

# **I-exceed Technology Solution Private Limited**



**SUMMER INTERNSHIP 2018**

---

## **Appzillon Chatbot and Voicebanking Service in Python**

---

*"Under the guidance of Mr.R Sundarrajan and Mr.R Balachandra"*

July 13, 2018

**Submitted by**

**Jayaprakash A**

3rd year B.Tech Undergraduate  
Computer Science And Engineering  
Indian Institute of Technology Palakkad

# Preface

It is a great opportunity for me to work in a highly esteemed company **I-exceed Technology Solutions Private Limited** at Bangalore. On the account of finish of my summer internship, I am submitting a project report on **Appzillon Chat-bot and Voice-banking Service in Python**. Subject to the limitation of resource, time and efforts every possible attempt has been made to solve the problem most efficiently. To the best of my knowledge the entire code was written by me and there were no violations of license.

The whole project has been divided into 5 chapters.

1. Introduction
2. Problem definition
3. Approach
4. Results and Discussion
5. Learning Outcomes

# Acknowledgement

By doing this project, we would like to thank all those who have been instrumental in helping us in achieving the desired outcome. We would like to thank **I-exceed Technology Solutions Private Limited** for providing us with this opportunity and having faith in our abilities. We would like to thank our mentors **Mr.R Sundarrajan** and **Mr.R Balachandra** for being a constant support and giving us a broader picture of the whole scenario. Their constant endeavor and guidance has helped us to complete the project within the stipulated deadline. We would like to thank Mr.Srinivas for providing us the administrative help and making our working environment comfortable. Last but not the least, we would like to thank our fellow interns for helping us out with all the problems that we faced and making our internship experience an enjoyable one.

# About the Industry

I-exceed is a digital transformation partner for leading financial institutions worldwide. Our flagship offerings include Appzillon Digital Banking, a suite of pre-built omni-channel digital banking solutions and Appzillon Digital Platform, an award winning low-code application development platform that facilitates rapid digital transformation.

## Appzillon Digital Platform

Appzillon Digital Platform with its low-code and automated development approach powers rapid digital transformation in enterprises. Appzillon's industry leading micro-app based architecture and comprehensive digital repository facilitate quick time to market of omni-channel applications with zero compromise on security, scalability and user experience.

## Appzillon Digital Banking

Appzillon Digital Banking is a comprehensive suite of omni-channel banking solutions that help banks create superior engagement levels with their customers. The solutions ensure consistent experience across customer touch points and address diverse banking areas such as retail banking, corporate banking and internal operations.

# Summer Internship 2018

## Project Approval Certificate

Machine Learning Department

I-exceed Technology Solutions Private Limited

The project entitled “Appzillon Chatbot and Voicebanking Service in Python” submitted by Jayaprakash A, Athul MA and Siddhardha SST is approved for Summer Internship 2018, at Machine Learning Department, I-exceed Technology Solutions Private Limited.

---

Mr. Sudhir Babu D  
Vice President  
I-exceed Technology Solutions

---

Mr. R Balachandra  
Project Mentor  
I-exceed Technology Solutions

---

Mr. R Sundarrajan  
Project Mentor

# Abstract

To succeed in the digital age, banks and other financial services providers must engage more deeply with their customers, personalising the experience to build loyalty. The need for a powerful customer acquisition platform to underpin this engagement is paramount.

In today's digital age the choice of banks and products is broader and customers engage with systems rather than people, sometimes many times a day and across multiple channels. Digital team calls 'Effortless Simplicity', an organising principle they discovered through research with their members and which drives them to create elegant, intuitive interfaces.

Mobile banking is the future because of its cost effectiveness and ability to reach out to customers in remote areas. The past few years have seen a huge surge in the use of chatbots for banking purposes. Some of the world's most prominent banks have integrated online chatbots into their websites and mobile apps, and the stage looks set for new banks to follow suit.

Our goal is to create a chatbot for banking sector. It must be capable enough of doing all banking sector activities. These days voice commands have revolutionized the way of using mobiles. Google's cloud-to-speech model has set very high benchmarks in this field. The next biggest in the near future is voice banking.

