacted individuals are by elements using tongues other than Arabic. DA is also known as "alamia" and is used in everyday life speech such as phone calls and family discussions.

1.2 Arabic Script

Arabic is both written and read from right to left. Arabic script is also used for writing several other languages around the world such as Persian (Farsi / Dari), Malay (Jawi), Uyghur, Kurdish, Urdu.

- **Letters** Arabic letters are written in the cursive form in both print and script (handwriting). They typically consist of two parts: letter form (رسم rasm) and letter mark (إعجام AijAm). The letter form is an important component in every letter. There is a total of 19 letter forms (See Figure 1.1). The letter marks, also called consonantal diacritics which are mainly dominated by hamzas, and dots. The Arabic alphabet is widely thought to have 28 letters (basic 28, sometimes substituting ٰ with ا) or 29 letters (basic 28 plus the Hamza-on-the-line letter constructed from the Hamza letter form).

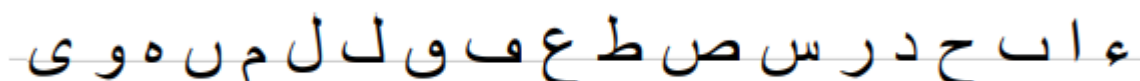


Figure 1.1: Letter forms of Arabic Language.

- **Arabic keyboard:** Different type bars (the equivalent of symbols) for different ligatures and letter shapes were used on typewriters and print press machines before computers. When typing on a typewriter, the correct letter shape has to be specified. In Roman script, this is a more complicated method of specifying capitals and small letters. Arabic data entering via keyboard is greatly simplified due to the encoding options available on modern computers. Arabic is simply graphemically entered in a logical sequence. Yamli, Google's ta3reeb, and Microsoft's Maren are just a few

examples of web tools that let users type in either a strict or loose romanization. A phonetic keyboard for Arabic is available on some operating systems [9].

- **Diacritics (التشكيل)** : The diacritics are a class of symbols in the Arabic script. Whereas letters are always written, diacritics are optional: written Arabic can be fully diacritized, partially diacritized, or entirely undiacritized. Except in religious texts, children’s educational books, and certain poetry, Arabic text is mostly undiacritized. In modern written Arabic, some diacritics are used to help readers disambiguate some words. There are three types of diacritics: Vowel, Nunation, and Shadda [9]. They are presented in Figure 1.2.

الحركة Vowel	التنوين Nunation	السكون No Vowel
بَ ba /ba/	بًا bā /ban/	بْ b. /b/
بُ bu /bu/	بُ bū /bun/	الشدة Double Consonant
بِ bi /bi/	بِ bī /bin/	بّ b~ /bb/

Figure 1.2: Types of Arabic diacritics [11]

- **Normalization:** Orthographic normalization is a basic task that Arabic Natural Language Processing (NLP) researchers always perform with the same goal in mind: to reduce noise and sparsity in the data and It doesn’t matter what task we are working on: preparing parallel text for machine translation, documents for information retrieval or text for language modeling. In Arabic, there are four letters that are so often misspelled using variants that researchers find it more useful to completely make these variants ambiguous (normalized). The following are the four letters in

order of most commonly normalized to least commonly normalized (the first two are what most researchers do by default, and the last two are less commonly applied).

1. The Hamza forms of Alef are normalised to Alef.
2. The Alif-Maqsura (ﺀ) is normalized to a Ya (ﻱ). In Egypt, but not in other Arab countries necessarily, a final Ya is often written dotless (i.e., as an Alif-Maqsura).
3. The Ta-Marbuta (ﺓ) is normalized to a Ha (ﻩ).
4. The non-Alif forms of Hamza (ﻮ̣ and ﻮ̣̣) are normalized to the Hamza letter (ﺀ).

1.3 Classical Arabic

Classical Arabic is the foundation of Arabic linguistic theory, and the educated Arabic reader understands it well. It differs from Modern Standard Arabic in numerous aspects, including its lexical, syntactic, morphological, phraseological, and semantic structure.

Classical Arabic is regarded as the language that evolved from the various Bedouin tribes of the Arabian Peninsula, as recorded in the pre-Islamic poetry [32]. Because the Quran was revealed in Arabic, it has a sacred and prestigious place not only among Arabs but also among all Muslims worldwide. All Muslims around the world must learn Arabic to be able to carry out their religious acts properly. Classical Arabic could have maintained its purity and linguistic features over 1500 years[5] due to its religious status as a language of Quran being recited daily and mostly five times a day (in the prayers) by all Muslims around the world. Furthermore, every Friday, all Muslims are obligated to assemble in their local mosques to hear the weekly-based oration (Xutbah) delivered in Classical Arabic. Therefore, all of the above factors helped Arabic maintain its dynamic utility for many years.

1.4 Modern Standard Arabic

Modern Standard Arabic is the Arab world's official language. The media and culture use it as their primary language. In terms of syntax, morphology, and phonology MSA

is based on Classical Arabic, the language of the Qur'an (Islam's Holy Book). MSA, on the other hand, is much more modern lexically. MSA is primarily a written language, not a spoken one. Most educated Arabic speakers can utilize MSA as a "lingua franca" to speak with each other regardless their nationalities or spoken native dialects. It is almost assimilated and understandable by all Arabs because the majority of Arabs are exposed to MSA through media, printed materials, religious practices, and certain work-related or social situations [70]. Arabic is a symbol of Arab unity, and it is the concerted ingredient that linguistically unifies Arab nations, since it is the official language of all Arab countries.

There are some differences between Classical Arabic and Modern Standard Arabic in terms of vocabulary, morphology, and syntax. Modern Standard Arabic includes a series of new lexical entries that are often borrowed or adapted from French (e.g. *secrétaire* /sikriti:r/ 'secretary', *informatique* /ma'lu:mijja:t/ 'Computer Science'). Morphologically, Modern Standard Arabic does not make much use of the case endings that are common in Classical Arabic. Syntactically, a new word order, namely subject-verb-object, emerged as a result of the contact between Arabic and other languages. It is now used as an alternative, in addition to the traditional word order subject-verb-object. The preposition *li* (which is equivalent to the English word *to*) is used instead of the traditional construct state, in which word order is critical. Therefore, the main characteristics of modern Standard Arabic are as follows: a new vocabulary enriched by a variety of French and English loan-words and assimilation idiomatic expressions; changes in syntactic and stylistic structure modeled by the French or English System; as well as sound pattern strongly affected by dialectal Arabic phonetics.

1.5 Arabic Phonology

Phonology is the study of the organization of natural language sounds, or phones. The phoneme, or smallest contrastive element in a language's sound system, is an essential element in phonology. Being contrastive implies that the language in issue contains a minimum pair including the phoneme: two words with different meanings that differ phonologically only in that phoneme. For the phonemes /q/ and /k/, for example, the MSA words قلب /qalb/ 'heart' and كلب /kalb/ 'dog' form a minimal pair. A phoneme

can be associated with several phones, or basic sounds, which are distributed according to phonotactics, or predictable rules. Allophones are the predictable phonemes connected with a phoneme. While Arabic does not have a phoneme /p/, which frequently leads to the notably Arabic-accented p-b mix in English speech, the phone [p] appears as an allophone of the phoneme /b/ in certain instances, such as before a voiceless phone: The term دبس /dibs/, which means molasses, is transcribed phonetically as [dips] but phonemically as /dibs/.

The problem in describing the phonology of MSA is that MSA is not natively spoken by any groups of Arabs, so no contemporary regional or social grouping can reasonably claim that its habits of performance represent a model of what is "correct". In the native grammatical tradition, /fasaha/ (= 'purity' and, by extension, 'correctness'), whether in phonology or any other aspect of the language, ultimately derives from historical precedent: that is, the classical Arabic as elaborated by the grammarians. Although we have a clear picture of what Arabic morphology, syntax, and vocabulary were like from an early period from the copious textual material that survives, we have little or no information on many aspects of how it was pronounced [47].

Like most Semitic languages, has a large consonantal vocabulary but a small vocalic system [78] [47]. In contrast to other Semitic languages, Arabic has retained the majority of the pharyngeal and emphatic consonants that were thought to exist in Proto-Semitic. MSA has 34 phonemes in its phonemic inventory, six of which are vowels. At the phonetic level, there would generally be more consonants and vowels. While MSA is a significantly simplified variant of CA in terms of lexicon and grammar, it is reasonable to infer that the differences between MSA and CA extend to phonology, specifically its phonemic inventory.

The phonemic inventory of MSA :

- **Consonants :**

Arabic consonantal phonemic inventory. The different manners of articulation are represented by rows, while the different places of articulation are represented by columns. Phoneme pairs are simple and emphatic variants.

Table 1.1: Arabic consonantal phonemic inventory

Manner \ Place		Place									
		Labial	Labio-dental	Inter-dental	Dental	Alveolar	Palatal	Velar	Uvular	Pharyngeal	Glotal
Stop	voiceless				ط ت			ك	ق		ء
	voiced	ب			ض د						
Fricative	voiceless		ف		س ص		ش	خ		ح	ه
	voiced			ظ ذ	ز			غ		ع	
Affricate	voiceless										
	voiced						ج				
Glide		و					ي				
Nasal		م				ن					
Trill						ر					
Liquid						ل					

The classification of consonants looks as follows [28]:

- labial الحروف الشفوية – sounds which are created with the help of upper and lower lips due to their conjunction : /b/ ب, /m/م, and /w/ و .
- labiodental الحروف الشفوية اللسانية – are created by upper teeth and lower lip: /f/ ف in Arabic .
- dental الحروف الاسنانية – are created by putting the tongue behind the upper teeth: /t/ ت, /t/ ط /d/د, /d/ض /s /س, /z/ز .
- interdental الحروف بين الاسنانية – are created by putting the tongue between the teeth: exist only in Arabic /θ/ ث, d ذ, Ḍ ظ .
- alveolar الحروف اللثوية – as the name says for itself, sounds are created with the help of alveolar ridge: /r/ ر, /l/ ل and /n/ ن .
- velar الحروف الطبقيّة – are created with the help of soft palate and back of tongue: /x/خ, and /k/ك .
- palatal الحروف الغارية - are made by soft palate and front part of tongue: /y/ي .

- glotal الحروف الخنجرية – are created only by air without any obstructions: /h/ ه .
- pharyngeal الحروف الحلقية – are created with the help of pharynx and tongue: /H/ ح .
- uvular الحروف اللهوية – created with the participation of uvula: they are represented by only one symbol /q/ ق, which exists just in Arabic.

• **Vowels الحركات:**

The height and backness of the tongue position are used to express vowels.

Arabic vocalic phonemic inventory. Vowels are represented in terms of height and backness of the position of the tongue.

Table 1.2: Arabic vocalic phonemic inventory

	Front	Central	Back
High	ي إ		و أ
Low		أ ا	

Finally MSA has two diphthongs: /ay/ and /aw/. Tables 1.1 and 1.2 present the various consonantal and vocalic (respectively) phonemes in MSA in terms of their articulatory features. In Table 1.1, the presence of a pair of phonemes in one cell, as in ‘t T’, indicates that they are plain and emphatic, respectively. Emphasis تفخيم (tafxiym) is a bass effect giving an acoustic impression of hollow resonance to the basic sound [47]. Emphasis together with the presence of eight consonants in the velar and post-velar region is what gives Arabic pronunciation its distinctive guttural quality [47]. Vowel phoneme pairs in Table 1.2 indicate length differences (short and long).

• **Characteristics Of The Arabic Letters صفات الحروف :**

We have studied the exit points of the letters (Makharij) previously. Now, it is fundamental to know their various characteristics. We call characteristic of a letter (sifat صفات), the state of this letter when pronounced. Previously, we saw

that multiple letters could leave the same exit point. As a result, when letters are pronounced, their characteristics allow them to be distinguished. The study of letter characteristics allows us to distinguish between several letters coming from the same exit point (makharij) and one another. It allows us to distinguish between thin (muraqaqa) and emphatic (mufakhama) letters, which is useful when using the "assimilations" (al idgham) rule, and it also allows us to improve our pronunciation. Indeed, for letters with the same exit point (for example, the د and the ت), it is sufficient to confuse a single characteristic so that one of these letters resembles the other. When a small amount of air is allowed to be blown out while pronouncing the د, it will sound like ت.

By learning these characteristics – Sifat, We will learn to respect all of the elements that make up these letters in order to preserve their purity until they can no longer be confused. Some letters will have a soft sound. Some will sound hard. Some will need breathing, some will not, etc.

The number of sifāt is a point of contention among scholars. Some have risen to 17 sifāt. Imam Ibn Al Jazary holds this viewpoint. Some have risen to 44 sifāt. Others excluded some characteristics (such as "inhiraf" and "leen"), and counted the ghunna among them, bringing their total to 14. Some of sifāt are listed below:

1. **Al Hams** – الهمس Al-hams is an Arabic word that means "whisper.". It is a flow of air And its letters are ten [23]:

ف ح ث ه ش خ ص س ك ت

These letters are compelled in this sentence:

فَحَثُّهُ شَخْصٌ سَكَّتْ

2. **Al Jahr** – الجهر Al-Jahr is the opposite of Al Hams . During the pronunciation of the following letters, the airflow is stopped.[23]

ا ب ج د ذ ر ز ض ط ظ ع غ ق ل م ن و ء ي

These letters are compelled in this sentence:

عَظَمَ وَزُنُ قَارِيٍّ غَضَّ ذِي طَلَبٍ جِد

3. **Ash-shidda** – الشدة al-shadda means the intensity. During the pronunciation of the following letters (8), the sound flow stops [23].

ت ك ب ط ق د ج أ

These letters are compelled in this sentence:

أَجِدُ قَطٍ بَكَتْ

4. **Al-istiala** – **الِاسْتِعْلَاءُ** Al-istiala is an Arabic word that means elevation . The upward pressure on the palate is caused by the letter's pronunciation [23]. This pressure is being applied to seven letters:

ظ خ ص ض غ ط ق

These letters are compelled in this sentence: **خُصَّ ضَغِطِ قِطٍ**

5. **As Safeer** – **الصَّفِيرِ** as-safir means whistling. It's an added sound that comes out from between the lips when you pronounce one of these three letters[23]. :

ص ز س

6. **Al Qalqala** – **الْقَلْقَالَةُ** Al qalqala is an Arabic word that indicates restlessness, instability, or disturbance. It is a strong rebound of the letter when sakin. It's a rule of Tajweed [23]. The letters of the Qalqala are:

ق ط ب ج د

These letters are compelled in this sentence:

قُطِبَ جَدٍ

7. **Al istitala** – **الِاسْتِطَالَةُ** al istitala means elongation. This is the lengthening of the sound when pronouncing the letter: ض [23].
8. **At tafashee** – **التَّفْشِي** at-tafashee means propagation. It is propagation or diffusion of the breath in the mouth during the pronunciation of the letter ش [23].
9. **Al-leen** – **اللَّيِّن** lit means gentle ness. It is a gentle and effortless pronunciation. The letters concerned are the waw و and ya ي having a sukoon and preceded by a fatha [23]. Example: **الَّذِي أَطْعَمَهُمْ مِنْ جُوعٍ وَأَمَّنَهُمْ مِنْ خَوْفٍ**.
10. **Al-itbaq** – **الِإِطْبَاقُ** The word al-itbaq is derived from a verb, which means "to stick". When pronouncing the following four letters, the tongue is glued to the palate [23].

ظ ط ض ص

• **Phonotactics** الصوتيات

The study of sound distribution patterns and restrictions within words (and sometimes across word boundaries) is referred to as "phonotactics". Derivational and inflectional morphology, as well as lexical root structure, are all influenced by phonotactic rules in Arabic. The Arabic grammarians discovered and described the majority of these rules and restrictions over a thousand years ago. The phonotactics of root morphology and the phonotactics of derivational and inflectional morphology are aspects of Arabic phonotactics. Morphophonemics is the study of how morphological processes interact with phonological structures and rules. The Arabic sound system relies on four phonological processes: assimilation (one sound absorbs or influences another), epenthesis (vowel insertion), deletion (of vowel or semivowel), and vowel shift (displacement of a vowel from one position in a word to another).

