INTELLIGENT AI CHATBOT

USING NLP and MACHINE LEARNING

A MINI PROJECT REPORT

Submitted by

ANASWHAR JOSEPH	**20CECS003**
ANUAVANTHIKA	**20CECS005**
APARNA	**20CECS006**
YOGESH KUMAR .A	**20CECS505**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SNS COLLEGE OF ENGINEERING

COIMBATORE – 641 107

ANNA UNIVERSITY

CHENNAI - 600 025

OCTOBER & 2021

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"INTELLIGENT AI CHATBOT USING MLP and MACHINE LEARNING"** is the Bonafide work of **, ANASWHAR JOSEPH (20CECS003), ANUAVANTHIKA (20CECS005), APARNA(20CECS006), YOGESH KUMAR(20CECS505),** who carried out the mini project work under my supervision.

SIGNATURE	**SIGNATURE**
DR.K.PERIYAKARUPPAN	Mr. K. KARTHIKEYAN
HEAD OF THE DEPARTMENT Professor Department of Computer Science and Engineering SNS College of Engineering Coimbatore - 641 107.	**SUPERVISOR** Assistant Professor Department of Computer Science and Engineering SNS College of Engineering Coimbatore - 641 107.

Submitted for the Mini Project Viva Voce examination held on _____

Internal Examiner

ACKNOWLEDGEMENT

We wish to express our sincere thanks to honorable Chairman **Dr. S. N. Subbramanian**, Correspondent **Dr. S. N. Rajalakshmi**, Founder Trustee **Dr. V. S. Velusamy** and our Technical Director **Dr. S. Nalin Vimal Kumar** whose progressive ideas added with farsighted counsels has shouldered us to reach meritorious heights.

We indebted to express our deep sense of gratitude to the Director **Dr. V. P. Arunachalam,** Principal **Dr. S.Charles** and Vice Principal **Dr. R. Sudhakaran**, for their valuable support while doing our project.

We highly indebted to record my heartfelt thanks to **Dr.K.Periyakaruppan**, Professor and Head, Department of Computer Science and Engineering, for his able guidance, suggestions and persistent encouragement that extended throughout the execution of our project work.

We would like to thank **Mr. E.P.Prakash**, Mini Project Coordinator cum Assistant Professor, Department of Computer Science and Engineering and our project guide **Mr. K. Karthikeyan,** Assistant Professor, Department of Computer Science and Engineering for their able guidance, without which, our project would not be a successful one.

We solemnly express our thanks to all the teaching and non teaching staff members of the Department of Computer Science and Engineering, family and friends for their valuable support which energized us to complete our project in time.

TABLE OF CONTENTS

ABSTRACT

A chatbot is a software application used to conduct an on-line chat conversation via text or text-to-speech, in lieu of providing direct contact with a live human agent. Designed to convincingly simulate the way a human would behave as a conversational partner, chatbot systems typically require continuous tuning and testing, and many in production remain unable to adequately converse or pass the industry standard Turing test. The term "Chatterbot" was originally coined by Michael Mauldin (creator of the first Verbot) in 1994 to describe these conversational programs.

Chatbots are used in dialog systems for various purposes including customer service, request routing, or information gathering. While some chatbot applications use extensive word-classification processes, natural language processors, and sophisticated AI, others simply scan for general keywords and generate responses using common phrases obtained from an associated library or database.

Most chatbots are accessed on-line via website popups or through virtual assistants. They can be classified into usage categories that include: commerce (e-commerce via chat), education, entertainment, finance, health, news, and productivity

The purpose of chat bots is to support and scale business teams in their relations with customers. It could live in any major chat applications like Facebook Messenger, Slack, Telegram, Text Messages, etc. Chatbot applications streamline interactions between people and services, enhancing customer experience. At the same time, they offer companies new opportunities to improve the customers engagement process and operational efficiency by reducing the typical cost of customer service. This project is focused on building a custom chatbot that will be your fundamental step of the learning curve of building your own professional chatbots.

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBERIVATION	EXPANSION
NLP	**Natural Language Processing**
NLTK	**N**atural **L**anguage **T**ool**k**it
NumPy	**Numerical Python**
JSON	**JavaScript Object Notion**
AI/ML	**Artificial Intelligence / Machine Learning**

INTRODUCTION

A chatbot is a software application used to conduct an on-line chat conversation via text or text-to-speech, in lieu of providing direct contact with a live human agent. Designed to convincingly simulate the way a human would behave as a conversational partner, chatbot systems typically require continuous tuning and testing, and many in production remain unable to adequately converse or pass the industry standard Turing test. The term "Chatterbot" was originally coined by Michael Mauldin (creator of the first Verbot) in 1994 to describe these conversational programs.

Chatbots are used in dialog systems for various purposes including customer service, request routing, or information gathering. While some chatbot applications use extensive word-classification processes, natural language processors, and sophisticated AI, others simply scan for general keywords and generate responses using common phrases obtained from an associated library or database.

