**VIT FAQ/Regulations ChatBot using Deep Neural Networking**

*submitted by*

**Pallav Gupta**

**(16BCE0941)**

**B. TECH**

# COMPUTER SCIENCE ENGINEERING



**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**ABSTRACT**

Nowadays it is the era of intelligent machine. With the advancement of artificial intelligent, machine learning and deep learning, machines have started to impersonate as human. The basic definition of chatbot is, it is a computer software program designed to simulate human conversation via text or audio messages.

Today’s AI systems can interact with users, understand their needs, map their preferences and recommend an appropriate line of action with minimal or no human intervention.

The basic foundation of chatbots is providing the best response of any query that it receives. The best response like answering the sender questions, providing sender relevant information, ask follow-up questions and do the conversation in realistic way.

The chatbot needs to be able to understand the intentions of the sender’s message, determine what type of response message (a follow-up question, direct response, etc.) is required, and follow correct grammatical and lexical rules while forming the response.

Some models may use additional Meta information from data, such as speaker id, gender, emotion. Sometimes, sentiment analysis is used toallow the chatbot to ‘understand’ the mood of the user by analysing verbal and sentence structuring clues.

**Keywords— Chatbot, Neural Network, Deep Learning, Intentions, Lexical, Meta-information.**

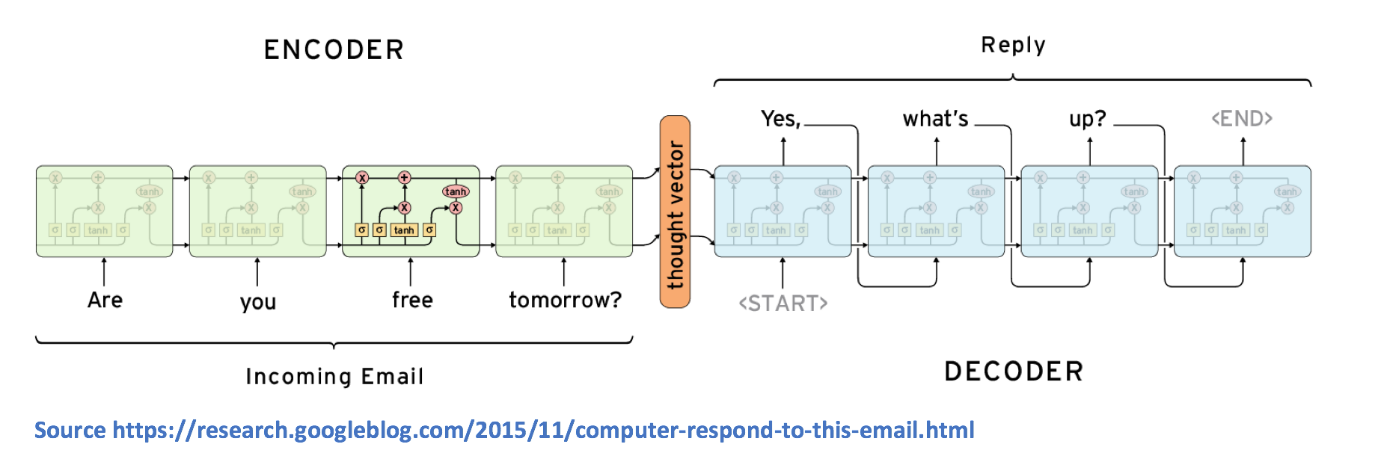
**LITERATURE SURVEY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Authors and Year (Reference)* | *Title (Study)* | *Concept* | *Implementation* | *Relevant Finding* |
| Nuruzzaman Mohammad,  Hussain Omar  2018/10/01 | **A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks** | This paper presented a survey on existing chatbots and techniques applied into it. It discusses the similarities, differences and limitations of the existing chatbots | . They compared 11 most popular chatbot application systems along with functionalities and technical specifications. | Presented why current chatbot models fails to take into account when generating responses and how this affects the quality conversation. |
|  | **Chatbot Development: YOUR GUIDE TO GETTING IT RIGHT** | They talked to several experts to get their thoughts on the key elements of chatbot development at every stage of the process. | Define Your Chatbot Strategy, Get Specific to Deploy Quickly, Integrating Your Chatbot, Go Live, Collect Data, Potential Pitfalls, Continue to Refine After Deployment. | Define Your Chatbot Strategy, Get Specific to Deploy Quickly, Integrating Your Chatbot, Go Live, Collect Data, Potential Pitfalls, Continue to Refine After Deployment |
|  | **A Neural Network based Vietnamese Chatbot** | A simple language model based on the seq2seq framework with attention decoder | A simple language model based on the seq2seq framework with attention decoder | Results showed that it can generate simple and basic conversations and extract knowledge from the generic dataset |
|  | **Improvement of Chatbot in Trading System for SMEs by Using Deep Neural Network** | These chatbots are used for answering questions in many businesses, providing customer information, providing train schedules, helping customer reservations, virtual assistants; serve as call centers to serve ten million customers automatically | . A deep learning based conversational artificial intelligence technique was used as tools for learning conversation between machine and customer. Conjunction with the convolution neural network technique by using Tensorflow training to improve the accuracy of these chatbots. | From the experimental results, using deep learning for chatbots learning, the accuracy is better than the traditional model. |