But the user voice cannot be sent to cloud models for text extraction. So create offline models for speech to text conversion that are useful for banking space. The models must be capable enough to predict text accurately.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Problem definition</b>	<b>2</b>
<b>3</b>	<b>Approach</b>	<b>3</b>
3.1	Creating chat bot . . . . .	3
3.1.1	Current model . . . . .	3
3.1.2	New model . . . . .	3
3.1.3	Spell check service . . . . .	4
3.2	Voice Layer . . . . .	5
3.2.1	Speech Recognition . . . . .	5
3.2.2	Voice Recording Application . . . . .	6
<b>4</b>	<b>Results and Discussions</b>	<b>8</b>
4.1	Spell Check Results . . . . .	8
4.2	Speech to text models . . . . .	9
4.2.1	Sphinx offline model . . . . .	10
4.2.2	Dataset . . . . .	10
4.2.3	Input format . . . . .	10
4.2.4	Streaming accuracy . . . . .	11
4.2.5	Discussion . . . . .	11
<b>5</b>	<b>Learning Outcomes</b>	<b>12</b>
<b>6</b>	<b>Summary</b>	<b>13</b>
<b>7</b>	<b>Future Scope</b>	<b>14</b>
7.1	Conclusion . . . . .	14
7.2	Future Work . . . . .	14
	<b>References</b>	<b>15</b>

# List of Figures

3.1	Flow chart depicting the working of chat bot . . . . .	3
3.2	Flow chart showing the functioning of spell check . . . . .	4
3.3	Diagram depicting flow of work in speech recognition model . . . . .	5
3.4	Image depicting the work flow in voice recording app . . . . .	7
3.5	Record App Screen . . . . .	7
4.1	Suggest action for a text . . . . .	8
4.2	Change action for a text . . . . .	9
4.3	Reject action for a text . . . . .	9



# List of Tables

4.1	Training and testing results of audio recognition models . . . . .	10
4.2	Results of audio recognition models on stream of data . . . . .	11



# Chapter 1

## Introduction

In this modern day world, people use their mobile phones for almost anything. From shopping to booking tickets everything is done using the handsets. People nowadays prefer mobile banking to physically going to banks. Gone are the days of standing in long queues at the bank and filling out paperwork to access general banking services. Customers served with a most personalized approach is the key to growth for banks. Without customer satisfaction, no organization can sustain for long in the market. In the banking industry, it is necessary to provide 24\*7 customer support.

Chatbots will help with tasks like resolving queries, options to update client KYC, provide information on new schemes and services around the clock. They ensure that the customer's queries are solved in the shortest period and never let customers feel that they are interacting with a machine.

Voice recognition apps have been present in the market for several years. However, until recently, beyond the particular regions like warehousing, speech recognition has not seen a bigger uptake through the small business society as a whole. With professional and high accuracy rates of apps accessible for mobile devices, is speech recognition. This is a service that business could make a wonderful use of today. A number of business customers have already experienced what speech recognition can provide.

We propose a way of creating banking chat bot along with a voice layer on top of it.

# Chapter 2

## Problem definition

In this chapter, we will discuss in detail about the problem statement given for the internship period.

The first problem given to us during the tenure was to create a chat bot service in python. Also to create small helper services like spell checker and auto complete for better experience while using chat bot. The flow of the chat bot should be as follows.

The user enters text while entering the text he/she must be helped with some predicted text (auto complete service). After the text entry is finished. Pass the text input through a spell checker layer. Finally the processed text is fed to the model to extract useful entities from the sentence for further processing.

- **The auto complete** service must be capable enough to suggest the user with the probable words he/she might enter. Suppose user has entered the characters 'acc', it must be able to predict the word is account and suggest the user. The same goes for predicting the sentence after few words have been entered.
- **The spell check** service should help user in correcting spelling mistakes after the text input. An additional feature of replacing abbreviations with their respective full forms also needs to be done. For example, UAOF to replaced with 'Universal account opening form'.
- Lastly the **predictor service**. This service must extract various entities present in the sentence(if any). These entities include product, client-name and branch details. It is these entities that are later used for further processing.

The next task given to us was adding a voice layer on top of the model. The voice input has to be taken through a microphone and predict the text spoken. The flow after this step is same as the one described earlier.

# Chapter 3

## Approach

In this chapter, we will discuss in detail about the approach towards completing the two different tasks, chat bot and the other voice layer. Let us begin with the chat bot.