– **Syllable Structure**

The structure of syllables is a part of the study of phonotactics and it forms the element of phonological word division concentrated on pronounceable segments of words and how they are divided, composed, and distributed, where each word consists of one or more syllables.

All Arabic syllables begin with CV or CVV and never with vowels or consonant clusters (two or more). For example نهر /nahr/: CVCC. CA and MSA have a similar syllable structure inventory, which is listed below.

– light syllable: CV(consonant-short vowel).

e.g.,-ma(مَ),-bi(بِ),-hu(هُ)

– heavy syllable: CVC(consonant-Short vowel-consonant) or CVV (consonant-long vowel).

e.g.,-faa(فَا),-dii(ضِي),-tab (تَب)

– super heavy syllable: CVVC(consonant-long vowel-consonant)or CVCC(consonant-short vowel-consonant-consonant)

e.g.,-riim(رِيم),nuun(نون),rabt(رَبَط)

- **Word Stress**

In Arabic, Stress is defined as the compression of a syllable within a word to make it clearer in perception and higher than other syllables within the same word. Stress, according to certain phonologists, refers to prominence or appearance. Traditional phonologists use terminology such as prod, height, rising, power of speech, emphasizing, and lengthening stir to describe stress. Stress in Arabic is determined by syllable structure and is automatic and it is assigned based on the structure of the words. This means that the phonological word's syllable structure determines the stress assignment. The syllable is stressed in Arabic when it contains a long vowel followed by a consonant (vvc) or a short vowel followed by two consonants or more (vcc) [15].

Stress and its location are predictable because rules can be created based on the word's structural patterns to pinpoint the syllable on which stress falls. Most linguists distinguish three levels of nonphonemic stress: primary, secondary, and weak.

- **Local variations of MSA**

Spoken varieties differ from Classical Arabic and Modern Standard Arabic (MSA) not only in grammar but also in pronunciation. Some of the common variations are shown in Table 1.3. In this table, example words are written as they will be pronounced by each dialect. Like in the first example , **ظابط** "Police Officer" is pronounced as shown in table. First example shows the MSA consonant (ظ) is pronounced as

/Ḍ/ in MSA, as */z/* in Egyptian Arabic (EGY) and Levantin arabic (LEV) includes the dialects of Syria, Lebanon, Jordan, Palestine. The second example shows that The MSA consonant **ث** */θ/* is pronounced as */t/* in LEV and EGY (or */s/* in more recent borrowings from MSA), e.g., **ثلاثة** 'three' is pronounced */θalāθa/* in MSA versus */talāta/* in EGY. The last example shows that the The MSA consonant **ذ** */d/* is pronounced as */z/* in EGY and **د** */d/* in Levantine.

Table 1.3: Local variation

EGY	GLF	LAV	MSA	NOR	English gross
زابط /ZābiT/	ظابط/ĎābiT/	زابط/ZābiT/	ظابط/ĎābiT/	ظابط/ĎābiT/	Police Officer
تلاتة /talāta/	ثلاثة/θalāθa/	تلاتة /talāta/	ثلاثة/θalāθa/	ثلاثة/θalāθa/	Three
هزا/hāza/	هذا/hāḏa/	هدا/hāda/	هذا/hāḏa/	هذا/hāḏa/	This

- EGY : Egyptian arabic includes dialects of Egypt and Sudan.
- LEV or LAV : Levantin arabic covers the dialects of Syria, Lebanon, Jordan, Palestine.
- GLF : Gulf arabic includes the dialects of Kuwait, United Arab Emirates, Bahrain, Saudi arabia and Qatar.
- NOR : North african arabic includes the dialects of Algeria, Morocco, Tunisia and Mauritania and Libyan arabic sometimes.

1.6 Conclusion

This chapter gave basic background about Arabic. We started with a linguistic description of Arabic script followed by a discussion of the two major classes of Arabic, dialectal and MSA. This is followed by Arabic Phonology where we presented Arabic consonants, vowels and word stress and also we showed the structure of syllables which is a subdivision of Phonotactics as well as we included some of the common variations among Arabic dialects and MSA.

The next chapter discusses the important concepts related to Automatic Speech Recognition.

Automatic Speech Recognition Overview

2.1 Introduction

With the success of Google Home, Amazon Echo, Siri, Cortana, and other voice assistants in recent years, they have become ubiquitous. These are the most well-known automatic speech recognition (ASR) examples. This category of applications starts with a sample of spoken audio in a certain language and converts the words uttered into text. As a result, they're often referred to as Speech-to-Text algorithms. For this reason, they are also called as Speech-to-Text algorithms. The traditional ASR system includes acoustic modeling, pronunciation lexicon, and language modeling modules. All of these systems work together to recognize speech, identify a specific speaker, detect multiple dialects, and so on, all from a single speech signal.

2.2 ASR Architecture

ASR system is composed of two major components: the front end and the decoder as shown in Figure 2.1. The spectrum representation of the voice waveform is extracted by the front end block. Mel Frequency Cepstral Coefficients are the most extensively used characteristics (MFCC). Based on the acoustic model, lexicon, and language model, the decoder block looks for the best match of word sequences for the input acoustic data.

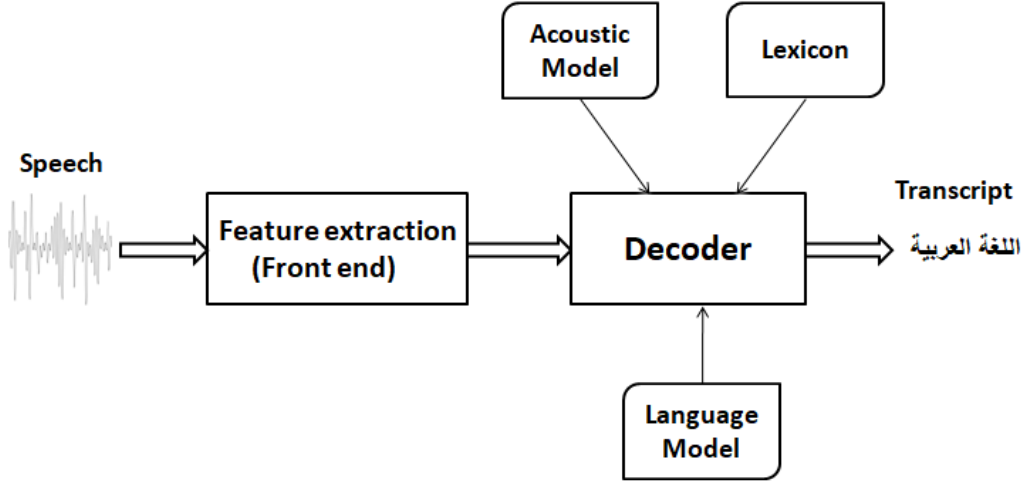


Figure 2.1: Diagram of automatic speech recognition system [36].

A sequence of speech vectors or observations "O" can be used to represent the input speech data. As a result, the difficulty in predicting the most likely word sequence w_h may be characterized by solving equation 2.1, often known as the fundamental equation of statistical speech recognition [26].

$$W_h = \underset{\mathbf{w}}{\operatorname{arg\,max}} P(w|o) \quad (2.1)$$

Where:

- o : represent the sequence of acoustic feature vectors (observations) .
- w : represent word sequence.
- $\operatorname{arg\,max}_{\mathbf{w}}$: is the search space, a function of the vocabulary.

The Bayes' rule is still used in the main stream of speech recognition to decompose equation 2.1 into the likelihood $P(o|w)$, the prior $P(w)$, and the denominator $P(o)$ which is independent of the word sequence and have no effect on the word sequence search. As a result, $P(o)$ is removed [26].

$$\begin{aligned} W_h &= \underset{\mathbf{w}}{\operatorname{arg\,max}} \frac{P(o|w) P(w)}{P(o)} \\ &= \underset{\mathbf{w}}{\operatorname{arg\,max}} P(o|w) P(w) \end{aligned} \quad (2.2)$$

The likelihood of the acoustic model is often referred to as $P(o|w)$, and the language model is referred to as $P(w)$. Since the probability of the language model $P(w)$ is independent of acoustics, it can be calculated independently and use different corpora.

The task of developing a speech recognition system is divided into two main modules, as shown in equation 2.2 : acoustic modeling and language modeling. The goal of acoustic modeling is to develop a model that can explain speech signals in the presence of an observation vector o . The acoustic observation $o = [o_1, o_2, \dots, o_T]$ vector is the result of front-end signal processing extracted from the raw waveform, which should ideally be invariant with respect to non-speech related factors like speaker factors, pronunciation variability, and environmental noise. However, in practice, the feature processing step will not be able to normalize all of the variability, so the acoustic models will have to share the task. On the other hand, before observing speech signals, language models should try to predict the prior distribution of the word sequence w . Traditional language models are based on the frequency of n-grams, which assume that the distribution of each word is dependent on the previous $n-k$ words, where k is the language model's history, also known as the order of language model. The remainder of this chapter provides an overview of a standard speech recognition system, and at the end will shed light on recent efforts in Arabic speech recognition.

2.3 Front-end Feature Extraction

Feature extraction is a crucial aspect of the speech recognition process since it extracts significant data from sample speech. The raw waveform is received in a continuous time and magnitude format. The signal processing front end's goal is to sample the raw acoustic waveform into feature vectors and extract the acoustic features that will be modelled by the acoustic modeling. The acoustic feature representation for speech recognition will be compact, preserving as much signal information as possible while minimizing variability across speakers and environmental acoustic conditions. It usually takes a frame of the speech signal every 16-32 milliseconds and updates it every 8-16 milliseconds [44],[79], as well as performing spectrum analysis. The regular frontend comprises the following algorithmic blocks, among others: Fast Fourier Transformation (FFT), Logarithm Calculation (LOG), Discrete Cosine Transformation (DCT), and Linear Discriminate Analysis

(LDA). To extract features from speech. MFCC is the most used approach and is based on the frequency domain using the Mel scale [79]. First, the speech signal is divided into time frames, each of which contains an arbitrary number of samples. In most systems, frame overlapping is used to create a smooth transition from one frame to the next. To eliminate discontinuities at the edges, each time frame is then windowed with a Hamming window. next, Fast Fourier Transformation (FFT) is calculated for each frame to extract frequency components of a signal in the time domain. The logarithmic Mel-Scaled filter bank is applied to the Fourier transformed frame. The mel scale is used to map the actual frequency to the frequency that human beings will perceive. The last step is to calculate Discrete Cosine Transformation (DCT) of the outputs from the filter bank. The overall procedure of MFCC extraction is shown on Figure 2.2.

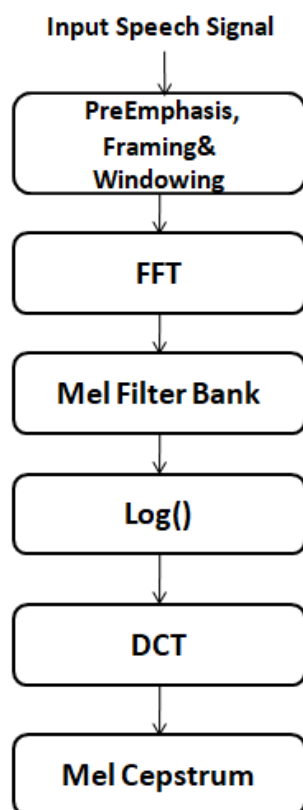


Figure 2.2: Process of extracting Mel-Frequency Cepstral Coefficient (MFCC) [82].

For each speech frame, a set of MFCC is computed. This set of coefficients is called an acoustic vector which represents the phonetically important characteristics of speech and is very useful for further analysis and processing in Speech Recognition

2.4 Acoustic Modeling

In voice (speech) recognition, acoustic modeling is the initial and important step. The acoustic model establishes a relation between acoustic data and language units. Due to feature extraction and statistical representation, the majority of the calculations are performed in acoustic modeling, which has the most impact on the recognition process. Extracted features are used to create statistical representations. The distribution of the extracted features with a specific sound has been designed in an Acoustic Model (AM) to establish the relation between the extracted features and the structures of the language module. Various feature extraction approaches have been reported, including those based on human perception and the working of voice producing systems. Because these systems pose issues in speech recognition, features were extracted for AM in speaker-independent mode recognition.

For the development of acoustic models, the selection of a classification method is also an important step. Many studies have been reported on acoustic modeling based on different classification techniques. The work reports the use of different classification methods such as based on hidden Markov model (HMM), discriminant training to optimize model parameters, artificial neural network (ANN), deep neural networks (DNNs) and sequence to sequence acoustic modeling.

2.4.1 Neural Network Acoustic Model

Neural network-based systems have attracted a big interest in voice recognition for acoustic modeling since the mid-1980s. One of the first successful applications was phoneme recognition [55], [77]. Then it was expanded to recognize isolated words [17]. Successful results were also achieved for continuous speech recognition, but only on small vocabularies [40]. At the same time, a hybrid HMM/ANN approach was developed [20], [14], [67], [63].

- **Deep Neural Network**

Following the success of hybrid HMM / ANN systems, the recent increase in computing resources has led to the development of deep neural networks (DNNs). These types of neural networks are made up of several hidden layers :

$$Y_{out} = h(M_n h(M_{n-1} \dots h(M_1 x))) \quad (2.3)$$

M_n is the weight matrix of layer n , while $h(\cdot)$ is the activation function. This approach has been shown to enhance the performance of speech recognition tasks compared to standard MLP with one hidden layer [45]. However, these types of networks are known to be difficult to train, especially when the amount of data is limited. Pre-training techniques have been developed to address this issue. These techniques are usually based on learning "good" intermediate representations using unsupervised generative models. These representations serve as a starting point for identification training. Approaches such as greedy layer-wise training and noisy auto-encoder approaches have been proposed. One of the most popular techniques in the speech community is the Deep Belief Networks approach.

The aim of this pretraining approach is to maximize the likelihood of the joint probability of data and labels. It is based on the limited Boltzmann machines framework. Other regularization methods, such as the dropout method, have also been presented. This approach is based on randomly setting to zero a certain amount of the weights on each training update. This approach has the effect of forcing the neurons to not rely on each other, hence enhancing the network's generalization capabilities.

- **Recurrent Neural Networks**

Recurrent Neural Networks (RNN) [30] are a type of neural network in which the connections between the units (or neurons) form a directed graph. To put it another way, an RNN-based model can use the predictions of previous examples to classify an example at a specific time. Bi-directional RNNs [72] have also been proposed, which are made up of two RNNs, one in each direction. As a result, at any given time, the prediction can access predictions in both directions.

Recurrent neural networks-based systems have been proposed in the context of HMM-based speech recognition. The alpha-net [21] is a recurrent neural network technique presented in the HMM framework. Also Robinson [69] suggested a recurrent network in which the network's output is calculated based on the current

input and a hidden state variable that is dependent on all prior inputs. The vanishing gradient problem is the main limitation of these models and that limits access to long-range context. Long-short-term memory networks (LSTM) [46] have been shown to help with this problem.

2.5 Language modeling

A probability distribution over word sequences is what a statistical language model is. The n-gram LM is the simplest model for assigning a probability to a sequence of words. An n-gram is a sequence of n words, a 2-gram (called a bigram) is a two-word sequence of words such as "Arabic Speech", and a 3-gram (called a trigram) is a three-word sequence of words for example "Automatic Speech recognition" [50]. The n-gram can be expressed as indicated in equation 2.4.

$$p(w) = \prod_{k=1}^K p(W_k | w_{k-1}, w_{k-2}, \dots, w_{k-n+1}) \quad (2.4)$$

This equation has two primary hyper-parameters; K is the number of words in W and n is the order of the LM: two for the bigram and three for the trigram LM.

