

# **Early Stages of Automatic Speech Recognition (ASR) in Non-English-Speaking Countries and Factors that Affect the Recognition Process**

By

**Dunya Yousufzai, Shoaib Naseri**

Submitted to the Kabul University and  
Department of Software Engineering

in partial fulfillment of the requirements for the degree of  
Bachelor of Computer Science

at the

Kabul University

March 2022



Author .....  
Kabul University and  
Department of Software Engineering  
March 1, 2022

Certified by .....  
Mohammad Rafi Bahez, Monograph Supervisor  
Associate Professor

Certified by .....  
Sayed Najmuddin Sadaat, Monograph Supervisor  
Associate Professor

Accepted by .....  
S. Hassan Adelyar  
Head of the Department (Distinguished professor), Software Engineering Department

Accepted by .....  
A. Wahid Samadzai  
Associate Professor, Software Engineering Department

# Early Stages of Automatic Speech Recognition (ASR) in Non-English-Speaking Countries and Factors that Affect the Recognition Process

by

Dunya Yousufzai, Shoaib Naseri

Submitted to the Kabul University and  
Department of Software Engineering  
on March 1, 2022, in partial fulfillment of the  
requirements for the degree of  
Bachelor of Computer Science

## Abstract

There has been a considerable stream in ASR over the past few decades, but it may seem strange why this field is still a subject for researchers to work on. There are many reasons, but somewhat because the discipline is created with the promise of human-level performance under pragmatic states and this is an inextricable problem. In addition, the increasing advancement of technology in various fields has caused a more compelling need for this field. Especially the establishment of such a system in the security sector in insecure third world countries such as Afghanistan is an urgent need. This paper began with the reflection of all the necessary knowledge about speech recognition and then suggested an unprecedented method for building an automated speech recognition (ASR) system in the Dari language using the two most powerful open-source engines CMUSphinx, from Carnegie Mellon University and DeepSpeech v0.9.3 /. These systems are much more impressive than early speech recognition systems. Using my own collected dataset, a speech-to-text model has been trained for the Dari language. Firstly, the dataset is filtered according to the task, then demonstrated the possible compatibility from the hidden Markov (HMM) models, the phoneme concept to RNN training. The system surpassed previously predicted results, as CMUSphinx stated, “for a typical 10-hour operation, the WER should be around 10% Finally, 3.3% WER was achieved with 10.3-hours of audio recording using CMUSphinx. 1% WER with DeepSpeech.

Monograph Supervisor: Mohammad Rafi Bahez  
Title: Associate Professor

Monograph Supervisor: Sayed Najmuddin Sadaat  
Title: Associate Professor

## **Acknowledgments**

We would like to express our warmest gratitude to our supervisors Mohammad Rafi Bahez and Sayed Najmuddin Sadaat for their valuable assistance in leading to the writing of this monograph. Also, we would like to thank our friends Zohra, Sanam, and Sara for their nice and useful comments, their supportive characters encouraged us to continue our studies and research with great passion through the development of this project. Also, we would like to sincerely thank all the members of the university who patiently helped us while working on this research.

# Contents

Figures.....	5
Tables.....	6
Acronym .....	7
1. Introduction.....	8
2. Background/History of Speech Recognition .....	10
2.1 Problem Statement .....	11
2.2 Objectives.....	12
2.3 Impact of the Problem.....	12
2.4 Project Scope.....	12
2.5 Existing System.....	13
2.6 Literature Review .....	13
2.7 Speech Recognition with CNN .....	16
3. Speech Recognition with CMUSphinx.....	21
4. Speech Recognition with DeepSpeech .....	26
4.1 Recurrent Neural Network .....	26
4.2 DeepSpeech.....	26
4.3 Dataset Preparation and Training the Data .....	27
5. Generic Introduction and Summary of the Development of Speech Recognition and Machine Translation App .....	29
5.7.1 Performance Requirements.....	34
5.7.2 Safety Requirements and Security Requirements.....	34
5.8.1 Maturity level 1: Initial .....	35
5.8.2 Maturity level 2: Managed.....	35
5.8.3 Maturity level 3: Defined.....	35
5.8.4 Maturity level 4: Quantitatively managed .....	35
5.8.5 Maturity level 5: Optimizing .....	36
6. Experimental Results .....	48
7. Conclusion.....	50

# Figures

Figure 1. Rarefaction and compression of air molecules.

Figure 2. Folders structure for each word.

Figure 3. Several 1-second audios record for each word.

Figure 4. Projection of Audio signal in time series domain.

Figure 5. The number of recordings for each voice command.

Figure 6. Demonstration of the performance of the model over a period of time.

Figure 7. List of the names of the recordings (utterance ids) and transcription for each audio file.

Figure 8. List of phones with it is a corresponding phonetic dictionary.

Figure 9. List of all CSV files (dev.csv, test.csv, train.csv).

Figure 10. The complete process of training.

Figure 11. The Waterfall Model (The credit of this image goes to SALEH OT).

Figure 12. Gantt chart.

Figure 13. Use Case diagram for SR and language translation.

Figure 14. Deployment diagram for SR and language translation.

Figure 15. DFD diagram of SR system.

Figure 16. Class diagram of SR system.

Figure 17. Sequence diagram for SR system.

Figure 18. The main page of SR and MT system.

Figure 19. The Speech Recognition page of SR and MT system.

Figure 20. The Complex Speech Training page of SR and MT system.

Figure 21. Tips for SR system.

Figure 22. Test page for SR system.

Figure 23. The Simple Speech Training page for SR system.

Figure 24. The language translator page for SR system.

## Tables

Table 1. Presents the results of fine-tuning process.

# Acronyms

- **AI** - Artificial Intelligence.
- **ANN** - Artificial Neural Network.
- **ASR** - Automatic Speech Recognition.
- **CNN** - Convolutional Neural Network.
- **DARPA** - Defense Advanced Research Projects Agency.
- **DFD** - Data Flow Diagram.
- **DP** - Deep Learning.
- **HMM** - Hidden Markov Model.
- **LSTM** - Long Short-Term Memory.
- **MT** – Machine Translation
- **RNN** - Recurrent Neural Network.
- **SR** - Speech Recognition.
- **SUR** - Speech Understanding Research.
- **WER** - word Error Rate.

# 1. INTRODUCTION

---

From prehistoric times until now, the exchange of information and considerable effort in interlocution has been and will be a considerable purpose to improve human understanding, so that hearing twiddles an important role in this process. Hearing hinges on a series of complex and intricate stages which convert sound waves in the air into electrical signals. The brain receives these signals with the help of the auditory nerve. Sound waves arrive in the outer ear and pass through a narrow canal called the ear canal, which conducts to the eardrum. As the sound waves enter, the eardrum starts vibrating. There are three tiny bones in the middle ear called the malleus, incus, and stapes that receive the vibration. The role of these bones in the middle ear is to amplify the sound vibrations and send them to the snail-shaped structure called the cochlea. Cochlea filled with fluid. When the fluid inside the cochlea twist and turns, a moving wave will be generated along the basal membrane. Hair cells - Sensory cells located above the basal membrane. The ions reach the top of the cell and secrete chemicals at the bottom called neurotransmitters. These chemicals attach to the auditory nerve and produce an electrical signal that eventually travels to the brain <sup>1</sup>. As you can see, comprehension and hearing are different at the human level, but this is where unprecedented achievements in SR (speech recognition) comes in. Automatic Speech Recognition (ASR) is a technology that allows a computer to identify the words that a person speaks into a microphone or telephone (Satori et al., 2007). The wide availability of devices equipped with microphones and powerful computing capabilities constitutes a great potential for using ASR systems (El Amrani et al., 2016).

In recent years, there have been significant advances in machine learning algorithms, which have provided the basis for the development of various handy applications. Deep learning uses cross-sectional studies that help applications such as speech recognition. It should be mentioned that different types of neural networks play a key role in the field of ASR by being at the core of machine learning. There has been comparatively enough recognition



research on the Persian language contrasted to the Afghanistan Dari language, although Dari is the main language but unfortunately no one has worked in this field so far. So, in this monograph, a thorough discussion about the steps of manual engineering processing, extremely modular and pliable ways which have support for a variety of HMM-based acoustic models, about numerous language models and probe tactics with respect to CMUSphinx and an end-to-end speech system has been made. By following these steps, you can build a model for your language that is still unpopular and unknown. Meanwhile, alleges disparate challenges: (i) a shrewd track to collect a large number of the dataset and filter it to productively put upon all of them must be found, (ii) it is necessary to be familiar and have enough information in this field, (iii) it is compulsory to have sufficient ability to switch between different types of algorithms and frameworks according to your needs (iv) and if you want to work with phonemes, you must create a phonetic dictionary for your own language even no one has worked on it yet. In the continuation of this article, the concept of a speech recognition system will be discussed. It begins with the Background/History of SR and by describing the basic complexity of the neural network and the process of training your own dataset with hard encryption, starting from scratch in Section 2, followed by a discussion on CMU Sphinx and how to prepare your own dataset (Section 3). And move on to the recurrent neural network (RNN), talking about DeepSpeech and preparing a dataset for this framework (Section 4). Then, It covers a simple speech recognition app (Section 5) And last it concludes with experimental results (Section 6), followed by conclusions (section 7).

## **2. Background/History of Speech Recognition**

### **The 1950S and 60S**

The first time the speech recognition industry started converting speech to text was only able to recognize numbers. That is, in 1952, Bell Labs developed a system called the Audrey system that could detect only digits. IBM established “Shoebox” ten years later. This system was better than the earlier system because it understood 16 words in English. Day by day, the industry of making advanced hardware has been developed. So, by the end of the ‘60s, the newer technologies were able to detect words that contain four vowels and nine consonants.

### **The 1970S**

There have been several advances in speech recognition since the 1970s. DARPA implemented the Speech Perception Research Program (SUR), which was one of the most positive and fundamental steps in the history of speech recognition. Carnegie Mellon's Harpy speech system was derived from this program and was able to recognize more than 1000 words. Also, another breakthrough was made by Bell Labs in the 1970s, which was enabled to interpret multiple choices.

### **The 1980S**

In the ‘80s the speech recognition systems improved more and were able to recognize a thousand words. Exactly it was the time that the most used and effective statistical method known as the “Hidden Markov Model (HMM)” was introduced. HMM is a model that works not only with words but with sentences too.

### **The 1990S**

In the 90s, BellSouth, an American telecommunication holding company introduced Voice portal (VAL) which was a dial-in interactive voice recognition system. The voice portal was the first system that could respond to questions.

## **The 2000S**

After years of study, research, and hard work, scientists have been able to achieve 80% accuracy in speech recognition. In addition, Google introduced Google Voice Search, which was a breakthrough. The system launched by Google can be used by people all over the world. The model that has been created by Google can provide an opportunity to predict the speaker's sentences. This model is trained by millions of data.

## **The Future**

After Google's Voice Search, Apple launched Siri. In the same way, amazon's Alexa was another achievement. The research and study of speech recognition will go on because today many advanced systems do not cover all languages over the world. The next goal is to achieve a good word error rate and human-level understanding.

## **2.1 Problem Statement**

The biggest problem and challenge that we can visualize in the development of an advanced system for speech recognition is not having a good resource, and specific algorithms to use. It means, today you can see different algorithms are used by different accredited companies. For example, Google uses HMM, DeepSpeech uses RNN, and...

Another factor is not having a system that covers all languages in the world. Furthermore, the languages are different and have different structures. The famous companies only offer services for people, which means they are not open-source. Recognition and understanding of spontaneous unrehearsed speech remain an elusive goal.

## **Research Questions**

1. What are the fundamental components, steps, algorithms, and semantic knowledge for speech-to-text conversion?
2. How can we carry out project/research on Automatic Speech Recognition (ASR) for non-English speaking countries using CMUsphinx?

