B.TECH PROJECT REPORT

on

DocVita: A Digital Assistant For Doctors

By

Shivam Bhosale

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

November 2018

DocVita: A Digital Assistant For Doctors

A PROJECT REPORT

Submitted in partial fulfillment of the

requirements for the award of the degrees

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted By:

Shivam Bhosale

Guided By:

Dr. Somnath Dey

INDIAN INSTITUTE OF TECHNOLOGY INDORE

November 2018

CANDIDATE'S DECLARATION

I hereby declare that the project entitled *DocVita: A digital assistant for doctors* submitted in partial fulfillment for the award of the degree of Bachelor of Technology in *Computer Science* completed under the supervision of Dr. Somnath Dey, IIT Indore is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

Signature and name of student with date

1.0 CERTIFICATE BY BTP GUIDE

It is certified that the above statement made by the student is correct to the best of my knowledge.

Signature of BTP guide with date and designation

PREFACE

This report on *DocVita: A digital assistant for doctors* is prepared under the guidance of Dr. Somnath Dey.

Through this report I have tried to give a detailed design of an innovative digital assistant for the doctors and tried to cover every aspect of the new design, if the design is technically and economically sound and feasible.

I have tried to the best of our abilities and knowledge to explain the content in a lucid manner. I have also added figures to make it more illustrative.

Shivam Bhosale
150001032
B.Tech, IVth Year
Discipline of Computer Science & Engineering
IIT Indore

ACKNOWLEDGEMENT

I wish to thank Dr. Somnath Dey for the kind support and valuable guidance. Furthermore I would also like to acknowledge with much appreciation the crucial role of our mentors Mr. Vikram Patel and Mr. Anmol Arora, who gave the permission to use all required equipment and the necessary materials to complete the project. I have to appreciate the guidance given by other supervisors, as well as the panels especially in our project presentation that has improved my presentation skills, thanks to their comments and advices.

It is their help and support, due to which I became able to complete the design and technical report. Without their support this report would not have been possible.

Shivam Bhosale
150001032
B.Tech, IVth Year
Discipline of Computer Science and Engineering
IIT Indore

ABSTRACT

In healthcare industry there are many areas that can use some sort of automation or technical innovation, so that their efficiency can be increased. The doctors and the patients are one of the keystones in the industry. This project was started with the inspiration to enable doctors so that they can manage both their patients and their clinics with minimal efforts.

In this project the team decided to make a digital assistant that can act and respond to the doctor to cater the needs of patient management and clinic management.

The main problem was divided into 4 subproblems:

- **Speech to text transcription**
- **Text processing (NLP)**
- **Inference engine**
- **Text to speech conversion**

We worked on each subproblem at a time and then integrated them to get a virtual assistant that can help doctors.

CONTRIBUTIONS

The major contributions that were made by me are as follows:

• In speech to text engine I explored online APIs available in Python libraries to do speech to text transcription. I studied and made the concepts clear in CMUSphinx which made us understand the type of problem we are dealing with and how to form Acoustic models, language model and dictionary for the problem statement.

• Approached the CNN and LSTM approaches using Keras.

• In the NLP engine part, I approached the problem with LSTM approach as sequence to sequence labelling using Keras.

• In Dialogflow part, I was responsible for back end handling of text input and building APIs for Inference engine.

• For security purposes I implemented the custom end point's back end handling using Node.js framework.

I would like to appreciate my partner contribution in the project. His major contributions involve:

• In speech to text engine he explored many online APIs to do speech to text transcription. Adding to it he also made CMUSphinx operational on our systems for offline speech recognition.

• He approached the CNN and LSTM approaches using Tensorflow and Pytorch.

• In the NLP engine part, He approached the problem with LSTM approach of multilabel classification using Keras.

• In Dialogflow part, He was responsible for front end handling of text input and send it in a meaningful format to the backend for interpretation.

• For security purposes He implemented the custom end point's front end using Angular 6 framework.

Contents

List of Figures

List of Tables

Chapter 1 : INTRODUCTION

In healthcare industry there are many areas where technological innovation or some sort of technical automation can be used, in order to improve efficiency and labour. Some of which can be :

- Limited time for communication between patients and doctors.
- Limited tools for doctors to manage their patients.
- Manual work in clinics such as prescription process, data management of patients in small clinics etc.
- Staying up to date on quick updations of vital information.

Hence, there seems to be a need of smart solutions in order to overcome certain basic problems, some of which are mentioned above, in an efficient way.

Chota Hospital decided to take on such problems aforementioned and provide some technological solutions to them. As a solution they came up with an android application which can help doctors to manage their patients as well as their clinics.

To take the same idea further and make it even more easy for doctors to use the team thought what if there would be an assistant always beside doctors, which can perform tasks that the doctor tells it to do. It should be as natural as doctor should feel like s/he is talking to a real person, performing all the tasks for him/her. As a result we decided to design a smart digital assistant that can respond and act on all the tasks that the doctor tells it to do.

1.1 Current State Of The Art Technologies

The current state of art technology includes smart assistants that are able to do specific day to day tasks. Some of the smart assistants that are very powerful and exists in the current market are, *Alexa (By Amazon)* [1], *Google Assistant (By Google)* [5], *Siri (By Apple)* [2], *Cortana (By Microsoft)* [10], *Bixby (By Samsung)* [17]

As we discussed above that these products are already available in the market, but all of them are only built around mostly casual home and daily tasks management. They are mainly used for schedule management, entertainment purposes i.e they are too generic and are not built for specific domain. However they show great potential for future application, but for now they are not quite capable to solve this problem.

Certain drawbacks that are shown by the smart assistants in case of our use case are as follows:

- The smart assistants show too generic behaviour which can lead to erroneous results.
- Unavailability of domain specific features.
- No common infrastructure to support medical information.

Chapter 2 : CONCEPT DESIGN

In order to overcome the limitations shown by current existing technologies in the previous chapter, we focussed to design a smart assistant that is capable of performing correct actions based on the input given to it. The major constraints upon which the assistant will be designed are as follows:

• The smart assistant is based on a particular domain i.e. medical field where terminologies are quite advanced.

• It should be precise enough so that it can be accepted in a low tolerance field such as medical industry.

• Proper features designed such that it can take actions upon the desired input given.