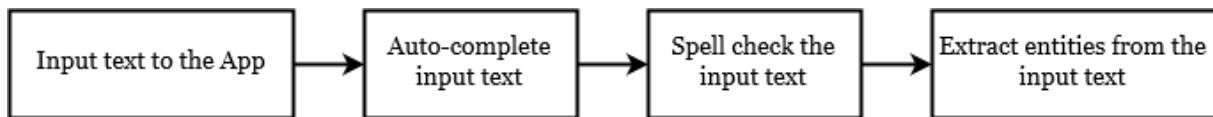


Figure 3.1: Flow chart depicting the working of chat bot

### 3.1 Creating chat bot

The company had a working chat bot service written in java. We were asked to improve the model and to create a python version of the chat bot. The java model uses Apache Open NLP library module. With this module named entity extraction was done.

#### 3.1.1 Current model

The current model used some built-in features of Open nlp for creating the model. The dataset to this model is set of sentences tagged manually. The features where the tag of the word, Parts of speech Tag of the present word and its n-previous and n-after words and various other features. Based on the set of these features the model was trained. There were no models ready for spell check and auto complete at that time.

#### 3.1.2 New model

After some research we found out that there were no ready made libraries in python that could help us with this task. An attempt to use certain features and create the model was made. This was done with the help of Natural Language Toolkit's chunker module. [1] However the results were not good as compared to the existing Apache's model.

The next idea was to use the Open nlp's features for creating the model along with some extra features. This was achieved by manually translating the Open nlp's source code to python. After adding some other features the model was created. The results were slightly

better than the existing one. The same model was built using different algorithms and results varied.

### 3.1.3 Spell check service

Spell check service was created from scratch. The main focus of this approach was to customize the spell checker to be able to cater the needs of banking industry. PyEnchant API was used. PyEnchant provides language bindings and wrapper classes to make the excellent Enchant spellchecker available as a Python module. It includes all the functionality of Enchant with the flexibility of Python and a nice 'Pythonic' object-oriented interface. It also has a special personal word list module that enables us to use list of words as our dictionary.

The first step in spell check was to identify spelling errors. For the dictionary, we created a personal word list from [NLTK](#) corpus. Any word not present in our dictionary was considered to be an error.

Now we have to predict possible suggestions for it. A word is a suggestion if it is made by altering minimum number of characters to form a meaningful word. Also for some acronyms we stored custom replacements for them. So when they are detected as errors, the custom replacement made would be suggested as a fix.