## **2.2 Objectives**

- ✓ To convert speech to text;
- ✓ To introduce efficacious ways for ASR;
- ✓ To have a comparative review between different frameworks;
- ✓ To understand the difference between different Algorithms;
- ✓ To know how to collect dataset;
- ✓ To know the importance of dataset;
- ✓ To help non-speaking countries in the field of ASR;
- ✓ To conduct new ideas in making an ideal society using ASR;
- ✓ To create ARPAbet for your own language;

## **2.3 Impact of the Problem**

- ✓ Improve security (Robots in wars instead of humans)
- ✓ Home Appliance
- ✓ People with disability
- ✓ Economic
- ✓ Time
- ✓ Ideal society

## **2.4 Project Scope**

The scope of this project is to deliver different models to detect 106 words in Dari Language; it will compare different frameworks. Furthermore, It shows which algorithms give a desirable output with their corresponding word error rates. This research led to the creation of a system that only requires specific parameters, and the users do not need technical and full information about the background procedures, they will only specify the parameters and start training.

## **2.5 Existing System**

There is not any existing system in the field of ASR in Afghanistan. A very small number of people researched Speech Recognition with a few limited words. We have famous systems in this field abroad, the well-known companies offer different services but, they are not covering Afghanistan's all native languages, and they are not open sources.

## **2.6 Literature Review**

“There are generally two categories of users who can benefit from adoption of speech as a control modality in parallel with others, such as the mouse, keyboard, touch-screen, and joystick. For novice users, functions that are conceptually simple should be directly accessible. For example, raising the voice output volume under software control on the desktop speakers, a conceptually simple operation, in some GUI systems of today requires opening one or more windows or menus, and manipulating sliders, check-boxes or other graphical elements. This requires some knowledge of the system’s interface conventions and structures. For the novice user, to be able to say raise the volume would be more direct and natural. For expert users, the GUI paradigm is sometimes perceived as an obstacle or nuisance and shortcuts are sought. Frequently these shortcuts allow the power user’s hands to remain on the keyboard or mouse while mixing content creation with system commands. For example, an operator of a graphic design system for CAD/CAM might wish to specify a text formatting command while keeping the pointer device in position over a selected screen element” (Acero, et al., 2001).

You can extract the basic use and importance of speech from the above example. It means, we need a speech recognition system in different areas, such as security, economics, home appliances, and ...

Afghanistan is one of the most vulnerable countries in many areas, especially in the field of security; many youngsters lose their lives every day, they become impaired while checking roadway mines. One of the alternatives to solve this problem is to use a trained robot (using speech systems). These are just a few examples of the usage of speech. Advancement in technology opened the gates to the improvements of different speech

recognition systems by authoritative companies in the world. Today many scientists researched speech recognition systems in different languages using several algorithms.

“Speech recognition, also known as automatic speech recognition (ASR), computer speech recognition, or speech-to-text, is a capability which enables a program to process human speech into a written format. While it’s commonly confused with voice recognition, speech recognition focuses on the translation of speech from a verbal format to a text one whereas voice recognition just seeks to identify an individual user’s voice. IBM has had a prominent role within speech recognition since its inception, releasing of “Shoebox” in 1962. This machine had the ability to recognize 16 different words, advancing the initial work from Bell Labs from the 1950s. However, IBM didn’t stop there, but continued to innovate over the years, launching VoiceType Simply Speaking application in 1996. This speech recognition software had a 42,000-word vocabulary, supported English and Spanish, and included a spelling dictionary of 100,000 words. While speech technology had a limited vocabulary in the early days, it is utilized in a wide number of industries today, such as automotive, technology, and healthcare. Its adoption has only continued to accelerate in recent years due to advancements in deep learning and big data. Research (link resides outside IBM) shows that this market is expected to be worth USD 24.9 billion by 2025” (IBM Cloud Education, 2020).

Speech recognition is a complex topic that for many years scientists have tried to developed different algorithms. Firstly, to understand the nature of the topic we need to know the hearing process, the language structure, grammar, and ...

“Automatic Speech Recognition (ASR) has historically been a driving force behind many machine learning (ML) techniques, including the ubiquitously used hidden Markov model, discriminative learning, structured sequence learning, Bayesian learning, and adaptive learning. Moreover, ML can and occasionally does use ASR as a large-scale, realistic application to rigorously test the effectiveness of a given technique, and to inspire new problems arising from the inherently sequential and dynamic nature of speech. On the other hand, even though ASR is available commercially for some applications, it is largely an unsolved problem—for almost all applications, the performance of ASR is not on par with

human performance. New insight from modern ML methodology shows great promise to advance the state-of-the-art in ASR technology” (Deng, et al., 2013).

Speech is a continuous audio stream where rather stable states mix with dynamically changed states. In this sequence of states, one can define more or less similar classes of sounds, or phones. Words are understood to be built of phones, but this is certainly not true. The acoustic properties of a waveform corresponding to a phone can vary greatly depending on many factors - phone context, speaker, style of speech and so on. The so-called coarticulation makes phones sound very different from their “canonical” representation. Next, since transitions between words are more informative than stable regions, developers often talk about diphones - parts of phones between two consecutive phones. Sometimes developers talk about subphonetic units - different substates of a phone. Often three or more regions of a different nature can be found. The number three can easily be explained: The first part of the phone depends on its preceding phone; the middle part is stable and the next part depends on the subsequent phone. That’s why there are often three states in a phone selected for speech recognition. Sometimes phones are considered in context. Such phones in context are called triphones or even quinphones. For example, “u” with left phone “b” and right phone “d” in the word “bad” sounds a bit different than the same phone “u” with left phone “b” and right phone “n” in word “ban”. Please note that unlike diphones, they are matched with the same range in waveform as just phones. They just differ by name because they describe slightly different sounds. For computational purpose it is helpful to detect parts of triphones instead of triphones as a whole, for example if you want to create a detector for the beginning of a triphone and share it across many triphones. The whole variety of sound detectors can be represented by a small amount of distinct short sound detectors. Usually we use 4000 distinct short sound detectors to compose detectors for triphones. We call those detectors senones. A senone’s dependence on context can be more complex than just the left and right context. It can be a rather complex function defined by a decision tree, or in some other ways. Next, phones build subword units, like syllables. Sometimes, syllables are defined as “reduction-stable entities”. For instance, when speech becomes fast, phones often change, but syllables remain the same. Also, syllables are related to an intonational contour. There are other

ways to build subwords - morphologically-based (in morphology-rich languages) or phonetically-based. Subwords are often used in open vocabulary speech recognition. Subwords form words. Words are important in speech recognition because they restrict combinations of phones significantly. If there are 40 phones and an average word has 7 phones, there must be  $40^7$  words. Luckily, even people with a rich vocabulary rarely use more than 20k words in practice, which makes recognition way more feasible. Words and other non-linguistic sounds, which we call fillers (breath, um, uh, cough), form utterances. They are separate chunks of audio between pauses. They don't necessarily match sentences, which are more semantic concepts. On the top of this, there are dialog acts like turns, but they go beyond the purpose of this document" (cmuweb)".

Speech is a set of waveforms that can be split into utterances, then by detecting those utterances we recognize the speech. Here, we will consider all possible numbers of the words that can join and form meaningful sentences.

For the feature engineering part, we usually divide speech into frames which is ten milliseconds in length, and from each frame, we extract a set of numbers (39) that form the speech. In addition, we need to create a model which is a series of complex mathematical calculations on numbers or features.

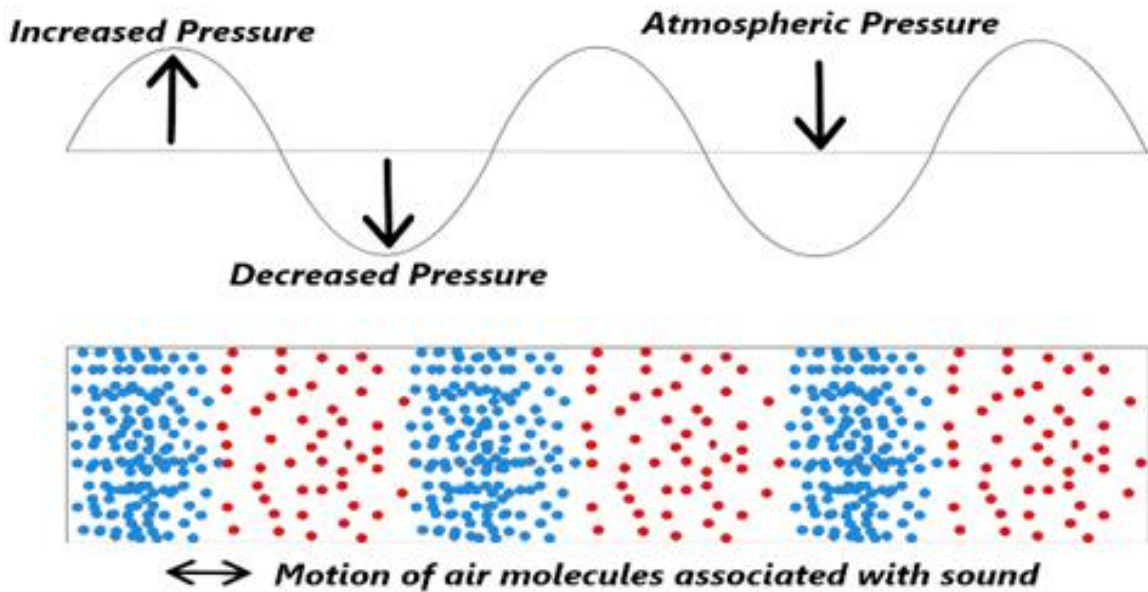
Today we have several systems in this area, for example CMUsphinx, DeepSpeech, Dragon and ...

These systems work with different algorithms, like CMUsphinx works with Hidden Markov Model, DeepSpeech works with RNN, and in the same way, we can create a model using CNN, but the important part here is to work on Dari language specifically because until today no one has worked here, no one has worked on Dari language structure, ARPAbet. Although some researchers worked on Arabic language and Persian language which is a little similar to Dari language but these systems cannot cover Dari speech system.

## **2.7 Speech Recognition with CNN**



Let's quickly find out what sound is? Sound is a longitudinal pressure wave composed of the impaction and dilution of air molecules, in a path equal to the application of energy. The areas where air molecules are forced to use energy to a more precise configuration than usual called impaction, and dilution are areas where air molecules are less packaged. The speed of a sound pressure wave in air is almost  $331.5 + 0.6 T_c$  m/s, where  $T_c$  is the temperature of Celsius (Acero, et al., 2001). Here we have to convert the analog signal to a digital signal, which is a segregated exhibition of a signal over a period of time. So, the kernel or core of speech-to-text conversion is the elicitation of various characteristics of the audio signal. Any physical element which is constant or variable in time is called a signal (Dib, 2019).



**Figure 1.** Rarefaction and compression of air molecules.

Convolutional neural networks (CNN) are the inspiration in the deep learning assembly. CNN utilizes a particular network frame, which is composed of intermittently called convolution and pooling layers. In CNN the input data required to be formed as multiple feature maps, which means it needs to organize speech feature vectors into feature maps (Abdel-Hamid et al., 2014). The CNN exploits domain knowledge about feature invariances within its structure (Abdel-Hamid et al., 2013). Recently CNN has met a significant research progress (Sainath et al) A convolutional neural network consists of an

input layer, hidden layers, and an output layer. The dot product is done by the hidden layer as the hidden layer is responsible to perform convolutions.

