

A PROJECT REPORT ON
AN INTELLIGENT VIRTUAL ASSISTANT USING DEEP
LEARNING

*Mini project submitted in partial fulfillment of the requirements for the
award of the degree of*

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY
(2018-2022)

BY

K. Pavan Kalyan	18245A1217
Umakanth Sahu	18241A1259
P. V. Asrith Reddy	18241A1246
K. Ajay Goud	18241A1224

Under the Esteemed guidance of
K. Sandeep

Asst Prof, Dept of IT



DEPARTMENT OF INFORMATION TECHNOLOGY
GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)
HYDERABAD



CERTIFICATE

This is to certify that it is a bonafide record of Mini Project work entitled “**An Intelligent Virtual Assistant using Deep Learning**” done by **K. Pavan Kalyan (18245A1217), Umakanth Sahu (18241A1259), P.V. Asrith Reddy (18241A1247), K. Ajay Goud (18241A1224)**, students of **B.Tech (IT)** in the Department of Information Technology, Gokaraju Rangaraju Institute of Engineering and Technology during the period 2016-2020 in the partial fulfillment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from GRIET, Hyderabad.

K. Sandeep
(Internal Project Guide)

Dr. K. Prasanna Lakshmi
(Head of the Department)

(Project External)

ACKNOWLEDGEMENT

We take the immense pleasure in expressing gratitude to our Internal guide **K. Sandeep, Asst Prof**, Information Technology, GRIET. We express our sincere thanks for his encouragement, suggestions, and support, which provided the impetus and paved the way for the successful completion of the project work.

We wish to express our gratitude to **Dr. K. Prasanna Lakshmi, P. Gopala Krishna**, our Project Co-coordinators **K. Archana** and **K. Swanthana**, for their constant support during the project.

We express our sincere thanks to **Dr. Jandhyala N Murthy**, Director, GRIET, and **Dr. J. Praveen**, Principal, GRIET, for providing us the conducive environment for carrying through our academic schedules and project with ease.

We also take this opportunity to convey our sincere thanks to the teaching and non-teaching staff of GRIET College, Hyderabad.



Name: K. Pavan Kalyan
Email: kpavankalyan8262@gmail.com
Contact No: 9505634625
Address: Zaheerabad, Sangareddy



Name: Umakanth Sahu
Email: umaksahu@gmail.com
Contact No: 9059866717
Address: Nizampet, Hyderabad



Name: P. V. Asrith Reddy
Email: asrith2000@gmail.com
Contact No: 7093529500
Address: Ramamurthy Nagar, Nellore.



Name: K. Ajay Goud
Email: Ajaykotha18@gmail.com
Contact No: 6302009608
Address: Manikonda, Hyderabad

DECLARATION

This is to certify that the project entitled “**An Intelligent Virtual Assistant using Deep Learning**” is a bonafide work done by us in partial fulfillment of the requirements for the award of the degree **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad.

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites, books and paper publications are mentioned in the Bibliography.

This work was not submitted earlier at any other University or Institute for the award of any degree.

<i>K. Pavan Kalyan</i>	<i>18245A1217</i>
<i>Umakanth Sahu</i>	<i>18241A1259</i>
<i>P. V. Asrith Reddy</i>	<i>18241A1246</i>
<i>K. Ajay Goud</i>	<i>18241A1224</i>

TABLE OF CONTENTS

Serial no	Name	Page no
	Certificates	ii
	Contents	v
	Abstract	viii
1	INTRODUCTION	1
1.1	Introduction to Project	1
1.2	Existing System	2
1.3	Proposed System	2
2	REQUIREMENT ENGINEERING	3
2.1	Hardware Requirements	3
2.2	Software Requirements	3
3	LITERATURE SURVEY	4
4	TECHNOLOGY	6
5	DESIGN REQUIREMENT ENGINEERING	11
5.1	Use Case Diagram	11
5.2	Class Diagram	12
5.3	Sequence Diagram	13
5.4	Deployment Diagram	14
5.5	Architecture	15

Serial no	Name	Page no
6	IMPLEMENTATION	16
6.1	Modules	16
6.1.1	Dataset	16
6.1.2	NLTK Utilities	17
6.1.3	Neural Network Model	18
6.1.4	Train Module	19
6.2	DOST GUI	24
7	SOFTWARE TESTING	25
7.1	Unit Testing	25
7.2	Integration Testing	26
7.3	Acceptance Testing	26
8	RESULTS	27
9	CONCLUSION AND FUTURE ENHANCEMENTS	31
10	BIBLIOGRAPHY	32

11. LIST OF FIGURES

S No	Figure Name	Page no
11.1	Use Case Diagram	11
11.2	Class Diagram	12
11.3	Sequence Diagram	13
11.4	Deployment Diagram	14
11.5	Architecture	15

ABSTRACT

Virtual assistants can recognize human language and commands which then result in the performing various tasks. Virtual assistants usually perform simple tasks for users, such as adding remembering items; providing information to be searched in a web browser; Calendar tasks etc.

The aim of the project is to create an Intelligent Virtual Personal Assistant (VPA) with a focus on user-based information. Our VPA will enhance the user interaction and provides better output to the input by using “Smart” Neural Network. This Virtual Assistant will be an advancement to **Cortana**, an assistant in Windows Operating System. It can outperform various tasks like opening major applications such as Google Chrome, Microsoft Edge, Microsoft Word etc., opening few important websites like Google, YouTube etc. just through your voice commands which are currently not supported by Cortana. Though these types of models are prevalent, our view is to work on this model and make this model generative in the future.

1. INTRODUCTION

1.1 Introduction to Project

Deep Learning is a subset of Machine Learning which is usually a neural network with three or more layers. It is also called as Hierarchical Learning or Deep-structured learning. This learning method uses the idea of ANN (Artificial Neural Network) and can consume large amounts of data by deepening the networks in specific way similar to human brain. Through this deeper network, the model gains the capability of extracting features from raw data and learn about the features in each consecutive layer. Even a single layered neural network can make approximate predictions but adding few hidden layers can increase the accuracy and optimize the Neural Network.

Deep Learning drives many services to improve automation and perform various tasks without human. There are many applications of Deep Learning in current world. Some of them are:

- Self-driving Cars
- Digital Assistants
- Voice enable TV remotes.
- Credit fraud detection etc.

Deep learning neural networks uses a combination of data inputs, weights, and bias. These elements work together in order to classify and recognize objects within the data. This series of computations through the network is called forward propagation. The two layers- input and output, of the deep neural network are called visible layers of the model. The input layer is used for feeding the input data for processing while the output layer gives the final classification.

There is another process called back propagation which uses algorithm to calculate errors in the predictions made and then adjusts the weights and biases of the function by moving in backward direction. Using both forward and backward propagation makes the model training more accurate and correct the errors subsequently.

In our project, we have used Deep Learning concepts in order to train the model using the datasets, which contains patterns and their corresponding classes. Now, whenever user gives a command, we classify the user sentence and perform the necessary action.

1.2 Existing System:

Current existing Virtual Assistants are Google Assistant, Alexa, Siri, and Cortana. Google Assistant works in android and other assistants are too specific to the underlying architecture. When we come to Cortana which is default Virtual Assistant to Windows, it cannot perform many tasks like opening applications and websites as of now. Although it is default virtual assistant, it is lacking with most of the features.

1.3 Proposed System

Our proposed model will be a hybrid model combination of Virtual Assistant and a Chatbot. We will be using NLTK library which stands for Natural Language Tool Kit as the name suggests it analyses the given input sentence and converts them into a form which is more likely to be used in Deep learning to build Neural Networks using PyTorch library. It can perform the abovementioned tasks such as opening applications, websites, obtaining current weather easily. Our project is a GUI- based application which helps the user to easily interact with the Application.

2. REQUIREMENT ENGINEERING

2.1 Hardware Requirements

- Processor – Intel Pentium (64-bit OS).
- Memory – 4GB RAM

2.2 Software Requirements

- Windows 10
- Visual Studio Code
- Python
- PyTorch
- Win32Com
- Tkinter
- PyInstaller

3. LITERATURE SURVEY

The evolution of these machines started with the question “I purpose to consider question, can machines think?” by Alan Turing, a computer scientist. He is the one who introduced Turing test which is still used to test the ability of a machine in fooling a human by masking itself as a human.

ELIZA is one of the first chatbots created by Joseph Weizenbaum in late 1960’s but it failed in Turing test, but it did break the ground with new ideas like keywords and programmed responses.

After a couple of years later in 1972, PARRY was created by Kenneth Colby. In Turing test with humans as psychiatrists only 48% told the difference between a human and Parry. Parry used a system of “emotional responses” with varying weights assigned to verbal inputs it stimulated a paranoid human. Colby is the first psychiatrist who drives the chatbots towards the field of mental illness and research.

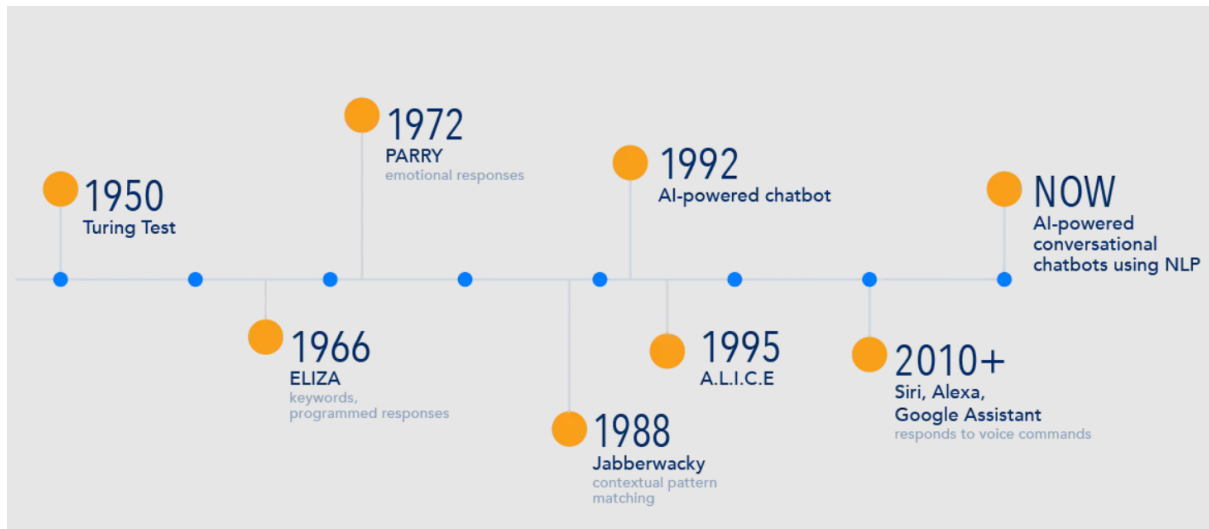
In the late 1980’s a chatbot named Jabberwacky was created by Rollo Carpenter whose purpose is to have fun with human in conversations. Contextual pattern matching an AI technique has led to further use of academic research.