Most chatbots are accessed on-line via website popups or through virtual assistants. They can be classified into usage categories that include: commerce (e-commerce via chat), education, entertainment, finance, health, news, and productivity

The purpose of chat bots is to support and scale business teams in their relations with customers. It could live in any major chat applications like Facebook Messenger, Slack, Telegram, Text Messages, etc. Chatbot applications streamline interactions between people and services, enhancing customer experience. At the same time, they offer companies new opportunities to improve the customers engagement process and operational efficiency by reducing the typical cost of customer service. This project is focused on building a custom chatbot that will be your fundamental step of the learning curve of building your own professional chatbots.

EMPATHY PHASE

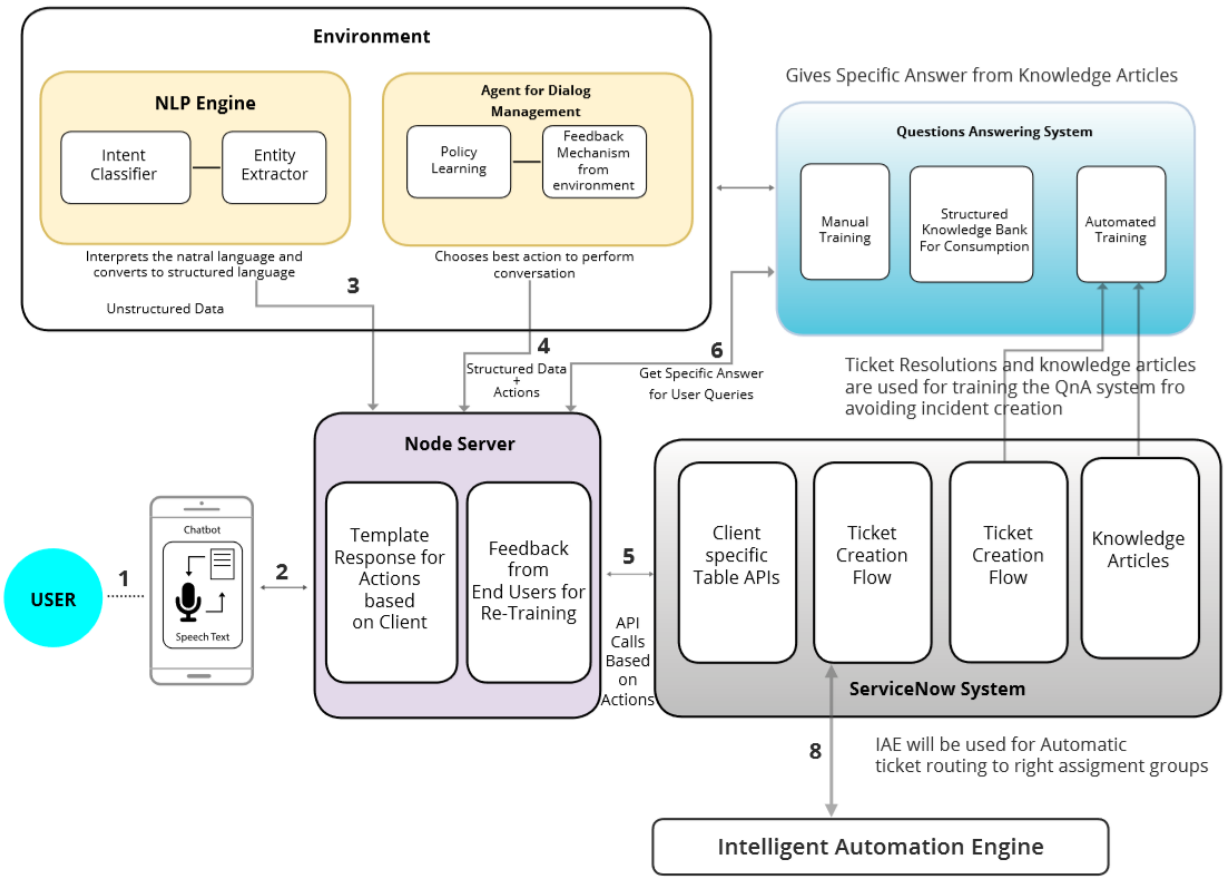
Existing System

The Existing Chatbots are very similar and converse in more formal way.

Proposed Systen

The purpose of chat bots is to support and scale business teams in their relations with customers. It could live in any major chat applications like Facebook Messenger, Slack, Telegram, Text Messages, etc. Chatbot applications streamline interactions between people and services, enhancing customer experience. At the same time, they offer companies new opportunities to improve the customers engagement process and operational efficiency by reducing the typical cost of customer service. This project is focused on building a custom chatbot that will be your fundamental step of the learning curve of building your own professional chatbots.

Product Architecture



1.0 Product Architecture

High-Level Approach

- User starts the conversation by typing a sentence into chatbot.
- JARVIS Will Analyze the user input.
- Then user will get an intelligent reply from the JARVIS

Primary goals

- Setting up an open-source project locally and handling the errors being faced
- Using multiple services to build up a new service over them.
- Having a real-world chatbot, to which you can literally chat like you chatting to a real person.