• Integrating it with a proper infrastructure which can hold medical information.

We decided to break the problems into subproblems and work on each subproblem.

• **Speech to Text Conversion Engine:** First black box will handle speech to text conversion. The estimated time assigned for this block was *1 month and 15 days.*

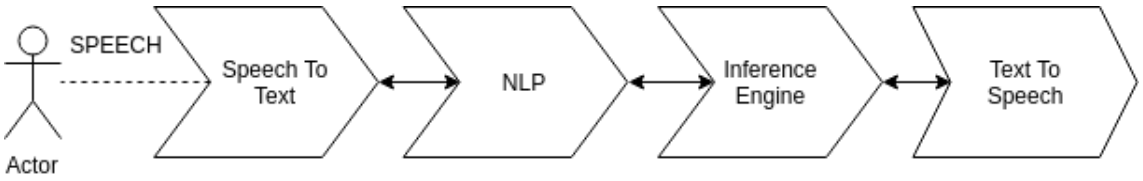
• **Text Processing (NLP):** Second black box was *Natural language processing* engine. This box will tokenize the text generated from first black box and also determine the intent of the conversation. Time estimated for this block was *1 month.*

• **Inference Engine:** Third black box is where all the logical decisions will be taken as per the tokens and intent generated by the NLP engine and also provide with the text response to drive the conversation forward. Time estimation for this box was about *1 month and 15 days.*

• **Text to speech conversion:** Fourth black box will convert the text response provided by the inference engine into speech. Time estimated for this black box was *1 month*.

Refer to *Figure 2.1* for a segregated flow of the main problem.

Figure 2.1: Concept Design Flow



Chapter 3 : EXPERIMENTS

3.0 Speech To Text Engine

This is the very first black box where the sound input by user is captured and converted into raw text. This is very important step in the entire process because if accuracy of this box is poor then it will give incorrect results as output. Also it is also very important in a very low tolerance field like medical. So our goal was to make it as accurate as possible. We set our benchmark at 85% accuracy for this black box.

3.0.0 Approach-1

We looked for online speech to text converter APIs such as *Google cloud, IBM Watson, Bing* etc. But they were not economically feasible for the product. Hence, we searched for offline open source speech to text convertors such as *CMUSphinx* [18], *Kaldi* [14]. We got about 62% accuracy with the existing model built on *CMUSphinx*. But since the accuracy was low we decided to build our own model.

3.0.1 Approach-2

To build our own model we first had to preprocess the data i.e. *.wav files* to feed it into any model. For that we used *Mel Frequency Cepstral Coefficients(MFCC)* [12] as feature vector. Then we implemented various models such as, *Hidden Markov Model, CNN model, LSTM model*.

The most successful implementation was LSTM model that we built for word recognition. But to interpolate the idea to convert all words demanded huge amount of training data that we could not afford to invest the time to gather. So we dropped the approach and worked on a pre existing model of *CMUSphinx* and adapted the model which required only 10 minutes of persons recording. Then we

built our custom language model and phonetic dictionary and got an accuracy of 97% for adapted CMU sphinx model.

3.0.2 Approach-3

The language model we built was not complex enough to support whole problem and since we did not have any pre existing data that we can use to build language model we decided to search for other options. We explored that *Google home [5], Alexa [1]* also provide with speech to text conversion API. We compared the trade off in *Table 3.1* ahead and decided to use the *Google Home APIs* for speech to text conversion.

3.1 Text Processing Engine (NLP)

What we wanted to acheive from NLP ?
From the text we wanted to know the context such that the inference engine can take decisions on it correctly.

How we preprocessed text ?
The textual information must be converted to actual numbers for feeding into the neural network. The text was preprocessed using 3 mainstream approaches which are as follows:

- **Bag Of Words Model (BOW)** [9]
- **Word2Vec Model** [11]
- **Doc2Vec Model** [8]

We preferred *Doc2Vec* model with the vision of future compatibility where the language or grammar was not hard coded such that we can get the text conversion quite accurate. Now the text is ready to be fed into the neural network.

3.1.0 Approach-1

Since, we explored neural network in our speech to text black box, we found it similar to the previous subproblem. Hence, we started implementing different neural network models right off the bat. We identified the type of problem and came to the conclusion to solve it using *LSTM* [3] [13]. We implemented both *bidirectional LSTM* and *unidirectional LSTM* to quantify the results in both cases. The results varied by 1-3% running different epochs. The overall accuracy was 91.4%-92%. Accuracy was measured by the context tags (the collection of certain tags define a context) that were the output for each sentences. The entire dataset preparation was done by us which included permutations of various texts that can be said by the doctor. Finally, it was hosted on cloud such that we can feed in texts and get contextual information.

Then we trained a *LSTM* which worked on sequence to one labelling technique i.e. we gave input a specific sequence of tokens and the model will give a single output which is actually the context of the sentence. Now the mapping of tags to the context was to be done such that we can build our inference engine. The previous *LSTM* model gave context tags as output which was input to this *LSTM* model for getting the correct contexts.

3.1.1 Approach-2

Google's NLP engine is *Dialogflow* [6] which lets us build an efficient model with small amount of data which was a drawback in our model. Due to small amount of data our accuracy was less as compared to *Dialogflow* engine. It was a purely business approach and also for better results to shift to Dialogflow. Existence of APIs made it easier to train the model overtime which was a difficult task in our manual model.

3.1.2 Security Layer

We used *Implicit OAuth 2.0* [4] flow for user authorization using custom endpoint for authorization and token generation.

The *OAuth* linking type supports two industry standard *OAuth 2.0* flows, the implicit and authorization code flows. We used *implicit OAuth 2.0* flow for user authorization using custom endpoint for authorization and token generation. The custom endpoint was built on *Angular 6* [7] framework.

In the implicit code flow, Google opens the designed authorization endpoint in the user's browser. After successful sign in, the user returns a long-lived access token to Google. This access token is now included in every request sent from the *Google Assistant* to *Actions On Google* to keep an authentication check on every request.

In the authorization code flow, there are two endpoints:

- The authorization endpoint, which is responsible for presenting the sign-in UI to the users that are not signed in and recording consent to the requested access in the form of a short-lived authorization code.