Creative labs created a chatbot named Dr. Sbaitso for the sake of MS-DOS in the starting of 1990’s. It is said to be the first chatbot that used Artificial Intelligence and included a complete voice-operated chat program method. The main aim of this is to show the machine as a psychologist with many responses and questions like,” what is the reason for that feeling?”

A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) was created by Richard Wallace in 1990’s. This is a universal language chatbot which utilized heuristic pattern matching. The feature of chatting with someone online is also included in it. It was designed to look like a young woman and to tell interesting facts about herself. XML schema artificial markup language (AIML) was used by this chatbot.

It was after 2001, we saw the predecessor of Siri and other applications with SmarterChild. In the span of 2010 and 2015 Siri, Google Assistant, Alexa, and Cortana began to take over the chatbots which can perform online searches, can respond to voice commands, and can do many multimedia activities.

From these what we learnt is most of the machines developed in start of the 21st century is mobile based application and all of them pretty good in that field, but we come to computers and when we come to windows, we have Cortana which have so many backdrops i.e., it cannot do many actions which are performed by Siri and google assistant our aim is to overcome those...



4. TECHNOLOGY

4.1 ABOUT PYTHON

Python is an interpreted high-level general-purpose programming language. This language has been created by Guido van Rossum in 1991. Its language constructs furthermore as its object-oriented approach aim to assist programmers write clear, logical code for little and large-scale projects. Python could be a widely used general-purpose, high level artificial language. Python is dynamically typed, and garbage collected. It supports multiple programming paradigms- procedural, object-oriented, and functional programming.

1. **Readable:** Python may be a very readable language.
2. **Easy to Learn:** Learning python is straightforward as this is often an expressive and high-level artificial language, which implies it's easy to grasp the language and thus easy to find out.
3. **Cross platform:** Python is offered and may run on various operating systems like Mac, Windows, Linux, Unix etc.
4. **Open Source:** Python may be an open-source programming language.
5. **Large standard library:** Python comes with an oversized standard library that has some handy codes and functions which we will use while writing code in Python.
6. **Free:** Python is liberal to download and use.
7. **Supports exception handling:** Python supports exception handling which implies we are able to write less error prone code and might test various scenarios which will cause an exception.
8. **Advanced features:** Supports generators and list comprehensions. we are going to cover these features later.
9. **Automatic memory management:** Python supports automatic memory management which implies the memory is cleared and freed automatically. you are doing not must bother clearing the memory.

4.2 WHAT CAN YOU DO WITH PYTHON

There are many applications of Python, here are few of them.

1. **Web development** – We create many Dynamic websites using Python. Flask and Django are famous web frameworks based on Python. By these frameworks we can write server-side code which helps in managing database, write business logic etc.
2. **Machine learning** Machine learning is a technique where a machine can learn and solve a particular problem on its own. Some applications of machine learning are Face recognition, Voice recognition, Stock market Prediction etc.
3. **Data Analysis** – Python is also used for Data analysis and data visualization in form of various charts.
4. **Scripting** – We can use Python for Scripting, which is writing small programs in order to automate simple tasks. E.g., Automating Sending an Email etc.
5. **Game development** – We can also develop games using python.
6. **Embedded Applications:** We can use python for Embedded Systems such as Raspberry Pi and make an IOT world.
7. **Desktop applications** – we can create desktop application using Tkinter or QT libraries.

4.3 A BRIEF OF THE LIBRARIES WE USED FOR THIS PROJECT

4.3.1 NUMPY

NumPy is a package for fundamental computing in Python. It provides multidimensional array object and various derived objects (such as matrices and masked arrays) and assortment of routines for fast operations on arrays which includes logical, mathematical, sorting, selecting, I/O, shape manipulation, discrete Fourier transforms, basic linear algebra, random simulation and much more.

4.3.2 PYTORCH

PyTorch facilitates building deep learning projects which is a Python library. Python is liked by all because it is easy to read and understand. PyTorch allows deep learning models to be expressed in idiomatic Python and emphasizes flexibility.

To be simple, it is like a NumPy but with strong GPU acceleration. Also, dynamic computation graphs are supported by PyTorch which allows you to change how the network behaves on fly, it is not like static graphs which are used in frameworks such as TensorFlow.

4.3.3 PYINSTALLER

PyInstaller is a python library that ties a Python application and all of its dependencies together into a single package. The user can directly run the app without installing Python and other dependencies. PyInstaller reads the complete python script written by you, analyses your code to discover all the modules required to be included in order to execute your program. Then it creates a copy of all the files required and bundles them into one single folder or file. PyInstaller also has option to include the console or just make it windowed application.

4.3.4 TKINTER

Python is used for developing GUI (Graphical User Interface). Out of all GUI methods, Tkinter is most popular and is by default shipped with python. Tkinter is very easy to learn and there are lot of tutorials for it. It provides an object-oriented interface to build GUI Application. GUI makes the application user-friendly than interacting with the CLI (Command Line Interface).

4.3.5 NLP AND NLTK

4.3.5.1 NLP

Natural Language Processing (NLP) is a process of understanding or manipulation of speech and text by any machine or software. In general humans interact with each other and understand their views but in NLP this is done by a computer rather than a human.

4.3.5.2 NLTK

NLTK (Natural Language Toolkit) is a bundle that contains programs and libraries for statistical language processing. NLTK is said to be one of the most powerful libraries of NLP. NLTK contain packaged that helps to interact with humans and give responses.

4.3.6 JSON

Json (JavaScript Object Notation) is an independent data format which is used for transmitting data between a server and a web application. These files are human-readable, lightweight, and can be easily edited using a text editor. Also, it is easy for machines to parse and generate this Json files. This notation uses elements such as Objects, Object members, arrays, values, and strings.

4.3.7 SPEECH RECOGNITION

It is very helpful for those users who are unable to see the GUI and to use the screens like others, it helps them to reach the state-of-the-art services and products. It is extremely flexible and acts as a wrapper for several popular speech APIs. The Web Speech API used by Google supports a default API key that is hard coded into the Speech Recognition library. It helps to get off without having a sign up for a device. All these makes it an excellent choice for a Python project.

4.3.8 WIN32COM

Using win32com module we will get a reference to the MS Office applications such as Word, Power point, Excel, Outlook etc. This library can also be used for re-opening already documents. It contains many useful functions that can automate heavy tasks using python very easily.

4.4 NEURAL NETWORKS

Neural networks (NN) are also called artificial neural networks (ANN). They are a subset of learning algorithms within the machine learning field. They are loosely based on the concept of biological neural networks like our human brain neural network.

“Neural networks are revolutionizing machine learning because they are capable of efficiently modeling sophisticated abstractions across an extensive range of disciplines and industries.”

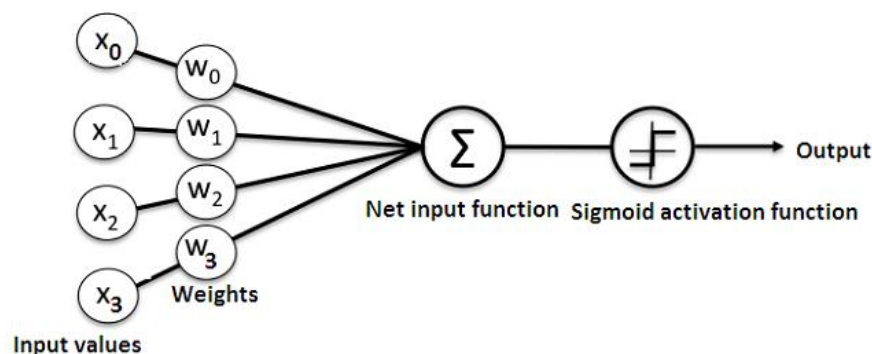
-Andrey Bulezyuk, German-based machine learning specialist.

Usually, an ANN constitutes of the following components:

- An Input layer to receive data.
- A hidden layer
- An output layer that gives the final classification of the data that is passed to it.
- Weights between the layers
- Each hidden layer is assigned a deliberate activation function. some of those functions are RELU function, Sigmoid activation function and etc.

We have so many types of neural networks but in this project, we are going to construct perception or feed forward neural networks which relays on data directly from front to back.

We often need back propagation to train the feed forward neurons, which gives the input with respective set of inputs and outputs. When the neuron receives the input, it is processed, and output is generated.



5. DESIGN REQUIREMENT ENGINEERING

5.1 Use case Diagram

In Unified Modeling Language, a use case diagram is a type of behavioral diagram. The main purpose of it is to show the graphical representation of the functionality of the system in terms of primary and secondary actors, their goals, and dependencies between them. It is used to show which system function is performed for which actor. Roles of the actors in the system can also be depicted.