Ideally, one can compute LM for an arbitrary order. However, The increasing value of n in n-gram can lead to sparsity. Therefore, zero probabilities are due to data sparsity, where n values for speech recognition applications are typically in the range of two to four. Maximum likelihood estimation, or MLE, is a simple method for estimating probabilities. We obtain the MLE estimate for the parameters of an n-gram model by obtaining counts from a corpus and normalizing the counts so that they lie between 0 and 1. For example, the following formula can be used to compute the bigram probability of a word w_k given a preceding word w_{k-1}

$$p(w_k | w_{k-1}) = \frac{C(w_{k-1}w_k)}{C(w_{k-1})} \quad (2.5)$$

For the general case of MLE n-gram parameter estimation:

$$p(w_k | w_{k-1}, w_{k-2}, \dots, w_{k-n+1}) = \frac{C(w_{k-n+1}, \dots, w_k)}{C(w_{k-n+1}, \dots, w_{k-1})} \quad (2.6)$$

Equation 2.6 (like Eq.2.5) estimates the n-gram probability by dividing the observed frequency of a particular sequence by the observed frequency of a prefix. This ratio is called a relative frequency. In the training text for some word sequences may be very low or even zero. To prevent the language model from assigning 0 probability to these unseen word sequences, we must reserve some probability mass from more frequent word sequences and allocate it to the unseen word sequences. This is referred to as smoothing or discounting. Normally, two major techniques are used to address this (i) back-off or interpolation [51] in which the model will distribute the probability mass unevenly to unseen word tokens in proportion to the probability that it is less than the lower-order n-gram, (ii) discounting [53] where the smoothing technique is based on assigning some of the probability distribution mass to n-gram sequences that were not seen in training. More advanced language model based on recurrent neural network have also been proposed [60], [56].

2.5.1 Evaluating a Language Model

One method of evaluating Language Model's performance is to embed it in an application and track the application's progress. This end-end evaluation of the Language Model is known as Extrinsic evaluation. In the case of speech recognition, for example, we can run it again with both LMs and see which one offers better transcription. This evaluation makes more sense intuitively because we can tell whether the specific component is improving the application. We can test the quality of the Language Model without relying on any application. This is commonly known as Intrinsic Evaluation. After splitting the corpus into train and test splits, we train two different N-Gram models on the training data and see which LM returns the highest probability for the test sentences. This type of evaluation with the test set is known as intrinsic evaluation. Perplexity (PP) is a quantifiable metric that is used for intrinsic evaluation. It is the inverse probability of the test set. Simply, the lower the perplexity the better the LM. PP can be expressed as shown in equation 2.7.

$$PP = \exp \left\{ -\frac{1}{k} \sum_{k=1}^k \log \left(p \left(w_k | w_{k-1}, w_{k-2}, \dots, w_{k-n+1} \right) \right) \right\} \quad (2.7)$$

2.6 Decoding

The acoustic models and language models are used in decoding for searching the recognition hypothesis that fits best to the models. The Viterbi algorithm [34] is used to find the most probable word sequence. A full breadth search is however infeasible in practice, therefore pruning using beam search technique [39] is usually used to efficiently infer the word sequence.

The following is how beam search is done. We feed a special start of sentence token $\langle sos \rangle$ to the decoder during the first decoding step. After that, the decoder generates a $P(y_1 | \langle sos \rangle, c_0)$ distribution over the G graphemes that make up the output vocabulary. We create a partial transcript and add it to our beam for each of the top k most likely graphemes. We extend each of these k partial transcripts by each of the G graphemes in the next decoding step, resulting in a total of kG candidates. Then, from this set of extensions, we keep only the k most likely partial transcripts. The transcript is considered complete and removed from the beam when the special end of sentence token $\langle eos \rangle$ is encountered. This repeats until there are no more partial transcripts.

2.7 Evaluation

An important issue in speech recognition is how to measure the performance of the system. A commonly metric is the word error rate(WER). WER reflects the number of corrections needed to transform the ASR output into the ground truth. Generally, a lower WER indicates a better quality ASR system. To calculate WER, we can use this formula

$$WER = \frac{I + S + D}{N} * 100\% \quad (2.8)$$



Figure 2.3: Example About WER

- Insertion (I): Words that are incorrectly added in the hypothesis transcript.
- Deletion (D): Words that are undetected in the hypothesis transcript.
- Substitution (S): Words that were substituted between reference and hypothesis.
- N : Total word number.

The example above shows how to calculate the WER. We can see that the ASR has made a few errors. It has inserted the word "حل", identified "سامية" as "سمية" and deleted the word "حالك" from the ground truth.

To calculate WER, we can use the formula in:(2.8). D is the number of deletions, I is the number of insertions, S is the number of substitutions and N is the number of words in the ground truth. In this example, the ASR output made 3 mistakes in total from 5 words in the ground truth. In this case, the WER would be $3 / 5 = 0.6$.

2.8 Classification of Speech Recognition System

2.8.1 Types of Speech Recognition System Based on Utterances

Speech recognition systems can be classified in different classes by describing what types of utterances they can recognize

Isolated Words

Isolated word recognition system which recognizes single utterances, or words. Isolated word recognition is appropriate for situations in which the user is required to provide a single word response or command, but it is unnatural for multiple word inputs. The main advantage of this type is that word boundaries are obvious and the words tend to be clearly pronounced, making it simple and easy to implement [54].

Connected Words

A connected words system works similarly to isolated words, but a connected words system allows separate utterances to be "run-together" with minimal pause between them [54]. The vocalization of a word or words that represent a single meaning to the computer is known as utterance.

Continuous Speech

In a continuous speech recognition system, users can speak almost naturally while the computer determines the content [54].

It is a computer dictation. In this closest words run together without a pause or any other separation between them. It is difficult to develop a continuous speech recognition system.

Spontaneous Speech

Spontaneous speech has a large number of definitions. Basically, it can be considered as speech that sounds naturally and is not repeated. The speaker starts an utterance and reaches a point where he cannot find the right word or thinks better about a word, and needs time to find a suitable alternative, they repeat a word as a kind of "race" on the second attempt. An ASR system with spontaneous speech capability should be able to handle a variety of non-standard grammatical speech features such as words running together "أمم" and "أبيه", and even slight stuttering [54].

2.8.2 Types of Speech Recognition System Based on Speaker Model

Each person has a particular voice, due to his unique physical body and personality. Speech recognition system is classified into two main categories as follows:

Speaker Dependent Models

Speaker dependent systems are designed to work with a specific type of speaker. They are more accurate for a specific speaker, but they may be less accurate for other types of speakers. These systems are typically less expensive, easier to develop, and more accurate. But these systems, are not flexible as speaker independent systems.

Speaker Independent Models

The Speaker Independent system can recognize a variety of speakers without any prior training. A speaker independent system is created to work with any type of speaker. It's used in Interactive Voice Response Systems (IVRS) that have to accept input from a large number of users. These systems are difficult to develop but they are very much flexible.

2.8.3 Types of Speech Recognition Based on Vocabulary

The complexity, processing, and recognition rate of an ASR system are all affected by the vocabulary size of the system. As a result, ASR systems are classified as follows:

- Small Vocabulary - 1 to 100 words or sentences.
- Medium Vocabulary - 101 to 1000 words or sentences.
- Large Vocabulary- 1001 to 10,000 words or sentences.
- Very-large vocabulary - More than 10,000 words or sentences .

2.9 ASR Tools

Speech recognition using deep learning is an enormous task that its success depends on the availability of a large training dataset. The existence of open-source deep learning enabled frameworks and Application Programming Interfaces (API) would increase the

development and research of ASR. There are different services and frameworks that provide developers with powerful deep learning abilities for speech recognition.

2.9.1 API Services

One of the most well-known applications is Cloud Speech-to-Text service from Google ¹ which converts Arabic speech or audio files to text using a deep-learning neural network algorithm. The Cloud Speech-to-Text service allows its translator system to directly accept the spoken word to be converted to text before translating them. For developers the service provides an API with multiple recognition features.

Another service is Microsoft Speech API from Microsoft ². Developers can use this service to create speech recognition systems using deep neural networks.

IBM ³ cloud furnish Watson service API for speech to text recognition support modern standard Arabic language.

2.9.2 Frameworks

We only consider frameworks that can be used locally and without restrictions for this work. Kaldi [66] is one such framework that proved to be the most successful open source ASR system in a previous study [35] it is an open source C++ toolbox that supports both traditional models (such as Gaussian mixing models) and deep neural networks [66].

Wav2Letter is an open-source speech recognition system developed by Facebook[27]. It's a deep neural network framework written entirely in C++ that makes use of the Array Firetensor library.

A more complete implementation was carried out by Mozilla with their DeepSpeech project [42] that can convert an audio to text and is implemented by Tensorflow based on Baidu's DeepSpeech Architecture. These models can be generated for any language.

¹<https://cloud.google.com/speech-to-text/>

²<https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/>

³<https://www.ibm.com/watson/services/speech-to-text/>

2.10 Conclusion

We gave a brief overview of the various aspects of modern speech recognition systems. We presented a literature overview on NN-based acoustic modeling. The language model covered the n-gram approach. Lastly, we described the frameworks and services that aid in the development of the ASR system. Finally, the final section shed some light on recent efforts in Arabic ASR.

In the next chapter, we present the architecture of our proposed method and give an overview about arabic speech recognition and mention most of techniques related to it.

A New Neural Network for Automatic ASR

3.1 Introduction

Research in speech recognition is progressing with many state-of-the-art results in recent times due to its huge applications that can be developed to help humans to improve their daily life tasks. Implementing speech recognition models can be done using several techniques, one of the modern techniques is using neural networks.

This chapter mentions the literature and related works in Modern Standard Arabic based on Neural networks and gives a comparison between them. Also discuss the architecture of our proposed model and the learning algorithm is going to be used in the empirical part of this research to train and optimize the automatic speech recognition model.

3.2 Related works

In this section, we will concentrate on neural network-based approaches, which are a significant class of discriminatory techniques that have inspired biological neural networks. We will go over the literature in terms of classification, steps, and AASR techniques.

3.2.1 General ASR

Dominique F & al [33] showed neural networks as the most extensively used artificial intelligence product, and they provided and studied acoustic and linguistic models of automatic speech recognition systems using the corpus "radio broadcast news". These

models are Deep Neural Networks (DNN) and Hidden Markov Model (HMM) models that were compared to conventional GMM/HMM models and found to have a significant relative improvement of 24%. Hinton et al [45] provide an overview of the use of deep neural networks, which consist of a large number of hidden layers that are trained using some novel techniques. The overview summarizes the findings of four different research groups who collaborated to reveal the benefit of a feed-forward neural network that takes a large number of frames of coefficients as input and outputs subsequent probabilities over HMM states. This method was studied as an alternative to using traditional HMMs and GMMs for speech recognition acoustic modeling. Akushkin et al [48] considered speech recognition in large-resource conditions for Russian language. Their system built on top of the Mozilla DeepSpeech framework and trained with nearly 1650 hours of YouTube crawled data. A similar approach to [48] was performed by Zesch and Agarwal [2]. They set out to train a German transcription system based on Mozilla DeepSpeech. A. Graves and al, combined CNN with LSTM and used beam search for word decoding, and they achieved good results.

Conventional automatic speech recognition systems use a modular, with separate modules for acoustic modeling, pronunciation lexicon, and language modeling that are all trained individually. End-to-end (E2E) models, on the other hand, are trained to convert acoustic features directly to text transcriptions, potentially optimizing all parts for the end task, E2E ASR has attracted interest from both academia and industry. The E2E system is based on a single deep neural network that can be trained to directly transcribe speech into labels (words, phonemes, etc).

3.2.2 Automatic Arabic Speech Recognition

In [80], authors divided the AASR system into four phases. First, the pre-processing phase. Second, the feature extraction stage. Third, decoding utilizing the language model, pronunciation dictionary, and acoustic model. Fourth, Post-processing results where the best hypothesis is produced. The phases work as follows : First, speech waveform is used as input in the pre-processing phase. Then, the output is a processed speech waveform and this is used as input in the feature extraction phase, where they have the feature vector as output and use it as input in the next phase, the decoding phase. In this phase, the acoustic model is employed along with a pronunciation dictionary. Then, the n-best

hypothesis - the output of the pronunciation dictionary phase is used in post-processing as input. As a result, the best hypothesis is generated from this work process.

Turab, Khatatneh and Odeh in [52] discussed the recognition of phonemes as it relates to speech recognition. To improve the results, the Gaussian low-pass filtering algorithm and a neural network were applied in the pre-processing stage. Furthermore, the stages of phoneme recognition are as follows: capturing a signal, sampling, quantification, and energy setting. Following that, a neural network is used to improve the results. Furthermore, this work demonstrates the improved impact in results after applying the Gaussian Low Pass filter to voice signals, resulting in noise reduction. Following that, in the training phase, the neural network was used to train the system to recognize speech signals. Zerari, Naima and al [81] present a basic end-to-end approach for recognizing an isolated Arabic word's spoken digit spelling in Arabic. The key properties of the natural speech signal were extracted using Mel frequency cepstral coefficients (MFCC) [4], which were then processed by a deep neural network capable of dealing with the non uniformity length of voice statement sequences. It first used Long Short Term Memory (LSTM) with three gates to decode the sequences of these features as a fixed vector, which was then passed to a multi-layer perceptron network for classification. The Arabic Digit Spoken dataset is used for training and testing, and it contains 8800 tokens from 88 native Arabic speakers (44 males and 44 females) [41]. This method achieves a 98.7% success rate.

The authors in [31] investigated the use of a neural network to recognize Arabic speech using a distributed word representation. As a result, the neural network achieves robust generalization that is capable of fighting the data sparseness problem in the best way possible. Many factors were investigated, including the n-gram order parameter experiment, output vocabulary, normalization method, model size, and parameters. The Arabic news broadcast and conversation broadcast were used in the evaluation. As a result, using an optimized neural network model over 4-gram, some improvement of up to 3.8% relative WER was achieved.

In [76] the recognition of three Arabic characters, sa (س), sya (ش), and tsa (ث) which have the same pronunciation, presents a difficult problem. Indonesian speakers pronounced these letters, however they had different makhraj in Arabic. From the 738 samples of the used dataset, he retrieved 13 MFCC feature vectors (248 samples of sa), (254 samples of sya), (and 236 samples of tsa). As a classifier, he employed a back-

propagation ANN model with a sigmoid activation function. He had a 92.42% accuracy rate.

With a 1200-hour speech corpus, AlHanai and al in [6] proposed an AASR system. Deep neural network (DNN) structure was used in the developed system, which included many techniques such as feed-forward, convolutional, time delay, recurrent long short-term memory (LSTM), highway LSTM (H-LSTM), and grid LSTM (GLSTM). The corpus evaluation produces the best results, with 18.3% WER using trained GLSTM models.

In [59], they developed the Arabic Loria Automatic Speech Recognition (ALASR) system, which is a voice recognition system based on Modern Standard Arabic (MSA) with an expansion based on Algerian dialects. Because it is mixed with French, the Algerian dialect is considerably distinct from the arabic dialects of the Middle East. The ALASR based on MSA produces good results (a WER of 14.02%), but it entirely collapses on an Algerian dialect data set (a WER of 89%), thus they created a new acoustic model by combining two models: one for MSA and one for French. The WER was reduced as a result of this combination, resulting in a more efficient rate of 65,45%.

The authors in [75] describe an ASR system in the framework of the 2016 Multi-Genre Broadcast (MGB-2) Challenge in Arabic. The fundamental aim behind this research was to integrate the use of GMM-derived features for training a DNN with time-delay neural networks for acoustic models to automatically phonetize Arabic words. The final system was a combination of five systems, and the outcome outperformed the best single LIUM ASR system by 9% in WER.