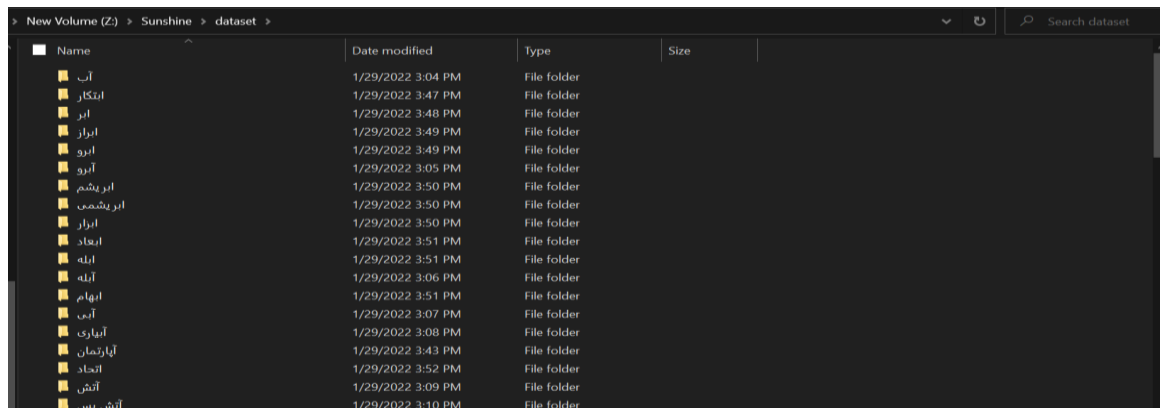
$$a \cdot b = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + a_3 b_3 + \dots + a_n b_n$$

and its activation function is commonly Rectifier (ReLU),  $F(x) = \max(0, x)$ , A clear approach to the rectifier is the analytic function  $f(x) = \ln(1 + e^x)$ .

The convolution operation is a linear operation, demonstrated by an asterisk, that consolidates two signals.

$$f[x, y] * g[x, y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot g[x - n_1, y - n_2]$$

Here in CNN the input with a shape of  $N_i \times I_h \times I_w + I_c$  passing through a convolution layer. Convolutional networks may include local and/or global pooling which Pooling layers deduct the dimensions of data. Figure 2 shows folders for each words and figure 3 shows several audios record for each word. Figure 4 demonstrates the projection of Audio signal in the time series domain, as 297,482 one-second recorded words (82.3 hours) have been put into specific folders so it is necessary to know the number of recordings for each voice command.



Name	Date modified	Type	Size
آب	1/29/2022 3:04 PM	File folder	
ابتکار	1/29/2022 3:47 PM	File folder	
ابر	1/29/2022 3:48 PM	File folder	
انبار	1/29/2022 3:49 PM	File folder	
آبرو	1/29/2022 3:49 PM	File folder	
آبرو	1/29/2022 3:05 PM	File folder	
آبرو	1/29/2022 3:50 PM	File folder	
آبرو	1/29/2022 3:50 PM	File folder	
آبرو	1/29/2022 3:50 PM	File folder	
آبرو	1/29/2022 3:51 PM	File folder	
آبرو	1/29/2022 3:51 PM	File folder	
آبرو	1/29/2022 3:06 PM	File folder	
آبرو	1/29/2022 3:51 PM	File folder	
آبرو	1/29/2022 3:07 PM	File folder	
آبرو	1/29/2022 3:08 PM	File folder	
آبرو	1/29/2022 3:43 PM	File folder	
آبرو	1/29/2022 3:52 PM	File folder	
آبرو	1/29/2022 3:09 PM	File folder	
آبرو	1/29/2022 3:10 PM	File folder	

**Figure 2.** Folders structure for each word.



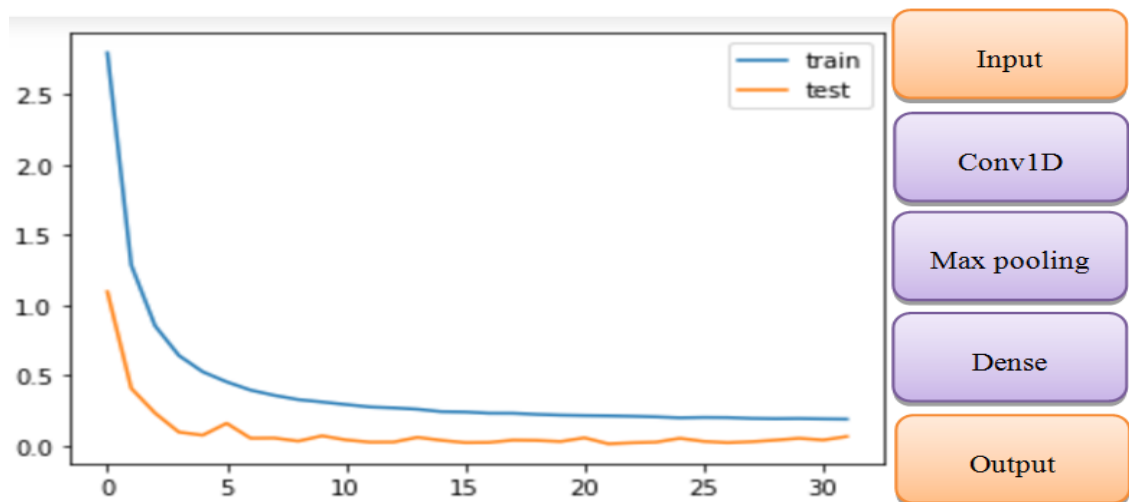
Two steps more for resampling and removing shorter commands of less than 1-second have been followed. All of the labels and all of the waves have been extracted in order to get output labels, then converted the output labels to integer encoded, chasing this conversion of the integer encoded Labels to a one-hot vector took place because it is a multi-classification problem. Afterward, reshaped the 2D array to 3D since the input to the conv1d has to be a 3D array.

[[[ 5.6007931e-16]

[7.9060451e-16]

[1.4276164e-15]]]

The model has been trained on 80% of the data and validated on the remaining 20% then the speech-to-text model has been built-in using conv1d. Conv1d is a convolutional neural network that carries out the convolution along one dimension. For model building, Keras functional API is preferred and used Adam for optimizer and categorical cross-entropy for the loss, before long early stopping and model checkpoints for the callbacks to stop training the neural network at the right time has been used so that it gives the possibility to save the best model after every epoch, and finally, the data on batch size 32 has been trained.



**Figure 6.** *Demonstration of the performance of the model over a period of time.*

After 63 epochs and have enough dataset with loss: 1.0385e-05 - accuracy: 1.0000 - val\_loss: 4.9401e-06 - val\_accuracy: 1.0000, performance of the model is not good.

The training has started with two words; then we added more words to the model until we reached 106 words; the higher the words, the lower the detection. Although we had a sufficient number of datasets still, we were not able to create an adequate model. We have gotten acceptable accuracy after training each data but detection is not good. By this experiment, we can ensure that accuracy cannot stand as the only factor for the recognition process.

**Drawbacks of this method:**

- (i) Takes lots of time (time-consuming);
- (ii) Contravention in some data (altered speaking manner) can ruin all of the datasets;
- (iii) Cannot deal with environmental noise and distorted acoustics and speech correlated noise;
- (iv) Low performance;
- (v) Having weakness in changing the sample rate of a large number of the dataset;

This method is suitable to create a model for 10 to 50 or maybe more with enough dataset, but for a larger vocabulary, you need to follow another way. So that the drawbacks were taken into consideration and fueled further research which led us to a good ASR model.

### **3. Speech Recognition with CMUSphinx**

The dominant technological approaches for speech recognition systems are based on pattern matching of statistical representations of the acoustic speech signal, such as HMM whole word and subword (e.g., phoneme) models (Bourlard & Morgan, 1994). Statistical Language Modeling (LM) is the evolution of probabilistic models that are qualified to predict the next word in the sequence according to the word before it, CMUSphinx uses an

acoustic model, a dictionary, and an n-gram language model, which determines the phonetic units in the word available in the dictionary (Hinton et al).

### **Speech recognition with CMUSphinx:**

Given the acoustic data:  $X=x_1, x_2, x_3 \dots x_k$ . Given the Word Sequence:  $W_r=wr_1, wr_2, wr_3 \dots wr_k$ . The target is to increase  $P(W_r/X)$ . In agreement with Bayes' Theorem:  $P(W_r/X) = (P(X/W_r) P(W_r))/P(X)$

Where:

$P(X|W_r)=\text{Acoustic model(HMMs)}$

$P(W_r)=\text{Language model.}$

$P(X)=\text{Constant for a complete sentence.}$

Sphinx2 uses dialog system language learning system and it is oriented on speech recognition in real time which makes it ideally suited for developing various mobile applications (Matarneh et al., 2017).

Sphinx3 represents semi continuous speech recognition acoustic model, adopted a common continuous model constructed on HMM (Matarneh et al., 2017). Hidden Markov modeling of speech assumes that speech is a piecewise stationary process, that is, an utterance is modeled as a succession of discrete stationary states, with instantaneous transitions between these states (Renals et al., 1994).

CMUSphinx 4 is the latest addition which is differently designed from the earlier Sphinx systems regarding flexibility, modularity, and algorithmic aspects. You can modify the language model from a statistical N-gram language model to a context-free grammar (CFG) or a stochastic CFG by shifting only one portion of the system, meaning the linguist. Similarly, it is feasible to run the system using continuous, semi-continuous, or discrete phase output distributions by adequate rectification of the acoustic scorer. The overall architecture of sphinx 4 consists of the front-end, decoder, and knowledge base, which decoder itself consists of the search manager, the linguist, and the acoustic scorer. "Sphinx-4 puts out a beam pruner that limits the scores to a configurable least possible amount close

to the best score, while also maintaining the total number of active tokens to a configurable maximum” (Lamere et al).

Now let’s move on to the work summary with CMUSphinx starting from the dataset, the database contains information that is required to extract statistics from the speech in form of the acoustic model. More than 10 hours of recorded words have been taken. Then filtered the dataset and prepared it for training. One of the factors that can affect the recognition process is the mismatch of the sample rate, all of your datasets must be 16 kHz (or 8 kHz, depending on the training data), you can use sox for this propose, “sox (Sound eXchange) is a cross-platform audio editing software. It has a command-line interface, and is written in standard C. It is free software, licensed under the GNU”<sup>19</sup>. Here it is momentous to prepare two dictionaries: “one in which legitimate words in the language are mapped to sequences of sound units (or sub-word units), and another one in which non-speech sounds are mapped to corresponding speech-like sound units”<sup>20</sup>. The file structure for the database is the following: you can name your folders and files whatever you want but be careful about extensions, I chose Sunshine.

**Sunshine:**

Sunshine.dic (Phonetic dictionary)

Sunshine.phone (Phone set file)

Sunshine.lm.DMP (Language model)

Sunshine.filler (List of fillers)

Sunshine\_train.fileids (List of files for training)

Sunshine\_train.transcription (Transcription for training)

Sunshine\_test.fileids (List of files for testing)

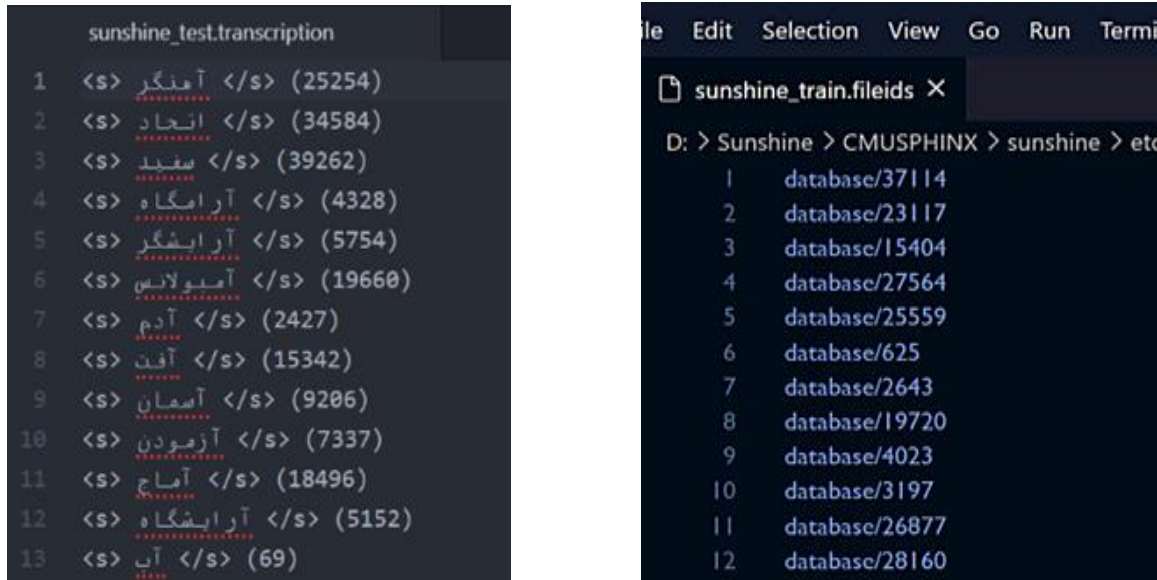
Sunshine\_test.transcription (Transcription for testing)

**\*.fileids:**

The Sunshine\_train.fileids and Sunshine\_test.fileids files are text files that list the names of the recordings (utterance ids) one by one, do not include audio file extensions in their content<sup>20</sup>.