**PROPOSED SYSTEM**

Chatbots that use deep learning are almost all using some variant of a **sequence to sequence** (Seq2Seq) **model**. In 2014, Ilya Sutskever, Oriol Vinyals, and Quoc Le published the seminal work in this field with a paper called “Sequence to Sequence Learning with Neural Networks”. This paper showed great results in machine translation specifically, but Seq2Seq models have grown to encompass a variety of NLP tasks.

A sequence to sequence model is composed of 2 main components, an encoder RNN and a decoder RNN. From a high level, the encoder’s job is to encapsulate the information of the input text into a fixed representation. The decoder’s is to take that representation, and generate a variable length text that best responds to it.



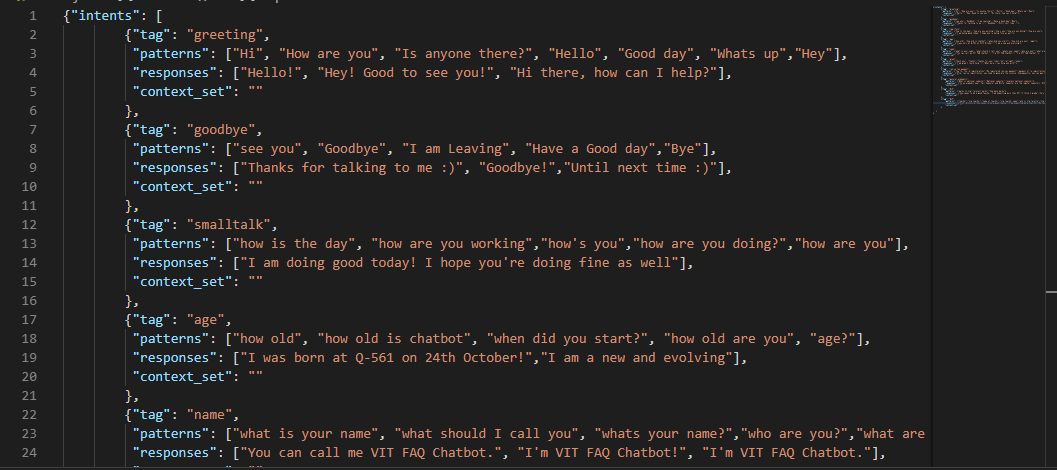
**Dataset Selection**

When thinking about applying machine learning to any sort of task, one of the first things we need to do is consider the type of dataset that we would need to train the model. For sequence to sequence models, we need a large number of conversation logs. From a high level, this encoder decoder network needs to be able to understand the type of responses (decoder outputs) that are expected for every query (encoder inputs).

**Dataset Creation**

A big part of machine learning involves dataset preprocessing. A JSON dataset is created for intent using tag, pattern, response, and context set.

Tag is used as an identifier for the creator to understand the topic to be taken upon. Pattern creates various pattern the user might ask and create an encoder. Response contains various response to a query selected randomly.



**Word Vectors**

To generate word vectors, we use the classic approach of a Word2Vec model. The basic idea is that the model creates word vectors by looking at the context with which words appear in sentences. Words with similar contexts will be placed close together in the vector space.

**Cleaning the Data**

As in any text analysis exercise, the data needs to be cleaned. One deviation we made from the typical data cleaning steps was that we preserved special symbols as most of them corresponded to emoticons. The only cleaning steps we did was to:

* Remove URLs, image references
* Remove stop words

This was done for all of the incoming messages and responses.

JSON file is creating a bunch of messages that the user is likely to type in and mapping them to a group of appropriate responses. The tag on each dictionary in the file indicates the group that each message belongs too. With this data we will train a neural network to take a sentence of words and classify it as one of the tags in our file. Then we can simply take a response from those groups and display that to the user. The more tags, responses, and patterns you provide to the chatbot the better and more complex it will be.