- The token exchange endpoint, which is responsible for two types of exchanges:

  – Exchanges an authorization code for a long-lived refresh token and a short-lived access token. This exchange happens when the user goes through the account linking flow.

  – Exchanges a long-lived refresh token for a short-lived access token. This exchange happens when Google needs a new access token because the one it had expired.

Although the implicit code flow is easier to implement, Google [4] recommends that access tokens issued using the implicit flow never expire, because using token expiration with the implicit flow forces the user to link their account again. If

there is a need of token expiration for security reasons, the auth code flow should be used instead.

Figure 3.1: OAuth 2.0 Implicit Flow

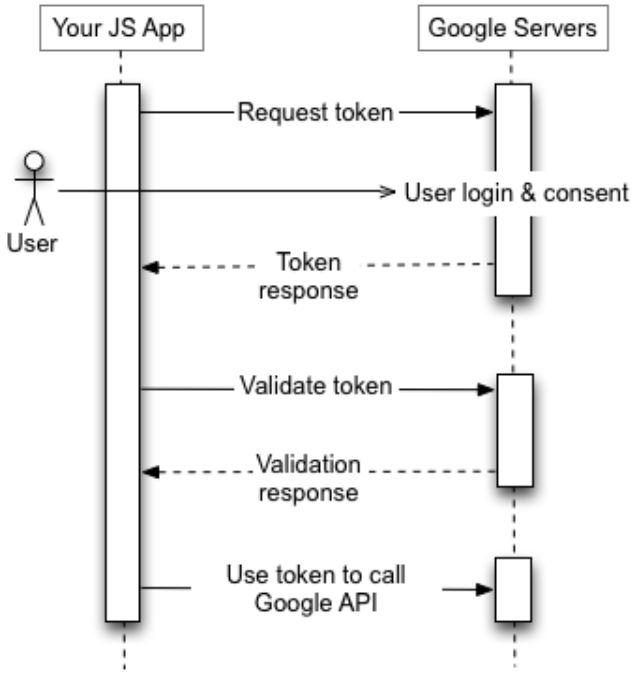


Table 3.1: Comparision between *Our Proposed Model* & *Dialogflow*

Our Proposed Model	**Actions on google**
Our *Speech To Text* model was still not up to the mark due to lack of training data. However given enough data we will have advantage and control to retrain model as we want if needed.	While *Google* has pretty good model to use. However since the model was too generic we will have to do some extra work in NLP engine to compensate for the error that it will generate.
Our NLP engine was fairly good and was scalable as well.	Dialogflow also lets us design our own model and train them as we wish.
Our models still lacked a platform such as web or android. So in order to use them we would have to build a platform.	On the contrary google already has their APIs made and ready. Also they can be integrated on any platform easily.

3.1.3 Our Proposed Model vs Action On Google

As we can see the trade off between both the options. We decided to opt for *Google Home* and *Dialogflow* approach since it had APIs ready for usage and scalable too. The only thing we were left with was to design a good NLP engine within the remaining time bounds.

3.2 Inference Engine

Up until now we have converted speech to text and also extracted information with the help of NLP engine i.e. *Dialogflow* model. In this black box we design our algorithm to take decisions according to the intents and parameters generated by NLP engine.

We also integrated it with *Docvita Android* [16] [15] app so that any changes made can be reflected on the app. Right now we are in the stage of building inference engine. We are approaching this problem in following steps:

- We think of a use-case/task/feature that doctor will like DocVita to do.
- Design flow of the conversation as natural as possible.
- Then we add required phrases and contexts in Dialogflow NLP engine.
- Train the NLP engine and check the output.
- Process the generated output to extract desired information and give proper response to user to keep conversation going.

3.3 Text To Speech

The text to speech conversion was not a difficult part as *Google Text To Speech* was an efficient solution which is also very flexible since we were using the *Speech To Text* provided by google. Presence of existing APIs made it easy to convert textual output from our inference engine to speech. So we decided to invest our saved efforts in building a better inference engine and more features in our digital assistant.

Chapter 4 : FINALIZED DESIGN

The finalized design includes the frameworks that we are going to use for development of each black box and link them together:

• Speech To Text: We used Actions on Google for text to speech conversion in our final model. Which is invoked with the help of Google Assistant with the help of explicit invocation *Talk to Docvita* or explicit invocation with phrase for example *Ask Docvita test to prescribe 2 tablets of paracetamol for fever*. First will only invoke the Docvita Assistant while next one will take direct action on the phrase used.

• Text Processing (NLP): The text generated from the previous box will go to our own built NLP model on Dialogflow. Then this model will provide intent and parameters and send it to fulfillment endpoint(Firebase server) to process.

• Inference Engine: Intent and parameters generated by NLP engine are sent to server to process. This done by inference engine deployed on Firebase server. Inference engine is built on Node.js platform. Here all the logic resides regarding what action we have to take given specific intent and set of parameters. The size of Inference engine is directly proportional to the number of functionalities the assistant has. Then as a response Inference engine return proper text and set of contexts to keep the conversation going.

• Text to Speech: In the end the response sent by the Inference is converted to speech by Actions on Google itself and if it is not the end of the conversation then it again waits for more input or it terminates the session if it's the end of the conversation.

Refer to *Figure 4.1* for the design flow of the problem statement.

The logic to handle the entire process is discussed below:

• The first input is speech that a person says. The speech input is fed forward by the Google Home hardware to cloud.

• The processing is done in cloud where speech is converted to text.

• After speech is converted to text, it is fed in the NLP engine Dialogflow to generate tags and textual value against those tags.

• These pair of inputs are fed in our inference engine where the algorithm checks for desired values against the tags. If all parameters are present then accordingly a textual response is given as output from Inference Engine to Actions on Google.

• This textual information is converted to speech by Actions on Google and we receive an audio ouput from Google Home.