**\*.transcription:**

The Sunshine\_train.transcription and Sunshine\_test.transcription files are text files listing the transcription for each audio file:



**Figure 7.** List of the names of the recordings (utterance ids) and transcription for each audio file.

One of the goals of this research paper is to guide you on how to create a phonetic dictionary for your own language. A phonetic dictionary provides the system with a mapping of vocabulary words to sequences of phonemes. To create a phonetic dictionary, you will come across ARPABET, ARPABET, or ARPAbet is developed by Advanced Research Projects Agency (ARPA) in the 1970s which is a collection of phonetic transcription codes. The purpose of developing ARPANET was to understand speech in the Research project. It demonstrated phonemes and allophones of General American English with preferable sequences of ASCII characters. So, creating a phonetic dictionary requires knowledge about your own language phonology. In this paper the Dari language phonology is used and listed all of the words and their pronunciation like "quick" /zu:d/ and "strength" /zo:r/, then I wrote down the words in English like دنیا - Dunya, after that I used Lexicon Tool which generates a pronunciation dictionary from a list of words in a form suitable for use with a speech recognizer, such as CMUSphinx. The Lexicon Tool



uses the CMUdict dictionary along with some simple normalization and inflection rules to identify a word and uses letter-to-sound rules when all else fails. So here is the output of my work.

آب	AH B	sunshine.phone - Notepad
ابتکار	EH B T AH K AH R	File Edit Format View Help
ابر	EY B ER	AA
ابراز	EY B R AH Z	AE
ابرو	AE B R OW	AH
أبرو	AA B R OW	AO
ابريشم	AE B R AH SH AH M	AY
ابريشمی	AE B R EH SH AH M IY	B
ابرار	AE B Z AH R	CH
ابعد	AE B AE D	D
إبله	AE B L AH	EH
آبله	AA B L AH	ER
ابهام	EH B HH AH M	EY
آبی	AE B IY	F
آبیاری	AH B AY AH R IY	G
آپارتمان	AH P AA R T AH M AH N	HH
اتحاد	EH T AH HH AH D	IH
آتش	AE T AH SH	IY
اتفاق	EH T AH F AH K	JH
اتمام	EH T M AH M	K
اتمسفير	AE T M AH S F IH R	L
آچار	AH CH AA R	M
آخر	AE K AH R	N
آداب	AE D AH B	NG
أدم	AE D AH M	OW

**Figure 8.** List of phones with it is a corresponding phonetic dictionary.

From these words, a phoneme file has been created and then created a language model. The language model tells the decoder which sequences of words are possible to recognize. There are lots of tools like SRILM which is the most advanced toolkit up to date, CMUCLMTK, IRSML, MITLM, web service such as Sphinx Knowledge Base Tool, and... but the problem is that some of them only support ASCII characters and English language and they do not support other languages, I personally prefer KenLM which can estimate, filters, and queries language models. Estimation is rapid and can be scaled on account of streaming algorithms. You can convert your model into binary format. After Setting up the training scripts and the format of database audio according to my needs, I started training in the Ubuntu environment and used Some additional scripts that will be launched if you choose to run them. These additional training steps can be costly in computation but improve the recognition rate. It's critical to test the quality of the trained database in order to select the best parameters, understand how your application performs,

and optimize the performance. To do that, you need decoding. The Decoder takes a model, tests part of the database and reference transcriptions and, estimates the quality (WER) of the model. Within the testing phase, use the language model with the description of the possible order of words in the language. Here the result of running the decoder:

```
SENTENCE ERROR: 3.2% (264/8313)  WORD ERROR RATE: 3.3% (272/8313)
```

## 4. Speech Recognition with DeepSpeech

### 4.1 Recurrent Neural Network

Recurrent neural networks have been a significant hub of research and advancement since the 1990s. They are designed to indoctrinate ordinal or time-varying samples. A recurrent net is a neural network with feedback (closed-loop) connections (Fausett, 1994). Examples include BAM, Hopfield, Boltzmann machine, and recurrent backpropagation nets (Hecht-Nielsen, 1990). The architectures span from fully interconnected to partially connected nets, which include multilayer feedforward networks with different input and output layers. Learning is an essential feature of neural networks and a leading feature that creates a handy application using a neural approach, in addition, a Real-time determination for optimization problems is frequently necessary for scientific and engineering problems, including signal processing (W. Senior et al).

### 4.2 DeepSpeech

DeepSpeech is an open-source voice recognition engine that is used to convert speech into text. It was using a recurrent neural network (RNN) to convert speech. To convert speech to text, a series of features must be extracted, so  $X_{t,k}^{(n)}$  represents the power of the  $k^{\text{th}}$  frequency bin in the audio frame at time  $t$ . The main purpose of Recurrent Neural Network is to transform an input order “x” into a string of character probabilities for the transcription “y” (Amodei et al). The RNN model in DeepSpeech is consists of 5 layers of hidden units

the first 3 layers are calculated by:  $ht^{(l)} = g(W^{(l)}h^{(l-1)} + b^{(l)})$ , The fourth layer is a bi-directional recurrent layer which intends to apply a limit sequence to label each component of the sequence based on the element's past and future contexts (Amodei et al). The fifth layer is a non-recurrent that takes the forward and backward units, finally, the output layer is a standard softmax function that returns the predicted character probabilities for each portion of the time  $t$  and character “k” (Hannun et al). For computation, DeepSpeech uses CTC loss to measure the error in prediction. “Connectionist temporal classification (CTC) is a kind of neural network output that links scoring function, for training recurrent neural networks (RNNs) like LSTM networks so that the timing is variable and it holds sequence problems”<sup>29</sup>.

### 4.3 Dataset Preparation and Training the Data

The dataset has been prepared, so 3 files needed to make ready the dataset: train.csv, dev.csv, test.csv. the CSV files included wav\_filename, wav\_filesize, and transcript, you can easily get the file size and audio file name using python. The following ratio for all audio files has been taken into account: 70 (training) – 20 (dev) – 10 (testing)! For training, you have to use Python 3.6, DeepSpeech, Tensorflow, and Mac or Linux environment.

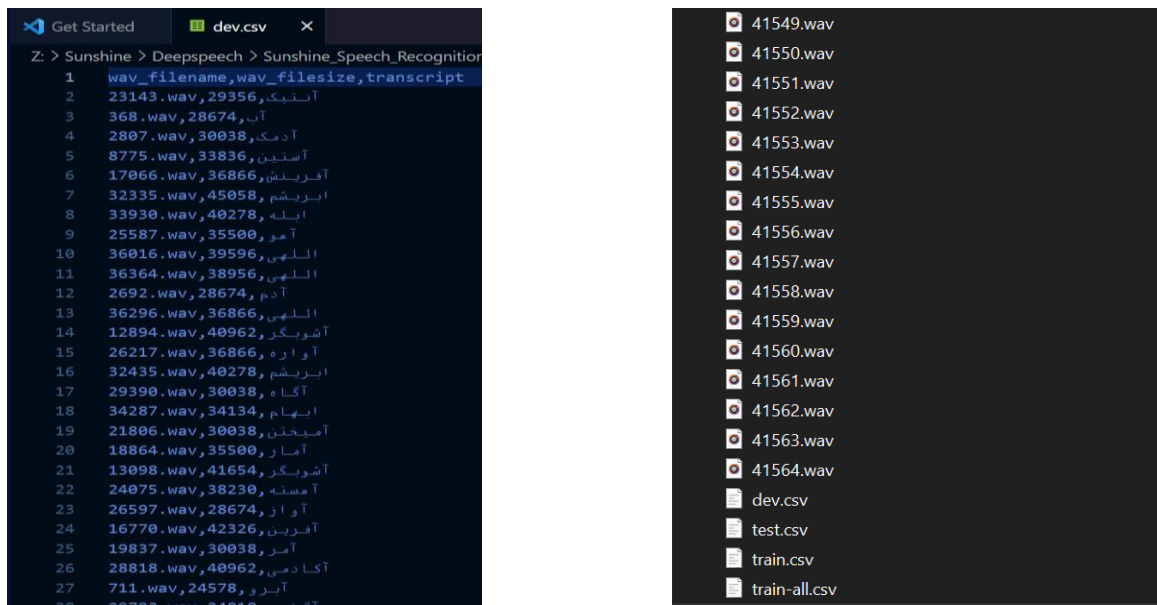


Figure 9. List of all CSV files (dev.csv, test.csv, train.csv).



## **5. Generic Introduction and Summary of the Development of Speech Recognition and Machine Translation App**

### **5.1 Purpose**

The project we're undertaking is to make a graphical user interface to train speech to text model and a translation model. It reduces the time to write code from zero-hero. It works as a framework and module.

### **5.2 Intended Audience**

This project has been developed for data scientists and non-data scientists. It means those who have the technical knowledge and those who do not have technical knowledge can use this application. This project is subjected to training speech-to-text and translation models without even writing one line of code. Therefore, someone who works in the industry of AI can directly install the app and use it. Those who do not have sufficient knowledge can also benefit from this app a lot. They can directly train their model by specifying the parameters they need.

### **5.3 Product Scope**

Most of the time developers investigate their time on learning speech recognition systems, so, the scope of this project is in-depth technical training models for speech recognition systems and language translations using the graphical user interface. This app helps users to directly train their models and only needs to gather enough datasets for the language they want to use.

#### **5.3.1 Included Functions**

- ✓ Dataset;
- ✓ Parameters;
- ✓ Training Panel;

- ✓ Speech panel;
- ✓ Language panel;
- ✓ Test panel;
- ✓ Command-line interpreter;

### **5.3.2 Excluded Functions**

Simple Speech Panel.

### **5.3.3 User Needs**

The users just need to install the app.

### **5.3.4 Assumptions and Dependencies**

This app is open-source, and users can get full control over the app. To have full control and run the app properly, the users should install python and relevant modules that are already specified in the 'requirement' file.

1. If the users do not create a dataset, they cannot train the model;
2. Users must have filtered the dataset to train the models;
3. Users must have enough dataset to train the model;
4. Users can not open two-panel together;
5. Users should specify the required parameters;

### **5.3.5 Benefits**

#### **Objectives and Goals**

The main goal of this project is to help developers to train speech and machine translation models directly for free without any cost.

#### **Current Systems**

We have lots of systems and services that can train language and speech models but this app has a perfect and foremost design and has different features (training model using GUI).

## 5.4 Operating Environment

The app will operate on all operating systems specifically Windows OS. A computer with a sound card and either a microphone or a headset is needed the most. For better proficiency, you need 8GB RAM, NVIDIA virtual GPU, a processor with high speed, free disk space.