Ahmed, Abdelrahman, and al [3] present the first complete end-to-end recipe for an Arabic BDRNN-based speech-to-text transcription system. This system contains a character-based decoder for search, which eliminates the need for a word lexicon. The Connectionist Temporal Classification (CTC) is also utilized, which employs an objective function to maximize the output character sequences given the input auditory data. It is made up of three parts: a BDRNNs acoustic model, a language model, and a character-based decoder. The training and decoding processes are also based on the Arabic grapheme. BDRNNs are trained using the CTC objective function, which eliminates the need for pre-segmented acoustic observations. In order to verify the performance with other word-based models, the test set will be evaluated at both the word and character levels. The recipe was tested using a corpus of 1200 hours of Aljazeera's multi-genre

broadcast programs. The WER for non-overlapped speech on the development set is 12.03%. For both well-resourced and low-resourced tasks, morph-based models outperform word-based models, but character-based models are similar to their performance in low-resource tasks, exceeding word-based models. Character-based models succeed in predicting unique word forms not seen in the training data. The word model outperforms the character-based model throughout the whole dataset, whereas the character-based model outperforms the word model by 6% in the under-resourced situation.

Othmane rifki [68] proposed an ASR system using the Wav2Vec2-XLSR-53 Model. He trained this model on the Arabic CommonVoice dataset, achieving a state of the art for Commonvoice arabic test set WER of 46.77%.

Mohamed BEN ALI [10] fine-tuned wav2vec2-large-xlsr-53 on Arabic Language using the Common Voice Corpus 5.1 dataset. As a result he had a 52% WER.

Mohammed Bakheet [13] fine-tuned Wav2vec2-large-xlsr-53 on Arabic using the train splits of Common Voice and Arabic Speech Corpus, and he achieved a result of 36.699% of WER.

Ali Safaya [71] finetuned the Arabic Hubert-Large Model on the Arabic CommonVoice dataset with a 2000h of training, acheiving a state of the art for commonvoice arabic test set WER of 17.68% and CER of 5.49%.

Table 3.1 gives a comparison between the previously described approaches for AASR.

Table 3.1: Comparison between Neural Net work-based Approaches

Approach	Aims	Method	Used DataSets	Result
(Zerari et al., 2018)	Recognize the arabic spoken digit spelling of an isolated arabic word	End to End approach	Consists of 8800 tokens by 88 individual (44 males and 44 females) arabic native speakers [41]	The performance achieves 98.77%
(Emami and Mangu, 2007)	Showing the use of distributed representations and neural network for AASR	Neural networks using distributed word representation	7M words of broadcast news and broadcast conversations acoustic transcripts released by LDC [57]	The result was a reductions of up to 0.8% absolute and 3.8% relative in WER.
(Ahmed et al., 2019)	End-to-End Lexicon Free Arabic Speech Recognition using Recurrent Neural Networks	Based on RNNs and Connectionist Temporal Classification (CTC)	The 1200 hours corpus of Al-jazeera multi-Genre broadcast programs[7]	They achieved a 12.03% WER for non-overlapped speech

Approach	Aims	Method	Used DataSets	Result
(AlHanai et al., 2016)	Development of ASR system for the 2016 arabic Multi-Genre BroadCast (MGB) challenge	Deep Neural Network (DNN)	The 2016 Multi-Genre Broadcast Challenge [7]	Produces results with 18.3% WER using trained GLSTM models.
(Menacer et al., 2017)	The development of ASR system for MSA (namely ALASR), to evaluate it on MSA data and then to test it on a corpus composed of Algerian dialect sentences	DNN approach for The acoustic model and the language model is a classical n-gram	Concerning the acoustic data: 63 hours extracted from Nemlar[58] and NetDC [24] corpora and transcripts of acoustic data plus the Gigaword [37] corpus for language model.	They reported a result of 14.02% in terms of WER.
(Tomashenko et al., 2016)	Develop an ASR system in the framework of the 2016 Multi-Genre Broadcast (MGB-2) Challenge in the Arabic language	The use of GMM-derived features for training a DNN plus time-delay neural networks for acoustic models	The 2016 Multi-Genre Broadcast Challenge [7]	LIUM ASR System achieved a 9% WER

Approach	Aims	Method	Used DataSets	Result
(Wahyuni, 2017)	Distinguish between the pronunciation of three different letters (sa, sya, and tsa) by Indonesian speakers which have the same sound (sa)	Extraction features using Mel-Frequency Cepstral Coefficients (MFCC) then use ANN for classification	738 data of three letters as: 248 data of sa (س), 254 data of sya (ش), 236 data of tsa (ث) Collected them by recording human voice with pronounces letters sa (س), sya (ش), tsa (ث) by depending on the makhras pronunciation of hijaiyah [76]	Average accuracy of 92.42%
Othmane Rifki	Develop an ASR system	Used the Wav2Vec2-XLSR-53 Model	CommonVoice dataset	He achieved a 46.77% WER
Mohamed Ben Ali	Develop an ASR system	Using the Wav2vec2-large-xlsr-53 Model	Common Voice Corpus 5.1 dataset	Produces results with 52% WER
Mohammed Bakheet	Developing an ASR system	He used the Wav2vec2-large-xlsr-53 Model	Two datasets have been used the CommonVoice data set and the Arabic Speech Corpus	He achieved a 36.69% WER
Ali Safaya	Develop an ASR system	He finetuned the Arabic Hubert-Large Model and used it	Common Voice 8.0	achieving WER of 17.68%

Approach	Aims	Method	Used DataSets	Result
Our Approach	Develop an ASR system for Modern standard Arabic	using end-to-end neural system based on Deepespeech's architecture	CommonVoice dataset	Our ASR System achieved a 24% WER

3.3 Arabic Speech Datasets for ASR

Automatic Speech Recognition (ASR) is a very niche field, and there are less resources available for it compared to other ML projects and fields. Table 3.2 describes some Arabic datasets for ASR.

Table 3.2: Datasets for Arabic Speech Recognition

Dataset	Arabic	Nb of speakers	Gender of speakers	Source	Size	Duration	Reference
Common Voice	MSA	1216	male and female	Direct recording	3 Go	Total Hours :137 ,val- idated hr.total: 85	(Ardila et al., 2019)
QASR	MSA	2000	male and female	The data is extracted from the Al Jazeera channel	unknown	2000 hours	(Mubarak et al., 2021)
ESCWA	The data includes international code alteration between Arabic and English	unknown	male and female	Collected at the meetings of the United Nations Economic and Social Commission for Western Asia (ESCWA) in 2021	unknown	2.8 hours.	(Chowdhury et al., 2021)
MGB-3	Arabic dialect (Egyptian)	unknown	male and female	YouTube Channels	1.7GO	16 hours.	(Ali et al., 2017)
MGB-2	Multi-Dialect Broadcast News Arabic Speech	unknown	male and female	Aljazeera TV programs	111.9GO	1200 hours.	(Ali et al., 2016)

Dataset	Arabic	Nb of speakers	Gender	Source	Size	Duration	Reference
Kalam'dz	Eight major subdialects of Algerian Arabic	4881	male and female	Web resources, Youtube, TV, Radio and other social media	unknown	104.4 hours	(Bougrine et al., 2017)
ALG-Darjiah	Algerian Arabic	109	male and female	Direct recording	unknown	4.5 hours	(Bougrine et al., 2016)
AMCAS Corpus	Three groups of Algerian dialects	735	male and female	Phones conversations	unknown	more than 72 hours	(Djellab et al., 2017)

3.4 Proposed Approach

As mentioned earlier in this chapter, end-to-end speech recognition has gained wide acceptance by numerous groups of researchers, for it proved to outperform most of the conventional methods of speech recognition. Furthermore, recent research shows that neural networks perform well in the field of speech recognition, especially when training RNNs using CTC loss function. In this section we will present RNN and CTC loss function.

3.4.1 Basic Structure of RNN

RNN [61] was created with the goal of processing sequence data. In the traditional neural network model, it has input layers, hidden layers and output layers. Each layer are fully connected, and the nodes between each layer are unconnected. But this common neural network is incapable of solving a wide range of issues. For example, if we want to predict the next word in a sentence, we usually need to use the previous word because the words before and after in a sentence are not independent.

The reason why RNN are called recurrent neural networks is that the current output of a sequence is also related to the previous output. The specific manifestation is that the network memorizes the previous information and uses it for the computation of the current output, i.e., the nodes between the hidden layers are no longer unconnected but connected, and the hidden layer's input comprises more than just the input layer's output. It also contains the outputs of the hidden layer from the previous moment. In theory, RNN can process sequence data of any length.

RNNs have input units, the input set is marked as $\{x_{(0)}, x_{(1)}, \dots, x_{(t)}, x_{(t+1)}, \dots\}$, and the output set of Output units is labeled as $\{y_{(0)}, y_{(1)}, \dots, y_{(t)}, y_{(t+1)}, \dots\}$. RNNs also contain hidden units. We mark their output set as $\{h_{(0)}, h_{(1)}, \dots, h_{(t)}, h_{(t+1)}, \dots\}$. These hidden units complete the most important work.

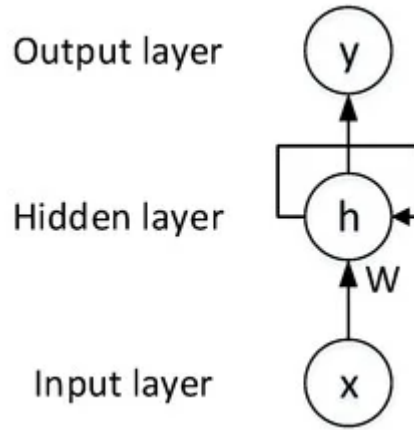


Figure 3.1: Basic structure of RNN [62].

$$h_t = f_w(h_{t-1}, x_t) \quad (3.1)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \quad (3.2)$$

$$y_t = W_{hy}h_t \quad (3.3)$$

Note that $h(t)$ is the new state and $h(t-1)$ is the old one. Each time the new state is computed by new function $f(w)$ with parameters old state $h(t-1)$ and input vector at some time step $x_{(t)}$. The unfold structure is shown below.

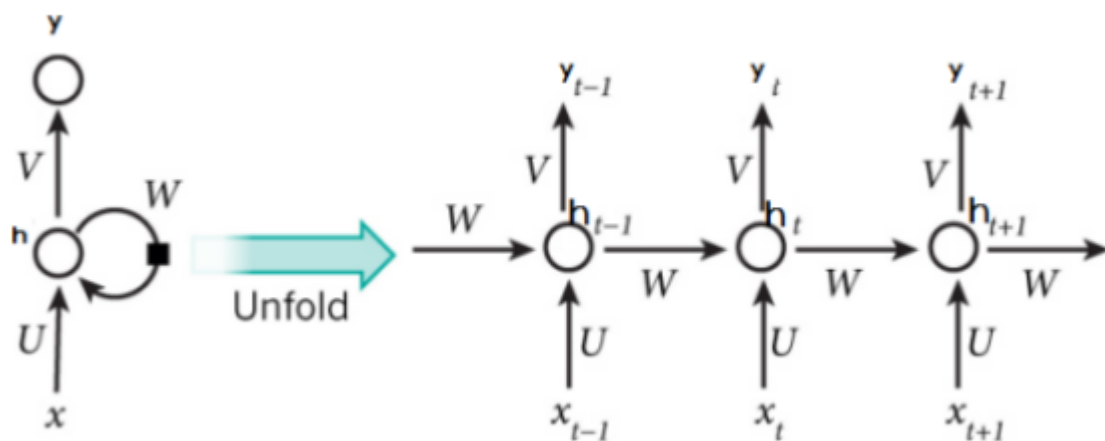


Figure 3.2: RNN unfold structure [73]

3.4.2 LSTM

The Long Short Term Memory network, or LSTM for short, is a type of RNN that can learn long-term dependency information. Hochreiter and Schmidhuber proposed LSTM in 1997, and it has recently been improved and promoted by Alex Graves. In many issues, LSTM has achieved considerable success.

Moreover, LSTM is designed to avoid long-term dependencies. A chained form of repeating neural network modules exists in all RNNs. This repeated module in a standard RNN has a very simple structure, such as a tanh layer. The structure of the LSTM is the same, but the structure of the repeated modules is different. Unlike a single neural network layer, there are four here to interact in a very special way. LSTM is a special network structure that consists of three "gate" structures. Figure 3.3 shows the structure of LSTM.

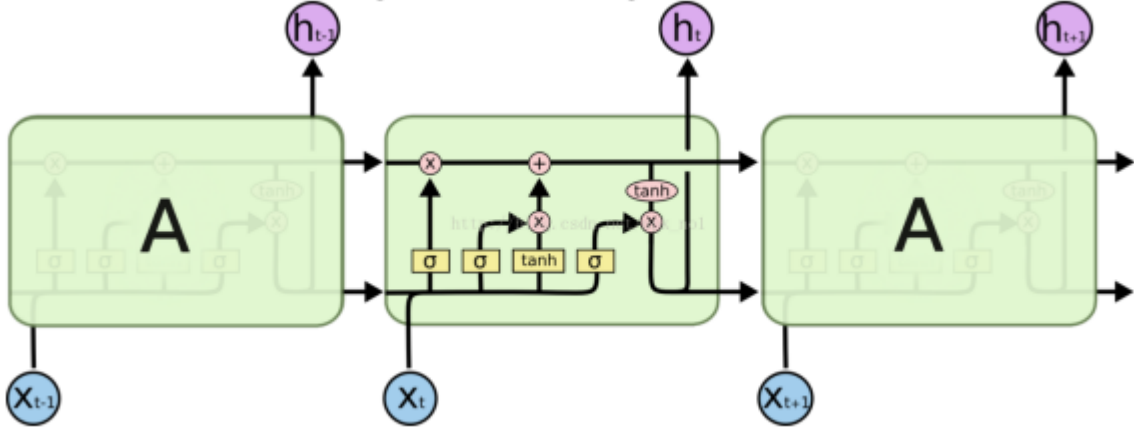


Figure 3.3: Structure of LSTM [16]

LSTM relies on some "gate" structures to allow information to selectively affect the state of each moment in the RNN. "Gate" structure is a sigmoid neural network and a bitwise multiplication operation. These two operations form a "gate" structure when combined. The reason this structure is called a gate is because the fully connected neural network layer uses sigmoid as the activation function to output a value between 0 and 1. It specifies how much data from the current input can pass through this structure, so its function is similar to that of a door. When you open the door (sigmoid output is 1), all information can pass and when the door is closed (sigmoid output is 0), no information can pass. There are six formulas shown below to describe the structure of a LSTM.

input gate:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (3.4)$$

forget gate:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (3.5)$$

candidate memory unit:

$$\check{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \quad (3.6)$$

current memory unit:

$$C_t = f_t C_{t-1} + i_t \check{C}_t \quad (3.7)$$

output gate

$$o_t = \sigma(W_f[h_{t-1}, x_t] + b_o) \quad (3.8)$$

output:

$$h_t = o_t \tanh(C_t) \quad (3.9)$$

3.4.3 Connectionist Temporal Classification

The first method for training end-to-end neural network models for ASR was connectionist temporal classification (CTC) [38]. CTC models are made up of a single recurrent network (sometimes with multiple layers) that produces one output every input time-step. A softmax layer sits at the top of the network, converting the inputs into a probability distribution over the vocabulary. In this section, we will assume that the vocabulary is the set of Arabic characters, plus a ‘blank’ unit, which is required for CTC and which will be explained shortly. We will represent the blank unit with an underscore in this section.

After feeding a set of speech features into the CTC model, we get one probability distribution per input time-step as the result. We will call the probability of observing label k at time t y_t^k . We define a path π as a sequence of labels of length T , where T is the length of both the input and output sequences. The probability of path π according to the CTC model is $p(\pi|x) = \sum_{t=1}^T y_t^{\pi_t}$, or the total probability of observing the labels that make up π .

The ability of CTC to handle the vast majority of cases in which the desired transcript is not the same length as the input sequence is its key feature. First, CTC makes an assumption about the ASR task: that the length of the desired output is less than or equal to the length of the input sequence. This assumption is a reasonable one for ASR when there are, as is most common, tens or hundreds of speech feature frames per second.