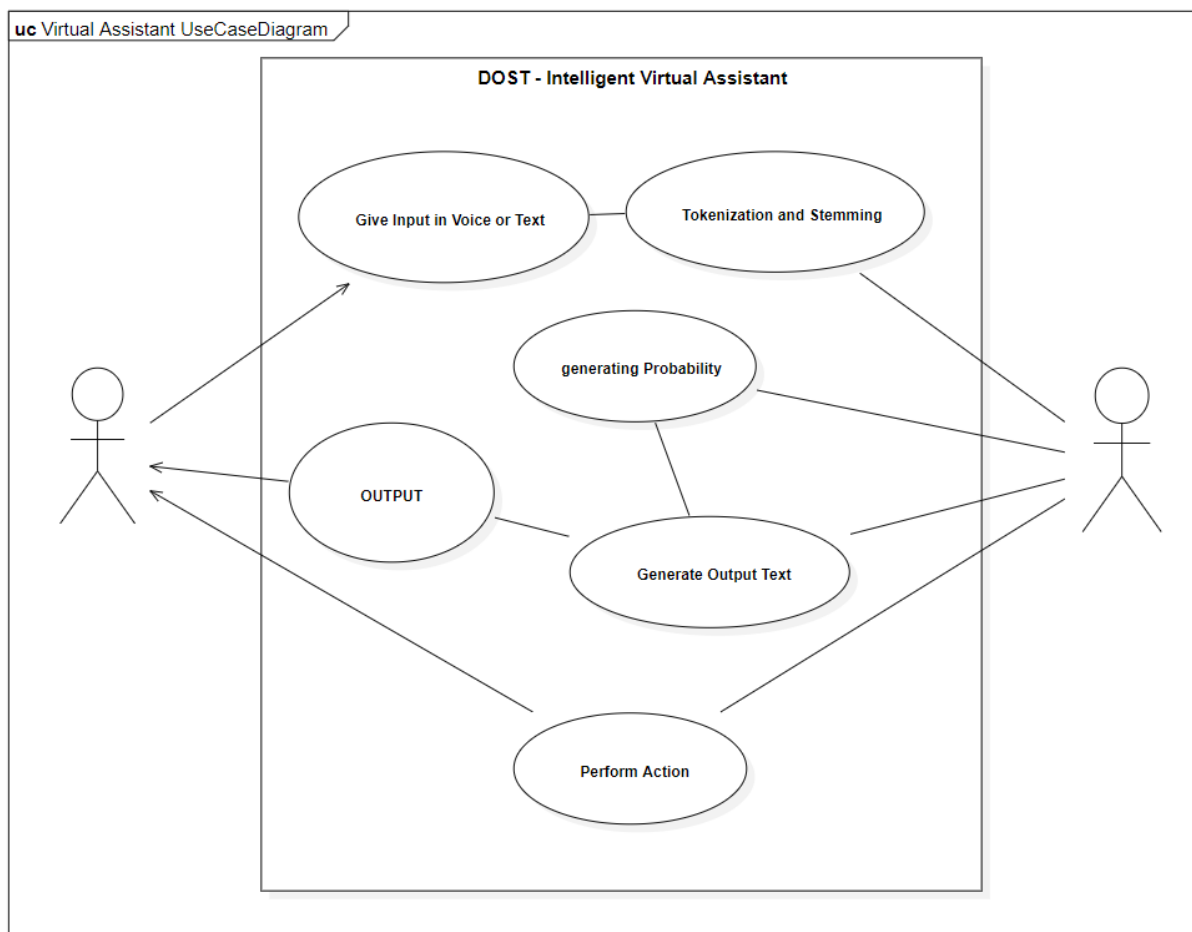


Fig 5.1: Use Case Diagram

5.2 Class Diagram

In Unified Modeling Language, a class diagram is a static structure diagram that shows the structure of the system with the help of classes, attributes, and operations(methods) present in the class, which class contains information is explained with the help of relationship among classes.

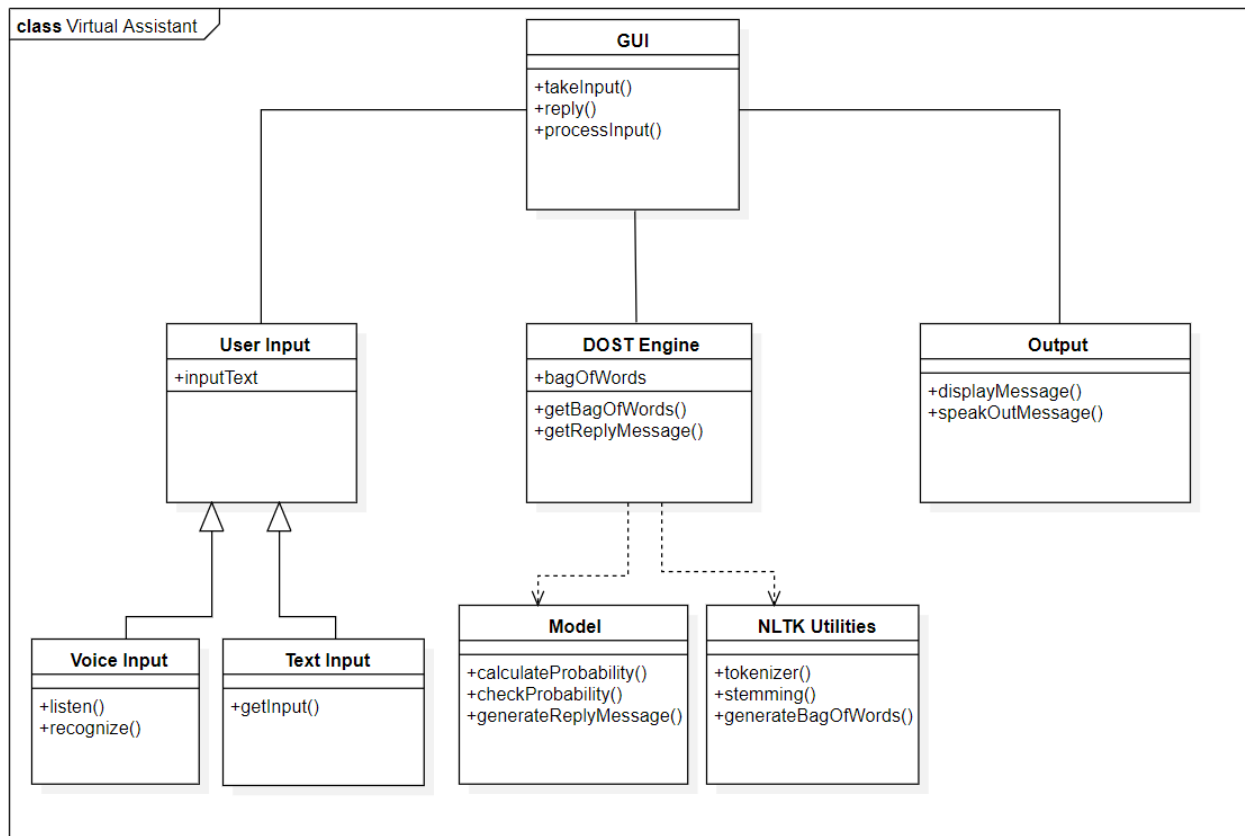


Fig 5.2: Class Diagram

5.3 Sequence Diagram

A sequence diagram is one of the interaction diagrams which shows how each process of the system operates with one another and in what order. It represents a timeline which begins at the top and descends gradually to mark the time sequence of the interactions. These diagrams are also called Event diagrams or Timing diagrams.

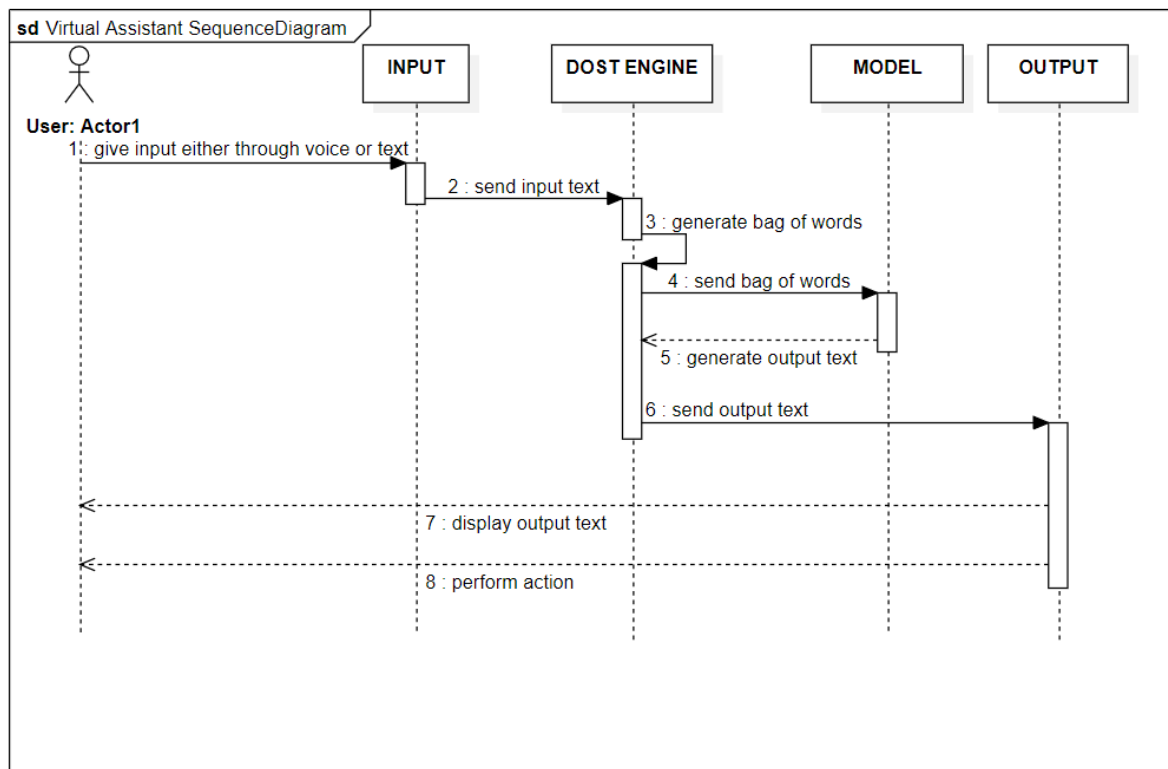


Fig 5.3: Sequence Diagram

5.4 Deployment Diagram

The deployment view of a system is represented by Deployment diagram which is related to the component diagram because the components are deployed using the deployment diagrams. A deployment diagram consists of Nodes which represents the physical hardware's of the system.