## 5.5 Functional Requirements

- 1) Sufficient Dataset
- 2) Language Recognition Panel
- 3) Audio Recorder
- 4) Appropriate parameters
  - ✓ Audio
  - ✓ Title
  - ✓ Test size
  - ✓ Random State
  - ✓ Loss
  - ✓ Optimizer
  - ✓ Metrics
  - ✓ Epochs
  - ✓ Sound device
  - ✓ Number of words
  - ✓ First language
  - ✓ Second Language
  - ✓ Bins
  - ✓ Length
  - ✓ Validation Split
- 5) Language Translation Panel
- 6) Graph Visualization

- 7) Find count of each label and plot bar graph
- 8) Show All Labels

## **5.6 System Features**

In order for the system to function well, the following features are needed.

### **5.6.1 Filtered Dataset**

This feature has a high effect on the recognition process, Users should create their own dataset and send the dataset through a filter, for example, the dataset should have a similar sample rate, the background noise should be lowered.

### **5.6.2 Visualize Amplitude**

To see the amplitude of a file.wav in a time series domain, a file path must be given with an appropriate title.

### **5.6.3 Test Size**

The number of data to be allocated to model testing.

### **5.6.4 Random State**

To prevent generating a new random value (the training and testing datasets would have different values) every time we run(execute) our code, the Random state is considered important to be specified. If we do not specify a random state the accuracy may change.

### **5.6.5 Loss**

It is calculated as the average of the absolute difference between the actual and predicted values.

### **5.6.6 Optimizer**



An optimizer is a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rate. Thus, it helps in reducing the overall loss and improving accuracy (Ayush Gupta, 2021).

### **5.6.7 Metrics**

To calculate the performance of the model, each machine learning deep learning or neural network pipeline uses different parameters to show the correctness of the model.

### **5.6.8 Epochs**

It states the number of times that the learning algorithm will work through the entire training dataset.

### **5.6.9 Number of Words**

It should be stated how many words we have in our dataset.

### **5.6.10 First Language**

From which language do we want to translate.

### **5.6.11 Second Language**

The language we want to get the translation.

### **5.6.12 Length**

It returns the length of the sentences

### **5.6.13 Learning Rate**

The learning rate is a configurable hyperparameter or a tuning parameter used in the training of neural networks (optimization algorithm) that states the step size at each iteration while moving toward a minimum of a loss function. It has a small positive value, often in the range between 0.0 and 1.0. The learning rate controls how quickly the model is adapted to the problem.

#### **5.6.14 Batch Size**

The batch size defines the number of samples that will be propagated or passed through the network.

#### **5.6.15 Validation Split**

It is used to validate the model performance during the training.

#### **5.6.16 Bins**

The number of bars we'd like to have in our chart.

### **5.7 Nonfunctional Requirements**

#### **5.7.1 Performance Requirements**

To increase the training process, you can use GPU. The graphics processing unit, or GPU, accelerates additional workloads in high-performance computing (HPC). GPUs are particularly designed to fasten computer graphics workloads

#### **5.7.2 Safety Requirements and Security Requirements**

It is a lightweight and open-source application, there are not any security issues because you will directly download and install the app, and the app is free of HTTP requests. Before installing the program, you can on your PC defender system and use antivirus software.

### **5.8 Software Quality Assurance Standards**

For the software quality assurance standard which specifically focuses on process improvement, we use Capability maturity model integration or CMMI.

### **5.8.1 Maturity level 1: Initial**

Have not decided which modules should be used.

Have not decided which panel should appear for the first time we run the project.

Some modules are missed.

The training model crashed unexpectedly.

unpredictable outcomes of processes involved.

### **5.8.2 Maturity level 2: Managed**

Functional requirements are managed (Dataset filtration, specified modules, training models, sound device and ...).

processes are planned and controlled.

projects are managed and implemented according to their documented plans (Assigned a developer for the whole project, used python programming language, used CNN and RNN algorithms).

This risk involved is lower than the Initial level, but still exists (some modules are deprecated).

Quality is better than the Initial level.

### **5.8.3 Maturity level 3: Defined**

Processes are well characterized and described using standards, proper procedures, methods, tools, etc.;

(Gathered all of the necessary functional and non-functional requirements; Started by creating two different classes for speech recognition and a class for language translation. Because the training follows a specific order, we use waterfall methodology; For the design, we created a prototype using Adobe XD);

Medium quality and medium risk involved (improper set of parameters);

The focus is process standardization;

### **5.8.4 Maturity level 4: Quantitatively managed**

Quantitative objectives for process performance and quality are set.

To train a good model, the parameters will be defined by the users. The users can choose appropriate parameters according to their database. To ease the work of some users with insufficient knowledge about speech recognition, another panel has been developed. It means that the parameters already specified by the developer, the users can directly start the training.

Higher quality of processes is achieved;

Lower risk;

### **5.8.5 Maturity level 5: Optimizing**

Created a requirement.txt file to get the updated version of the modules.

A page to help the users with the process.

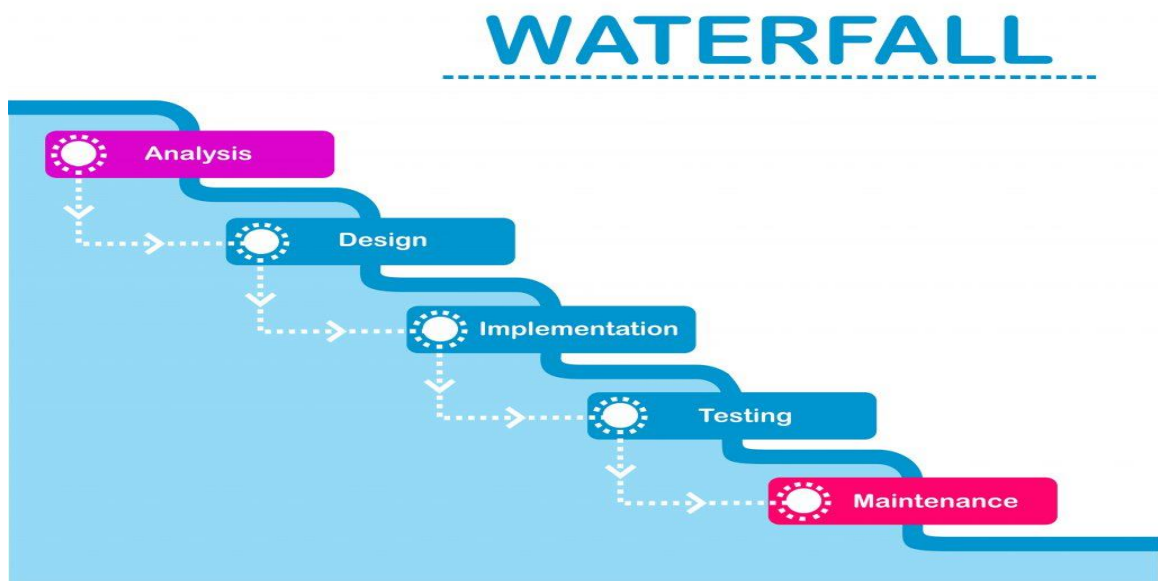
Has given full control for the users.

## **5.9 Business Rules**

*The users need to open the folder and take into account the structure of the folder, there is already folders and files for the dataset, if they want to change the name of the folders and files they will also need to bring some changes in the code, else they can put their datasets in the specified folders and files.*

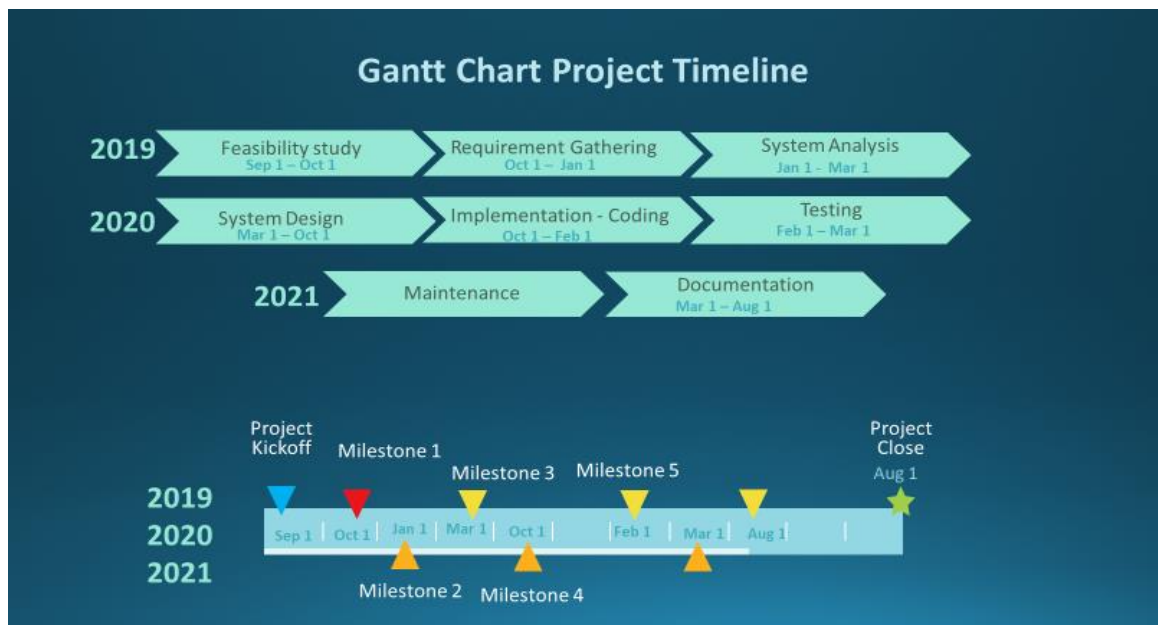
## **5.10 Methodology**

We can use different software process models, such as scrum, agile, and ... But we use waterfall model for our project as we do not need to go back in their afore stage.



**Figure 11.** *The Waterfall Model (The credit of this image goes to SALEH OT).*

## 5.11 Gantt Chart



**Figure 12.** *Gantt chart*

## 5.12 Tools

The tools that should be used through the development of this project are Python programming language and all the modules and libraries that are developed for the use of Machine learning, deep learning, and neural network algorithms such as (TensorFlow, nltk, matplotlib, and ....)

## 5.13 Scenarios

### 5.13.1 Scenario 1

**Purpose:** The scenario describes Sunshine Speech recognition and language translation App.

**Individuals:** Everyone who has the ability to use the computer.

**Equipment:** pc

**Scenario:**

1. Users can open the application (The full app will be released in GitHub).
2. Users should read the article in ResearchGate before starting the training. [ What kind of information required?] (The information has given in the paper).
3. Users should have a browser. [Can the users open any browsers?] (Yes, no constraints in browsers).
4. Users can check the help button to know about the project team. [ Is it significant?] (No, but if they want to contribute in the future, they can check out the about).
5. Users can navigate to other pages. [ Is there any information regarding the pages?] (Yes, each page has a specific name).

### 5.13.2 Scenario 2

**Purpose:** The scenario describes how to train speech to text.

**Individuals:** Everyone who has a little info about Neural networks.

**Equipment:** Computer (PC), any OS, Python, and python modules.

**Scenario:**

1. Users can run the game and open the speech recognition panel.
3. Users will see a page with the necessary parameters that should be inserted.
4. There are three different taps [Is there a tip to guide the users before inserting the parameters?] (yes, users can click the tips button and understand how to insert the parameters).
5. When the users start the training, they can witness their data in figures.
6. Users can test their model [ How the can test?] (By navigating to the test page).
7. Users can record their data, then play their voice, and finally, by clicking the predict button, they can see the output on the screen.

**5.13.3 Scenario 3**

**Purpose:** The scenario describes how to train the dataset directly without specifying the parameters.

**Individuals:** Everyone.

**Equipment:** computer (PC), python, python modules, dataset, any OS.

**Scenario:**

1. Users can navigate to the speech recognition page.
2. Users can click to start simple training.
3. Users just need to put their datasets in the dataset folder. [ Can the users put all the voice in one folder?] (No, each word should be put in a separate folder).
4. After preparing the dataset, users can train their data. [Are there any parameters that need to get changed?] (No, the parameters already specified, but the users have the ability to change the parameters as the project is open source, all the codes will be under the control of the users).
6. Users can record their data, then play their voice, and finally, by clicking the predict button, they can see the output on the screen.