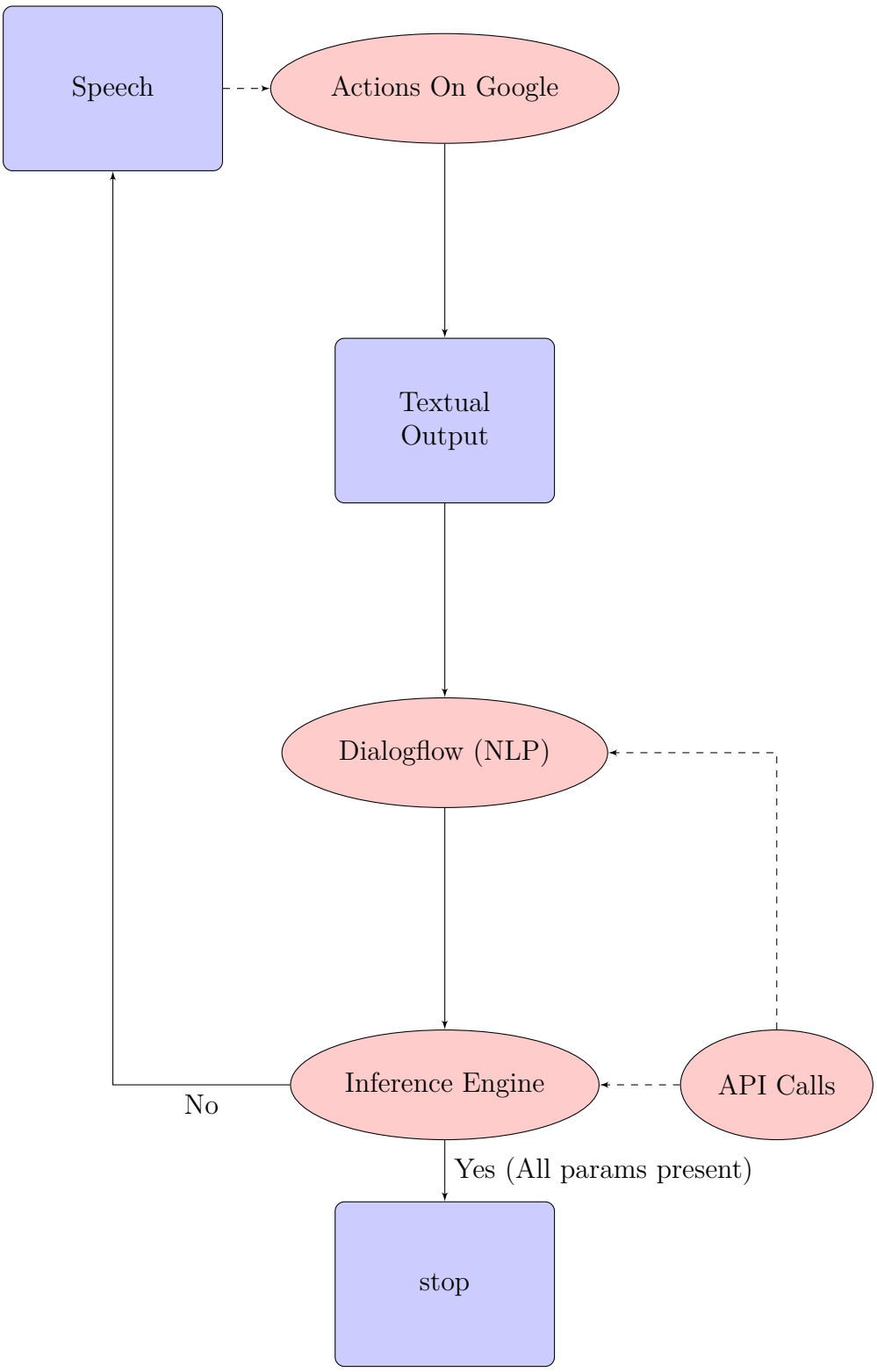


Figure 4.1: Design Flow

Chapter 5 : RESULT & ANALYSIS

All the results and conclusions from all the experiments we did throughout the project are compiled and structured in this chapter

5.0 Speech To Text

The online models that we tested are:

- Google API
- IBM Watson
- Microsoft BING
- Houndify

WORKING

The online available APIs take recorded audio in ".wav" format as input and produces the transcription of the audio in textual format as output. All the computation is done in the cloud space of respective APIs.

5.0.0 CMUSphinx (Open Source)

CMUSphinx [18] is one of the most popular software, provided by the Carnegie Mellon University in USA, for voice to text transcription.

WORKING

The working of CMUSphinx is discussed below:

- CMUSphinx is good in adapting to a particular voice, as a result it is actually an adaptation model.

• For adaptation purpose, we need recordings of the person and a pre-defined corpus which is the vocabulary in a particular context. This corpus is the dictionary which is nothing but all the unique words in a specific context.

• Dictionary consists of mapping of words to the phonemes, which is the smallest syllable in a word and defines how the word is pronounced.

• Finally, Sphinx requires Language model and Acoustic model. Language model keeps a stat of probability of words occuring in the spoken sentences using n-gram technique. Acoustic model is particular to a person. It consists of numerical representation of each phoneme. This representation in done using HMM (Hidden Markov Model).

• The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multidimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. It is only the outcome, not the state visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model.

5.0.1 Manual Models

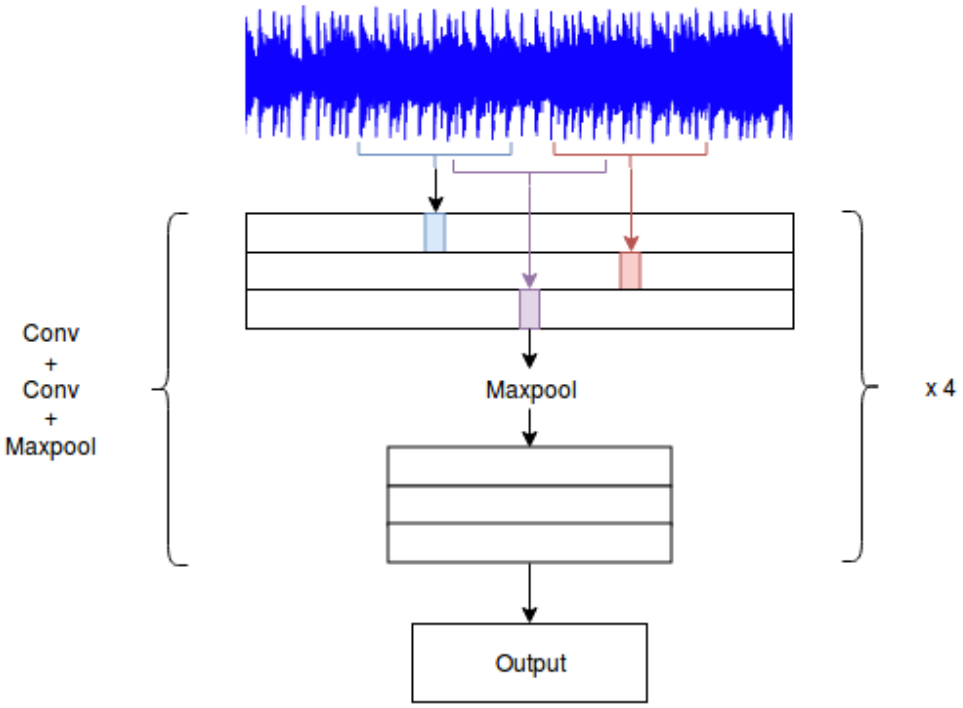
Some of the models that we built for manual speech to text transcription are:

- Convolutional Neural Network (CNN)
- Long Short Term Memory (LSTM)

WORKING OF CNN

The well-known application of CNN is image classification, where a fixed dimension image is fed into a network along with different channels (RGB in the case of a color image) and after various steps of convolution, pooling and fully connected layers, network outputs class probabilities for the image. We want to do the same, but here instead of an image, we have sound clips. A quick search on Google Scholar provide a lot of research papers, which discuss the implementation of CNN on a sound dataset. A paper I found particularly interesting and quite relevant is [20]. I borrowed the idea of dataset (feature extraction) preparation for CNN from this paper. For example: how to get equal size segments from varying length audio clips and which audio feature(s) we can feed as a separate channel (just like RGB of a color image) into the network. Once we have the initial dataset ready for the CNN. We can train as deep network (composed of different layers) as we want.

Figure 5.1: CNN for sound waves



As shown in *Figure 5.1*, the sound waves are pooled and the most prominent features are extracted from MFCC transformed sound waves.

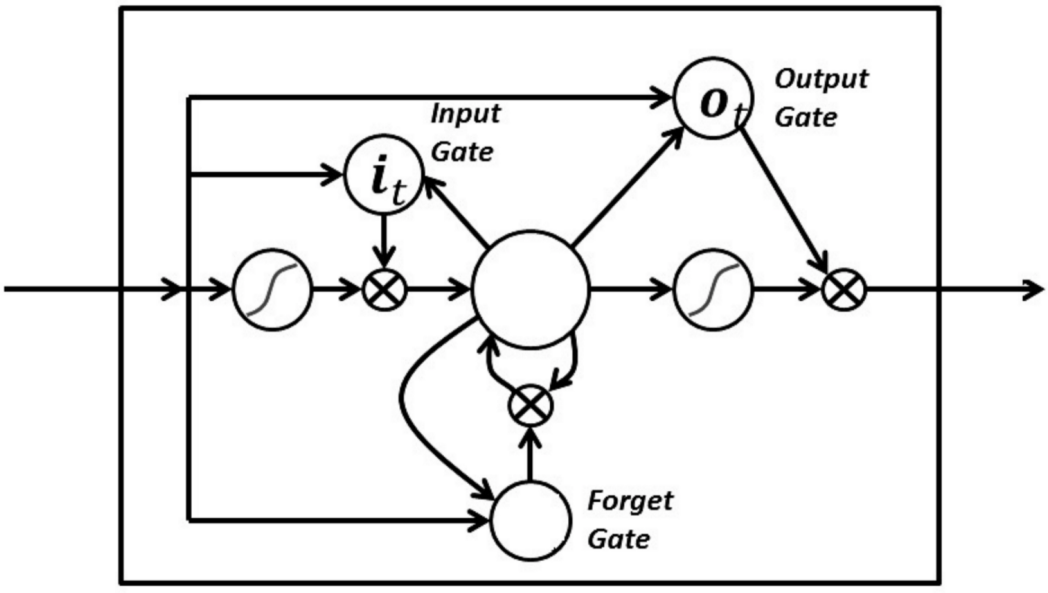
WORKING OF LSTM

Long Short-Term Memory (LSTM) networks are an extension for recurrent neural networks, which basically extends their memory. Therefore it is well suited to learn from important experiences that have very long time lags in between. The units of an LSTM are used as building units for the layers of a RNN, which is then often called an LSTM network.LSTM's enable RNN's to remember their inputs over a long period of time. This is because LSTM's contain their information in a memory, that is much like the memory of a computer because the LSTM can read, write and delete information from its memory. This memory can be seen as

a gated cell, where gated means that the cell decides whether or not to store or delete information (e.g if it opens the gates or not), based on the importance it assigns to the information. The assigning of importance happens through weights, which are also learned by the algorithm. This simply means that it learns over time which information is important and which not. In an LSTM there are three gates: input, forget and output gate. These gates determine whether or not to let new input in (input gate), delete the information because it isnâĂŹt important (forget gate) or to let it impact the output at the current time step (output gate). The gates in a LSTM are analog, in the form of sigmoids, meaning that they range from 0 to 1. The fact that they are analog, enables them to do backpropagation with it.

Refer to *Figure 5.2* for an overview of LSTM gates in a LSTM network.

Figure 5.2: LSTM gates



5.0.2 Actions on Google

Actions on google provide a platform for speech to text conversion with APIs to interact.

5.1 Analysis for Speech to Text models

The points that should be taken into consideration for speech to text models' analysis are:

Scalability:

• Manual models are scalable and can be deployed on cloud easily.

• CMUSphinx is difficult to be hosted on a cloud and run scripts to get textual output from an audio as an input. It includes overhead which affects the performance.

• Actions on Google is a ready to go solution available to us, as the entire model built by google is hosted on cloud which is easier to access and interact with.

Support & Maintenance:

• Manual models need lots of support and maintenance as they need to tuned manually depending upon the use-case and there is problem of redeployment of updated model.

• CMUSphinx is comparatively easier to tune but it still has an overhead of retraining and adding new words to vocabulary as CMUSphinx does not work for out of vocabulary words and also retraining the acoustic model and language model depending on the vocabulary. Similar as above, there is a problem of redeployment.

• Actions on Google needs very less support from admin side as the model tunes itself. There is no problem of redeploying the updated model.

Future Updates & Improvements:

• Manual models can cope with newer use cases but to train the model up to the mark for market deployment needs huge amount of data which was lacking in our case.