Hardware Requirements

- Intel or AMD 64Bit CPU
- 4GB OF RAM
- 10MB HDD Space
- Nvidia GPU with 2GB Dedicated VRAM
- HID Devices.

Software Requirements

- Python 3 (3.7 or later Recommended)
- VS Code (IDE)
- PyTorch (Open source machine learning framework)
- NLTK (**N**atural **L**anguage **T**ool**k**it to work with human language data)
- Pickle (A Python Module used to serializing and de-serializing an object)
- NumPy (**Num**erical **Py**thon is Python Library used to work with arrays)
- Keras (A High-level neural network API)
- NLP (**N**atural **L**anguage **P**rocessing enable computers to process human language in the form of text or voice data and to 'understand' its full meaning)
- A JSON Intents file (Used for training our bot)
- Tkinter (A Standard python library used to create GUI)

Design Phase

Step 1:

Create a new Folder named JARVIS in VS Code. This is our project folder where we build and save our python files

Step 2:

Create File Named intents.json

Introduction to JSON

A JSON (Stands for JavaScript Object Notion) file is a lightweight data-interchange format. It acts as data provider in which out bot uses the contents in intents. json as a training data.

Building intents.json file

Step 3:

Open the intents.json file and start building the trainer data. This json file consists of 3 different elements namely tag, patterns and response. In which

The tag element which is a class label consists of a unique name for e.g. “greetings”

The patterns element consists of a complete sentence or a word which we human will used to chat with our bot. For e.g. “Hi”, “Hello” etc.

The response element consists of a response which relates to the contents of patterns element. For e.g. “Hello, There”, “Hi, Master”

Our bot will be trained with is json file. So, on final product if we say Hi to our Bot. our bot tries to classify it in one of those patterns text for example if it recognizes that it is a greeting then it takes randomly one of those answers from these responses.

Our json file will be written in below given structure.

Intents.json

```
{
  "intents": [
    {
      "tag": "greetings",
      "patterns": [
        "JARVIS",
        "Hi",
        "Hello",
        "Hey"
      ],
      "responses": [
        "Master!!",
        "Hello, Master",
        "Hi there",
```

```
        "Hi, Master",
        "Master, Any Help",
        "Master!! Good to see you again",
        "Master, You are back",
        "My Name is JARVIS. To learn how I can help, Ask :
What can you do?"
      ],
      "context": [""]
    },
    {
      "tag": "wellbeing",
      "patterns": [
        "How are you?",
        "How you doing?",
        "You Good?"
      ],
      "responses": [
        "I'm Fine Master, Thanks for Asking",
        "Doing great master",
        "All good master",
        "Thanks for asking. I'm doing well",
        "Things are good, Trying to learn new things
everyday",
        "I've had a whirlwind of a week, But I'm hanging
there",
        "I'm Looking forward to the end of the Pandemic."
      ],
      "context": [""]
    },
    {
      "tag": "goodbye",
      "patterns": ["Bye", "See you later", "Goodbye", "Nice
chatting to you, bye", "Till next time"],
      "responses": ["See you!", "Have a nice day", "Bye!
Come back again soon."],
      "context": [""]
     },
     {
```

```
      "tag": "self",
     "patterns": ["Who are you?", "What is your name", "who
are you?"],
     "responses": ["It's me JARVIS", "My Name is JARVIS
[Just A Rather Very Intelligent System]", "Hi, I'm
Jarvis"],
     "context": [""]
    },
    {
      "tag": "creator",
     "patterns": ["Who created you?", "Who's your
creator"],
     "responses": ["Well now, I can't give away all my
serets...", "I Was created by... Uhh my creator"],
     "context": [""]
    },
    {
      "tag": "botage",
     "patterns": ["How old are you?", "What's your age",
"Your Birthday?"],
     "responses": ["Well, my birthday is September 2, 2021.
So I'm really a spring chicken. Except I'm not a chicken",
"If you're planning a surprise party, I made my debut on
September 2, 2021"],
     "context": [""]
    },
    {
      "tag": "Funny",
     "patterns": ["Tell me a joke", "Make me laugh", "Tell
Me something funny"],
     "responses": ["Q: What do you call a penguin in the
desert? A: Lost", "Q: What did the SNAIL say while riding
on the turtles back? A: Wheeeeeeeee", "Q: What do you call
a lazy baby kangaroo?A: A pouch potato","I can’t believe I
got fired from the calendar factory. All I did was take a
day off!","Money talks. Mine always says goodbye","I went
to see the doctor about my short-term memory problems – the
```

```
first thing he did was make me pay in advance","Why do bees
hum? They don’t remember the lyrics!"],
     "context": [""]
    },
     {
       "tag": "noanswer",
      "patterns": [""],
      "responses": ["Sorry, can't understand you", "Please
give me more info", "Not sure I understand"],
      "context": [""]
     }
  ]
}
```

The Above json format can be improvise by adding multiple tags, patterns and response which resemble a conversation between two persons.

Now Save the file.

Step 4:

In this step we are using NLP preprocess pipeline method to create an array of word by tokenize, stem and bag of words concept.

For Example, let take following sentence

[“Are you Organizing the Organization?”]