Second, CTC is based on a simple function that maps the sequence of outputs to a ‘labeling’, which is the actual transcript of the input speech. This function handles cases in which the desired transcript is shorter than the input sequence. Consider a simple training example: a sequence of four speech feature frames and the word *مهر*. In CTC,

there are several different ways that the model could take in these 4 frames and correctly produce the labeling نهر. They are: نهر, نه-ر, ن-هر, نهر, نهر, نهر, نهر, and نهر. The mapping removes all blanks and repeated labels so that each of these output sequences map to the same labeling, نهر.

The CTC loss function is used to train CTC models, and it maximizes the likelihood of the desired labeling by marginalizing over every possible output sequence that condenses into that labeling. This is possible in part because of the way CTC models are structured: the outputs are conditionally independent given the input. Given an input x and ground-truth labeling y , the CTC objective function which we want to maximize is:

$$O(x, Y) = \sum_{\pi \in Y} p(\pi|x) \quad (3.10)$$

where Y is the set of paths that condense to the labeling y . A variant of the forward-backward algorithm specifically designed for CTC can compute both this objective function and the best labeling for a new input [38].

3.4.4 Model Architecture

The suggested model is going to be following the concept of Mozilla DeepSpeech [42], because it is an end-to-end neural system that can be simply trained. We use a deep recurrent neural network (RNN), which can be trained end-to-end using supervised learning. It extracts Mel-Frequency Cepstral Coefficients [49] as features and directly outputs the transcription, without requiring the forced alignment on the input or any external source of knowledge like a Grapheme to Phoneme (G2P) converter a neural network that consists of six layers. The input layer contains m inputs, each of which corresponds to m MFCC features extracted from a short audio signal. The second and third layers are identical to the first. Concerning the first three layers, the nodes in the neighbouring layers are fully connected and the ReLU (Rectifier Linear Unit) is used as an activation function : $\text{ReLU}(x) = \max(0, x)$. The fourth layer is a recurrent neural network in this case LSTM. The result is sent to the fifth layer having ReLU as the activation function. The final layer of the model is also a non-recurrent layer which outputs standard logits corresponding to predicted character probabilities of each time slice and character in the alphabet. SoftMax activation function is further applied on this layer. The number of

nodes in the output layer corresponds to the number of alphabets in the language for instance with Arabic alphabets, the number of nodes in the output layer is 36 denoting أ-ي , where the output value of each node is proportional to the probability of the respective character of the alphabet. Further, the Connectionist Temporal Classification (CTC) loss function [38] is used to maximize the probability of the correct transcription. Finally, the model is optimized using Adam's optimization. Dropout is applied to all five layers with the probability of 0.4 in order to prevent the neural network from overfitting. The ASR architecture used in our experiments is illustrated in Figure 3.4.

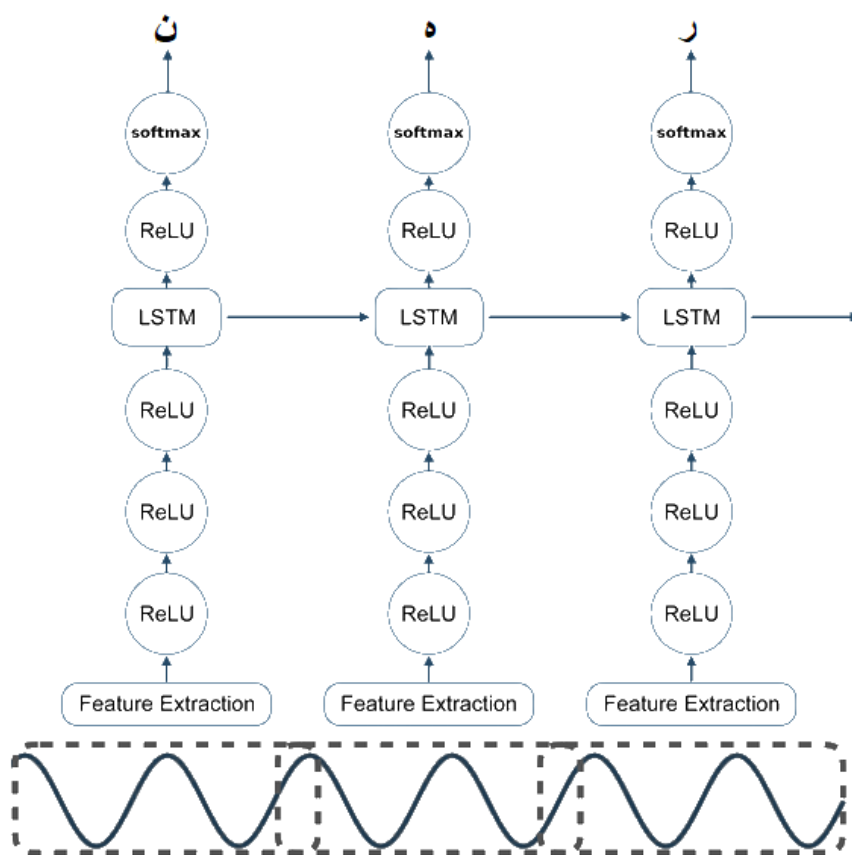


Figure 3.4: Architecture of DeepSpeech [42]

The output is an acoustic model, mapping acoustic features to probabilities of alphabets. The language model is used to convert these probabilities into a sequence of meaningful words. The language model is trained over our corpus's transcripts, which then assigns probabilities to valid Arabic words and phrases. Once a language model has been learned from the training data, the trained language model gives valid sentences a higher probability than invalid sentences. A 5-gram language model is trained using

kenLM [43].

The Figure 3.5 shows the paths and components for training and use after the model is developed. Features are extracted from the audio input and with this the model is trained, based on the results reflected in WER or loss function the weights and biases are adjusted to improve the model. For a working model, the decoder outputs results reflected in audio and text.

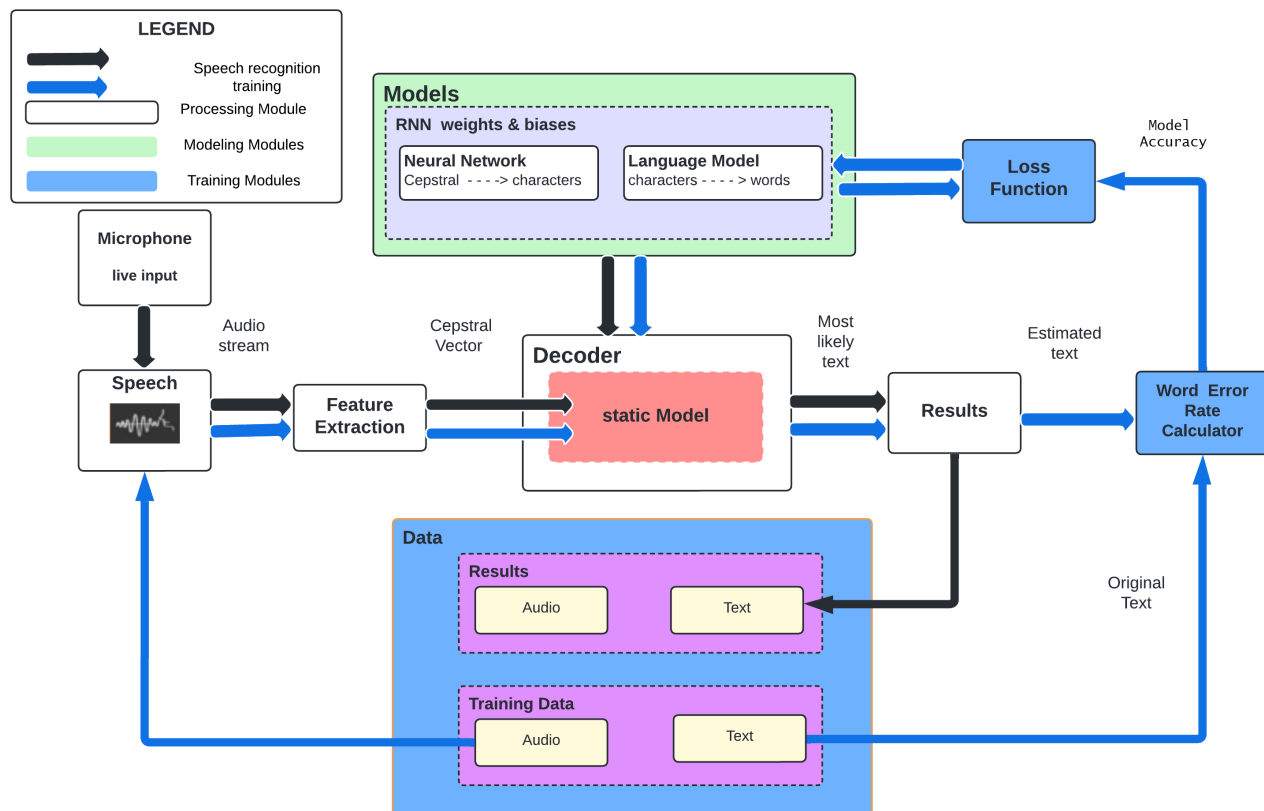


Figure 3.5: Diagram of our Proposed Approach

3.5 Conclusion

This chapter reviewed surveys and journal papers about Automatic Speech Recognition and mentioned most of the techniques related to it, and showed that the field of Automatic Speech Recognition is broadly investigated for Modern Standard Arabic and dialectical arabic. Therefore, this research adopts an approach based on machine learning to develop a speech recognition model for Modern Standard Arabic.

Implementation and Experiment

4.1 Introduction

This chapter is carried out to train and test DeepSpeech model for the Arabic language (MSA). The reason why we choose Deep Speech as the core toolkit is that deep learning is the most popular algorithm in recent years. Deep Speech is already implemented by Mozilla. The source code for DeepSpeech is available on Github (<https://github.com/mozilla/DeepSpeech/>). It has a pre-trained model for Mandarin and English. But it doesn't have pre-trained Arabic model. This experiment uses our corpus as the main corpus to train, and we will describe this corpus in section 4.3. The Mozilla Deep Speech is implemented based on a standard deep speech algorithm.

Section 4.2 shows a description of the files required to set up the entire system.

Section 4.3 provides a description of how each portion of the corpus was processed to match requirements.

Section 4.5 contains all the steps to process the experiment.

4.2 Preliminaries

In order to train the Deep Speech system, there are few files must be provided. Deep Speech is implemented by Mozilla under a DeepSpeech folder. The system's directory has the following structure:

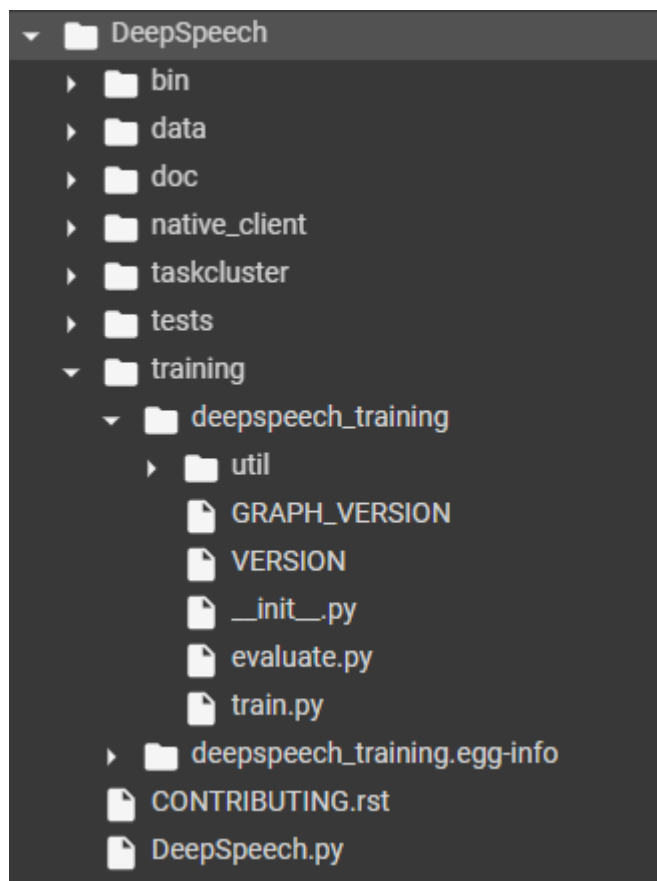


Figure 4.1: The structure of DeepSpeech directory

The main script is `DeepSpeech.py` located in the project's root directory. For its list of command line options, you can find in `flags.py`, which contains all the options, like dropout rate, learning rate and so on. The details are placed in the appendix.

`util` file contains all the script for preparing the experiment and evaluating the final results. The following files are included in it:

- `check_characters.py`: checking what characters are contained in the csv file.
- `evaluate.py`: evaluate the result after testing.
- `gpu.py`: check GPU usage.
- `config.py`: check the configuration.
- `stm.py`: calculate the running time.
- `taskcluster.py`: check the process.
- `flags.py`: contains all the flags.

4.3 Data Collection and Processing

The first main task in speech recognition is data collection and processing. The speech dataset used for training contains audio utterances and their corresponding transcriptions. Speech dataset can be of reading, conversational, or spontaneous speech. To create a read speech corpus, the following minimal steps must be done. 1) Prepare large text data (sentences) that cover the majority of the target language’s vocabulary. 2) Record the sentences spoken by a variety of speakers (native, fluent, various ages and genders) in various environments (noise, clean, outdoor/indoor). A conversational speech corpus can be created by recording and transcribing a speaker’s normal conversation on any topic. Commonly, these raw recordings are referred to as uncleaned dataset. To prepare a clean dataset, post processing is required. Steps in the post-processing process include preparing the transcriptions of audio utterances, cleaning audio samples by removing too noisy, repeated words, etc, and synchronization of audio and text transcriptions. The entire process is time-consuming and costly (recording setup, incentives to speakers, hosting of data, etc). Mozilla has a good open-source initiative to create a free speech corpus [12], the Common Voice Corpus 9.0, has recorded speech data of 20217h covering 93 languages. Common Voice Corpus 9.0 has over 2953h of recorded speech for the English language, while it has only 139h for the Modern standard Arabic language out of which only 88h is validated. In our work, we collected the dataset from Common Voice (CV)(cv-5.1 and cv-6.1) speech corpus for training Arabic ASR. The summary of the corpora, total words, unique words, total duration, total utterances, and average duration of each utterance is listed in Table 4.1

Table 4.1: Detail of the Corpus

Dataset	Total words	Total unique words	Total utterance	Average duration (sec)
CommonVoice (train)	166723	27925	29875	4
CommonVoice (dev)	8415	3505	1918	3.75
CommonVoice (test)	8270	3430	1911	3.4

4.3.1 Preprocessing

All MP3 audio clips are converted into wave format, and have a mono audio channel with a sample rate of 16,000 Hz and a depth of 16 bit for each value so that they can be read by our DeepSpeech pipeline. Data was divided into three sets training, validation and test. Diacritics are used to provide a phonetic guide i.e. to show the correct pronunciation because the normal Arabic text does not provide enough information about it. Because DeepSpeech is a character level system, adding these representations may have an impact on the acoustic model's performance. More alphabets signify more possibilities of prediction. We decided to remove all existing diacritics from the dataset. We cleaned the transcriptions by removing commas, punctuation, special characters, etc. Using shell script. As shown in Figure 4.2.

```
[ ] %%writefile prepro.sh
dir=/content/drive/MyDrive/cv-corpus-5.1-2020-06-22/ar/clips

for type in train test dev; do
  cat $dir/$type.csv \
  | awk -F ',' '{if(NR == 1 || $3 !~ /[a-zA-Z]/) print $0}' \
  | sed 's/[~]//g' \
  | sed 's/ٲ/ٲ/g' \
  | sed 's/ٳ/ٳ/g' \
  | sed 's/ٴ/ٴ/g' \
  | sed 's/ٵ/ٵ/g' \
  | sed 's/ٶ/ٶ/g' \
  | sed 's/ٷ/ٷ/g' \
  | sed 's/ٸ/ٸ/g' \
  | sed 's/٩/٩/g' \
  | sed 's/[ٲٳٴٵٶٷٸ٩]//g' \
  | sed "s/$(printf %b '\u0670')//g" \
  | sed "s/$(printf %b '\u06d6')//g" \
  | sed "s/$(printf %b '\u06d7')//g" \
  | sed "s/$(printf %b '\u06d8')//g" \
  | sed "s/$(printf %b '\u06da')//g" \
  | sed "s/$(printf %b '\u06db')//g" \
  > $dir/$type.clean.csv
done
```

Figure 4.2: Preprocessing CommonVoice Arabic text

4.3.2 Forced Aligning

To make it possible for the model to train on the data, it is necessary to align the transcriptions of the audio file and save it as (comma-separated values) file to feed it to the model, each of the CSV file rows contains three values the first column represents

audio filename, the second represents audio file size and the third column represents the respected text to each audio file.