**5.13.4 Scenario 4**

**Scenario:** The scenario describes how to train a language translation model.

**Individuals:** Those who have little knowledge in RNN.

**Equipment:** Computer (PC), any OS, Filtered dataset, python, python modules.

**Scenario:**

1. Users can run the program and navigate to the language translator page.
2. Users need to prepare a dataset for the language translation model.
3. Users can put the words or sentences with their meanings in the words.txt file [can the users change the name of the file "words.txt"?] (No, they cannot change, if the users want to have a unique name for their datasets, they have to change the name of word.txt in the source code).
4. [Do users need to specify the parameters?] (Yes, the parameters are important to be specified).
5. After specifying the accurate and suitable parameters according to the database, the users can start the training and create their own language translator model.

## 5.14 System and Interface Design

### 5.14.1 Use Case Diagram (Use Case Specification)

**Name of use cases:** Read the article, Start the program, Navigate to SR page, Navigate to MT page, Start training, Start simple training, Insert parameters, Predict, Record, Play,

**Actor:** Trainers

**Summary:** The trainers will train models to convert speech to text and to translate a language.

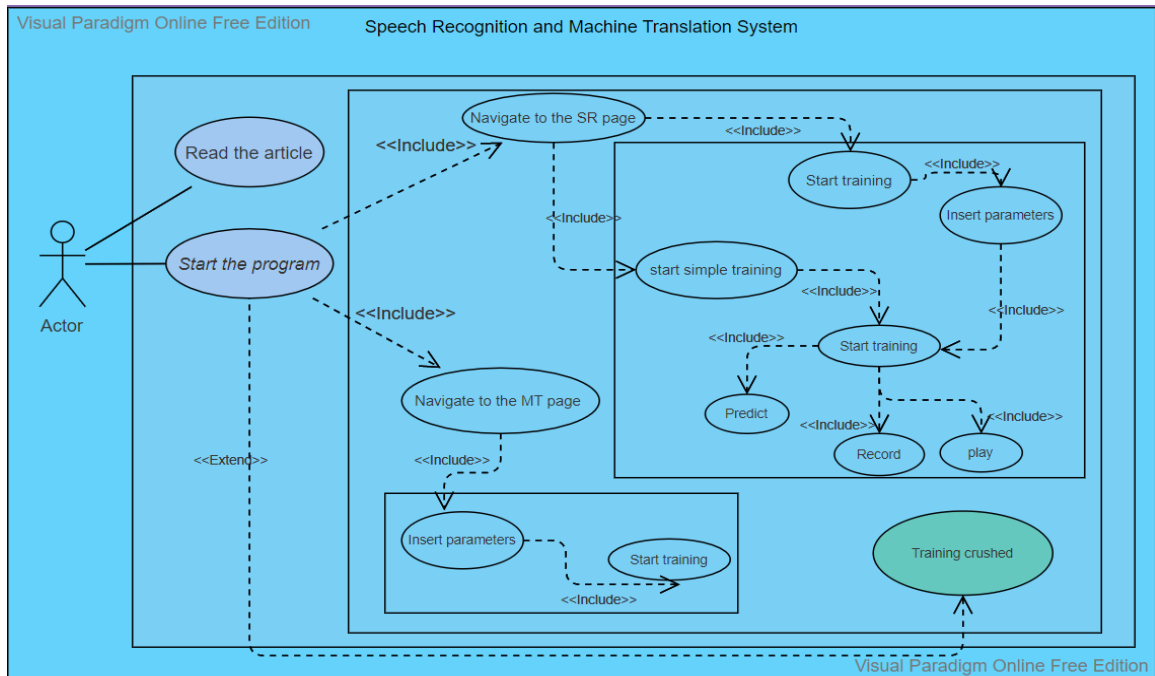
**Precondition:** Users must have filtered dataset and python + Python modules.

**Trigger:**

**Course of events**

1. Users can click the read article button to read the full article.
2. Users can navigate to SR or MT page.
3. Users should insert accurate parameters.
4. Users can train a speech recognition model directly without specifying parameters.
5. Users can record and play their audio.
6. Users can use their recorded audio and the trained model for the prediction.

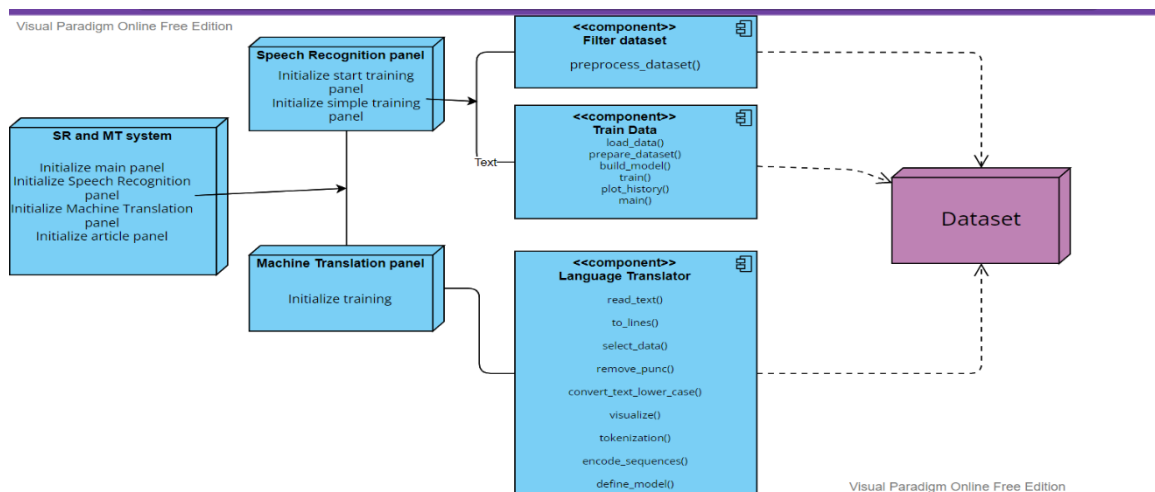




**Figure 13.** Use Case diagram for SR and language translation.

### 5.14.2 Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of the system, where the software components are deployed.



**Figure 14.** Deployment diagram for SR and language translation.

### 5.14.3 Data Flow Diagram (DFD)

A data flow diagram (DFD) is a way of representing a flow of data of a process or system. A data flow model is diagrammatic representation of the flow and exchange of information within a system.

We have different levels in DFD (but mostly the first three levels of DFD are being used by developers)

Level 0

Level 1

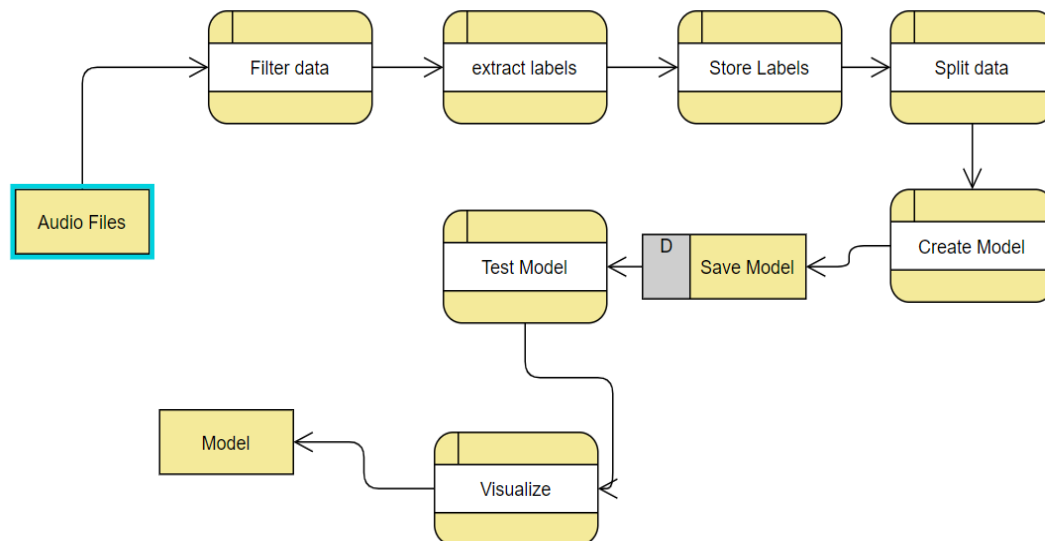
Level 2

⋮

Level 0 (A context-level DFD (level 0) is the most basic form of DFD. It aims to show how the entire system works at a glance. There is only one process in the system and all the data flows either into or out of this process.).

Level 1 (aim to give an overview of the full system. They look at the system in more detail. Major processes are broken down into sub-processes).

Level 2 (level 2 data flow diagram (DFD) offers a more detailed look at the processes that make up an information system than level 1 DFD does, we used level one for our project).



**Figure 15.** DFD diagram of SR system.

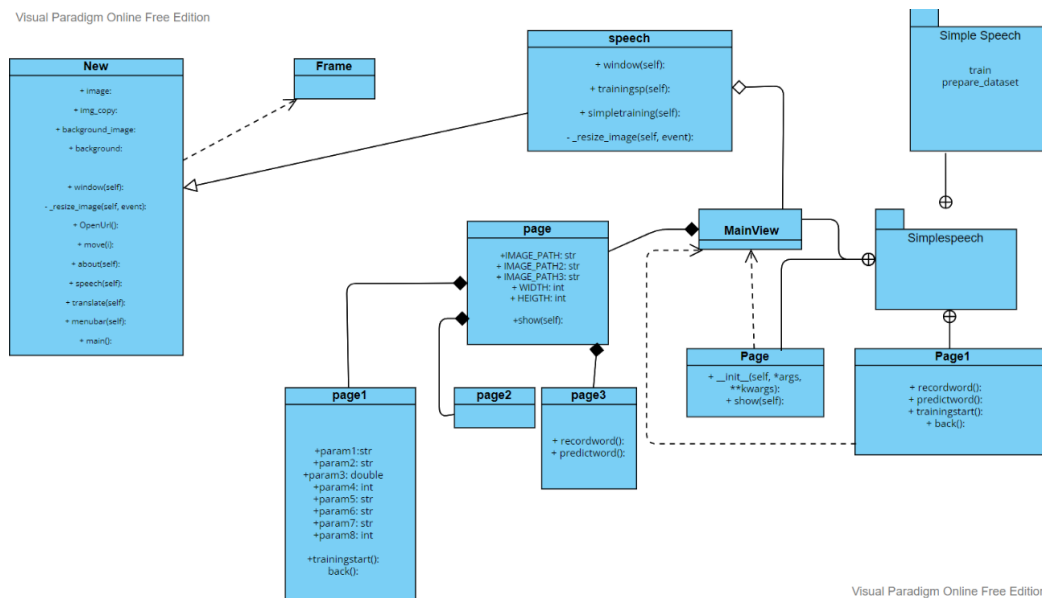
### 5.14.4 Class Diagram

Shows the static structure of classifiers in a system

Diagram provides a basic notation for other structure diagrams prescribed by UML

Helpful for developers and other team members too

Business Analysts can use class diagrams to model systems from a business perspective



**Figure 16.** Class diagram of SR system.

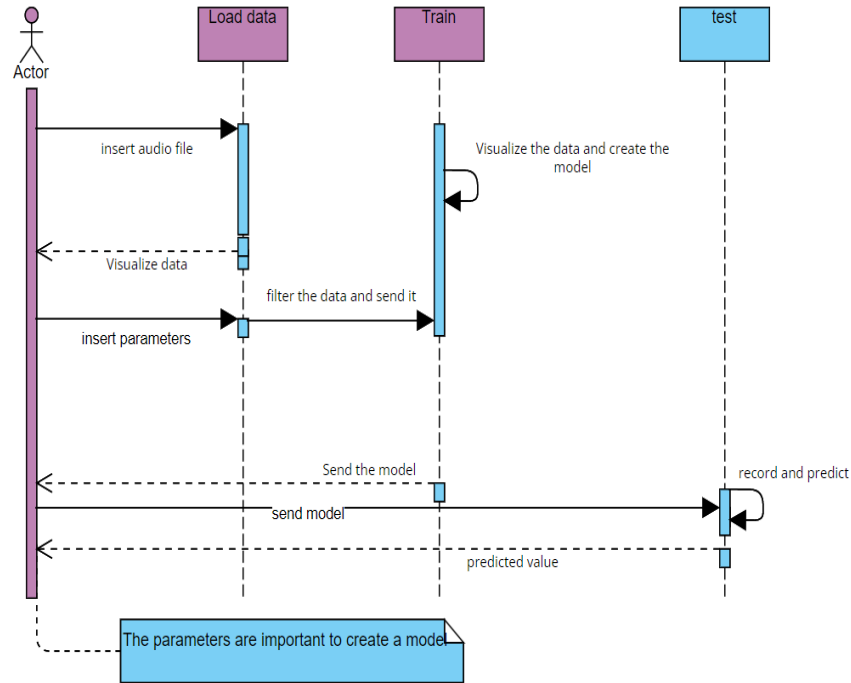
### 5.14.5 Sequence Diagram

Sequence diagrams are a type of UML diagram that shows how objects in A system or classes within code interact with each other, this diagram shows the sequence of events.