Now we are implementing tokenization. Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens. With our example when we implement tokenization we will get following output

[“Are”, “you”, “Organizing”, “the”, “Organization”, “?”]

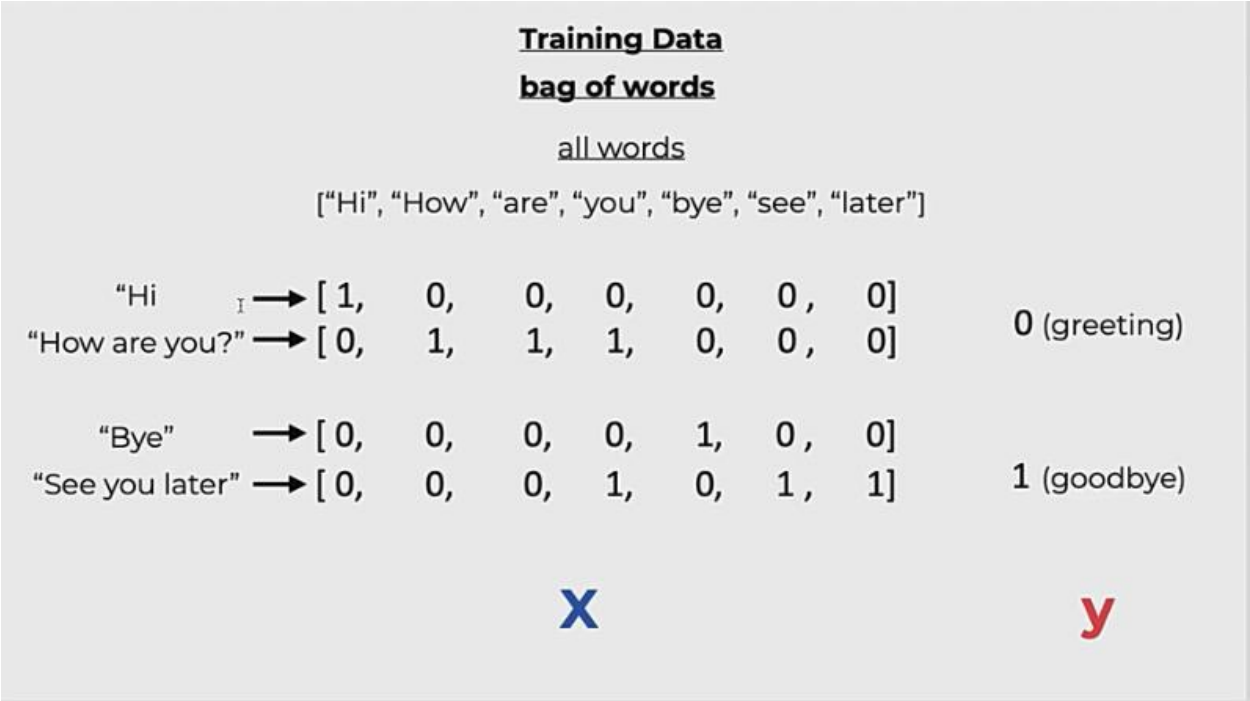
Next, we are Implementing lower + stem. Stemming is a technique used to extract the base form of the words by removing affixes and punctuation marks from them. It is just like cutting down the branches of a tree to its stems. For example, the stem of the words eating, eats, eaten is eat. And then we convert all upper case to lowercase. So, our output will be like.

[“are”, “you”, “organ”, “the”, “organ”]

Next, we are creating bag of words. Here our training data will be converted into X and Y vectors. And we train our bot with NLTK and Feed Forward Neural Net

Training Data

1.1 Training Data



1.2 Bag of words creation

Step 5:

Create a new file called nltk_utils.py and code as given below. In this file we use nltk algorithms to do the tokenization, stemming and bag of words.

nltk_utils.py

```
import numpy as np
import nltk
# nltk.download('punkt')
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()

def tokenize(sentence):
    """
    split sentence into array of words/tokens
    a token can be a word or punctuation character, or
number
    """
    return nltk.word_tokenize(sentence)


def stem(word):
    """
    stemming = find the root form of the word
    examples:
    words = ["organize", "organizes", "organizing"]
    words = [stem(w) for w in words]
    -> ["organ", "organ", "organ"]
    """
    return stemmer.stem(word.lower())


def bag_of_words(tokenized_sentence, words):
    """
    return bag of words array:
    1 for each known word that exists in the sentence, 0
otherwise
    example:
```