All of what was mentioned above has been done and the resulted CSV file totaled at 29875 records represents the audio files as long as their transcription is all aligned together.

Figure 4.3 shows a preview of the dataset and each of its columns.

1	wav_filename	wav_filesize	transcript
2	common_voice_ar_19221910.wav	142124	هل أخذت هذه الصورة حديثا
3	common_voice_ar_19221908.wav	130604	لماذا لا تأكل الخضروات
4	common_voice_ar_19221906.wav	163628	لا أعسل شعري بالصابون في الصباح
5	common_voice_ar_19221931.wav	171308	حضر المؤتمر مئة و خمسون ديبلوماسي
6	common_voice_ar_19221930.wav	140588	هل لديك أي أفلام رصاص
7	common_voice_ar_19221932.wav	165164	من المهم أن تتذكر من هم أصدقاؤك
8	common_voice_ar_19221965.wav	117548	منزلك رائع
9	common_voice_ar_19221935.wav	126764	هل سجلوا أسماءهم للتصويت
10	common_voice_ar_19221960.wav	119084	أحب جيرانك
11	common_voice_ar_19221963.wav	109868	هل ستساعدني
12	common_voice_ar_19221968.wav	86828	مرحبا بك
13	common_voice_ar_19221969.wav	134444	جاء ليطلب مساعدتنا
14	common_voice_ar_19221967.wav	94508	لنن هذا
15	common_voice_ar_19221966.wav	180524	شكرا لمجبتك كل هذه المسافة لتودعني
16	common_voice_ar_19221983.wav	126764	هو أبح تارو الأكبر
17	common_voice_ar_19221984.wav	119084	تتعجب الشمس كل يوم
18	common_voice_ar_19221982.wav	140588	البارحة كانت ليلة عيد ميلادي
19	common_voice_ar_19221999.wav	153644	يمكنك أن تذهب إلى أي مكان تريد
20	common_voice_ar_19221998.wav	145964	تلك الدمية هدية من صمتي
21	common_voice_ar_19222003.wav	125228	إنها تكثر من الكلام

Figure 4.3: Aligned Dataset Preview

4.4 Operation Environment

The implementation of the proposed methodology is executed by adopting a variety of tools and concepts and choosing the proper platform for developing the speech recognizer, all of the tools are briefly described below.

4.4.1 Jupyter Notebook

The Jupyter Notebook is an open source web application known as a computational notebook that allows researchers to create and share documents with live code, equations, visualizations, and text. The name, Jupyter, stands for Julia (Ju), Python (Py), and R combined. A Jupyter notebook can be executed locally or on a remote server (cloud). Each document is composed of multiple cells, each of which contains script language or

markdown code. This technology makes it easier to share and replicate scientific works [65].

4.4.2 TensorFlow

TensorFlow is a large-scale machine learning system that works in a Heterogeneous environments, It maps the nodes of a dataflow graph across multiple machines in a cluster, as well as multiple computational devices within a single machine, such as multicore CPUs, general-purpose GPUs, and custom-designed ASICs known as Tensor Processing Units (TPUs). TensorFlow allows developers to try out new optimizations and training algorithms. TensorFlow supports a wide range of applications, with a focus on training and inference on deep neural networks. TensorFlow is used in production by several Google services. It was released as an open-source project, and become widely used for machine learning research [1].

4.4.3 Google Colaboratory (Colab)

Google Colaboratory also known as Colab, is a project aimed at disseminating machine learning education and research. Colaboratory notebooks are based on Jupyter and function similarly to Google Docs objects in that they can be shared and users can work together on the same notebook. Colaboratory makes available Python 2 and 3 runtimes pre-configured with machine learning and artificial intelligence libraries like TensorFlow, Matplotlib, and Keras.

Colab operates under Ubuntu 17.10 64 bits and it is composed of an Intel Xeon processor (not specified) with two cores @2.3 GHz and 13 GB RAM. It is equipped with an NVIDIA Tesla K80 (GK210 chipset), 12 GB RAM, 2496 CUDA cores @560 MHz [22]. The proposed model is going to be computed using Colab which is free to use the Jupyter notebook environment.

4.5 Experiment Step

First, the master branch of the DeepSpeech repository was cloned using the following command :

```
[ ] ! git clone https://github.com/mozilla/DeepSpeech --branch v0.7.4
```

Figure 4.4: Cloning DeepSpeech Repository

Next, all the pre-requested softwares were installed on the machine, including tensor-flow, python, and so on. Then we can build the Language Model (LM) by using Kenlm, using vocabulary.txt, which contains all the textual data. The command is shown below:

```
! %cd /content/DeepSpeech/data/lm
! python3 generate_lm.py --input_txt vocabulary.txt --output_dir . \
--top_k 500000 --kenlm_bins /content/kenlm/build/bin/ \
--arpa_order 5 --max_arpa_memory "85%" --arpa_prune "0|0|1" \
--binary_a_bits 255 --binary_q_bits 8 --binary_type trie
```

Figure 4.5: Building Language Model

Finally, the machine can start training the data with the following command:

```
!python3 '/content/DeepSpeech/DeepSpeech.py' \
--drop_source_layers 1\
--checkpoint_dir '/content/drive/MyDrive/checkpoints'\
--alphabet_config_path '/content/DeepSpeech/data/alphabet.txt' \
--train_files '/content/drive/MyDrive/cv-corpus-5.1-2020-06-22/ar/clips/tr
--dev_files '/content/drive/MyDrive/cv-corpus-5.1-2020-06-22/ar/clips/dev.
--test_files '/content/drive/MyDrive/cv-corpus-5.1-2020-06-22/ar/clips/test
--export_dir '/content/drive/MyDrive/export' \
--n_hidden 512 \
--train_batch_size 80 \
--dev_batch_size 80 \
--test_batch_size 64 \
--dropout_rate 0.4\
--learning_rate 0.0001\
--summary_dir '/content/drive/MyDrive/summary' \
--early_stop true\
--scorer '/content/drive/MyDrive/lm/Arabic_scorer.scorer'\
--es_epochs 5\
--max_to_keep 15\
```

Figure 4.6: Model Training

In this command, the batch size for training, validation and testing is 80,80,64. The numbers of hidden layers are 512. In order to avoid overfitting, we also set the dropout rate as 0.4. After finishing all the experiment steps, the Model begins to train. The training and testing took approximately 6 hours. In the next section, we will show the final results.

4.6 Results of Experiment

In order to evaluate the results, we use WER as the standard to evaluate. WER stands for word error rate. This standard is computed by the formula below.

$$\text{Word Error Rate} = 100 * \frac{\text{Insertion} + \text{Substitution} + \text{Delete}}{\text{NumberofWords}} \quad (4.1)$$

The key for training is to make small modifications to the parameters to see if the results are good or not. Each time, we use the training set and validation set to train the model. Then the model will be tested by test set, and it will output the WER. The best WER we got is 0.243 (24.3%), which is a low value. Figure 4.7 shows some examples of the recognition.

```
Test on /content/drive/MyDrive/cv-corpus-5.1-2020-06-22/ar/clips/test.clean.csv - WER: 0.243055, CER: 0.176742, loss: 22.196198
-----
Best WER:
-----
WER: 0.000000, CER: 0.000000, loss: 70.906418
- wav: file:///content/drive/MyDrive/cv-corpus-5.1-2020-06-22/ar/clips/common_voice_ar_20755031.wav
- src: "لا يحزنهم الفزع الأكبر وتلقاهم الملائكة هذا يومكم الذي كنتم توعدون"
- res: "لا يحزنهم الفزع الأكبر وتلقاهم الملائكة هذا يومكم الذي كنتم توعدون"
-----
WER: 0.000000, CER: 0.000000, loss: 59.960743
- wav: file:///content/drive/MyDrive/cv-corpus-5.1-2020-06-22/ar/clips/common_voice_ar_19204945.wav
- src: "أعتقد أنه من الأفضل لك أن تستريح إنك تبدو مريضاً"
- res: "أعتقد أنه من الأفضل لك أن تستريح إنك تبدو مريضاً"
-----
WER: 0.000000, CER: 0.021739, loss: 56.487614
- wav: file:///content/drive/MyDrive/cv-corpus-5.1-2020-06-22/ar/clips/common_voice_ar_19203958.wav
- src: "عندما كنت في الثانوية استيقظت السادسة كل صباح"
- res: "عندما كنت في الثانوية استيقظت السادسة كل صباح"
-----
WER: 0.000000, CER: 0.000000, loss: 52.445404
- wav: file:///content/drive/MyDrive/cv-corpus-5.1-2020-06-22/ar/clips/common_voice_ar_19651951.wav
- src: "كان نصف حالة من التبغ بقيمة ثلاثة أرطال"
- res: "كان نصف حالة من التبغ بقيمة ثلاثة أرطال"
-----
WER: 0.000000, CER: 0.000000, loss: 52.185123
- wav: file:///content/drive/MyDrive/cv-corpus-5.1-2020-06-22/ar/clips/common_voice_ar_20401421.wav
- src: "عند دخولي إلى الغرفة كان مستغرقاً في قراءة كتاب"
- res: "عند دخولي إلى الغرفة كان مستغرقاً في قراءة كتاب"
```

Figure 4.7: Results

Figure 4.8 presents WER/epoch

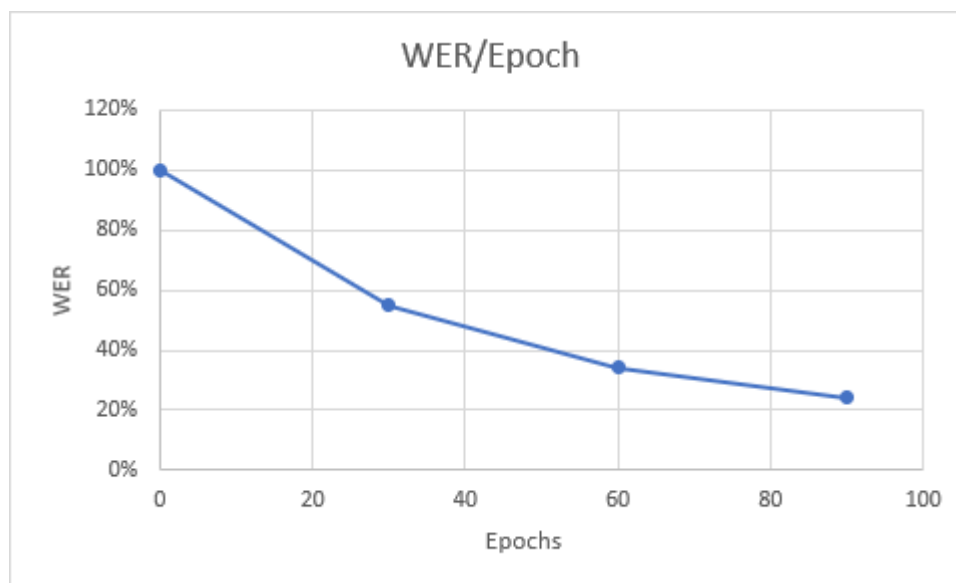


Figure 4.8: Evolution of WER

4.6.1 Comparison of Results

The table below 4.2 gives a comparison between our DeepSpeech model and some previous works on Arabic ASR:

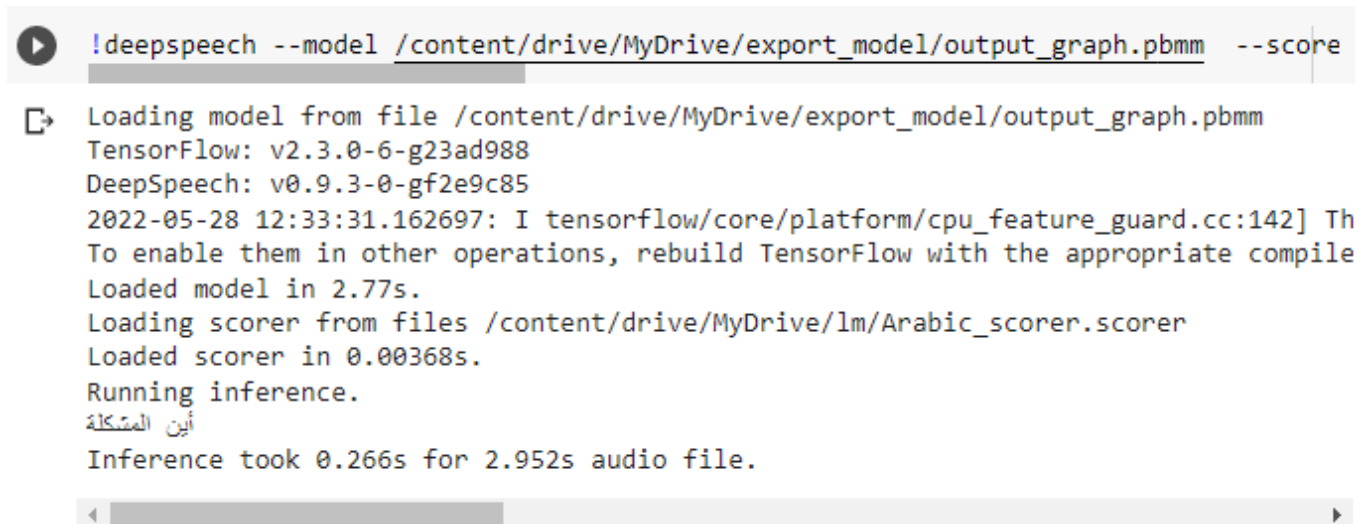
Table 4.2: Comparison between ASR systems for Modern Standard Arabic

Model	Dataset	WER	CER
Safaya's hubert-large-arabic-ft	CommonVoice	17.68%	5.49%
Our DeepSpeech Arabic model	CommonVoice	24.3%	17.6%
Bakheet's wav2vec large xlsr53	CommonVoice and arabic speech corpus	36%	-
Othmane's Wav2Vec2 XLSR53	CommonVoice	46.77%	-

It's clearly visible that our model reduced the WER by 11.7% compared to Bakheet's Wav2Vec model. the safaya's model trained on 2000h dataset and tested on CommonVoice dataset. Thus, to achieve better results than safaya's model we need a large dataset and a powerful GPUs.