There are five steps to draw a sequence diagram

- 1) Define who will initiate the interaction;
- 2) Draw the first message to a sub-system;
- 3) Draw a message to another sub-system;
- 4) Draw return message to actor;

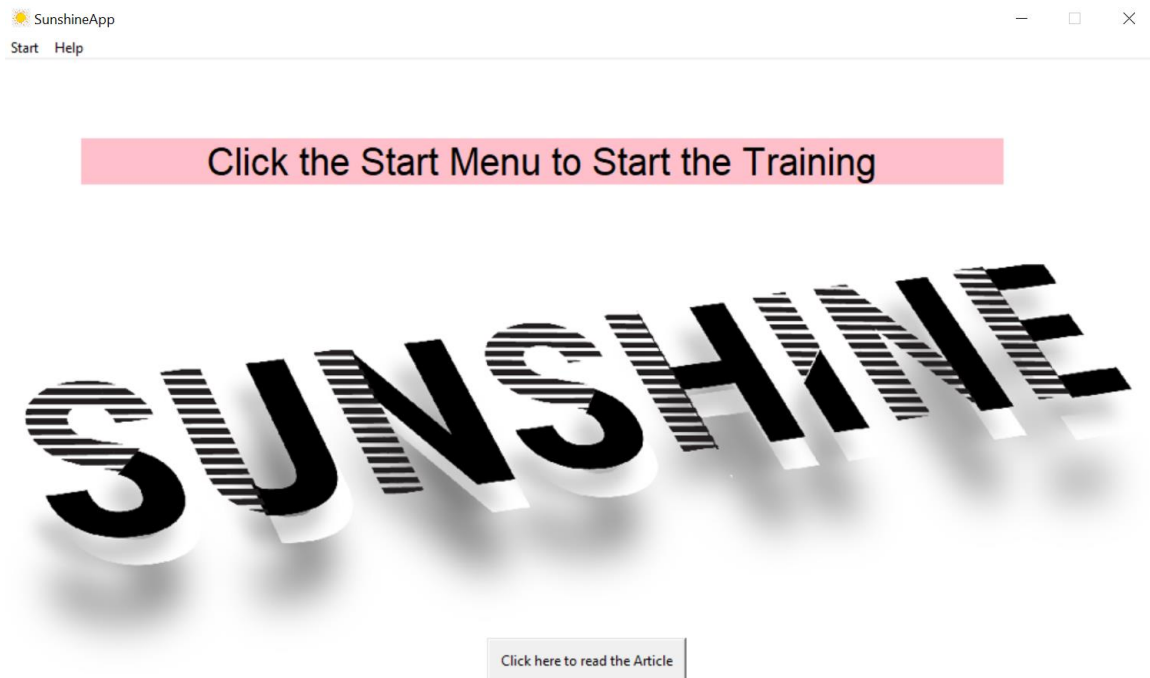
5) Send/Respond to an anonymous actor;



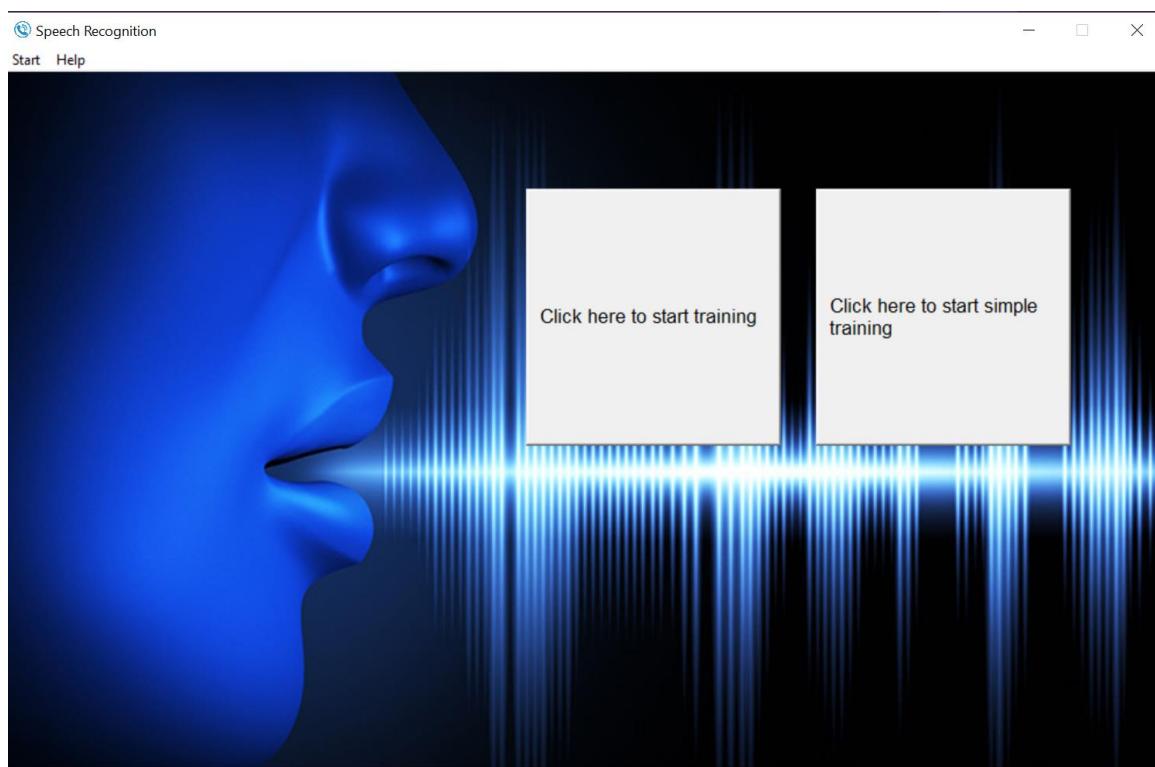
**Figure 17.** Sequence diagram for SR system.

#### 5.14.6 System Design

On the first page, there is a button that by clicking that button the page of the speech recognition paper will appear in a browser. In the menu bar, we have help and start, by clicking the start menu the labels for the speech recognition and machine translation will get appeared. when the users click the speech recognition, a page with two buttons will appear, the first button will open the complex speech recognition system and the second button navigate the users to the page that starts training directly. The complex training of speech recognition and language translation requires the insertion of suitable parameters by the users. There are also three other buttons, the record button, the play and predict button which will predict the newly inserted data.



**Figure 18.** *The main page of SR and MT system.*



**Figure 19.** *The Speech Recognition page of SR and MT system.*

Speech Training

Training Tips Test

Welcome, Please Write the Appropriate Parameters

/subfolder/file.wav/

Title

Test Size 0.0

Random State 0

Loss

Optimizer

Metrics

Epochs 0

Back Start training

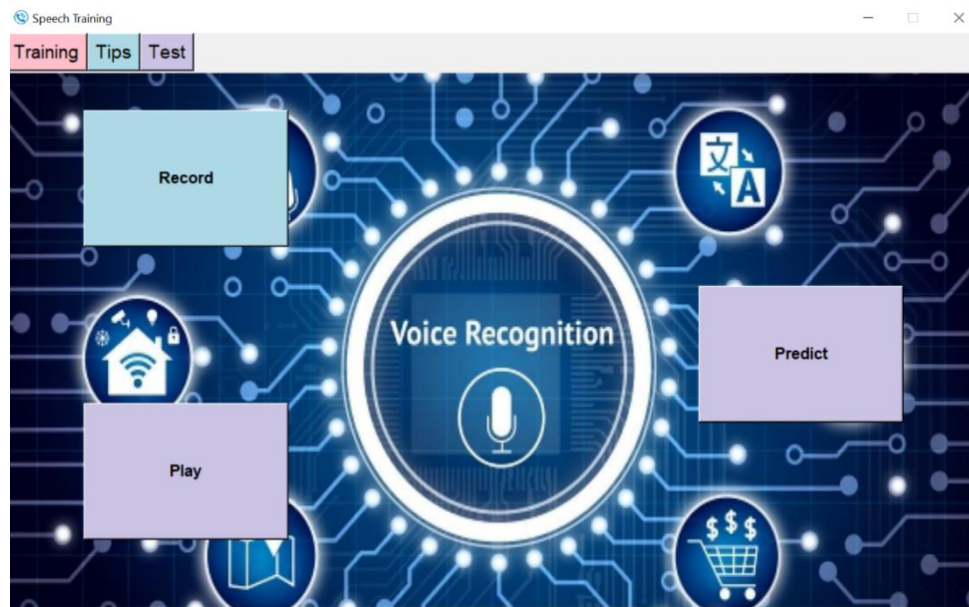
**Figure 20.** *The Complex Speech Training page of SR and MT system.*

Speech Training

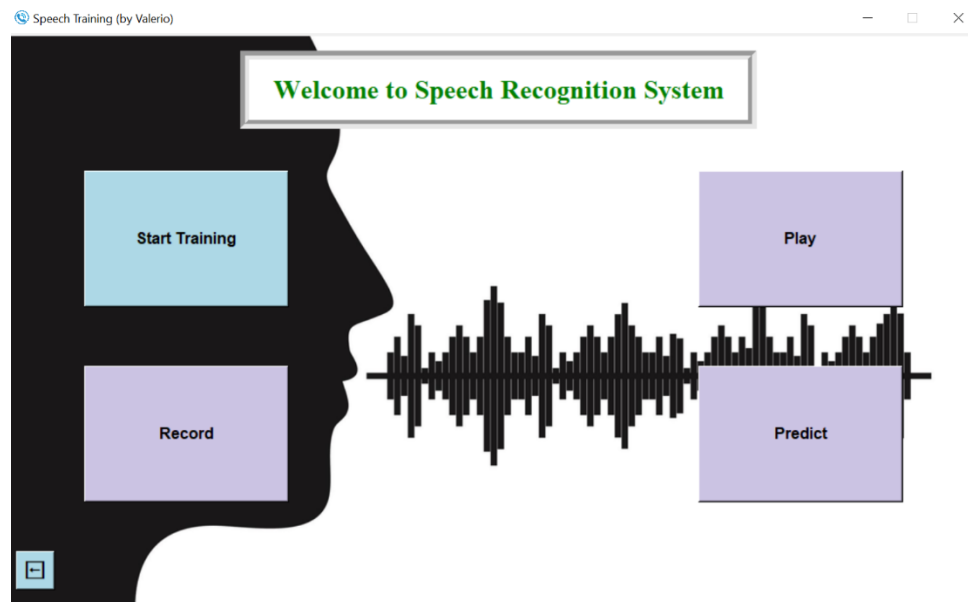
Training Tips Test

Try to choose each parameter carefully  
Do not put  
Comma;  
semicolon;  
Spaces;  
&  
Quotation Mark;

**Figure 21.** *Tips for SR system.*



**Figure 22.** *Test page for SR system.*



**Figure 23.** *The Simple Speech Training page for SR system.*

Welcome, Please Write the Appropriate Parameters			
Number of words	0	Test Size	0.0
1st Language		Random State	0
2nd Language		Learning Rate	0.0
Bins	0	Loss	
Length	0	Epochs	0
Language		Batch Size	0
Length	0	Validation Split	0.0
Language		Start training	

**Figure 24.** *The language translator page for SR system.*

## 6. Experimental Results

The major components and topics within the space of ASR are: 1) feature extraction; 2) acoustic modeling; 3) pronunciation modeling; 4) language modeling; and 5) hypothesis search (Deng & Li, 2013).