• CMUSphinx can also adapt to huge use cases but the overhead to train the model increases manifold.

• Actions on Google free tier provides limited use cases (valid until now for our case) but it also includes an enterprise edition for extension purpose.

Accuracy:

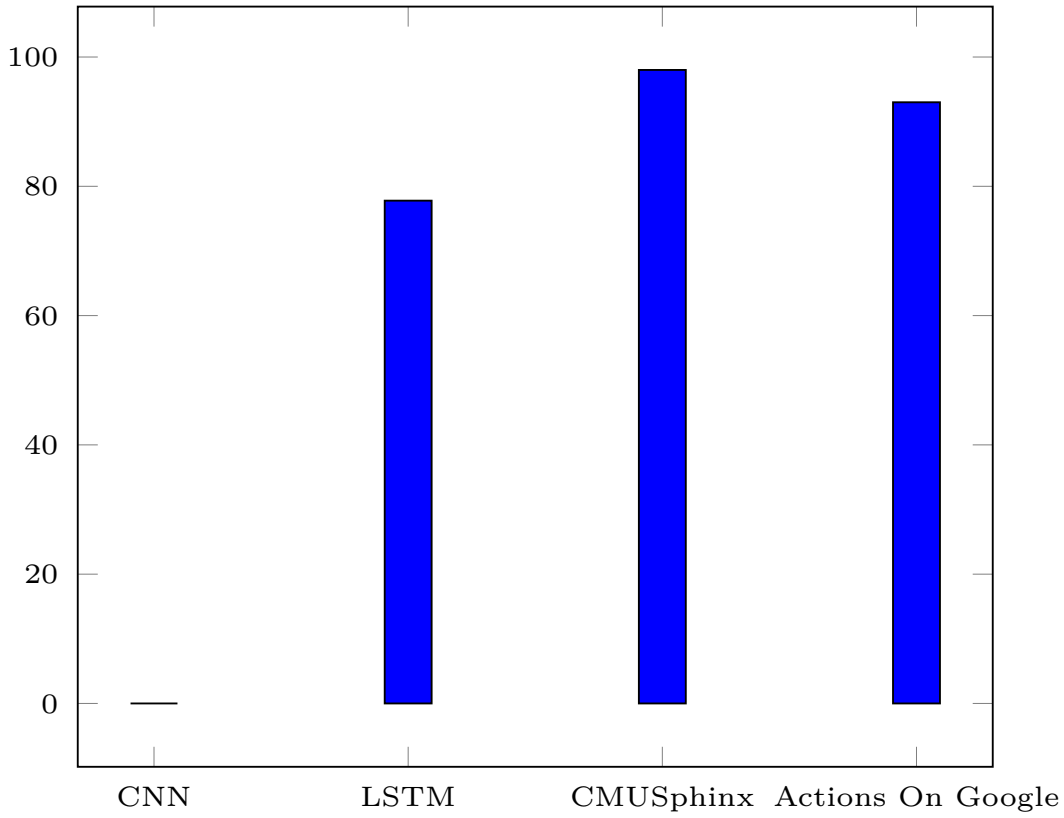


Figure 5.3: Accuracy comparision between different models

Refer to *Figure 5.3* for accuracy of different models that we have implemented.

5.2 Natural Language Processing (NLP)

The textual output is now needed to be processed for taking meaningful actions. We made our own NLP engine which is analogous to sequence to sequence classification in LSTM model.

How the output is interpreted and how the input data was created ?

We trained a LSTM which worked on sequence to one labelling technique i.e. we gave input a specific text and the model will give a single output which is actually the context of the sentence. Now the mapping of tags to the context was to be done such that we can build our inference engine. The previous LSTM model gave context tags as output which was input to this LSTM model for getting the correct contexts.

5.2.0 Inputs for LSTM

We used different methods for encoding words into meaningful format such that the sentence can be fed into the LSTM network.

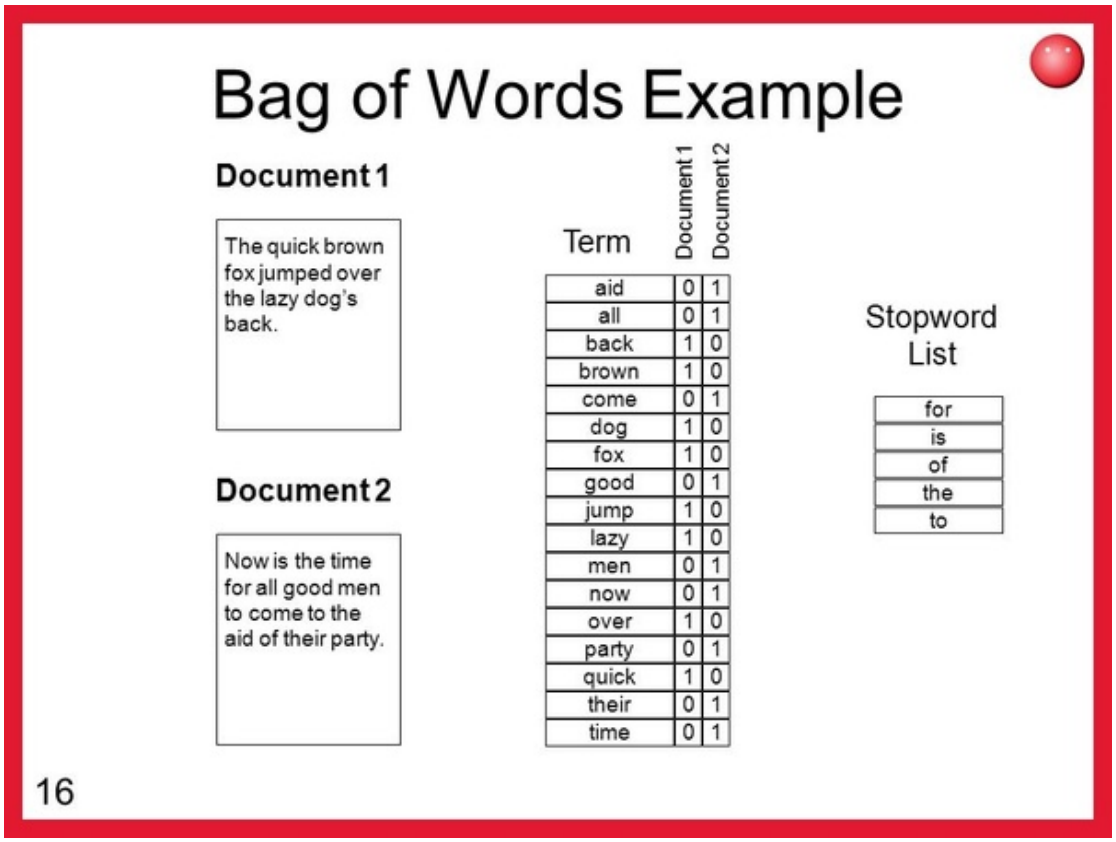
Bag Of Words (BOW)