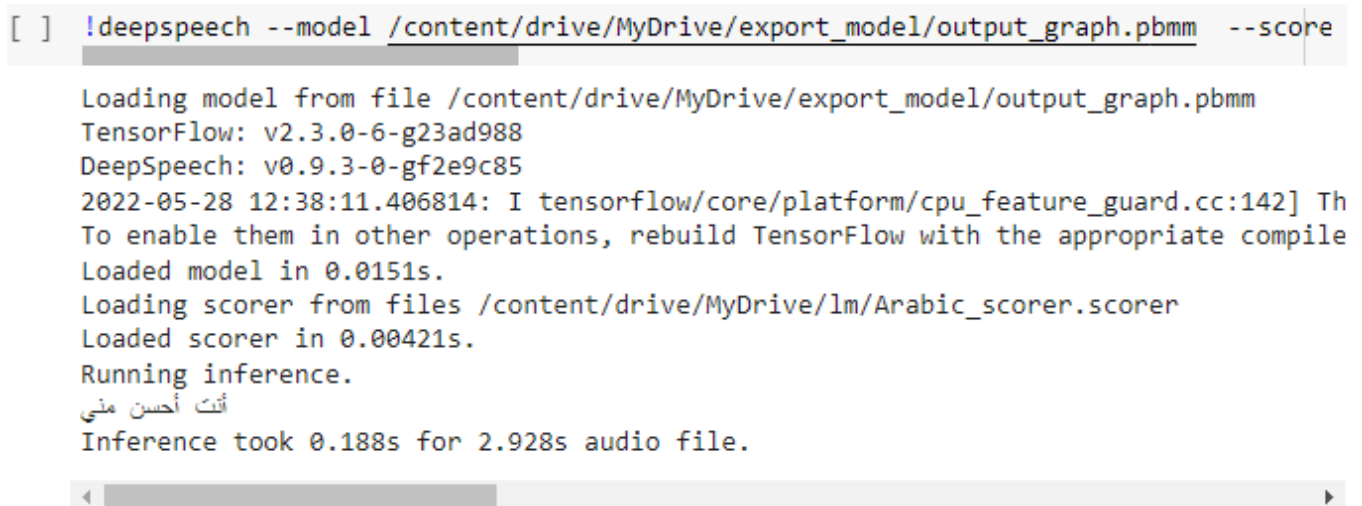
4.7 Deployment

After we have trained and evaluated our model, we are ready to use it for inference where spoken phrases, utterances are assessed by our trained model and a text transcription provided. Some examples are shown in Figures 4.9, 4.10, 4.11.



```
!deepspeech --model /content/drive/MyDrive/export_model/output_graph.pbmm --score
Loading model from file /content/drive/MyDrive/export_model/output_graph.pbmm
TensorFlow: v2.3.0-6-g23ad988
DeepSpeech: v0.9.3-0-gf2e9c85
2022-05-28 12:33:31.162697: I tensorflow/core/platform/cpu_feature_guard.cc:142] Th
To enable them in other operations, rebuild TensorFlow with the appropriate compile
Loaded model in 2.77s.
Loading scorer from files /content/drive/MyDrive/lm/Arabic_scorer.scorer
Loaded scorer in 0.00368s.
Running inference.
أين المشكلة
Inference took 0.266s for 2.952s audio file.
```

Figure 4.9: Example 1



```
[ ] !deepspeech --model /content/drive/MyDrive/export_model/output_graph.pbmm --score
Loading model from file /content/drive/MyDrive/export_model/output_graph.pbmm
TensorFlow: v2.3.0-6-g23ad988
DeepSpeech: v0.9.3-0-gf2e9c85
2022-05-28 12:38:11.406814: I tensorflow/core/platform/cpu_feature_guard.cc:142] Th
To enable them in other operations, rebuild TensorFlow with the appropriate compile
Loaded model in 0.0151s.
Loading scorer from files /content/drive/MyDrive/lm/Arabic_scorer.scorer
Loaded scorer in 0.00421s.
Running inference.
أنت أحسن مني
Inference took 0.188s for 2.928s audio file.
```

Figure 4.10: Example 2

```
[ ] !deepspeech --model /content/drive/MyDrive/export_model/output_graph.pbmm --score
Loading model from file /content/drive/MyDrive/export_model/output_graph.pbmm
TensorFlow: v2.3.0-6-g23ad988
DeepSpeech: v0.9.3-0-gf2e9c85
2022-05-28 12:39:43.745512: I tensorflow/core/platform/cpu_feature_guard.cc:142] Th
To enable them in other operations, rebuild TensorFlow with the appropriate compile
Loaded model in 0.0155s.
Loading scorer from files /content/drive/MyDrive/lm/Arabic_scorer.scorer
Loaded scorer in 0.00544s.
Running inference.
قال لا أبالي
Inference took 0.196s for 2.592s audio file.
```

Figure 4.11: Example 3

4.8 Conclusion

This chapter covered the implementation of our proposed approach for this research. We demonstrated some of the essential tools and their underlining technology and features needed to apply the methodology of this research. Moreover, we presented the results by implementing the suggested setup and configuration to fulfill the paramount goal of this research which is using neural networks (end to end approach) to train ASR model for the Modern Standard Arabic language.

General conclusion

This research addressed the possibility of designing ASR system with end to end neural network for the Modern Standard Arabic. We started with the collection of related data and then went through the process of designing the represented dataset. A suitable learning algorithm was chosen, and the structure of the model was built by training the DeepSpeech model. And evaluating the produced results from the machine learning model.

From the beginning of this research, there have been a lot of hurdles to overcome, especially the two major steps (collecting a dataset and training the model). Processing of the collected data were achieved, and training the model took time because of the shortage of computation power.

We designed the first DeepSpeech model for Modern standard Arabic using Common-Voice dataset, which will be very helpful for relevant future researches, and developing Modern Standard Arabic recognizer to convert spoken speech to corresponding text form.

The results were evaluated to measure the performance of the proposed model. The Google Colab platform was used for training and validation. The results showed that the model has potentials in converting spoken modern standard Arabic to text format, the model reached 24.3% in term of WER and 17.6% in term of CER.

This was our first experience in dealing with real world projects in the field of deep learning, this allowed us to improve our knowledge and skills in this field.

We have more ideas, we plan to try in the future in order to improve the performance of the model that we worked with, including increasing the dataset size by combining more datasets. Preprocessing the audio to improve its quality is another development also that would improve the overall performance of the model. While we did the inference

the model predicted different transcripts for the same audio which could have resulted from the quality of the various audio files passed to the model. A solution to this issue would be to implement further preprocessing such as noise cancellation on the audio file before passing them to the model for the transcript.

Bibliography

- [1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). {TensorFlow}: A system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283.
- [2] Agarwal, A. and Zesch, T. (2019). German end-to-end speech recognition based on deepspeech. In *KONVENS*.
- [3] Ahmed, A., Hifny, Y., Shaalan, K., and Toral, S. (2019). End-to-end lexicon free arabic speech recognition using recurrent neural networks. In *Computational Linguistics, Speech And Image Processing For Arabic Language*, pages 231–248. World Scientific.
- [4] Al-Anzi, F. S. and AbuZeina, D. (2017). The capacity of mel frequency cepstral coefficients for speech recognition. *International Journal of Computer and Information Engineering*, 11(10):1149–1153.
- [5] AL I FA RG HA, L. (2012). Statistical and symbolic paradigms in arabic computational linguistics. *Arabic language and linguistics*, page 35.
- [6] AlHanai, T., Hsu, W.-N., and Glass, J. (2016). Development of the mit asr system for the 2016 arabic multi-genre broadcast challenge. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 299–304. IEEE.
- [7] Ali, A., Bell, P., Glass, J., Messaoui, Y., Mubarak, H., Renals, S., and Zhang, Y. (2016). The mgb-2 challenge: Arabic multi-dialect broadcast media recognition. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 279–284. IEEE.

-
- [8] Ali, A., Vogel, S., and Renals, S. (2017). Speech recognition challenge in the wild: Arabic mgb-3. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 316–322. IEEE.
- [9] Ali, A. M. A. M. (2018). Multi-dialect arabic broadcast speech recognition.
- [10] ALI, M. B. (2021). Speech recognition on common voice arabic. <https://huggingface.co/mohamed1ai/wav2vec2-large-xls-ar>. Accessed 07 June 2022.
- [11] Alqrainy, S. and Rashaideh, H. (n.d). Toward developing a universal and standard system for transliteration and transcription arabic language.
- [12] Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F. M., and Weber, G. (2019). Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*.
- [13] Bakheet, M. (2021). Speech recognition on common voice arabic. <https://huggingface.co/mohammed/wav2vec2-large-xlsr-arabic>. Accessed 07 June 2022.
- [14] Bengio, Y. (1993). A connectionist approach to speech recognition. In *Advances in Pattern Recognition Systems Using Neural Network Technologies*, pages 3–23. World Scientific.
- [15] Betti, M. J. and Ulaiwi, W. A. (2018). Stress in english and arabic: A contrastive study. *English Language and Literature Studies*, 8(1):83–91.
- [16] Biswal, A. (2022). Recurrent neural networks (rnn) tutorial: Types, examples, lstm and more. <https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn>. Accessed: 23 April 2022.
- [17] Bottou, L., Soulié, F. F., Blanchet, P., and Lienard, J.-S. (1989). Experiments with time delay networks and dynamic time warping for speaker independent isolated digits recognition. In *First European Conference on Speech Communication and Technology*.
- [18] Bougrine, S., Cherroun, H., Ziadi, D., Lakhdari, A., and Chorana, A. (2016). Toward a rich arabic speech parallel corpus for algerian sub-dialects. In *The 2nd Workshop on Arabic Corpora and Processing Tools*, pages 2–10.

- [19] Bougrine, S., Chorana, A., Lakhdari, A., and Cherroun, H. (2017). Toward a web-based speech corpus for algerian dialectal arabic varieties. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 138–146.
- [20] Boulard, H. and Wellekens, C. J. (1990). Links between markov models and multilayer perceptrons. *IEEE Transactions on pattern analysis and machine intelligence*, 12(12):1167–1178.
- [21] Bridle, J. S. (1990). Alpha-nets: A recurrent ‘neural’network architecture with a hidden markov model interpretation. *Speech Communication*, 9(1):83–92.
- [22] Carneiro, T., Da Nóbrega, R. V. M., Nepomuceno, T., Bian, G.-B., De Albuquerque, V. H. C., and Reboucas Filho, P. P. (2018). Performance analysis of google colab as a tool for accelerating deep learning applications. *IEEE Access*, 6:61677–61685.
- [23] Characteristics of letters (2022). Sifaat al huruf. <https://en.al-dirassa.com/characteristics-of-letters-sifaat-al-huruf-tajweed-rules/>. Accessed 16 april 2022.
- [24] Choukri, K., Nikkhou, M., and Paulsson, N. (2004). Network of data centres (netdc): Bnsc-an arabic broadcast news speech corpus. In *LREC*.
- [25] Chowdhury, S. A., Hussein, A., Abdelali, A., and Ali, A. (2021). Towards one model to rule all: Multilingual strategy for dialectal code-switching arabic asr. *arXiv preprint arXiv:2105.14779*.
- [26] Clark, A., Fox, C., and Lappin, S. (2012). *The handbook of computational linguistics and natural language processing*, volume 118. John Wiley & Sons.
- [27] Collobert, R., Puhersch, C., and Synnaeve, G. (2016). Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv preprint arXiv:1609.03193*.
- [28] Consonants (n.d). Contrastive analysis of Arabic and English consonants General classification. <https://thesisleader.com/essays/consonants-in-arabic-and-english/>. Accessed 16 March 2022.
- [29] Djellab, M., Amrouche, A., Bouridane, A., and Mehallegue, N. (2017). Algerian modern colloquial arabic speech corpus (amcase): regional accents recognition within

- complex socio-linguistic environments. *Language Resources and Evaluation*, 51(3):613–641.
- [30] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.
- [31] Emami, A. and Mangu, L. (2007). Empirical study of neural network language models for arabic speech recognition. In *2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 147–152. IEEE.
- [32] Ferguson, C. A. (1996). *Sociolinguistic perspectives: Papers on language in society, 1959-1994*. Oxford University Press.
- [33] Fohr, D., Mella, O., and Illina, I. (2017). New paradigm in speech recognition: deep neural networks. In *IEEE international conference on information systems and economic intelligence*.
- [34] Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.
- [35] Gaida, C., Lange, P., Petrick, R., Proba, P., Malatawy, A., and Suendermann-Oeft, D. (2014). Comparing open-source speech recognition toolkits. In *11th International Workshop on Natural Language Processing and Cognitive Science*.
- [36] Goldenthal, W. D. (1994). *Statistical trajectory models for phonetic recognition*. PhD thesis, Massachusetts Institute of Technology.
- [37] Graff, D. (2003). Arabic gigaword ldc2003t12. *CD-ROM. Linguistic Data Consortium, Philadelphia*.
- [38] Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.
- [39] Greer, K., Lowerre, B., and Wilcox, L. (1982). Acoustic pattern matching and beam searching. In *ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 7, pages 1251–1254. IEEE.

-
- [40] Haffner, P. (1992). Connectionist word-level classification in speech recognition. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 621–624. IEEE.
- [41] Hammami, N. and Bedda, M. (2010). Improved tree model for arabic speech recognition. In *2010 3rd International Conference on Computer Science and Information Technology*, volume 5, pages 521–526. IEEE.
- [42] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al. (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.
- [43] Heafield, K. (2011). Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pages 187–197.
- [44] Hermansky, H. (1990). Perceptual linear predictive (plp) analysis of speech. *the Journal of the Acoustical Society of America*, 87(4):1738–1752.
- [45] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97.
- [46] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- [47] Holes, C. (2004). *Modern Arabic: Structures, functions, and varieties*. Georgetown University Press.
- [48] Iakushkin, O., Fedoseev, G., Shaleva, A., Degtyarev, A., and Sedova, O. (2018). Russian-language speech recognition system based on deepspeech. In *Proceedings of the VIII International Conference on Distributed Computing and Grid-technologies in Science and Education (GRID 2018)*.
- [49] Imai, S. (1983). Cepstral analysis synthesis on the mel frequency scale. In *ICASSP'83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 8, pages 93–96. IEEE.

-
- [50] Jurafsky, D. and Martin, J. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall series in artificial intelligence. Pearson Prentice Hall.
- [51] Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401.
- [52] Khatatneh, K. et al. (2014). A novel arabic speech recognition method using neural networks and gaussian filtering. *International Journal of Electrical, Electronics & Computer Systems*, 19(1).
- [53] Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *1995 international conference on acoustics, speech, and signal processing*, volume 1, pages 181–184. IEEE.
- [54] Kurzekar, P. K., Deshmukh, R. R., Waghmare, V. B., and Shrishrimal, P. P. (2014). Continuous speech recognition system: a review. *Asian Journal of Computer Science and Information Technology*, 4(6):62–66.
- [55] Lang, K. (1988). The development of the time-delay neural network architecture for speech recognition. *Technical Report CMU-CS-88-152*.
- [56] Lecorvé, G. and Motlicek, P. (2012). Conversion of recurrent neural network language models to weighted finite state transducers for automatic speech recognition. Technical report, Idiap.
- [57] Liberman, M. and Cieri, C. (1998). The creation, distribution and use of linguistic data: the case of the linguistic data consortium. In *proceedings of the 1st International Conference on Language Resources and Evaluation (LREC)*, pages 159–164.
- [58] Maamouri, M., Graff, D., and Cieri, C. (2006). Arabic broadcast news speech. *Linguistic Data Consortium, Philadelphia*.
- [59] Menacer, M. A., Mella, O., Fohr, D., Jouvét, D., Langlois, D., and Smaïli, K. (2017). Development of the arabic loria automatic speech recognition system (alasar) and its evaluation for algerian dialect. *Procedia Computer Science*, 117:81–88.

- [60] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010a). Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari.
- [61] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010b). Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari.
- [62] Miyamoto, R. (2020). Recurrent neural networks. <https://www.rkenmi.com/posts/rnn-recurrent-neural-networks>. Accessed: 23 April 2022.
- [63] Morgan, N. and Bourlard, H. (1995). Continuous speech recognition. *IEEE signal processing magazine*, 12(3):24–42.
- [64] Mubarak, H., Hussein, A., Chowdhury, S. A., and Ali, A. (2021). Qasr: Qcri al-jazeera speech resource—a large scale annotated arabic speech corpus. *arXiv preprint arXiv:2106.13000*.
- [65] Perkel, J. M. (2018). Why jupyter is data scientists’ computational notebook of choice. *Nature*, 563(7732):145–147.
- [66] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hanemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al. (2011). The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number CONF. IEEE Signal Processing Society.
- [67] Renals, S., Morgan, N., Bourlard, H., Cohen, M., and Franco, H. (1994). Connectionist probability estimators in hmm speech recognition. *IEEE transactions on speech and audio processing*, 2(1):161–174.
- [68] Rifki, O. (2021). Speech recognition on common voice arabic. <https://huggingface.co/othrif/wav2vec2-large-xlsr-arabic>. Accessed 07 June 2022.
- [69] Robinson, A. J. (1994). An application of recurrent nets to phone probability estimation. *IEEE transactions on Neural Networks*, 5(2):298–305.
- [70] Ryding, K. C. (2005). *A reference grammar of modern standard Arabic*. Cambridge university press.