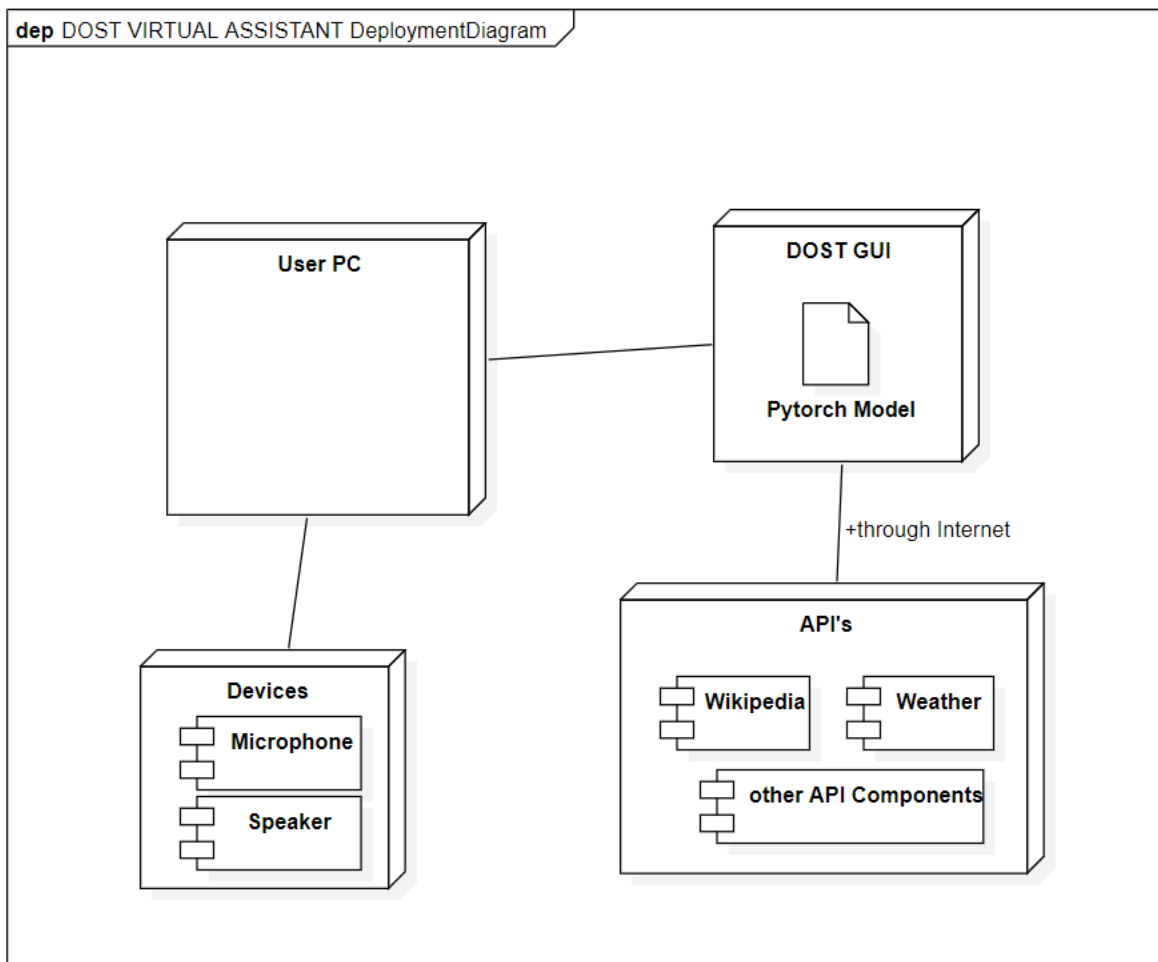


Fig 5.5: Deployment Diagram

5.5 Architecture

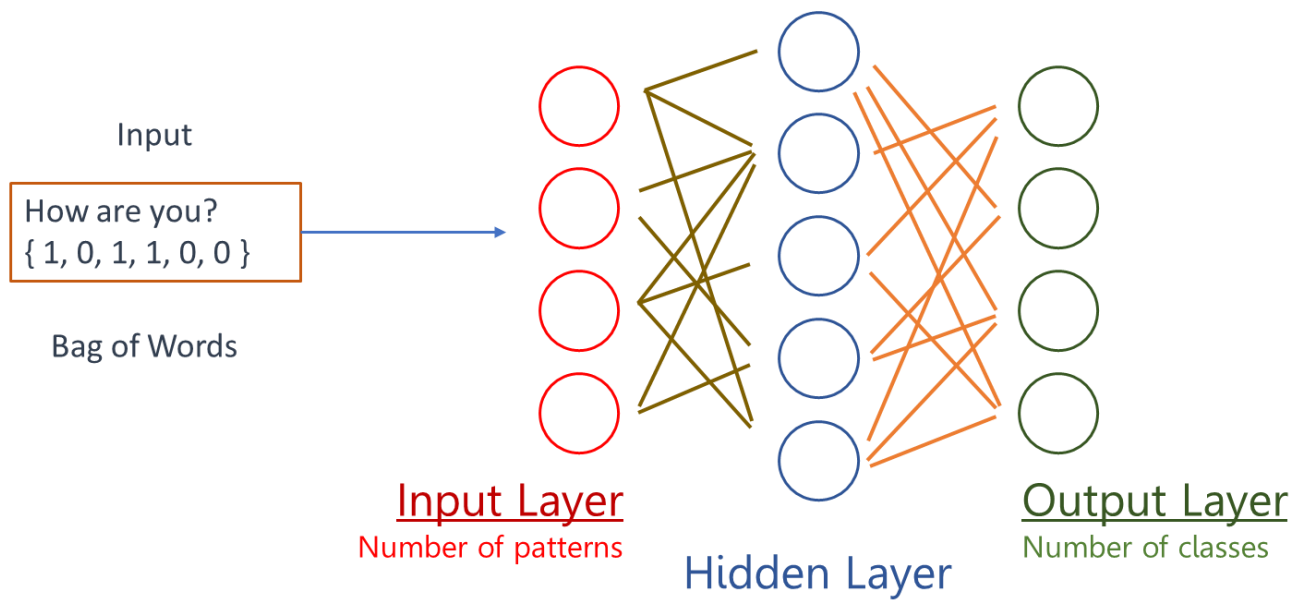


Fig 5.5.1: 3-Layered Model Architecture Diagram

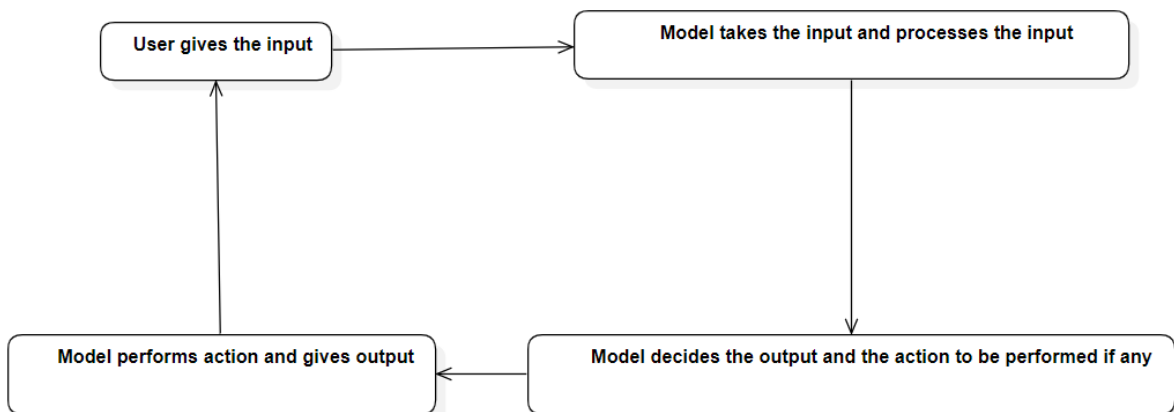


Fig 5.5.2: Block Diagram

6. IMPLEMENTATION

6.1 Modules

6.1.1 Dataset

The following figure shows the snapshot of the intents.json file which is used as the dataset for this project.

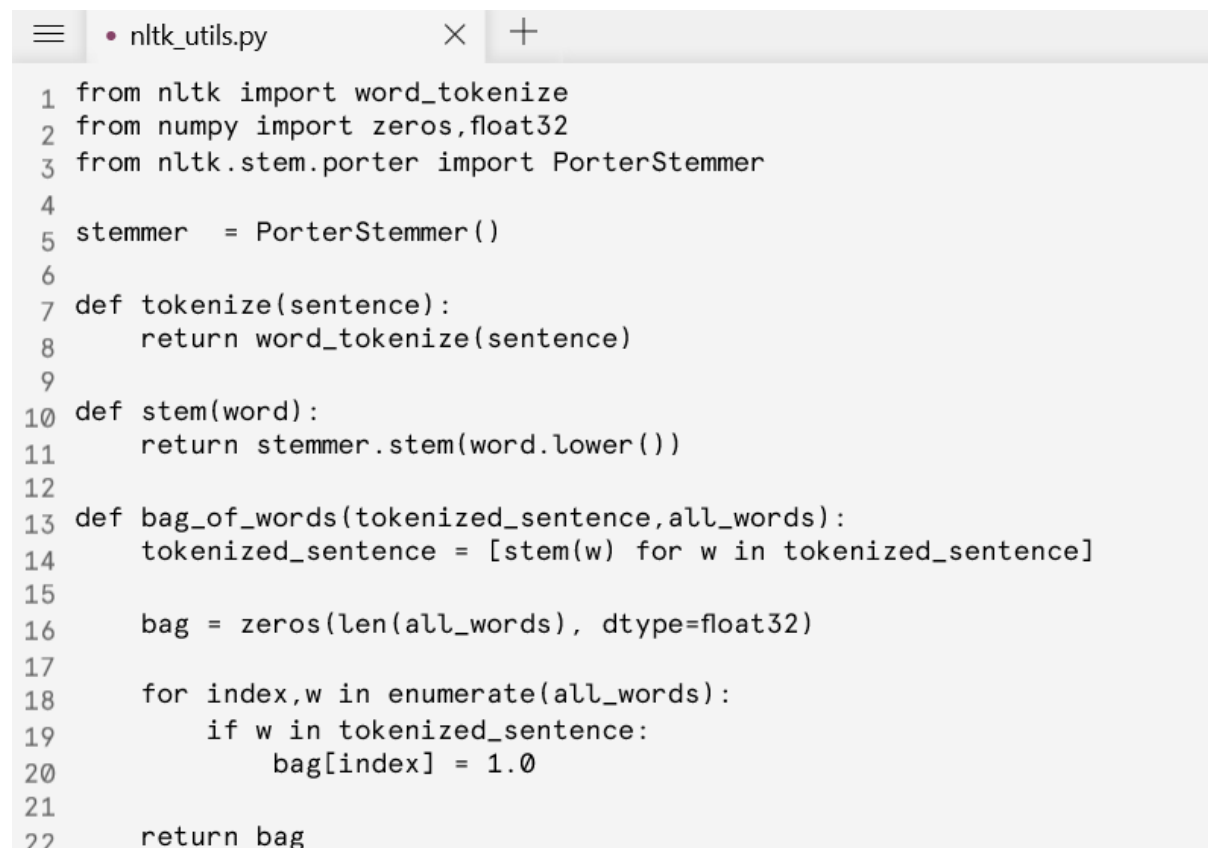


```
1 {
2   "intents": [
3     {
4       "tag": "greeting",
5       "patterns": [
6         "Hi", "Hey", "How are you", "Hello", "Good day"
7       ],
8       "responses": [
9         "Hey :-)", "Hi there, what can I do for you?", "Hi there, how can I help?", "Hello"
10      ]
11    },
12    {
13      "tag": "youtube",
14      "patterns": [
15        "open youtube", "open youtube in browser", "youtube", "youtube.com", "open youtube.com", "open yt", "yt"
16      ],
17      "responses": [
18        "A: webbrowser.open('www.youtube.com')"
19      ]
20    },
21    {
22      "tag": "facebook",
23      "patterns": [
24        "open facebook", "open facebook in browser", "facebook", "facebook.com", "open facebook.com", "open
fb", "fb"
25      ],
26      "responses": [
27        "A: webbrowser.open('www.facebook.com')"
28      ]
29    }
30  ]
31 }
```

Fig 6.1.1.1: Dataset

These intents.json file contains all the actions and the responses that are supported by our application. Adding new action or responses can be easily done just by adding the category and training the data.