• BOW model is a simple frequency based model which counts the number of words in a sentence after removing the stopwords and converts each sentence to an array of numbers (frequency) which is the size of vocabulary chosen.

• The drawback of BOW model is that, firstly, it places a limit on the vocabulary size and secondly, the ordering of words is lost from a sentence and there is also loss of information due to removal of stopwords.

Refer to *Figure 5.4* for a brief overview of how the model works.

Figure 5.4: BOW example



Word2Vec

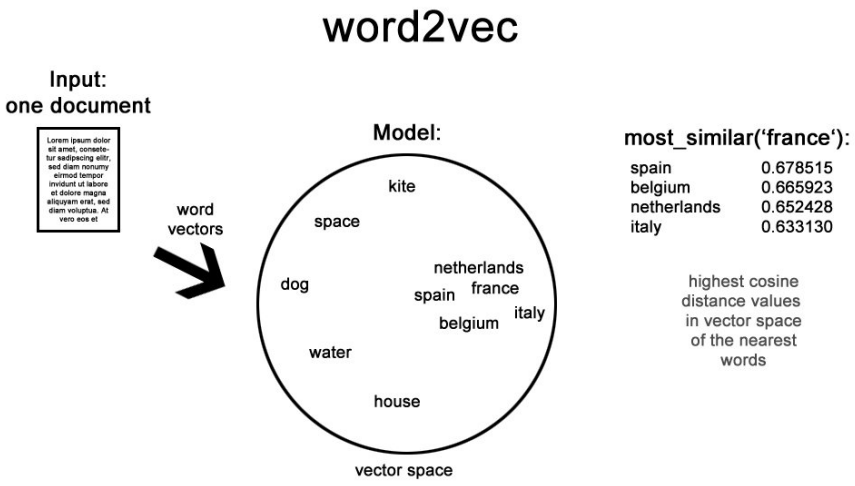
• Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.

• The drawback of Word2Vec model is that, it vectorizes words which is very useful but it does not vectorizes a sentence. The benifit of vectorizing a sentence is that 2 sentences of same meaning will have same vector representation in the

vector space which is of added benifit for getting information out of text.

Refer to *Figure 5.5* for a brief overview of how the model works.

Figure 5.5: Word2Vec Example



Doc2Vec

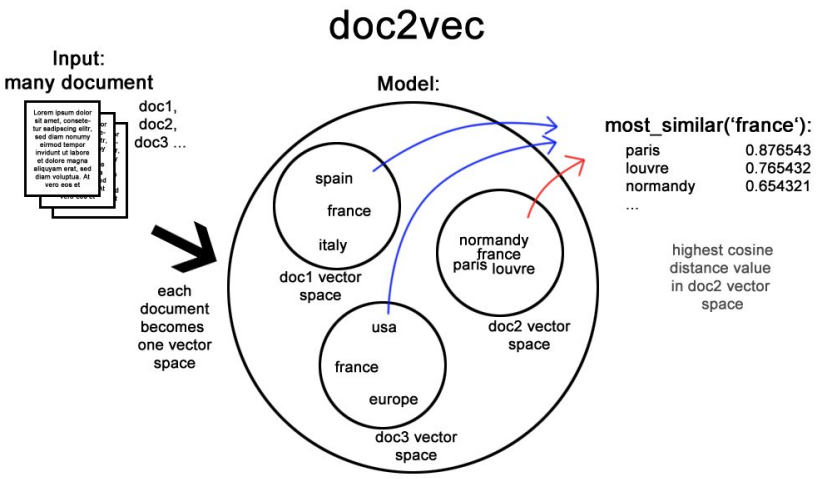
• Doc2Vec is an extension of Word2Vec and it also vectorizes each sentence as each sentence can act as a document unlike each word in case of Word2Vec.

Refer to *Figure 5.6* for a brief overview of how the model works.

5.2.1 CNN vs LSTM

We first tried CNN approach to classify our texts into classes, which is actually a multiclass classification. The problem with it is that, since there are many words, the output vector should be of length of the vocabulary which is actually not fixed. As a result, the problem of sparse matrix comes into play where most of the entries are 0 and only a few of them actually greater than 0, also we had a problem of vanishing gradient which is quite often in case of sigmoid and tanh functions,

Figure 5.6: Doc2Vec Example



hence, we had to discard the idea of CNN and we exploited LSTM model. The benefit of LSTM over CNN [19] is that the problem of vanishing gradient resolves because it uses RELU activation functions. Secondly, the problem of sparse matrix is also resolved though not fully, as the LSTM model works on sequence to sequence technique where for each input in a sequence there is an output. For eg. When translating one language to other language or generating subtitles. There are also limitations in LSTM model as the number of input and output sequence should be same. It can't be possible that if we input a sequence of length 10 the output sequence is of length 5. Hence, to make the output sequence of length 10 we need to pad the sequence with 5 more values. (generally 0)

5.2.2 Dialogflow

- Dialogflow is NLP engine provided by Google which has availability of API.
- Advantage of Dialogflow is that it is easily scalable and can be integrated with Actions on Google.
- Another advantage of Dialogflow is that it can be used to overcome the drawback of speech to text from google because of its too generic behaviour as out of context terms can be handled in NLP engine.

Refer to *Figure 5.7* for approximate accuracies of LSTM & Dialoglow.