- [71] Safaya, A. (2022). Speech recognition on common voice arabic. <https://huggingface.co/asafaya/hubert-large-arabic-ft>. Accessed 20 June 2022.
- [72] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.
- [73] Stuart, V. (2018). How do neural networks "remember"? https://persagen.com/2018/08/31/memory_in_rnn.html. Accessed: 23 April 2022.
- [74] Szmigiera, M. (2022). The most spoken languages worldwide in 2022. <https://www.statista.com/statistics/266808/the-most-spoken-languages-worldwide/#statisticContainer>. Accessed 02 June 2022.
- [75] Tomashenko, N., Vythelingum, K., Rousseau, A., and Esteve, Y. (2016). Lium asr systems for the 2016 multi-genre broadcast arabic challenge. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 285–291. IEEE.
- [76] Wahyuni, E. S. (2017). Arabic speech recognition using mfcc feature extraction and ann classification. In *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, pages 22–25. IEEE.
- [77] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339.
- [78] Watson, J. (2007). *The Phonology and Morphology of Arabic*. The Phonology of the World's Languages. OUP Oxford.
- [79] Xie, L. and Liu, Z.-Q. (2006). A comparative study of audio features for audio-to-visual conversion in mpeg-4 compliant facial animation. In *2006 International Conference on Machine Learning and Cybernetics*, pages 4359–4364. IEEE.
- [80] Yu, D. and Deng, L. (2015). Deep neural networks. In *Automatic Speech Recognition*, pages 57–77. Springer.
- [81] Zerari, N., Abdelhamid, S., Bouzgou, H., and Raymond, C. (2018). Bi-directional recurrent end-to-end neural network classifier for spoken arab digit recognition. In *2018*

2nd International Conference on Natural Language and Speech Processing (ICNLSP), pages 1–6. IEEE.

- [82] Zhu, L., Chen, L., Zhao, D., Zhou, J., and Zhang, W. (2017). Emotion recognition from chinese speech for smart affective services using a combination of svm and dbn. *Sensors*, 17(7):1694.

Appendix A

This appendix reports the whole flags file, which contains all the parameters for Deep Speech to set up.

```
1 f . DEFINE_string ( ' train_files ' , '' , ' comma separated list
   of files specifying the dataset used for training . Multiple
   files will get merged . If empty , training will not be run
   . ' )
2 f.DEFINE_string ( ' dev_files ' , '' , ' comma separated
3 list of files specifying the dataset used for
4 validation . Multiple files will get merged . If
5 empty , validation will not be run . ' )
6
7 f.DEFINE_string ( ' test_files ' , '' , ' comma separated
8 list of files specifying the dataset used for
9 testing . Multiple files will get merged . If empty
10 , the model will not be tested . )
11 f.DEFINE_string ( ' feature_cache ' , '' , ' path where
12 cached features extracted from — train_files will
13 be saved . If empty , caching will be done in
14 memory and no files will be written . )
15 f.DEFINE_integer ( ' feature_win_len ' , 32 , ' feature
16 extraction audio window length in milliseconds ' )
17 f.DEFINE_integer ( ' feature_win_step ' , 20 , ' feature
18 extraction window step length in milliseconds ' )
```



```
1 f.DEFINE_integer ( 'audio_sample_rate ' , 16000 , ' sample rate
   value expected by model ' )
2
3 # Global Constants
4 # =====
5 f.DEFINE_integer ( 'epochs ' , 75 , 'how many epochs (complete
   runs through the train files ) to train for ' )
6 f.DEFINE_float ( 'dropout_rate ' , 0.05 , ' dropout rate for
   feedforward layers ' )
7 f.DEFINE_float ( 'dropout_rate2' , -1.0 , ' dropout rate for
   layer 2 – defaults to dropout_rate ' )
8 f.DEFINE_float ( 'dropout_rate3' , -1.0 , ' dropout rate for layer
   3 – defaults to dropout_rate ' )
9
10 f.DEFINE_float ( 'dropout_rate4' , 0.0 , ' dropout rate for
   layer 4 – defaults to 0.0 ' )
11 f . DEFINE_float ( 'dropout_rate5 ' , 0.0 , ' dropout rate for
   layer 5 – defaults to 0.0 ' )
12 f . DEFINE_float ( 'dropout_rate6 ' , -1.0 , ' dropout rate for
   layer 6 – defaults to dropout_rate ' )
13 f . DEFINE_float ( ' relu_clip ' , 20.0 , ' ReLU clipping value
   for non – recurrent layers ' )
14 # Adam optimizer parameters
15 f . DEFINE_float ( 'beta1' , 0.9 , 'beta 1 parameter of Adam
   optimizer ' )
16 f . DEFINE_float ( 'beta2 ' , 0.999 , 'beta 2 parameter of
17 Adam optimizer ' )
18 f . DEFINE_float ( ' epsilon ' , 1e -8 , ' epsilon parameter of
   Adam optimizer ' )
19 f.DEFINE_float( 'learning_rate ' , 0.001 , 'learning rate of Adam
   optimizer ' )
```

```
1 # Batch sizes
2 f.DEFINE_integer ( 'train_batch_size ' , 1 , 'number of elements
   in a training batch ')
3 f.DEFINE_integer ( 'dev_batch_size ' , 1 , ' number of elements
   in a validation batch ')
4 f.DEFINE_integer ( 'test_batch_size ' , 1 , ' number of elements
   in a test batch ')
5 f.DEFINE_integer ( ' export_batch_size ' , 1 , ' number of
   elements per batch on the exported graph ')
6 # Performance
7 f.DEFINE_integer('inter_op_parallelism_threads' , 0 , 'number of
   inter-op parallelism threads – see tf.ConfigProto for more
   details. USE OF THIS FLAG IS UNSUPPORTED')
8 f.DEFINE_integer('intra_op_parallelism_threads' , 0 , 'number of
   intra-op parallelism threads – see tf.ConfigProto for more
   details. USE OF THIS FLAG IS UNSUPPORTED')
9 f.DEFINE_boolean('use_allow_growth' , False , 'use Allow Growth
   flag which will allocate only required amount of GPU memory
   and prevent full allocation of available GPU memory')
10 f.DEFINE_boolean('load_cudnn' , False , 'Specifying this flag
   allows one to convert a CuDNN RNN checkpoint to a checkpoint
   capable of running on a CPU graph.')
11 f.DEFINE_boolean('train_cudnn' , False , 'use CuDNN RNN backend
   for training on GPU. Note that checkpoints created with this
   flag can only be used with CuDNN RNN, i.e. fine tuning on a
   CPU device will not work') f.DEFINE_boolean('
   automatic_mixed_precision' , False , 'whether to allow
   automatic mixed precision training. USE OF THIS FLAG IS
   UNSUPPORTED. Checkpoints created with automatic mixed
   precision training will not be usable without mixed precision
   .')
```

```
1 # Sample limits
2 # Sample limits
3 f.DEFINE_integer('limit_train', 0, 'maximum number of elements
   to use from train set - 0 means no limit')
4 f.DEFINE_integer('limit_dev', 0, 'maximum number of elements to
   use from validation set - 0 means no limit')
5   f.DEFINE_integer('limit_test', 0, 'maximum number of
   elements to use from test set - 0 means no limit')
6 # Checkpointing
7 f.DEFINE_string('checkpoint_dir', '', 'directory from which
   checkpoints are loaded and to which they are saved - defaults
   to directory "deepspeech/checkpoints" within user\'s data
   home specified by the XDG Base Directory Specification')
8   f.DEFINE_string('load_checkpoint_dir', '', 'directory in
   which checkpoints are stored - defaults to directory "
   deepspeech/checkpoints" within user\'s data home
   specified by the XDG Base Directory Specification')
9   f.DEFINE_string('save_checkpoint_dir', '', 'directory to
   which checkpoints are saved - defaults to directory "
   deepspeech/checkpoints" within user\'s data home
   specified by the XDG Base Directory Specification')
10  f.DEFINE_integer('checkpoint_secs', 600, 'checkpoint saving
   interval in seconds')
11  f.DEFINE_integer('max_to_keep', 5, 'number of checkpoint
   files to keep - default value is 5')
12  f.DEFINE_string('load_train', 'auto', 'what checkpoint to
   load before starting the training process. "last" for
   loading most recent epoch checkpoint, "best" for loading
   best validation loss checkpoint, "init" for initializing
   a new checkpoint, "auto" for trying several options.')
```

```
1 f.DEFINE_string('load_evaluate', 'auto', 'what checkpoint
   to load for evaluation tasks (test epochs, model export,
   single file inference, etc). "last" for loading most
   recent epoch checkpoint, "best" for loading best
   validation loss checkpoint, "auto" for trying several
   options.')
```

```
2
3 # Transfer Learning
4
5 f.DEFINE_integer('drop_source_layers', 0, 'single integer
   for how many layers to drop from source model (to drop
   just output == 1, drop penultimate and output ==2, etc)')
```

```
6
7 # Exporting
8
9 f.DEFINE_string('export_dir', '', 'directory in which
   exported models are stored - if omitted, the model won\'t
   get exported')
```

```
10 f.DEFINE_boolean('remove_export', False, 'whether to remove
   old exported models')
```

```
11 f.DEFINE_boolean('export_tflite', False, 'export a graph
   ready for TF Lite engine')
```

```
12 f.DEFINE_integer('n_steps', 16, 'how many timesteps to
   process at once by the export graph, higher values mean
   more latency')
```

```
13 f.DEFINE_boolean('export_zip', False, 'export a TFLite model
   and package with LM and info.json')
```

```
14 f.DEFINE_string('export_file_name', 'output_graph', 'name
   for the exported model file name')
```

```
15 f.DEFINE_integer('export_beam_width', 500, 'default beam
   width to embed into exported graph')
```

```
1
2 # Model metadata
3 f.DEFINE_string('export_author_id', 'author', 'author of the
   exported model. GitHub user or organization name used to
   uniquely identify the author of this model')
4 f.DEFINE_string('export_model_name', 'model', 'name of the
   exported model. Must not contain forward slashes.')
5 f.DEFINE_string('export_model_version', '0.0.1', 'semantic
   version of the exported model. See https://semver.org/.
   This is fully controlled by you as author of the model
   and has no required connection with DeepSpeech versions')
6 def str_val_equals_help(name, val_desc):
7     f.DEFINE_string(name, '<{}>'.format(val_desc), val_desc)
8     str_val_equals_help('export_contact_info', 'public contact
   information of the author. Can be an email address, or a
   link to a contact form, issue tracker, or discussion
   forum. Must provide a way to reach the model authors')
9     str_val_equals_help('export_license', 'SPDX identifier of
   the license of the exported model. See https://spdx.org/
   licenses/. If the license does not have an SPDX
   identifier, use the license name.')
10    str_val_equals_help('export_language', 'language the model
   was trained on – IETF BCP 47 language tag including at
   least language, script and region subtags. E.g. "en-Latn-
   UK" or "de-Latn-DE" or "cmn-Hans-CN". Include as much
   info as you can without loss of precision. For example,
   if a model is trained on Scottish English, include the
   variant subtag: "en-Latn-GB-Scotland".')
11    str_val_equals_help('export_min_ds_version', 'minimum
   DeepSpeech version (inclusive) the exported model is
   compatible with')
```

```

1   str_val_equals_help('export_max_ds_version', 'maximum
    DeepSpeech version (inclusive) the exported model is
    compatible with')
2   str_val_equals_help('export_description', 'Freeform
    description of the model being exported. Markdown
    accepted. You can also leave this flag unchanged and edit
    the generated .md file directly. Useful things to
    describe are demographic and acoustic characteristics of
    the data used to train the model, any architectural
    changes, names of public datasets that were used when
    applicable, hyperparameters used for training, evaluation
    results on standard benchmark datasets, etc.')
3
4   # Reporting
5   f.DEFINE_integer('log_level', 1, 'log level for console
    logs - 0: DEBUG, 1: INFO, 2: WARN, 3: ERROR')
6 f.DEFINE_boolean('show_progressbar', True, 'Show progress for
    training, validation and testing processes. Log level should
    be > 0.')
7 f.DEFINE_boolean('log_placement', False, 'whether to log device
    placement of the operators to the console')
8 f.DEFINE_integer('report_count', 5, 'number of phrases for each
    of best WER, median WER and worst WER to print out during a
    WER report')
9 f.DEFINE_string('summary_dir', '', 'target directory for
    TensorBoard summaries - defaults to directory "deepspeech/
    summaries" within user\'s data home specified by the XDG Base
    Directory Specification')
10 f.DEFINE_string('test_output_file', '', 'path to a file to save
    all src/decoded/distance/loss tuples generated during a test
    epoch')

```

```
1
2 # Geometry
3 f.DEFINE_integer('n_hidden', 2048, 'layer width to use when
   initialising layers')
4 # Initialization
5 f.DEFINE_integer('random_seed', 4568, 'default random seed that
   is used to initialize variables')
6 # Early Stopping
7 f.DEFINE_boolean('early_stop', False, 'Enable early stopping
   mechanism over validation dataset. If validation is not being
   run, early stopping is disabled.')
8 f.DEFINE_integer('es_epochs', 25, 'Number of epochs with no
   improvement after which training will be stopped. Loss is not
   stored in the checkpoint so when checkpoint is revived it
   starts the loss calculation from start at that point')
9 f.DEFINE_float('es_min_delta', 0.05, 'Minimum change in loss to
   qualify as an improvement. This value will also be used in
   Reduce learning rate on plateau')
10 # Reduce learning rate on plateau
11 f.DEFINE_boolean('reduce_lr_on_plateau', False, 'Enable reducing
   the learning rate if a plateau is reached. This is the case
   if the validation loss did not improve for some epochs.')
12 f.DEFINE_integer('plateau_epochs', 10, 'Number of epochs to
   consider for RLROP. Has to be smaller than es_epochs from
   early stopping')
13 f.DEFINE_float('plateau_reduction', 0.1, 'Multiplicative factor
   to apply to the current learning rate if a plateau has
   occurred.')
14 f.DEFINE_boolean('force_initialize_learning_rate', False, 'Force
   re-initialization of learning rate which was previously
   reduced.')
```

```
1 # Decoder
2 f.DEFINE_boolean('utf8', False, 'enable UTF-8 mode. When this is
   used the model outputs UTF-8 sequences directly rather than
   using an alphabet mapping.')
3 f.DEFINE_string('alphabet_config_path', 'data/alphabet.txt', '
   path to the configuration file specifying the alphabet used
   by the network. See the comment in data/alphabet.txt for a
   description of the format.')
4   f.DEFINE_string('scorer_path', 'data/lm/kenlm.scorer', 'path
   to the external scorer file created with data/lm/
   generate_package.py')
5   f.DEFINE_alias('scorer', 'scorer_path')
6   f.DEFINE_integer('beam_width', 1024, 'beam width used in the
   CTC decoder when building candidate transcriptions')
7   f.DEFINE_float('lm_alpha', 0.931289039105002, 'the alpha
   hyperparameter of the CTC decoder. Language Model weight
   .')
8   f.DEFINE_float('lm_beta', 1.1834137581510284, 'the beta
   hyperparameter of the CTC decoder. Word insertion weight
   .')
9   f.DEFINE_float('cutoff_prob', 1.0, 'only consider characters
   until this probability mass is reached. 1.0 = disabled
   .')
10  f.DEFINE_integer('cutoff_top_n', 300, 'only process this
   number of characters sorted by probability mass for each
   time step. If bigger than alphabet size, disabled.')
11
12 # Inference mode
13 f.DEFINE_string('one_shot_infer', '', 'one-shot inference mode:
   specify a wav file and the script will load the checkpoint
   and perform inference on it.')
```