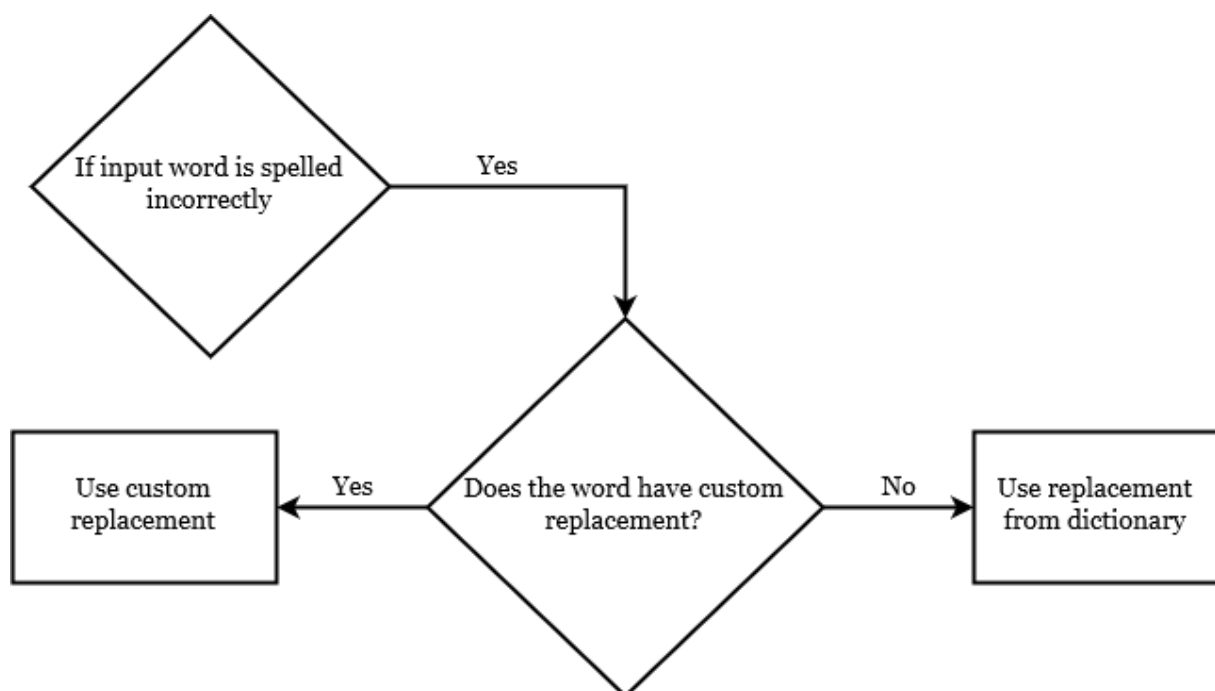


Figure 3.2: Flow chart showing the functioning of spell check

## 3.2 Voice Layer

The main aim of this task to use/create an offline model for speech to text conversion.

### 3.2.1 Speech Recognition

This was highly important as bank do not prefer voice to sent to cloud services for conversion due to privacy violation issues. According to state of art resources there is only one offline speech to text model Open Source Speech Recognition Toolkit.

[Sphinx](#) is a lightweight speech recognition engine, specifically tuned for hand held and mobile devices, though it works equally well on the desktop. This model was tested with voice samples. The results were too bad. It was unable to detect most frequently used words correctly. Due to its high inaccuracy, we had to drop the model.

The task now is to create a speech recognition model. We will now discuss the flow of extracting words from spoken text. The first step is to create a model that can identify word from single word audio files. Next step is to figure out the pauses or silence between any two words in an audio stream. The last step is to integrate both the things, so that for an audio stream, word is detected followed by silence detection and this is done repeatedly to be able to extract the entire text.

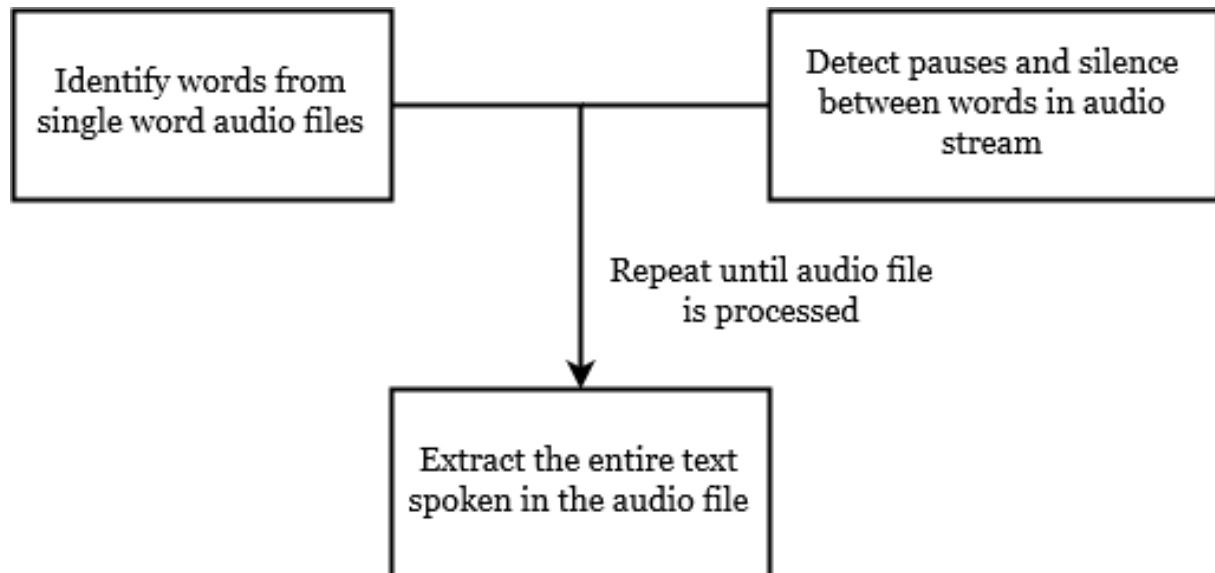


Figure 3.3: Diagram depicting flow of work in speech recognition model

**Recognising a single word.** This task of recognising a single word from audio is termed as Key Word Spotting. This model must be able to detect a word from an audio clip(contains

only one word). A neural network was built for this purpose where the input node is the audio stream and the out node has the words along with their prediction scores. This network was created with the help of tensorflow[2]. Google's cloud to speech models were built using Deep Neural Network, however this model was built using Convolutional Neural Network. The architecture of CNN used for KWS is taken from a research paper[3].