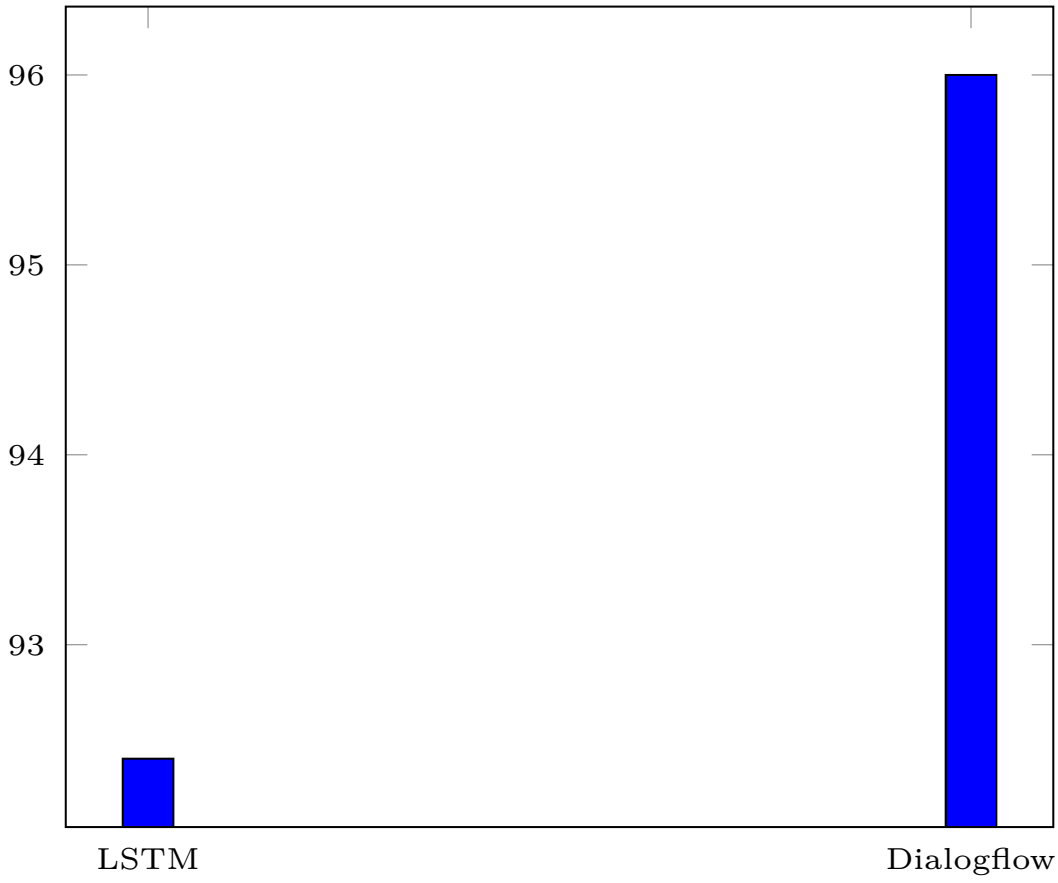


Figure 5.7: Accuracy plot between Dialogflow & LSTM

Chapter 6 : CONCLUSIONS & SCOPE FOR FUTURE WORK

For the time being, because of lack of data, there is a tradeoff between complexity, accuracy and scalability. Avalability of APIs made it easy to build the minimum viable product for deployment in the market. Avalability of data will make manual models also scalable and future proof.

The future scope of our word includes data collection for efficient model training. API building for interaction with deployed manual model and Optimizing NLP model for better accuracy.

Bibliography

[1] Developed By Amazon. Amazon Alexa. https://developer.amazon.com/alexa, 2014. [Smart assistant by Amazon].

[2] Developed By Apple. Siri. https://www.apple.com/siri/, 2010. [Smart assistant by Apple].

[3] FranÃğois Chollet. keras. https://github.com/fchollet/keras, 2015.

[4] Actions On Google. OAuth 2.0 Implicit Flow. https://developers.google.com/actions/identity/oauth2, 2012. [Authentication flow by Google].

[5] Developed By Google. Google Assistant. https://store.google.com/product/google_home, 2016. [Smart assistant by Google].

[6] Developed By Google. Dialogflow. https://dialogflow.com/docs/getting-started, 2017.

[7] Developed By Google. Angular 6. https://angular.io/, 2018. [Framework for webapps].

[8] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[9] Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. Bag-of-words as target for neural machine translation. *CoRR*, abs/1805.04871, 2018.

[10] Developed By Microsoft. Cortana. https://www.microsoft.com/en-in/windows/cortana, 2013. [Smart assistant by Microsoft].

[11] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[12] Lindasalwa Muda, Mumtaj Begam, and I. Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques. *CoRR*, abs/1003.4083, 2010.

[13] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[14] Daniel Povey et al. Kaldi ASR. `http://kaldi-asr.org/`, 2013. [Offline speech recognition (open source)].

[15] Developed By Chota Hospital Pvt.Ltd. Docvita App. `https://play.google.com/store/apps/details?id=in.chotahospital.docvita&hl=en_IN`, 2018. [Docvita App for patients].

[16] Developed By Chota Hospital Pvt.Ltd. DocvitaMD App. `https://play.google.com/store/apps/details?id=in.chotahospital.docvitamd`, 2018. [DocvitaMD App for doctors].

[17] Developed By Samsung. Bixby. `https://www.samsung.com/global/galaxy/apps/bixby/`, 2017. [Smart assistant by Samsung].

[18] Carnegie Mellon University. CMUSphinx. `https://cmusphinx.github.io/wiki/tutorial/`, 2015. [Offline speech recognition system by CMU].

[19] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of CNN and RNN for natural language processing. *CoRR*, abs/1702.01923, 2017.

[20] Boqing Zhu, Kele Xu, Dezhi Wang, Lilun Zhang, Bo Li, and Yuxing Peng. Environmental sound classification based on multi-temporal resolution CNN network combining with multi-level features. *CoRR*, abs/1805.09752, 2018.

Appendix A : LSTM & Dialogflow

Dialogflow and LSTM are very much similar in our approach:

• In Dialogflow the text is the input and after processing the text, Dialogflow generates a stream of tokens (parameters).

• This stream of tokens are again fed into the model which further determines the intent in which the text is fed as input.

• Dialogflow has a plus point that when determining the intent along with the stream of tokens we have input and output contexts.

• Basically, Dialogflow does a sequence to one classification.