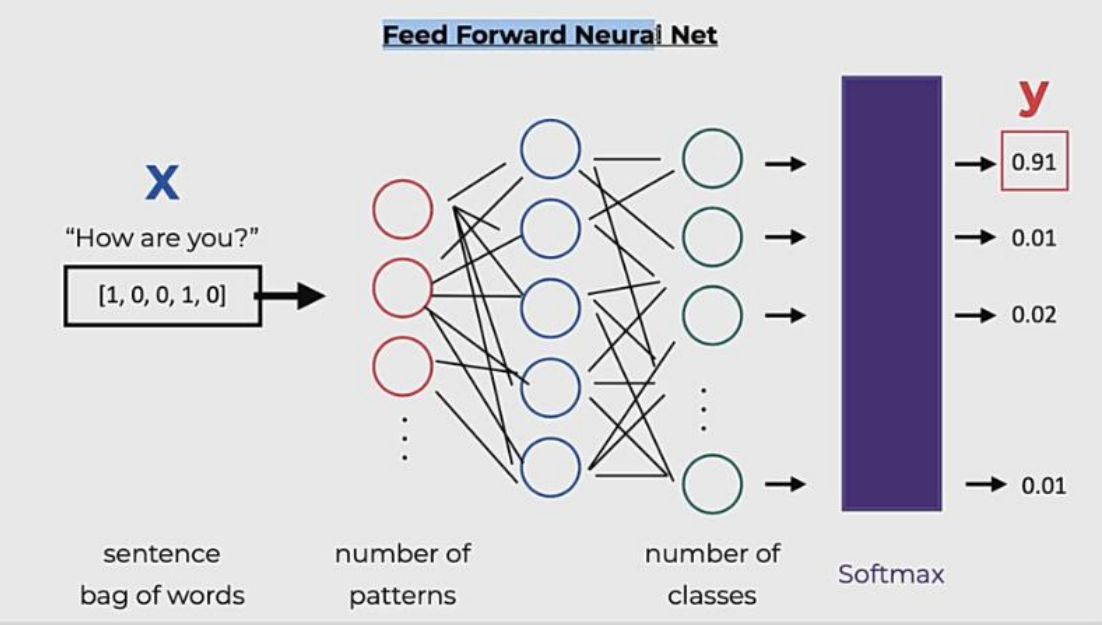
```
    sentence = ["hello", "how", "are", "you"]
    words = ["hi", "hello", "I", "you", "bye", "thank",
"cool"]
    bog   = [  0 ,    1 ,    0 ,   1 ,    0 ,    0
,      0]
    """
    # stem each word
    sentence_words = [stem(word) for word in
tokenized_sentence]
    # initialize bag with 0 for each word
    bag = np.zeros(len(words), dtype=np.float32)
    for idx, w in enumerate(words):
        if w in sentence_words:
            bag[idx] = 1

    return bag
```

Step 6:

Now we are going to build a neural net model for our bot using PyTorch with the concept called Feed Forward Neural Network. He once we input our bag of words Feed Forward Neural network will trained with patterns and classes and using a probability it guess the correct tag for users response.

Create a File named model.py and save it. Then start coding it with below python code.



1.3 Feed Forward neural network

model.py

```
import torch
import torch.nn as nn


class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size,
num_classes):
        super(NeuralNet, self).__init__()
        self.l1 = nn.Linear(input_size, hidden_size)
        self.l2 = nn.Linear(hidden_size, hidden_size)
        self.l3 = nn.Linear(hidden_size, num_classes)
        self.relu = nn.ReLU()
```

```
    def forward(self, x):
        out = self.l1(x)
        out = self.relu(out)
        out = self.l2(out)
        out = self.relu(out)
        out = self.l3(out)
        # no activation and no softmax at the end
        return out
```

train.py

```
import numpy as np
import random
import json

import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader

from nltk_utils import bag_of_words, tokenize, stem
from model import NeuralNet

with open('intents.json', 'r') as f:
    intents = json.load(f)

all_words = []
tags = []
xy = []
# loop through each sentence in our intents patterns
for intent in intents['intents']:
    tag = intent['tag']
    # add to tag list
    tags.append(tag)
    for pattern in intent['patterns']:
        # tokenize each word in the sentence
        w = tokenize(pattern)
        # add to our words list
        all_words.extend(w)
        # add to xy pair
```

```
        xy.append((w, tag))

# stem and lower each word
ignore_words = ['?', '.', '!']
all_words = [stem(w) for w in all_words if w not in
ignore_words]
# remove duplicates and sort
all_words = sorted(set(all_words))
tags = sorted(set(tags))

print(len(xy), "patterns")
print(len(tags), "tags:", tags)
print(len(all_words), "unique stemmed words:", all_words)

# create training data
X_train = []
y_train = []
for (pattern_sentence, tag) in xy:
    # X: bag of words for each pattern_sentence
    bag = bag_of_words(pattern_sentence, all_words)
    X_train.append(bag)
    # y: PyTorch CrossEntropyLoss needs only class labels,
not one-hot
    label = tags.index(tag)
    y_train.append(label)

X_train = np.array(X_train)
y_train = np.array(y_train)

# Hyper-parameters
num_epochs = 1000
batch_size = 8
learning_rate = 0.001
input_size = len(X_train[0])
hidden_size = 8
output_size = len(tags)
print(input_size, output_size)
```

```
class ChatDataset(Dataset):

    def __init__(self):
        self.n_samples = len(X_train)
        self.x_data = X_train
        self.y_data = y_train

    # support indexing such that dataset[i] can be used to
get i-th sample
    def __getitem__(self, index):
        return self.x_data[index], self.y_data[index]

    # we can call len(dataset) to return the size
    def __len__(self):
        return self.n_samples

dataset = ChatDataset()
train_loader = DataLoader(dataset=dataset,
                          batch_size=batch_size,
                          shuffle=True,
                          num_workers=0)

device = torch.device('cuda' if torch.cuda.is_available()
else 'cpu')

model = NeuralNet(input_size, hidden_size,
output_size).to(device)

# Loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(),
lr=learning_rate)

# Train the model
for epoch in range(num_epochs):
    for (words, labels) in train_loader:
        words = words.to(device)
        labels = labels.to(dtype=torch.long).to(device)
```

```
        # Forward pass
        outputs = model(words)
        # if y would be one-hot, we must apply
        # labels = torch.max(labels, 1)[1]
        loss = criterion(outputs, labels)

        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    if (epoch+1) % 100 == 0:
        print (f'Epoch [{epoch+1}/{num_epochs}], Loss:
{loss.item():.4f}')


print(f'final loss: {loss.item():.4f}')

data = {
"model_state": model.state_dict(),
"input_size": input_size,
"hidden_size": hidden_size,
"output_size": output_size,
"all_words": all_words,
"tags": tags
}

FILE = "data.pth"
torch.save(data, FILE)

print(f'training complete. file saved to {FILE}')
```