To create the best possible models for a language and appraise the performances of the models, different experiments have been used by modifying the percentage of invisible audio files during the training. In this paper, you saw developing a speech recognition from scratch used CNN (conv1d) then went through the two most powerful open-source frameworks which use different neural network algorithms with their pros and cons, for each application different number of the dataset have been used, for example with CNN 297482 sec audios, for CMUSphinx 10 hours recorded files have been used. The performance of the model with CMUSphinx and DeepSpeech speech was satisfactory.



In the first experiment, each neural network in cycles of 66 epochs has been trained, evaluating the resulting network after every cycle, but noticed a decrease in performance which can most likely be attributed to overfitting and underfitting. By lowering the number of the words to 50 40 30 20 10 5 3 and up to 2 the performance of the models got better. Then I got a subset of my own collected dataset and start training with CMUSphinx with the following configuration:

```
CFG_HMM_TYPE='.cont;  
$CFG_FEATURE="s2_4x";  
$CFG_NUM_STREAMS=4;  
$CFG_INITIAL_NUM_DENSITIES=256;  
$CFG_FINAL_NUM_DENSITIES=256;  
$CFG_N_TIED_STATES=2000;  
$CFG_MMIE="yes";  
$CFG_G2P_MODEL='yes';  
$DEC_CFG_VERBOSE=1
```

After training, I got two different folders with different files under the name of model architecture and model parameter following WER of 3.3% and noticed an increase in performance and got high accuracy, CMUSphinx is the best approach for speech recognition because with a fewer number of the dataset you can create a good model, also probabilistic works well as the problems with creating Speech-to-text model are the altered speaking manner, homophone, homograph and distorted acoustic like pray/prey. The drawback with CMUSphinx is creating a phonetic dictionary as ARPABET does not support other languages, you need to create one for your own language.

In addition, DeepSpeech has been used, models are trained for 33 epochs with a learning rate of 0.00095 on the full Dari dataset, it returned 1% WER that can be gratified for the created model. The drawbacks of DeepSpeech are: (i) require Linux or mac environment,

(ii) usage of some old versions of the libraries, look at synopsis of studies in (Srinivasamurthy et al).

Ultimately, we can recap the factors that affect the recognition process: 1. Environmental noise, i.e., stationary or nonstationary additive noise; 2. Distorted acoustics and speech correlated noise; 3. Different microphones; 4. limited frequency bandwidth; 5. Altered speaking manner; 6. Homophone; 7. Homograph; 8. Phonetic dictionary; 9. Language model; 10. Dataset; 11. Sample rate; 12. Neural network algorithms; 13. The number of channels of the incoming audio; 14. language constraints, (Chollet et al).

*Table 1. Presents the results of fine-tuning process.*

Models	WER	CER	loss	Learning rate
CNN (conv1d)	-	-	0.1867	0.0001
DeepSpeech	1	0.25	0.71	0.00095
CMUSphinx	3.3	3.2	-	-

## 7. Conclusion

According to the above, it can be concluded that neural network algorithms work well with ASR, I investigated the performance of an ASR based on CNNs, RNN, and HMM. This system was based on CMUSphinx from Carnegie Mellon University and DeepSpeech. DeepSpeech introduces an end-to-end deep learning-based speech system. DeepSpeech is able to exceed in performance than existing state-of-the-art recognition pipelines and CMUSphinx has the flexibility in the usage of various kinds of acoustic and language representations, after that I created a phonetic dictionary for the Dari language, Finally, this project can conduce to subsequent studies and works on building the language model for different languages including Dari.

## References

1. <https://www.nidcd.nih.gov/health/journey-of-sound-video>.
2. Satori, H., Harti, M., Chenfour, N. 2007. *Introduction to Arabic Speech Recognition Using CMUSphinx System*
3. El Amrani, M.Y., Hafizur Rahman, Wahiddin, M.R., Shah, A. 2016. *Building CMU Sphinx language model for the Holy Quran using simplified Arabic phonemes*.
4. Dib, M. 2019. *Automatic Speech Recognition of Arabic Phonemes with Neural Networks*.
5. Abdel-Hamid, O., Mohamed, A.R., Jiang, H., Deng, L., Penn, G., & Yu, D. 2014. *Convolutional Neural Networks for Speech Recognition*, IEEE/acm transactions on audio, speech, and language processing, VOL. 22, NO. 10.
6. Abdel-Hamid, O., Deng, L., & Yu, D. 2013. *Exploring convolutional neural network structures and optimization techniques for speech recognition*.
7. Sainath, T.N., Mohamed, A., Kingsbury, B., & Ramabhadran, B. 2013. *Deep convolutional neural networks for LVCSR*. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 8614-8618.
8. Zafar, I., Tzanidou, G., Burton, R., Patel, N., Araujo, L. 2018. *Hands-On Convolutional Neural Networks with TensorFlow*.
9. Nagajyothi, D., Siddaiah, P. 2018. *Speech Recognition Using Convolutional Neural Networks*. International Journal of Engineering & Technology.
10. Bourlard, H., Morgan, N. 1994. *Connectionist speech recognition a hybrid approach*.
11. Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., & Kingsbury, B. 2012. *Deep neural networks for acoustic modeling in speech recognition*. IEEE Signal Processing Magazine.
12. Schultz T., & Waibel, A. 2001. *Language-independent and language-adaptive acoustic modeling for speech recognition*. Speech Communication, vol. 35.

13. Dahl, G., Yu, D., Deng, L., & Acero, A. 2011. *Large vocabulary continuous speech recognition with context-dependent DBN HMMs*. IEEE Int. Conf. Acoust., Speech, Signal Process.
14. Leonard, R.G., & Doddington, G.R. 1984. *A database for speaker-independent digit recognition*. in Proceedings of the International Conference on Acoustics, Speech and Signal Processing, vol. 3. IEEE.
15. Dhankar, A. 2017. *Study of deep learning and CMU sphinx in automatic speech recognition*.
16. Matarneh, R., Maksymova, S., Lyashenko, V., Belova, N. 2017. *Speech Recognition Systems: A Comparative Review*. IOSR Journal of Computer Engineering (IOSR-JCE), Volume 19, Issue 5, Ver. IV.
17. Renals, S., Morgan, N., Bourlard, H., Cohen, M., & Franco, H. 1994. *Connectionist probability estimators in HMM speech recognition*. IEEE Transactions on Speech and Audio Processing.
18. Lamere, P., Kwok, P., Gouvea, E., Raj, B., Singh, R., Walker, W., & Wolf, P. *The CMU Sphinx-4 Speech Recognition System*.
19. <https://en.wikipedia.org/wiki/SoX>.
20. <https://cmusphinx.github.io/wiki/>.
21. W. Senior, A., & J. Robinson, A. 1995. *Forward backward retraining of recurrent neural networks*. NIPS.
22. Graves, A., Mohamed A.R., & Hinton, G. 2013. *Speech recognition with deep recurrent neural networks*.
23. Schuster, M., & K. Paliwal, K. 1997. *Bidirectional Recurrent Neural Networks*. IEEE Transactions on Signal Processing, vol. 45.
24. Behbahani, Y.M., Babaali, B., & Turdalyuly, M. 2016. *Persian sentences to phoneme sequences conversion based on recurrent neural networks*.

25. Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., Chen, J., Chen, J., Chen, Z., Chrzanowski, M., Coates, A., Diamos, G., Ding, K., Du, N., Elsen, E., Engel, J., Fang, W., Fan, L., Fougner, C., Gao, L., Gong, C., Hannun, A., Han, T., Vaino Johannes, L., Jiang, B., Ju, C., Jun, B., Le Gresley, P., Lin, L., Liu, J., Liu, Y., Li, W., Li, Dongpeng Ma, Z., Narang, S., Ng, A., Ozair, S., Peng, Y., Prenger, R., Qian, S., Quan, Z., Raiman, J., Rao, V., Satheesh, S., Seetapun, D., Sengupta, S., Srinet, K., Sriram, A., Tang, H., Tang, L., Wang, C., Wang, J., Wang, K., Wang, Y., Wang, Z., Wu, S., Wei, L., Xiao, B., Xie, W., Xie, Y., Yogatama, D., Yuan, B., Zhan, J., & Zhu, Z. *Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin*. Baidu Silicon Valley AI Lab1, 1195 Bordeaux Avenue, Sunnyvale CA 94086 USA Baidu Speech Technology Group, No. 10 Xibeiwang East Street, Ke Ji Yuan, Haidian District, Beijing 100193 CHINA.
26. Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., & Y. Ng, A. 2014. *Deep Speech: Scaling up end-to-end speech recognition*. arXiv:1412.5567v2 [cs.CL].
27. Rˆopke, W., Rˆadulescu, R., Efthymiadis, K., & Nowˆe, A. 2019. *Training a Speech-to-Text Model for Dutch on the Corpus Gesproken Nederlands*.
28. <https://deepspeech.readthedocs.io/en/r0.9/>.
29. [https://en.wikipedia.org/wiki/Connectionist\\_temporal\\_classification](https://en.wikipedia.org/wiki/Connectionist_temporal_classification).
30. Deng, L., & Li, X. 2013. *Machine learning paradigms for speech recognition: An overview*. IEEE Trans. Audio, Speech, Lang. Process., vol. 21, no. 5.
31. Srinivasamurthy, N., & Narayanan, S. 2003. *Language-Adaptive Persian Speech Recognition*.
32. Sun R., Giles C. L. 2001. *Sequence learning: from recognition and prediction to sequential decision making*. IEEE Intelligent Systems.
33. Medsker L.R. 2001. *Recurrent neural network*.

34. Demuynck, K., Roelens, J., Compernelle, D.V., & Wambacq, P. 2008. *An open-source speech recognition and automatic annotation kit*. Ninth Annual Conference of the International Speech Communication Association.
35. J. Robinson, A. 1994. *An Application of Recurrent Nets to Phone Probability Estimation*. IEEE Transactions on Neural Networks, vol. 5, no. 2.
36. Shneiderman, B. 2000. *The Limits of Speech Recognition*.
37. Dr. Reddy, R. 2001. *Spoking language processing*.
38. V. Shannon, R., Zeng, F.G., Kamath, V., Wygonski, J., & Ekelid, M. 1995. *Speech Recognition with Primarily Temporal Cues*. Science, New Series, Vol. 270, No. 5234.
39. Chollet, F. 2017. *Deep Learning with Python*. Manning Publications Co.
40. Logan, B., & Salomon, A. 2001. *A music similarity function based on signal analysis*.
41. The credit for the first figure goes to Rolphe Frédéric Fehlmann, it is designed by Dunya Yousufzai but the concept was taken from Rolphe Frédéric Fehlmann.