6.1.2 NLTK Utilities



```
1 from nltk import word_tokenize
2 from numpy import zeros, float32
3 from nltk.stem.porter import PorterStemmer
4
5 stemmer = PorterStemmer()
6
7 def tokenize(sentence):
8     return word_tokenize(sentence)
9
10 def stem(word):
11     return stemmer.stem(word.lower())
12
13 def bag_of_words(tokenized_sentence, all_words):
14     tokenized_sentence = [stem(w) for w in tokenized_sentence]
15
16     bag = zeros(len(all_words), dtype=float32)
17
18     for index, w in enumerate(all_words):
19         if w in tokenized_sentence:
20             bag[index] = 1.0
21
22     return bag
```

Fig 6.1.2.1: nltk_utility.py

This module contains few functions that are used in other files frequently.

Porter Stemmer is a class which provides many functions to find the root of words after removing verb and tense part from given sentence. We are creating an instance of Porter Stemmer class named stemmer.

Tokenize function: Given input variable called sentence, this function divides the sentence into words and return it.

E.g.: when we pass “**Hi, how are you?**” the output is [**'Hi', ',', 'How', 'are', 'you', '?'**].

Stem function: This function takes input word and returns the root word form.

E.g.: root word of “Organize” is “Organ”.

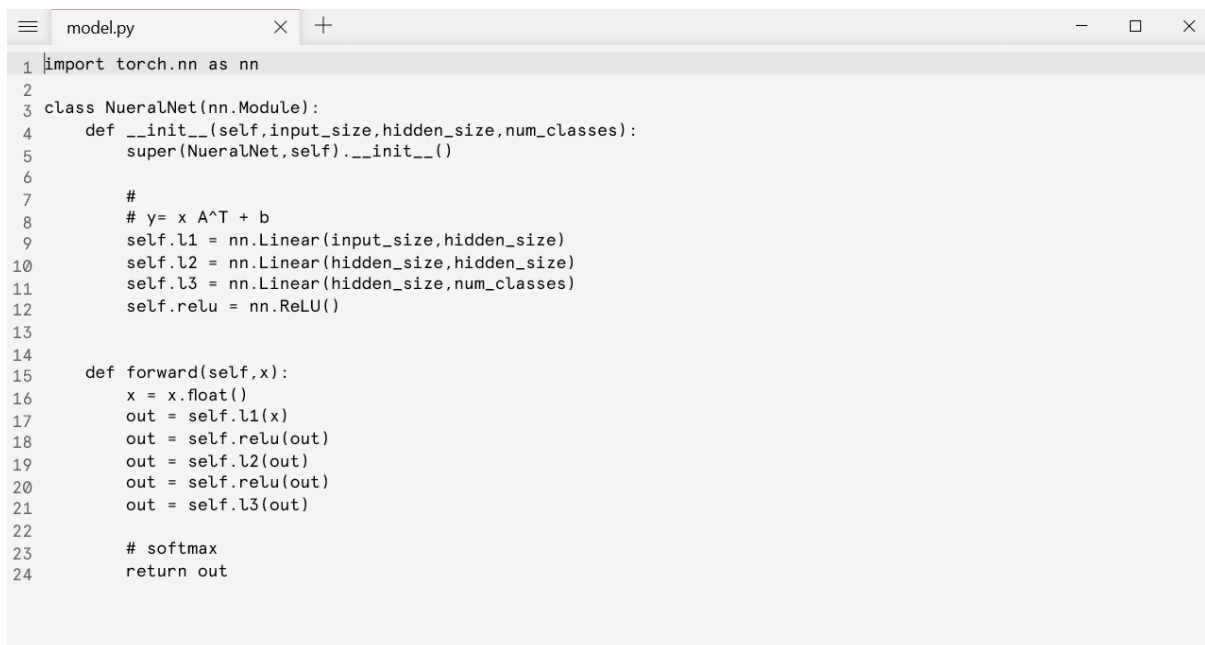
Bag of words function: This function takes input parameters user input sentence which is tokenized and all words which is enumeration of all the words present in **intents.json** file. Then we will stem the tokenized words. Now, we create a bag which consists only of zeros.

Now we iterate over all words array and if we find any word match with our user input words, we then mark 1 in our bag array and return it.

The output for sentence “How is the weather likely today?” is.

```
[ 1.  1.  0.  0.  1.  0.  1.  0.
.
.
0.  0.  0.  1.  0.  0.  0.  1.]
```

6.1.3 Neural Network Model



```
1 import torch.nn as nn
2
3 class NueralNet(nn.Module):
4     def __init__(self, input_size, hidden_size, num_classes):
5         super(NueralNet, self).__init__()
6
7         #
8         # y= x A^T + b
9         self.l1 = nn.Linear(input_size, hidden_size)
10        self.l2 = nn.Linear(hidden_size, hidden_size)
11        self.l3 = nn.Linear(hidden_size, num_classes)
12        self.relu = nn.ReLU()
13
14
15    def forward(self, x):
16        x = x.float()
17        out = self.l1(x)
18        out = self.relu(out)
19        out = self.l2(out)
20        out = self.relu(out)
21        out = self.l3(out)
22
23        # softmax
24        return out
```

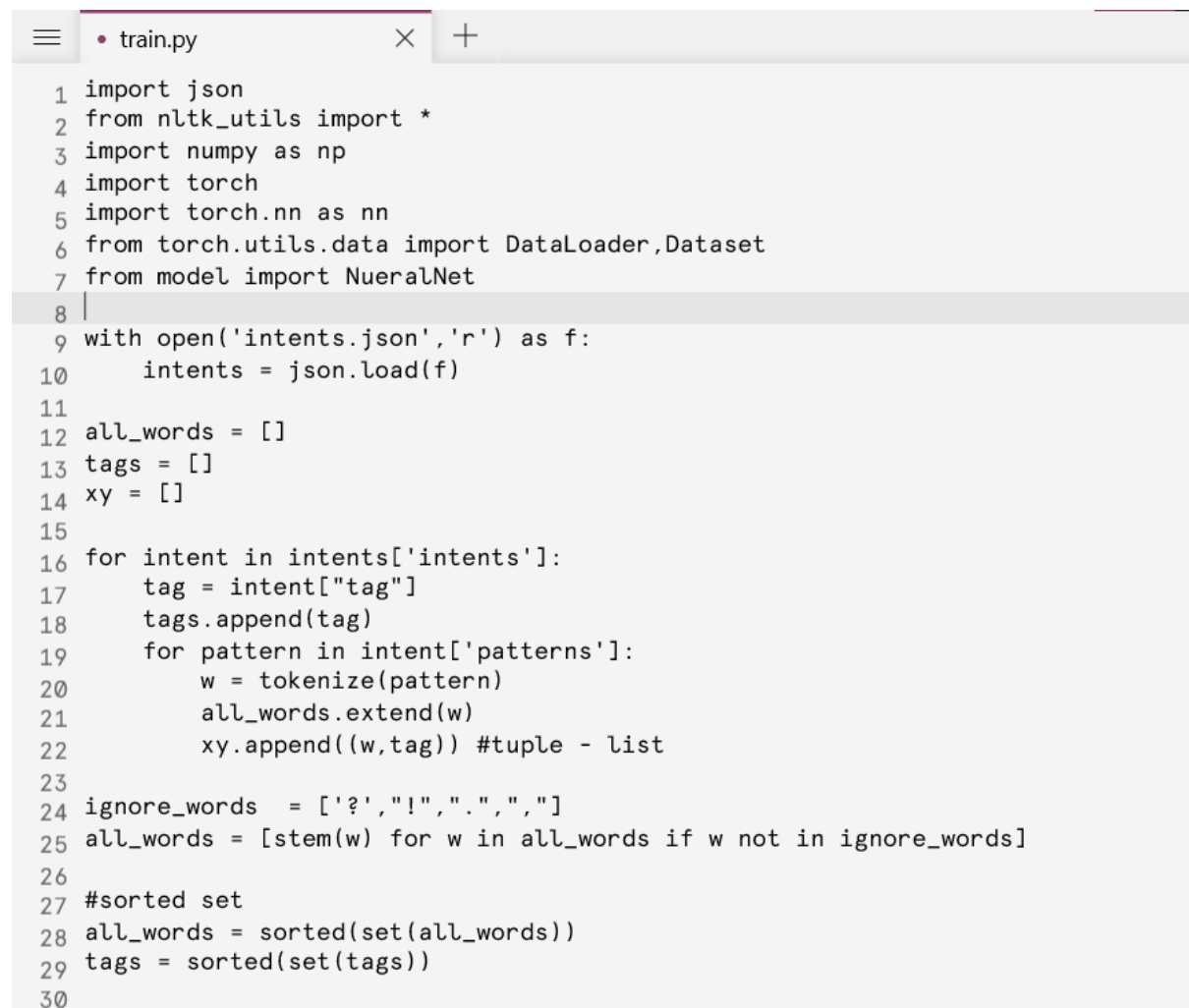
Fig 6.1.3.1: Neural Network Model

We define our own Neural Network by inheriting `nn.Module` class which is the base class for all Neural networks. Our `NeuralNet` class consists of 3 layers that is l1, l2, l3 with their corresponding input, hidden, and output sizes. We apply Linear transformation to the incoming data: which uses the formula $y = x A^T + b$.

Now we create an instance of **RELU** which stands for Rectified Linear Unit function. That is, it gives output as 0 when negative values are passes and return the number when number is >0 .

forward function: It performs data transformation of variable `x` to `y` by passing it through our neural network and returns the output.

6.1.4 Train Module



```
1 import json
2 from nltk_utils import *
3 import numpy as np
4 import torch
5 import torch.nn as nn
6 from torch.utils.data import DataLoader, Dataset
7 from model import NueralNet
8
9 with open('intents.json', 'r') as f:
10     intents = json.load(f)
11
12 all_words = []
13 tags = []
14 xy = []
15
16 for intent in intents['intents']:
17     tag = intent["tag"]
18     tags.append(tag)
19     for pattern in intent['patterns']:
20         w = tokenize(pattern)
21         all_words.extend(w)
22         xy.append((w, tag)) #tuple - list
23
24 ignore_words = ['?', '!', '.', ',']
25 all_words = [stem(w) for w in all_words if w not in ignore_words]
26
27 #sorted set
28 all_words = sorted(set(all_words))
29 tags = sorted(set(tags))
30
```

Fig 6.1.4.1: Loading the data set.