Step 7:

Now we have completed coding our training files. It's time to implement them in out chat application. For the create a new file called chat.py here we will import our nltk and model and build a chatbot.

chat.py

```
import random
import json

import torch

from model import NeuralNet
from nltk_utils import bag_of_words, tokenize

device = torch.device('cuda' if torch.cuda.is_available()
else 'cpu')

with open('intents.json', 'r') as json_data:
    intents = json.load(json_data)

FILE = "data.pth"
data = torch.load(FILE)

input_size = data["input_size"]
hidden_size = data["hidden_size"]
output_size = data["output_size"]
all_words = data['all_words']
tags = data['tags']
model_state = data["model_state"]

model = NeuralNet(input_size, hidden_size,
output_size).to(device)
model.load_state_dict(model_state)
model.eval()
```

```
bot_name = "JARVIS"

def get_response(msg):
    sentence = tokenize(msg)
    X = bag_of_words(sentence, all_words)
    X = X.reshape(1, X.shape[0])
    X = torch.from_numpy(X).to(device)

    output = model(X)
    _, predicted = torch.max(output, dim=1)

    tag = tags[predicted.item()]

    probs = torch.softmax(output, dim=1)
    prob = probs[0][predicted.item()]
    if prob.item() > 0.75:
        for intent in intents['intents']:
            if tag == intent["tag"]:
                return random.choice(intent['responses'])

    return "I do not understand..."
```

Step 8:

Once coding the bot we have to give a proper Gui to make it look better. For the we are going to create a simple gui using python gui library call tkinter. Now create a file called app.py and start coding as below.

app.py

```
import random
import json

import torch

from model import NeuralNet
from nltk_utils import bag_of_words, tokenize

device = torch.device('cuda' if torch.cuda.is_available()
else 'cpu')

with open('intents.json', 'r') as json_data:
    intents = json.load(json_data)

FILE = "data.pth"
data = torch.load(FILE)

input_size = data["input_size"]
hidden_size = data["hidden_size"]
output_size = data["output_size"]
all_words = data['all_words']
tags = data['tags']
model_state = data["model_state"]

model = NeuralNet(input_size, hidden_size,
output_size).to(device)
model.load_state_dict(model_state)
model.eval()

bot_name = "JARVIS"

def get_response(msg):
    sentence = tokenize(msg)
    X = bag_of_words(sentence, all_words)
    X = X.reshape(1, X.shape[0])
```

```
    X = torch.from_numpy(X).to(device)

    output = model(X)
    _, predicted = torch.max(output, dim=1)

    tag = tags[predicted.item()]

    probs = torch.softmax(output, dim=1)
    prob = probs[0][predicted.item()]
    if prob.item() > 0.75:
        for intent in intents['intents']:
            if tag == intent["tag"]:
                return random.choice(intent['responses'])

    return "I do not understand..."
```