The details of the data-set and the results obtained from training would be discussed in the next chapter4.2.2. The next step was to run this model over a stream of audio with more than one word. For this step we had to make certain assumptions.

- Each word is 1 second long.
- Between any two words spoken there is a fixed interval of around 0.1 second.

These assumptions form the base of the subsequent steps. Also we need to encourage the fact that these assumptions are not too much vague. They are based on normal pace of speech. So now the audio stream is read per second for recognizing the word and the data for the following 0.1 second is discarded. This is how we programmed the model to work. The results of this step would be discussed in the coming chapter. This ends our discussion on the model generation and its architecture.

The same model needs to be extended to be able to detect various other words too. The data is however limited. A model was created using some pronunciation samples along with custom recorded voice of some individuals. The model was too bad. The main reason is lack of adequate data. We will discuss the steps taken to overcome this shortcoming in the next subsection.

### 3.2.2 Voice Recording Application

The above described shortcoming lead to creation of Voice Recording API. It is a web-app that uses system's microphone to capture audio. The user is asked to speak some sentences. The media file is captured as an audio blob and saved to disk through an Ajax request. Before saving the audio file to disk, the voice goes through a scanning layer. This layer detects if user has spoken the required text or some garbage. It is done to ensure that user do not speak unnecessary words and this might waste the disk space.



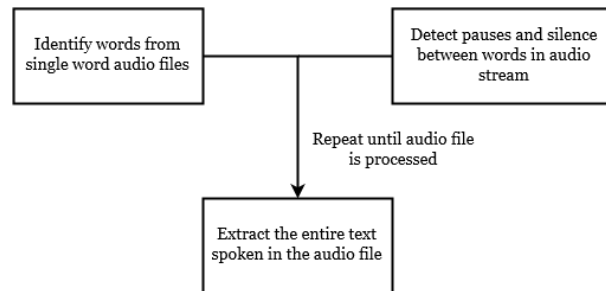


Figure 3.4: Image depicting the work flow in voice recording app

The scanning is done by using Google cloud speech model. The recorded audio is converted to required format(will be discussed in next chapter 4.2.2) and then the audio is sent for prediction. If predicted text contains the sentence to be spoken, it is saved to disk else it would be rejected. The app has been deployed and shared in the organization for collection of voice samples.