**Loading JSON Data**

Start by importing some modules and loading in json data. Make sure that json file is in the same directory as your python script.

**Extracting Data**

Now its time to take out the data we want from our JSON file. We need all of the patterns and which class/tag they belong to. We also want a list of all of the unique words in our patterns, so let’s setup some blank lists to store these values.

Now its time to loop through our JSON data and extract the data we want. For each pattern we will turn it into a list of words using *nltk.word\_tokenizer*, rather than having them as strings. We will then add each pattern into our docs\_x list and its associated tag into the docs\_y list.

**Word Stemming**

Stemming a word is attempting to find the root of the word. For example, the word "thats" stem might be "that" and the word "happening" would have the stem of "happen". We will use this process of stemming words to reduce the vocabulary of our model and attempt to find the more general meaning behind sentences.

**Bag of Words**

Now that we have loaded in our data and created a stemmed vocabulary we have to create a **bag of words**. As we know neural networks and machine learning algorithms require numerical input. So out list of strings wont cut it. We need some way to represent our sentences with numbers and this is where a bag of words comes in. What we are going to do is represent each sentence with a list the length of the amount of words in our models vocabulary. Each position in the list will represent a word from our vocabulary. If the position in the list is a 1 then that will mean that the word exists in our sentence, if it is a 0 then the word is nor present. We call this a bag of words because the order in which the words appear in the sentence is lost, we only know the presence of words in our models vocabulary.

As well as formatting our input we need to format our output to make sense to the neural network. Similarly to a bag of words we will create output lists which are the length of the amount of labels/tags we have in our dataset. Each position in the list will represent one distinct label/tag, a 1 in any of those positions will show which label/tag is represented.

**Developing a Model**

Now that we have preprocessed all of our data we are ready to start creating and training a model. For our purposes we will use a fairly standard feed-forward neural network with two hidden layers. The goal of our network will be to look at a bag of words and give a class that they belong too (one of our tags from the JSON file).

We will start by defining the architecture of our model. Keep in mind that you can mess with some of the numbers here and try to make an even better model! A lot of machine learning is trial an error.

**Training & Saving the Model**

Now that we have setup our model its time to train it on our data! To do these we will **fit** our data to the model. The number of epochs we set is the amount of times that the model will see the same information while training.

**Making Predictions**

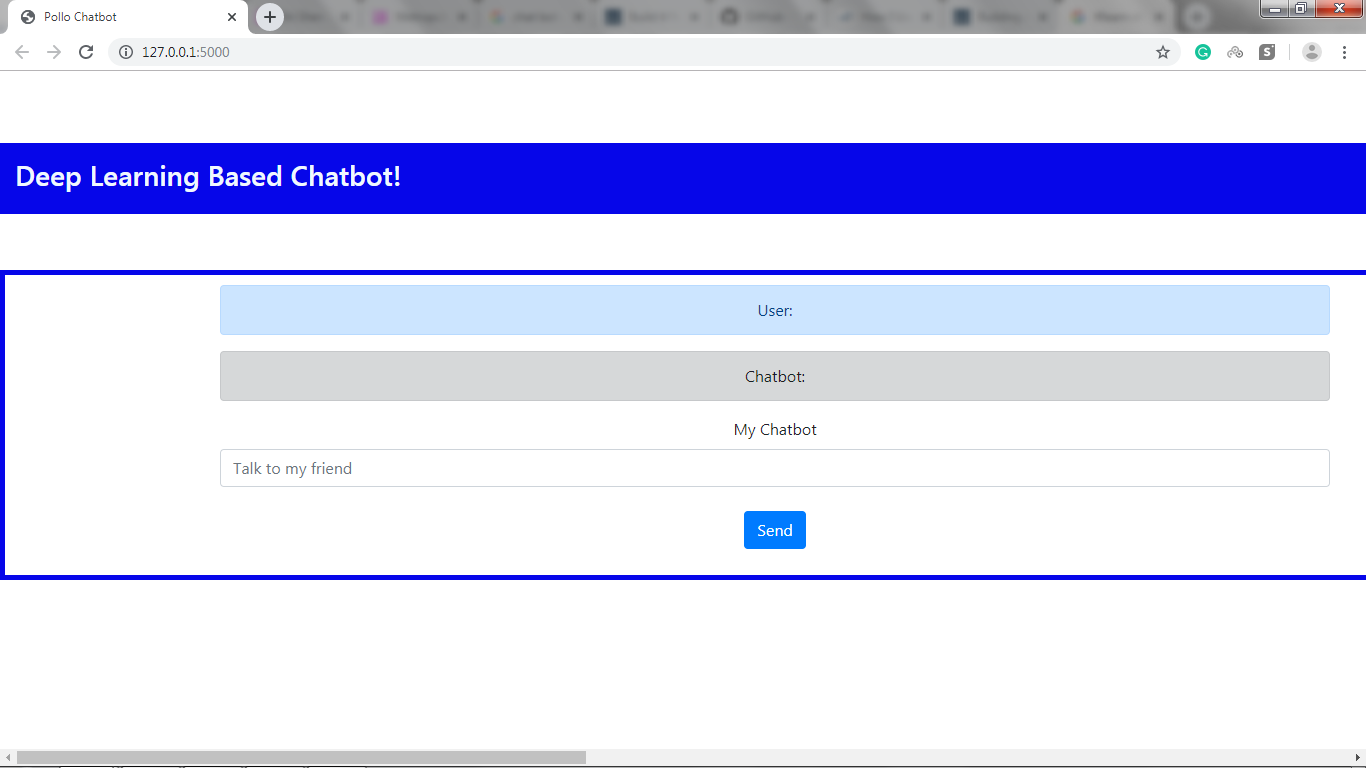
Now its time to actually use the model! Ideally we want to generate a response to any sentence the user types in. To do this we need to remember that our model does not take string input, it takes a bag of words. We also need to realize that our model does not spit out sentences, it generates a list of probabilities for all of our classes. This makes the process to generate a response look like the following:

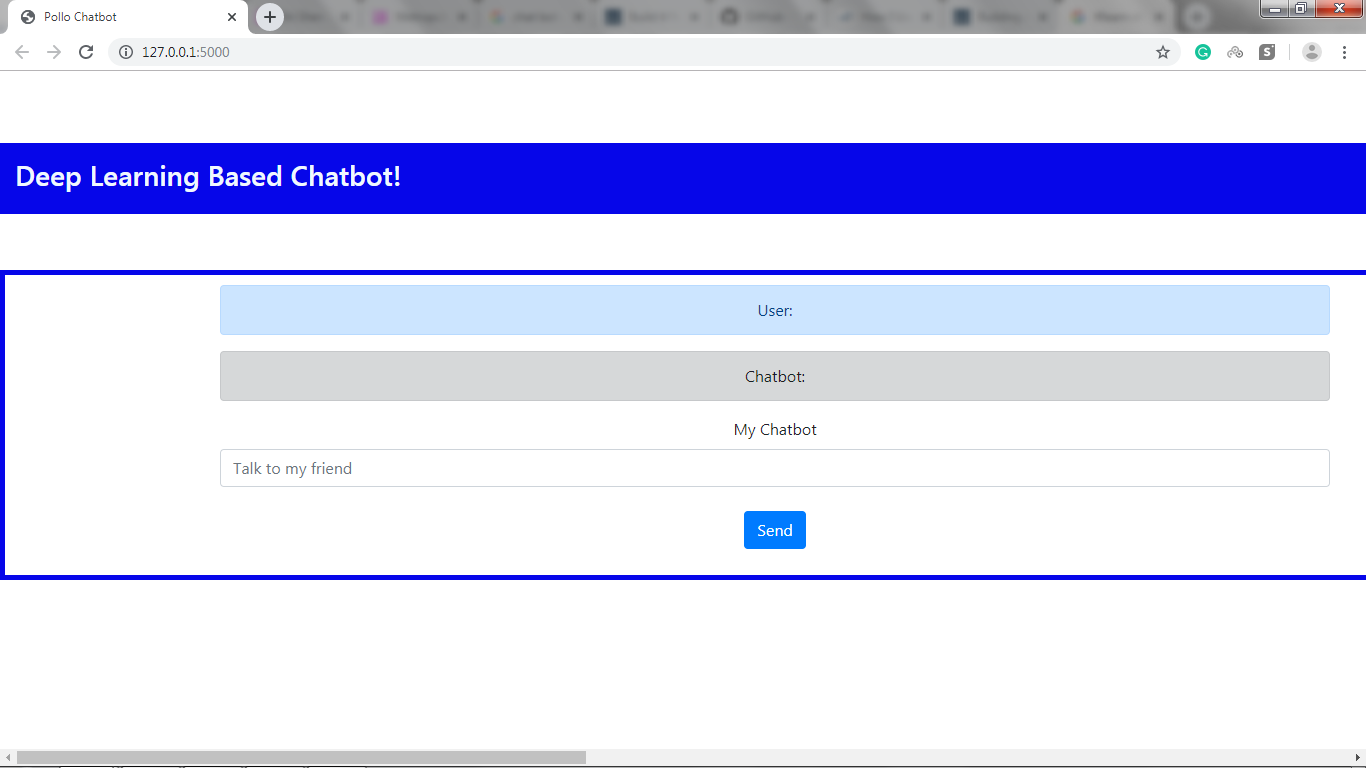
– Get some input from the user  
– Convert it to a bag of words  
– Get a prediction from the model  
– Find the most probable class  
– Pick a response from that class