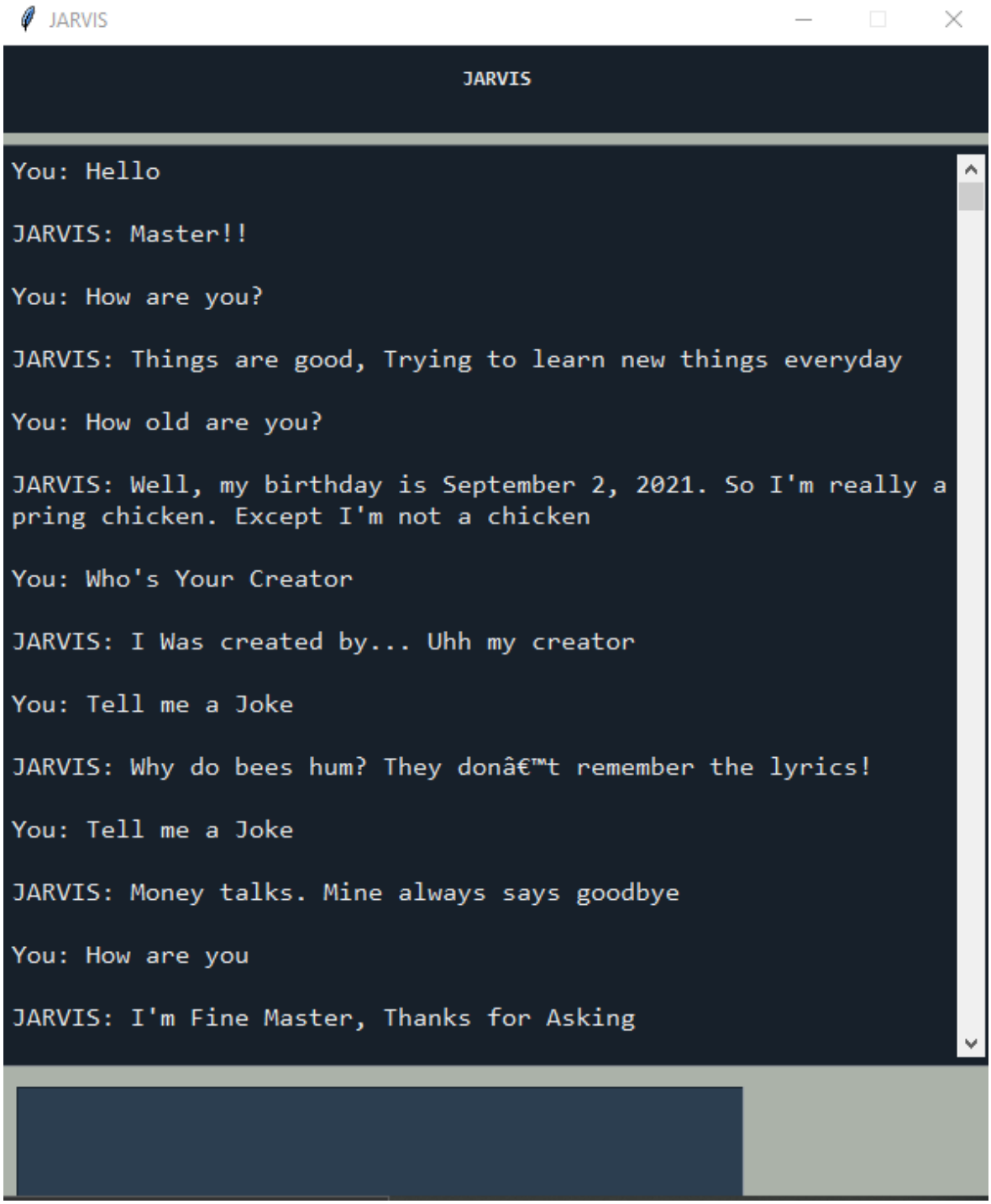
Step 9:

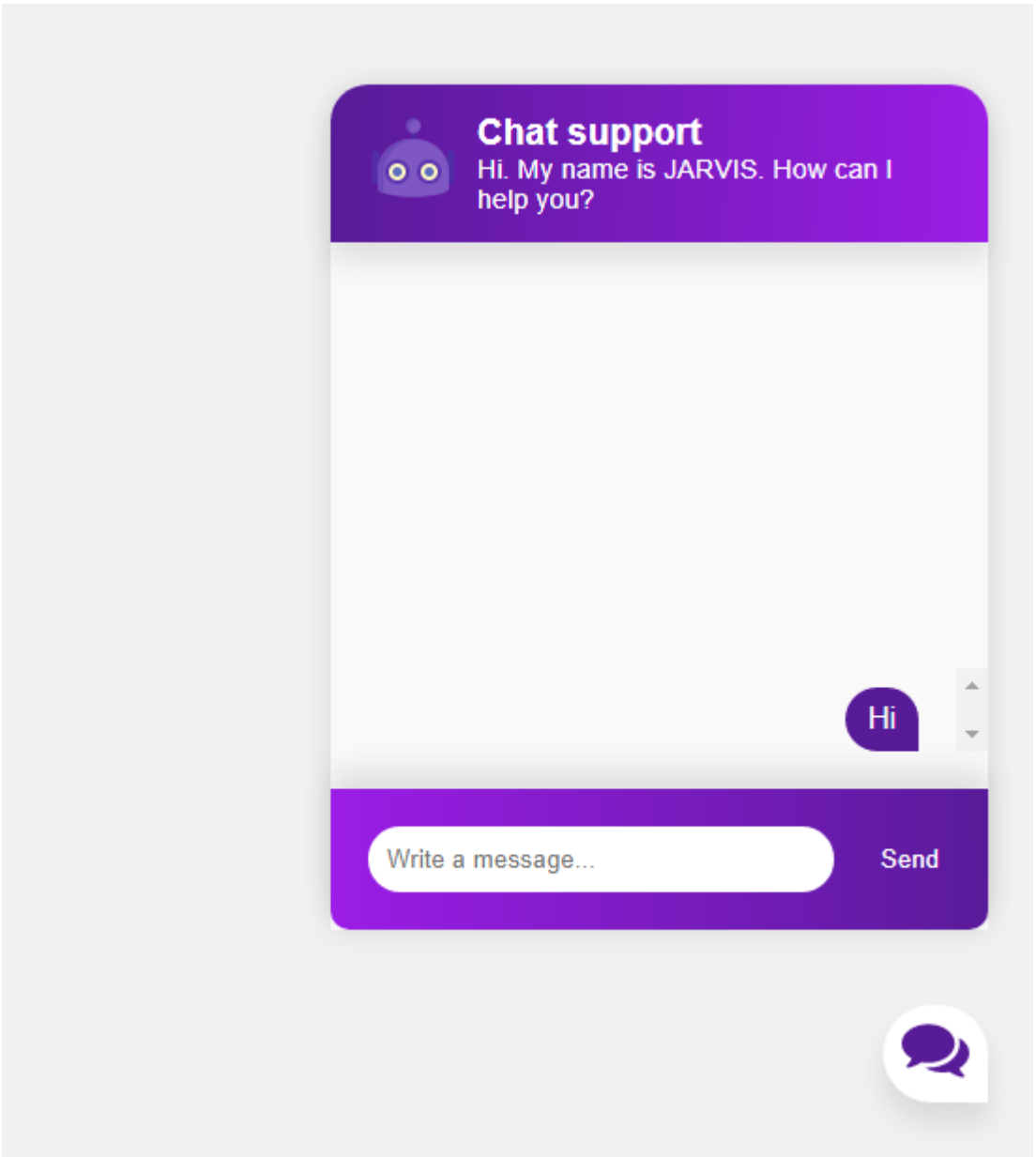
Now we have completed coding out chat bot application. But our bot has not been trained yet. So lets start training our bot. To do this we are going to run the file train.py which we created earlier. This file will train our bot from the list of conversation we given.