We load our dataset, and we tokenize and stem each of the patterns and store it in tuple form of each word and its corresponding tag. After which we ignore the words which contain any punctuations and then convert it into sorted set.

```

31
32 #training part
33 x_train,y_train = [],[]
34
35 for (pattern_sentence, tag) in xy:
36     bag = bag_of_words(pattern_sentence,all_words)
37     x_train.append(bag)
38
39     label = tags.index(tag)
40     y_train.append(label)
41
42
43 x_train = np.array(x_train)
44 y_train = np.array(y_train)
45
46
47 class ChatDataset(Dataset):
48     def __init__(self):
49
50         self.n_samples = len(x_train)
51         self.x_data = x_train
52         self.y_data = y_train
53
54     def __getitem__(self, idx):
55         return self.x_data[idx],self.y_data[idx]
56
57     def __len__(self):
58         return self.n_samples

```

Fig 6.1.4.2: Training dataset

Now we create x_train and y_train arrays and load the training data and then convert it into NumPy array. **Chat Dataset** extends Dataset class which loads our training data into this class for further training our model.

```

60 # Hyper Parameters
61
62 batch_size = 8
63 hidden_size = 8
64 output_size = len(tags)
65 input_size = len(x_train[0])
66 learning_rate = 0.001
67 num_epochs = 1000
68 device = 'cpu'
69
70 dataset = ChatDataset()
71 train_loader = DataLoader(dataset = dataset, batch_size= batch_size, shuffle=True, num_workers= 0 ) # 0
72
73 model = NueralNet(input_size,hidden_size,output_size).to(device)
74
75
76
77 # Loss and optimizer
78 criterion = nn.CrossEntropyLoss()
79 optimizer = torch.optim.Adam(model.parameters(),lr=learning_rate)
80
81 for epoch in range(num_epochs):
82     for (words, labels) in train_loader:
83         words = words.to(device)
84         labels = labels.long() #
85         labels = labels.to(device)
86         #forward
87
88         outputs = model(words)
89         loss = criterion(outputs, labels)
90
91         # backward and optimizer step
92         optimizer.zero_grad()
93         loss.backward()
94         optimizer.step()
95     if (epoch+1)%100 == 0:
96         print(f'epoch {epoch+1}/{num_epochs}, loss = {loss.item():.4f}')
97
98 print(f'final loss, loss = {loss.item():.4f}')

```

Fig. 6.1.4.3 Training Data

```

D:\Mini Project>py train.py
138 138
35 ['Calculator', 'Chrome', 'Command Prompt', 'Edge', 'Excel', 'Google', 'Micro
soft Paint', 'Notepad', 'Powerpoint', 'Word', 'age', 'amazon', 'date', 'day', '
duckduckgo', 'facebook', 'family', 'fine', 'flipkart', 'funny', 'goodbye', 'gre
eting', 'instagram', 'love', 'made by', 'myself', 'other assistants', 'quora',
'task Manager', 'thanks', 'time', 'twitter', 'weather', 'wikipedia', 'youtube']

epoch 100/1000, loss = 0.1202
epoch 200/1000, loss = 0.0016
epoch 300/1000, loss = 0.0004
epoch 400/1000, loss = 0.0008
epoch 500/1000, loss = 0.0000
epoch 600/1000, loss = 0.0000
epoch 700/1000, loss = 0.0000
epoch 800/1000, loss = 0.0000
epoch 900/1000, loss = 0.0000
epoch 1000/1000, loss = 0.0000
final loss, loss = 0.0000
training complete, file saved to data.pth

```

Fig. 6.1.4.4 Training Console Output

```

chatpy
21 def weather():
22     res = requests.get('https://ipinfo.io/')
23     data = res.json()
24
25     location = data['city']
26
27     complete_api_link = "http://api.openweathermap.org/data/2.5/weather?appid=
4578c4231438a2b339c6b12f30f78648&q="+location
28     api_link = requests.get(complete_api_link)
29     api_data = api_link.json()
30
31     temp_city = ((api_data['main']['temp']) - 273.15)
32     temp_city = "{:.2f}".format(temp_city)
33     weather_desc = api_data['weather'][0]['description']
34     hmdt = api_data['main']['humidity']
35     wind_spd = api_data['wind']['speed']
36     date_time = datetime.datetime.now().strftime("%d %b %Y | %I:%M:%S %p IST")
37
38     return f"\nWeather Stats for - {location} || {date_time}\nCurrent temperature is : {temp_city}
deg C\nCurrent weather desc : {weather_desc}\nCurrent Humidity : {hmdt}\nCurrent wind
speed : {wind_spd} kmph"
39

```

Fig. 6.1.4.5 Weather Implementation

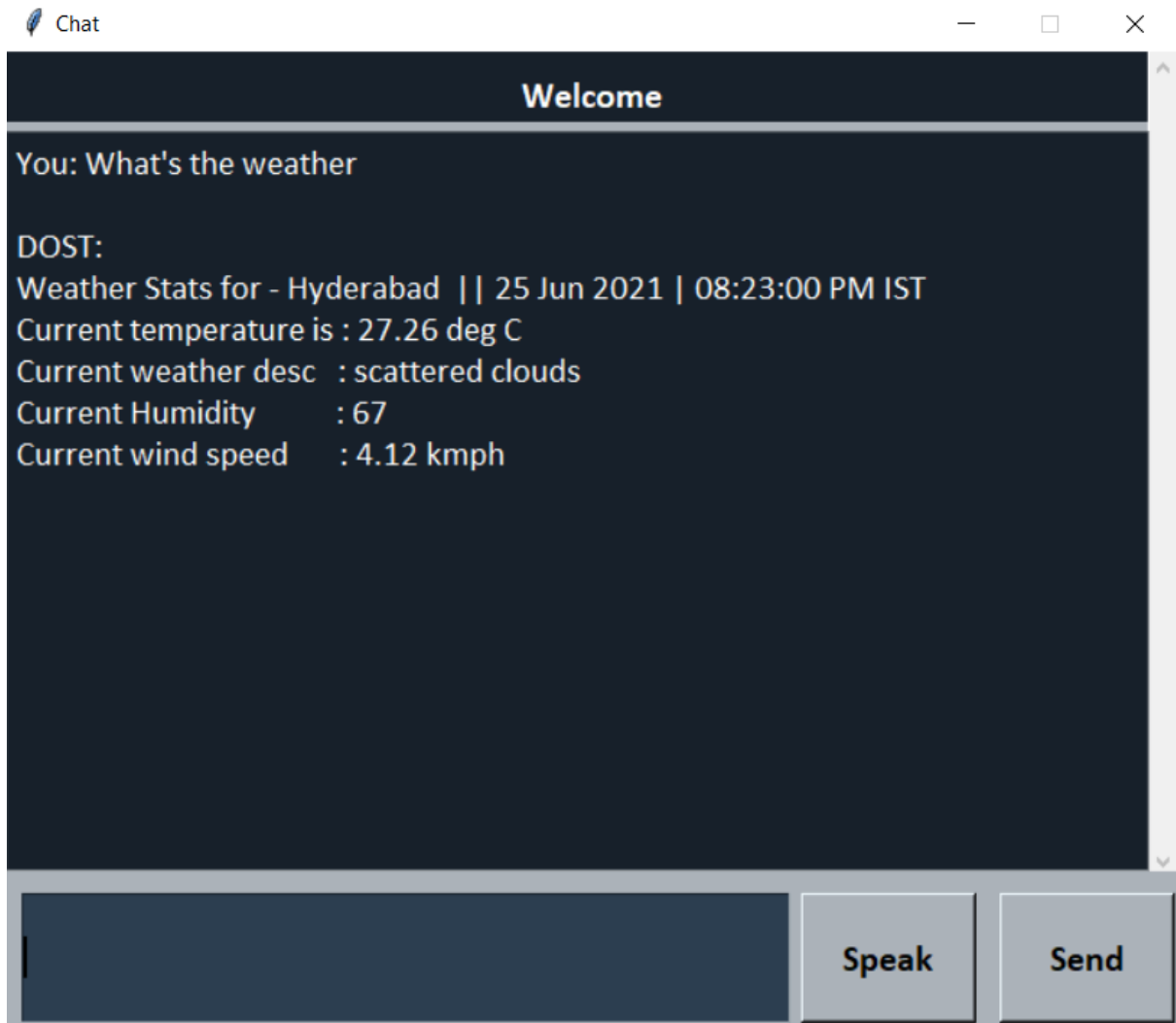


Fig: 4.1.4.6 Weather Output

```
chat.py
def openWord():
    word = win32.Dispatch('Word.Application')
    word.Visible = 1

    doc = word.Documents.Add()
    word.ActiveDocument.ActiveWindow.View.Type = 3

    openUP("word")

def openExcel():
    excel = win32.Dispatch('Excel.Application')
    excel.Visible = 1
    excel.Workbooks.Add()
    openUP("excel")

def openPPT():
    ppt = win32.Dispatch('Powerpoint.Application')
    ppt.Visible = 1
    ppt.Presentations.Add()
    openUP("powerpoint")
```