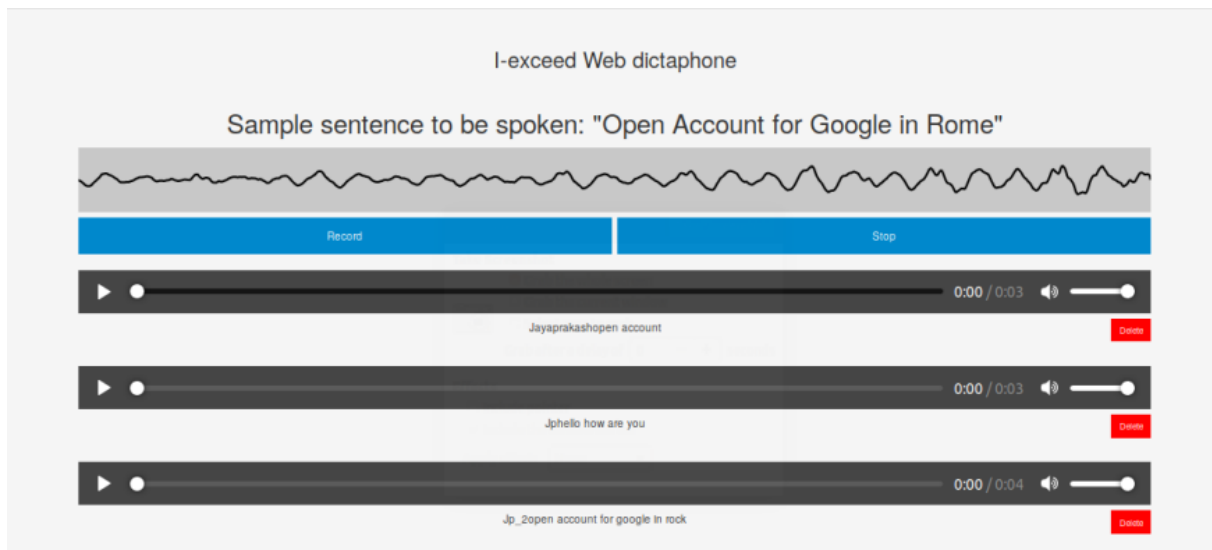


Figure 3.5: Record App Screen

# Chapter 4

## Results and Discussions

In this chapter, we will discuss in detail about the results of chat bot spell check module. This would be continued by discussion on tensorflow model and VR application. We shall begin the discussion with spell check module.

### 4.1 Spell Check Results

We have discussed in the previous chapter<sup>3</sup> about the spell check service. As per the guidelines we spell check service works on three different action types.

- **Suggest:** In this action level, only suggestions are made to sentence. It responds with an OK status in case there are no errors.

### Spell Checker

Sentence:

Action: ☒ Suggest ☐ Change ☐ Reject

Status: SUGGESTION

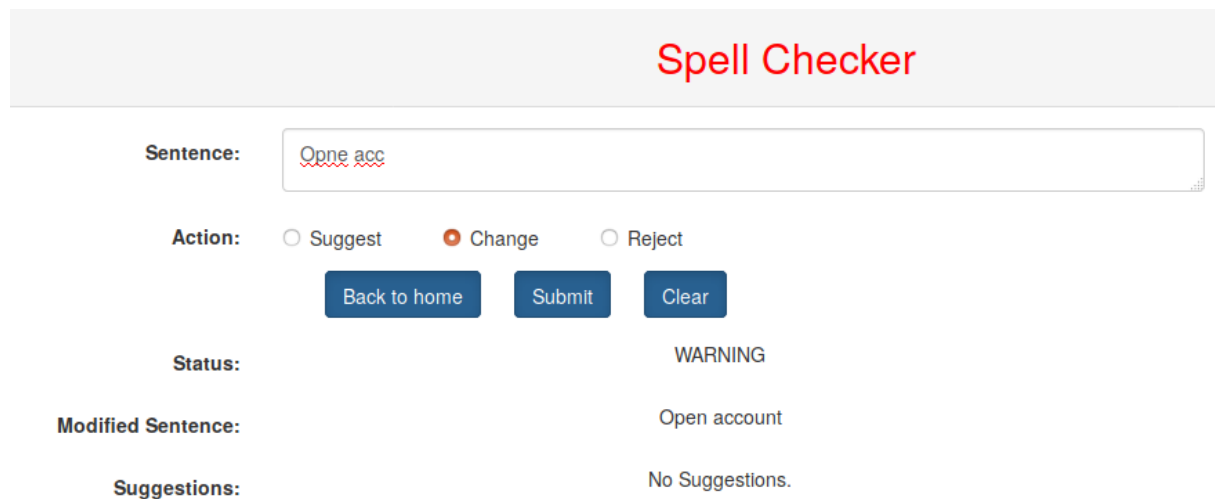
Modified Sentence: No changes made to sentence.

Suggestions:

Wrongly Spelt Words	Possible Replacements
Opne	Open
acc	account, ABC

Figure 4.1: Suggest action for a text

- **Change:** In this action level in case of errors they are replaced with most probable suggestion and warning is given.



**Spell Checker**

Sentence:

Action: ☐ Suggest ☒ Change ☐ Reject

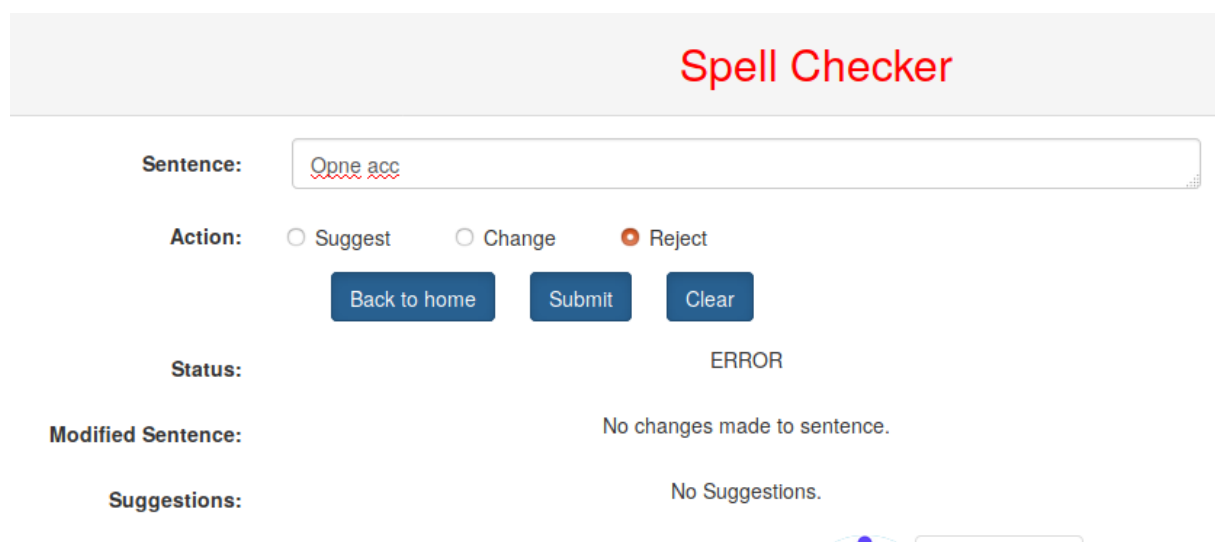
Status: WARNING

Modified Sentence: Open account

Suggestions: No Suggestions.

Figure 4.2: Change action for a text

- **Reject:** In this action level it checks for errors, accepts if no mistakes else a reject status is sent. This allows us only to see if any errors are present or not.



**Spell Checker**

Sentence:

Action: ☐ Suggest ☐ Change ☒ Reject

Status: ERROR

Modified Sentence: No changes made to sentence.

Suggestions: No Suggestions.

Figure 4.3: Reject action for a text

## 4.2 Speech to text models

We have created two models, one using speech dataset and the other using voice samples of limited number of individuals. In this section we discuss in brief about the dataset, input format and certain results obtained after training the models. But we would first start with sphinx offline model, and discuss the reason behind dropping the idea.

### 4.2.1 Sphinx offline model

Sphinx provides us with an offline model for speech to text conversion. We used Speech Recognition library of python on top of sphinx model to use the model. This library accepts audio either as a saved file or through microphone. It predicts the text from this audio source. The testing was not quite satisfactory. The model worked well for certain accents but failed for Indian accent. It failed to detect common English words. Hence it was rejected.

We will try to provide a reason for poor performance of sphinx model. The sphinx acoustic models are trained on telephone speech, recorded deliberately for the offline model. The model works only for certain kind of audio files.

### 4.2.2 Dataset

The dataset for the first model is speech dataset. Speech Commands [dataset](#), which consists of 65,000 WAVE audio files of people saying thirty different words. This data was collected by Google and released under a CC BY license. The dataset for the second model was the audio recordings of limited number of people.

### 4.2.3 Input format

The input format is quite simple. All the audio recordings of the word are saved in a single directory with the directory name being the word itself. The audio files should be in 16-bit little-endian PCM-encoded WAVE format. Also the files should have a sample rate of 16,000.

The model is trained with a learning rate of 0.01 for 15,000 steps, and then a fine-tuning pass of 3,000 steps with a 10x smaller rate. Table 4.1 gives a broad idea of the training and testing results of the two models.

Table 4.1: Training and testing results of audio recognition models

Model name	Speech dataset	Custom dataset
Some words	Stop, one, two, eight, bird, visual, learn,.. etc	Open, account, for, Google
No. of training files	≈ 2400 per word	≈ 10 per word
No. of training steps	18000	1800
Time taken	≈ 7hours	≈ 15 minutes
Training accuracy	94.03%	100.00%
Testing accuracy	81.93%	≈ 30%

#### 4.2.4 Streaming accuracy

As discussed earlier using a fixed silence duration between two words, the model was made to predict words from an audio stream. The model worked well.