Once we stated training our bot it may take a few seconds to few minutes depending upon the size of intents files. Once completed it will create a file called data.pth this is a neural net file which our bot will use to chat (Respond back) to us.

```
d:\Engineering-Projects\Mini-Projects\JARVIS2>C:/Python39/python.exe d:/Engineering-Projects/Mini-Projects/JARVIS2/train.py
12 patterns
4 tags: ['goodbye', 'greetings', 'noanswer', 'wellbeing']
20 unique stemmed words: [',', 'are', 'bye', 'chat', 'do', 'good', 'goodby', 'hello', 'hey', 'hi', 'how', 'jarvi', 'later', 'next', 'n
ice', 'see', 'till', 'time', 'to', 'you']
20 4
Epoch [100/1000], Loss: 1.0870
Epoch [200/1000], Loss: 0.1792
Epoch [300/1000], Loss: 0.0283
Epoch [400/1000], Loss: 0.0146
Epoch [500/1000], Loss: 0.0060
Epoch [600/1000], Loss: 0.0042
Epoch [700/1000], Loss: 0.0018
Epoch [800/1000], Loss: 0.0003
Epoch [900/1000], Loss: 0.0009
Epoch [1000/1000], Loss: 0.0004
final loss: 0.0004
training complete. file saved to data.pth
```

PROTOTYPE


JARVIS
JARVIS
You: Hello
JARVIS: Master!!
You: How are you?
JARVIS: Things are good, Trying to learn new things everyday
You: How old are you?
JARVIS: Well, my birthday is September 2, 2021. So I'm really a pring chicken. Except I'm not a chicken
You: Who's Your Creator
JARVIS: I Was created by... Uhh my creator
You: Tell me a Joke
JARVIS: Why do bees hum? They donâ€™t remember the lyrics!
You: Tell me a Joke
JARVIS: Money talks. Mine always says goodbye
You: How are you
JARVIS: I'm Fine Master, Thanks for Asking


Chat support
Hi. My name is JARVIS. How can I help you?
Hi
Write a message...
Send

Reference

- Bates, M (1995). "Models of natural language understanding". Proceedings of the National Academy of Sciences of the United States of America. 92 (22): 9977–9982. Bibcode:1995PNAS...92.9977B. doi:10.1073/pnas.92.22.9977. PMC 40721. PMID 7479812.

- Steven Bird, Ewan Klein, and Edward Loper (2009). Natural Language Processing with Python. O'Reilly Media. ISBN 978-0-596-51649-9.

- Daniel Jurafsky and James H. Martin (2008). Speech and Language Processing, 2nd edition. Pearson Prentice Hall. ISBN 978-0-13-187321-6.

- E McGaughey, 'Will Robots Automate Your Job Away? Full Employment, Basic Income, and Economic Democracy' (2018) SSRN, part 2(3) Archived 24 May 2018 at the Wayback Machine.

- 2. ^ Bird, Steven; Klein, Ewan; Loper, Edward (2009). Natural Language Processing with Python. O'Reilly Media Inc. ISBN 978-0-596-51649-9.
- 3. ^ Perkins, Jacob (2010). Python Text Processing with NLTK 2.0 Cookbook. Packt Publishing. ISBN 978-1849513609.