Fig: 4.1.4.7 Opening Office Applications

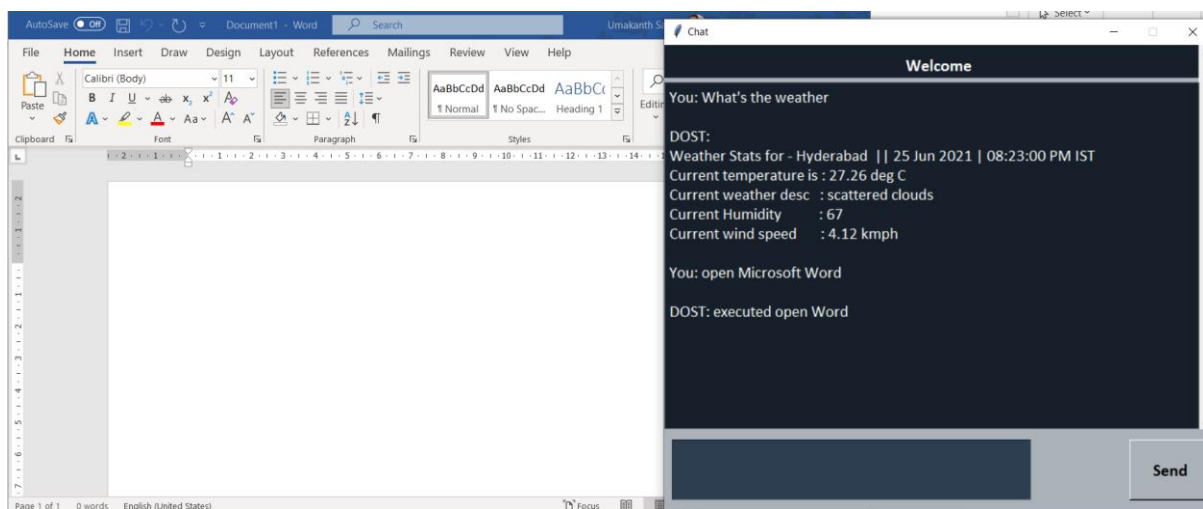


Fig: 4.1.4.8 Word opening

6.2 DOST GUI

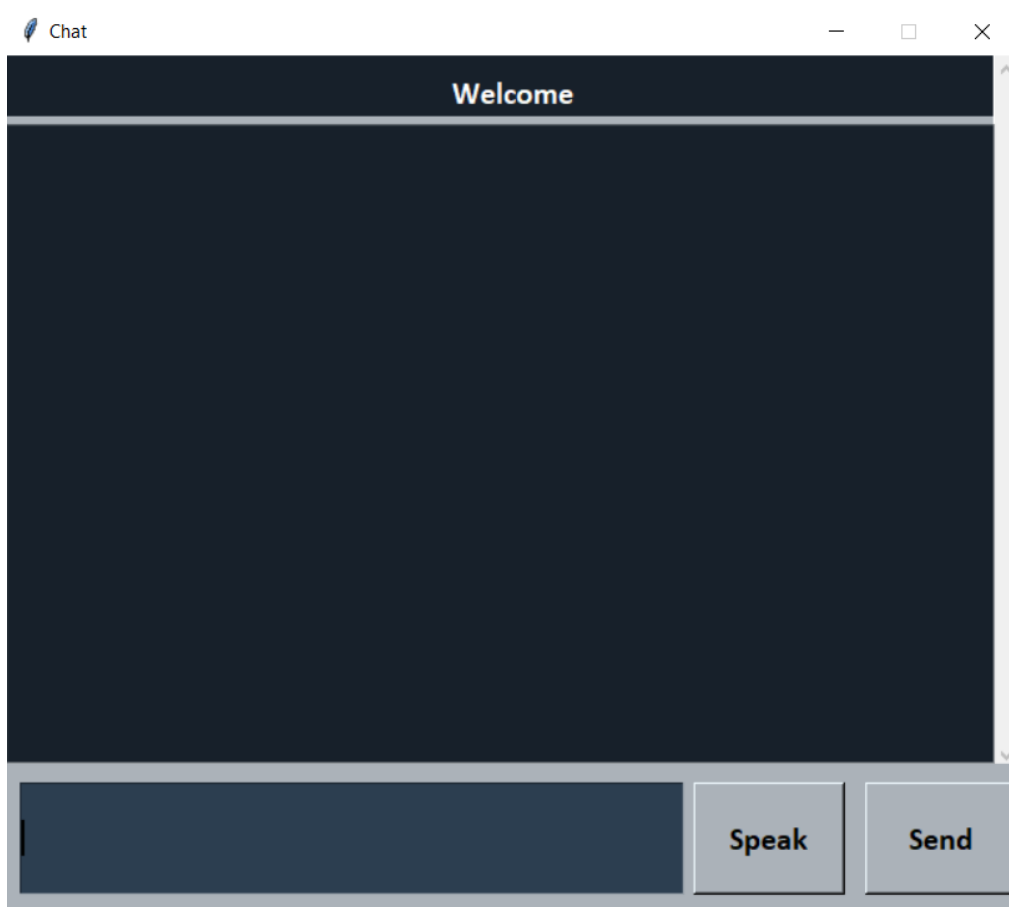


Fig 6.2.1: DOST GUI

```
app.py
31 class ChatApplication:
32     def __init__(self):
33         self.window = Tk()
34         self._setup_main_window()
35
36     def run(self):
37         self.window.mainloop()
38
39     def _setup_main_window(self):
40         self.window.title("Chat")
41         self.window.resizable(width=False, height=False)
42         self.window.configure(width=650, height=550, bg=BG_COLOR)
43
44         #Welcome head label
45         head_label = Label(self.window, bg=BG_COLOR, fg= TEXT_COLOR, text="Welcome", font=FONT_BOLD, pady=10)
46         head_label.place(relwidth=1)
47
48         #divider
49         line = Label(self.window, width=450, bg=BG_GRAY)
50         line.place(relwidth=1, rely=0.07, relheight=0.012)
51
52         #Scroll bar
53         scrollbar = Scrollbar(self.window) #
54         scrollbar.pack(side="right", fill=Y)
55         scrollbar.place(relheight=0.83, relx=0.974)
56
57         #text widget
58         self.text_widget = Text(self.window, width=20, height=2, bg=BG_COLOR, fg=TEXT_COLOR, font=FONT, padx=5, pady=
5, yscrollcommand=scrollbar.set)
```

Fig. 6.2.2 app.py code

7. SOFTWARE TESTING

7.1 Unit Testing:

The testing done for the modules constructed from the system design is called Unit testing. Testing is done with respective inputs for those modules. Modules are assembled into larger unit during this. Phase level testing has been done during project design and coding. We have tested our module interface in order to ensure the proper flow of information in and out of the program unit during the test. The temporary output data that is generated is ensured that it maintains its integrity throughout the algorithm's execution. At last, all paths have been tested for error handling.

We have used the following scenarios to check the behaviour of our model:

- Tested input for each class label for “Statement Coverage” and “Branch Coverage”
- Tested model for no input
- Tested when large input is given
- Tested when there is no Internet Connectivity

After testing, we have noticed the issues when there is no internet connectivity. The App displays the complete stack trace of the Exception. The above issues we handled by appropriate exceptions.

7.2 Integration Testing:

System testing is usually done to find errors resulting from unanticipated interaction between system components and sub-system. Before diverting the software to customers source code must be tested to detect and rectify the possible errors once it is generated. A series of test cases must be developed to uncover all the possibly existing errors. There are many different software techniques for this. These software techniques provide guidance in designing a test that exercise the internal logic and input-output domains to uncover errors in the program function, performance, and behaviour. Majorly software is tested using two methods. (i) Testing the internal program logic using the test case design techniques is called White Box Testing. (ii) Testing the software requirements using the test case design techniques is called Black Box Testing. These two will help in finding the maximum number of errors with minimal time and effort.

We have used the following scenarios to check the behaviour of our model:

- Tested for model and app modules integration.
- Tested for errors when there is missing module.

7.3 Acceptance Testing:

This testing process refers to verification and validation. We must try to acknowledge the system specifications and meet the client's requirements. Hence, we must verify and validate the product to make sure everything is in place as per the client's requirements. Verification is performed to ensure that the software correctly implements a specific functionality awhile Validation is done to ensure if the customer requirements are properly met or not by the product. Verification of the project was carried out to ensure that the project met all the requirements and specifications as mentioned in the project abstract. We have ensured that our project is up to the standard as we planned at the beginning of our project development.

8. RESULTS

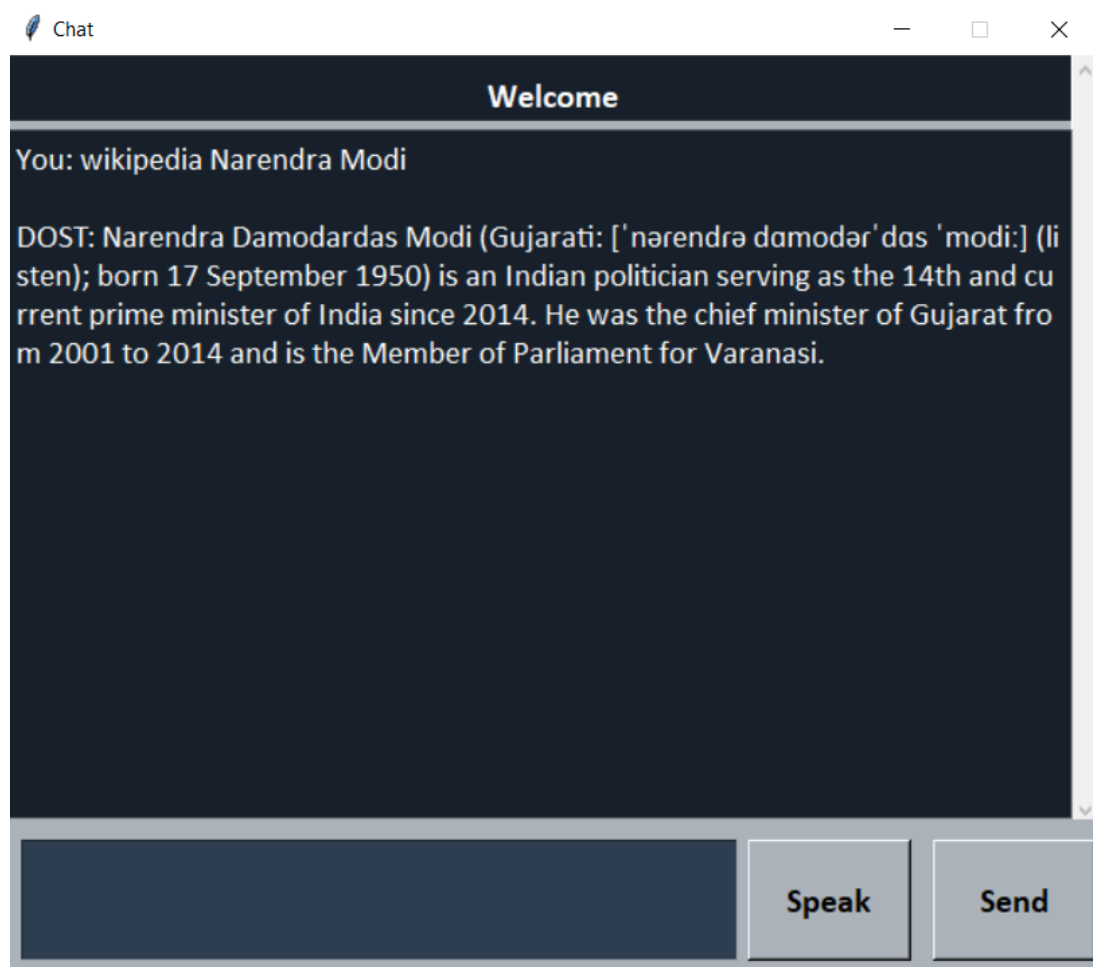


Fig: 8.1: Wikipedia Results

When we give input as “Narendra Modi Wikipedia” through voice after pressing the speak button, we get summary about Mr. Narendra Modi extracted from Wikipedia. This result has been retrieved using the “wikipedia” python module. When there is no internet connectivity then app says no Internet Connectivity.