Table 4.2: Results of audio recognition models on stream of data

Model	Audio clip duration	Total no of words	Correctly predicted	Incorrectly predicted	Accuracy
Trained on speech dataset	65 seconds	62	41	21	66.12%
Trained on custom dataset	65 seconds	62	6	56	9.67%

#### 4.2.5 Discussion

In this subsection we will try to provide a possible explanation for the low testing and very high training accuracy for the model trained on custom dataset.

The very high training accuracy can be credited to the existence of very less training data. The model has been trained repeatedly on the same data for over thousand steps and has over fitting occurred. Probably the model could be over fit to the validation set too.

So, outliers become much more dangerous and testing accuracy reduces. Noise in general becomes a real issue, could be from the training data itself. We observed that the model was only able to predict text properly from the voices of the people which was used to train the model. Even this prediction was not fully accurate. The model failed miserably when tried to predict other people's voice. All the above reasons contribute to the very less testing accuracy.

# Chapter 5

## Learning Outcomes

The internship journey was a very interesting and also good learning experience. The project given was not any project cooked up for the sake of internship. This task was a very important one for the company. It plays the foundation step to the further activities in the company.

I will begin with the formal academic learning. By the end of the project, I was able to understand in depth about chatbots. The way it works. The main section where machine learning was involved to extract entities from text. The other external services like spell check and auto suggest that plays a significant role in shaping a chat bot service. Due to direct involvement the core features and details were clearly understood and successfully implemented.

The next main learning was the voice section. I was able to capture the essence of speech recognition and able to visualize the working of it. I was amazed by the fact that a voice recognition to detect a single word extracts as many 9 lakh weights. It was a very important learning and would be extremely helpful for me in successive machine learning projects.

The other skill learnt was deploying an application to web server. This was the first time I did it and it was quite interesting. The flask application was deployed to Web Server Gateway Interface server. Since Apache server configurations was very new to me, it was quite problematic at the start. But now the app has been deployed successfully. These were some of the important skills learnt during the internship period.

# Chapter 6

## Summary

In this section we will summarize all the tasks done during the course of project. The main aim of the project was to create a chat bot service and voice banking service.

The first task of creating chat bot service for banking sector was achieved using named entity extraction model. The model is fed with set of utterances which are tagged manually. The model uses certain features like pos-tag, n-gram and n-previous and after words. Using this features the model is created. So, the input is given to the chat bot with auto suggest help. After passing the spell check layer, the text is fed to model for entity extraction. Based on the various elements extracted further processing is done.

The second task to investigate about speech to text offline models. The only available offline model as of now is Sphinx. But the model works poorly for most of the words spoken in general context. A model was created that detect the word from an audio file containing a single word. The model gave pretty good results. Along with an assumption of fixed interval between two words, the model was able to predict the words from an audio stream containing multiple words.

# Chapter 7

## Future Scope

### 7.1 Conclusion

The chat bot works quite well and is able to extract entities with a very high accuracy. Also the speech to text model is fine as it is able to predict the single word audio files quite decently. For internal testing and voice layer for present applications is done using Azure Bing service. As per the constraints of time and resources a pretty good research and model was created.

### 7.2 Future Work

The main chat bot could be improved by training the model with more utterances and be able to perform equally good. The second important task would be to create a full working the speech to text model. The model needs to be trained to detect other words as the scope is limited currently. The current model is able to detect words from an audio stream but with a fixed silence duration between any two words.

The audio recognition model needs a fix. It should be able to work for varying amount silence duration. The model can detect silence and also can detect words. If the silence detection and word detection are done properly and repeatedly in a loop, the final speech to text model would be ready. Embedding this model to the main speech to text model would finish all the required tasks.



# References

- [1] E. K. Steven Bird and E. Loper, “Extracting information from text.” <https://www.nltk.org/book/ch07.html>. 3
- [2] “Audio recognition.” [https://www.tensorflow.org/tutorials/audio\\_recognition](https://www.tensorflow.org/tutorials/audio_recognition). 6
- [3] C. P. Tara N. Sainath, “Audio recognition.” [https://www.isca-speech.org/archive/interspeech\\_2015/papers/i15\\_1478.pdf](https://www.isca-speech.org/archive/interspeech_2015/papers/i15_1478.pdf). 6