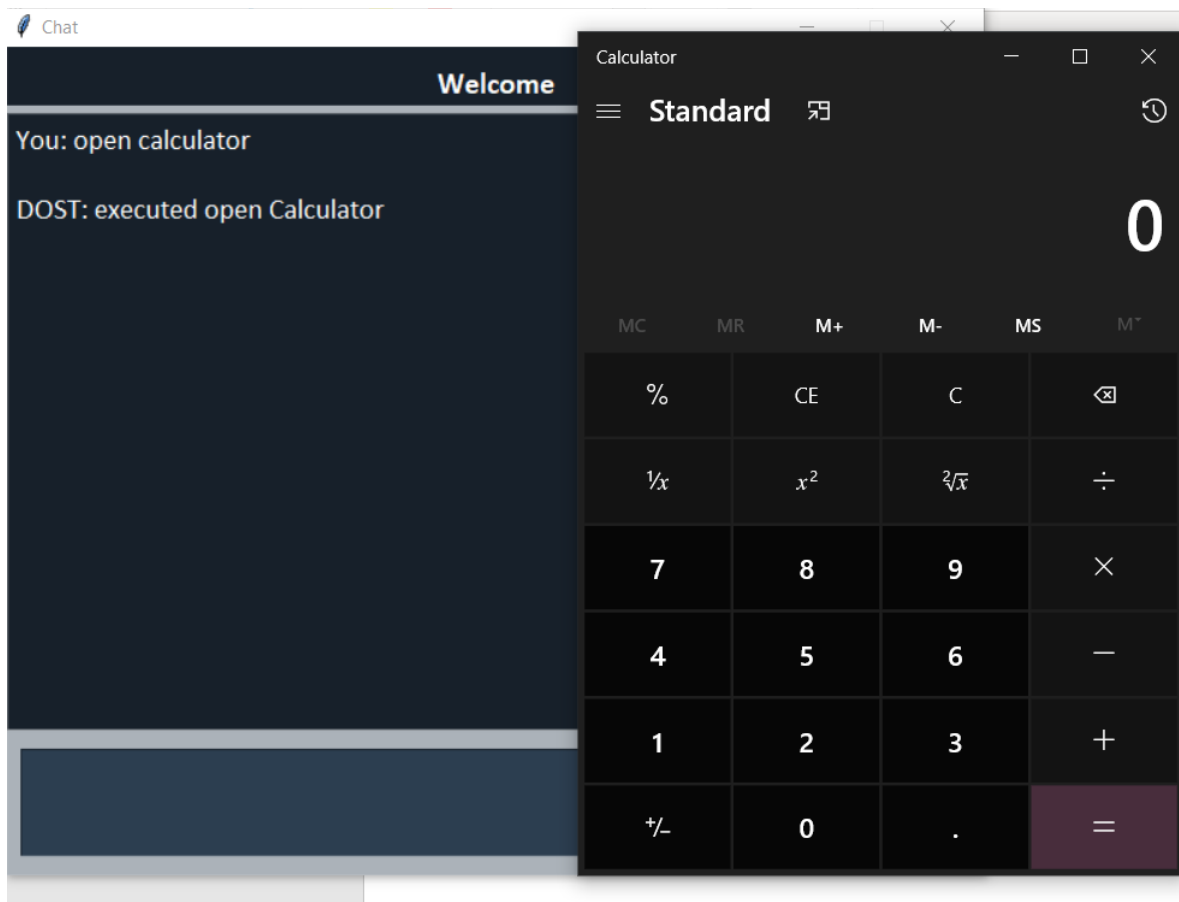


Fig8.2: Opening Application Calculator

We give input as “Open Calculator” or just say “Calculator”, then python instantly opens the Windows native Calculator app. We can also directly ask the DOST App for the simple calculations like addition, subtraction, multiplication, division to which our DOST application gives answer right away on the screen.

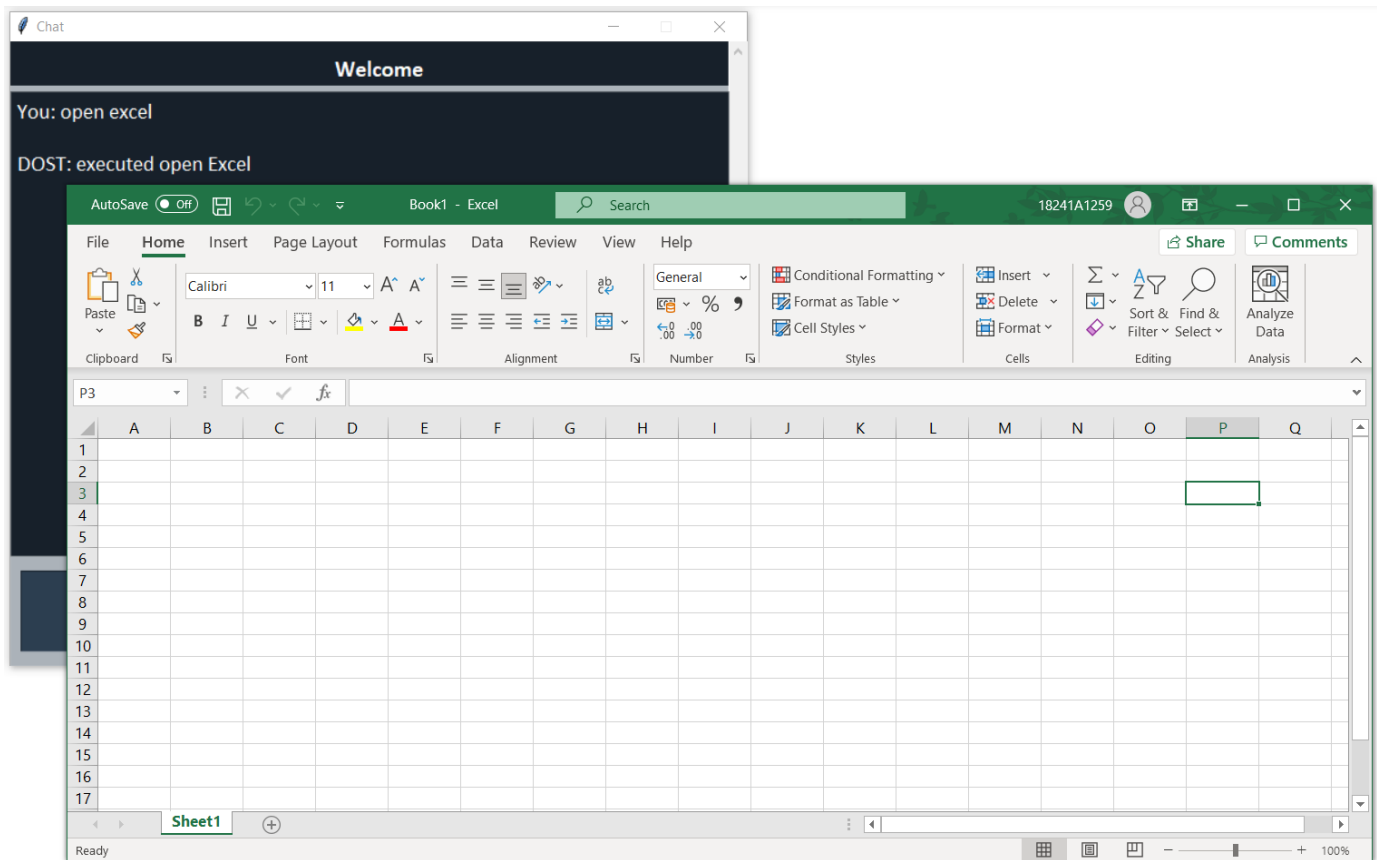


Fig 8.3: Opening Office Applications - Excel

When we say, “Open Microsoft Excel”, Our DOST Application instantly opens an Excel file for you. A new Excel is opened through WinCom32 python module. If Microsoft App is not installed on your PC, then it says that there is no application found.

```
D:\Mini Project>py train.py
138 138
35 ['Calculator', 'Chrome', 'Command Prompt', 'Edge', 'Excel', 'Google', 'Micro
soft Paint', 'Notepad', 'Powerpoint', 'Word', 'age', 'amazon', 'date', 'day', '
duckduckgo', 'facebook', 'family', 'fine', 'flipkart', 'funny', 'goodbye', 'gre
eting', 'instagram', 'love', 'made by', 'myself', 'other assistants', 'quora',
'task Manager', 'thanks', 'time', 'twitter', 'weather', 'wikipedia', 'youtube']

epoch 100/1000, loss = 0.1202
epoch 200/1000, loss = 0.0016
epoch 300/1000, loss = 0.0004
epoch 400/1000, loss = 0.0008
epoch 500/1000, loss = 0.0000
epoch 600/1000, loss = 0.0000
epoch 700/1000, loss = 0.0000
epoch 800/1000, loss = 0.0000
epoch 900/1000, loss = 0.0000
epoch 1000/1000, loss = 0.0000
final loss, loss = 0.0000
training complete, file saved to data.pth
```

Fig 8.4: Epoch and Loss

The above picture depicts the output of the training file. Each epoch line shows the loss and number of trainings done. Finally, loss is calculated and the model is saved to a pytorch model file.

9. CONCLUSION AND FUTURE ENHANCEMENTS

9.1 Conclusion

In this project, we built a virtual assistant which helps to do specific actions for the sake of users with the help of Deep Learning. In the end of the project, we have understood Deep Learning concepts such as forward and backward propagation, Adam's algorithm, RELU and Sigmoid functions etc. We have also understood Natural Language Processing-how words are tokenized and stemmed etc. During the deployment phase of the project, we have faced an issue of converting the main project into an executable file. Later when we resolved this issue, we found out that the output exe file was enormous. Later we fixed.

All the issues by referring to the actual documentation of PyInstaller. Finally, this may not be the best virtual assistant, but we succeeded in showing how the virtual assistants are built.

9.2 Future Enhancements

When we come to the further enhancements, the accuracy and effectiveness of the model is directly proportional to the training of model. You must include large chunks of data to train the model. There is still a large scope to improve the GUI. We can work on the machine to make it a mobile friendly application. We can also make it run in the background and activate the GUI using Voice Command which many of the voice assistants are currently doing.

10. BIBLIOGRAPHY

1. Creating a chat bot using Pytorch [Video file]. Retrieved from <https://www.youtube.com/watch?v=RpWeNzfSUHw>
2. <https://docs.python.org/3/library/json.html>
3. <https://pytorch.org/docs/stable/index.html>
4. <http://timgolden.me.uk/pywin32-docs/html/com/win32com/HTML/QuickStartClientCom.html>
5. <https://pyinstaller.readthedocs.io/en/stable/>
6. <https://docs.python.org/3/library/tkinter.html>
7. <https://docs.python-requests.org/en/master/index.html>