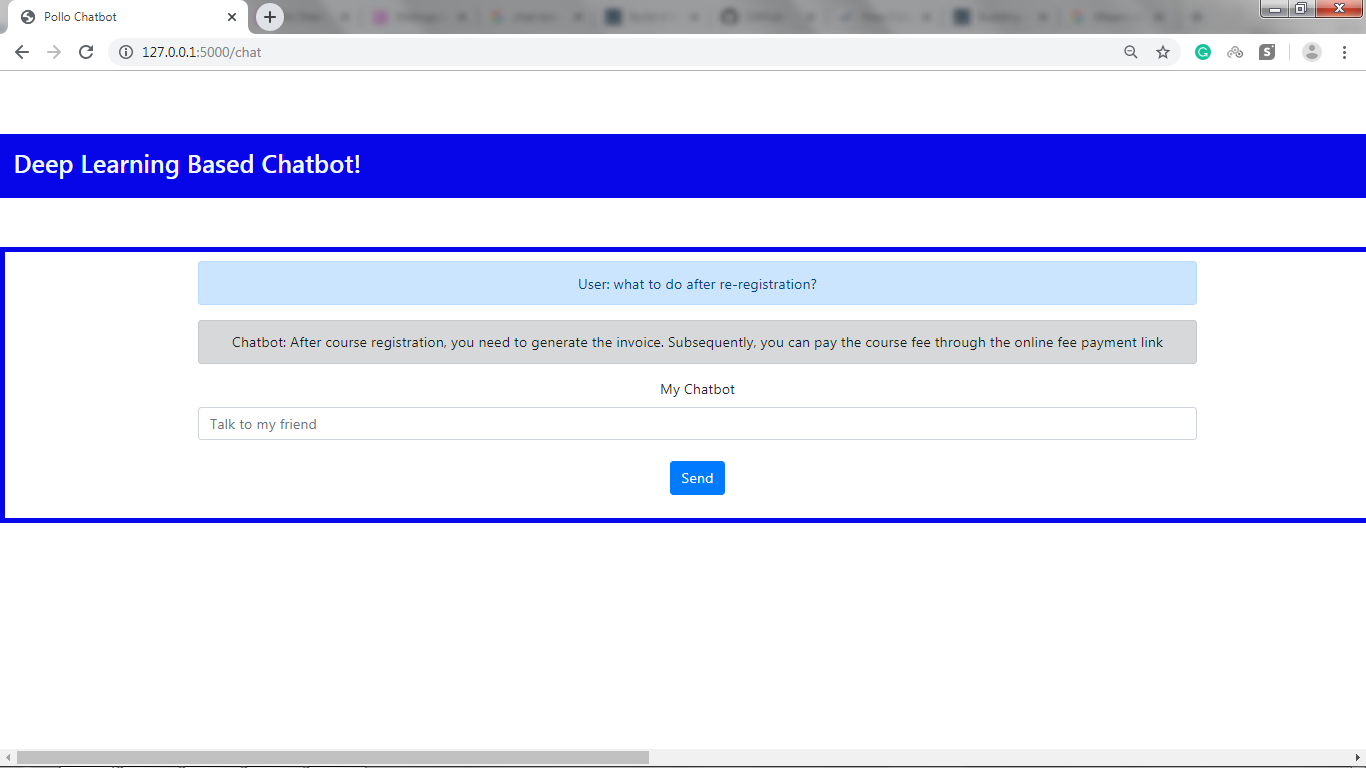
The *bag\_of\_words* function will transform our string input to a bag of words using our created words list. The *chat* function will handle getting a prediction from the model and grabbing an appropriate response from our JSON file of responses.

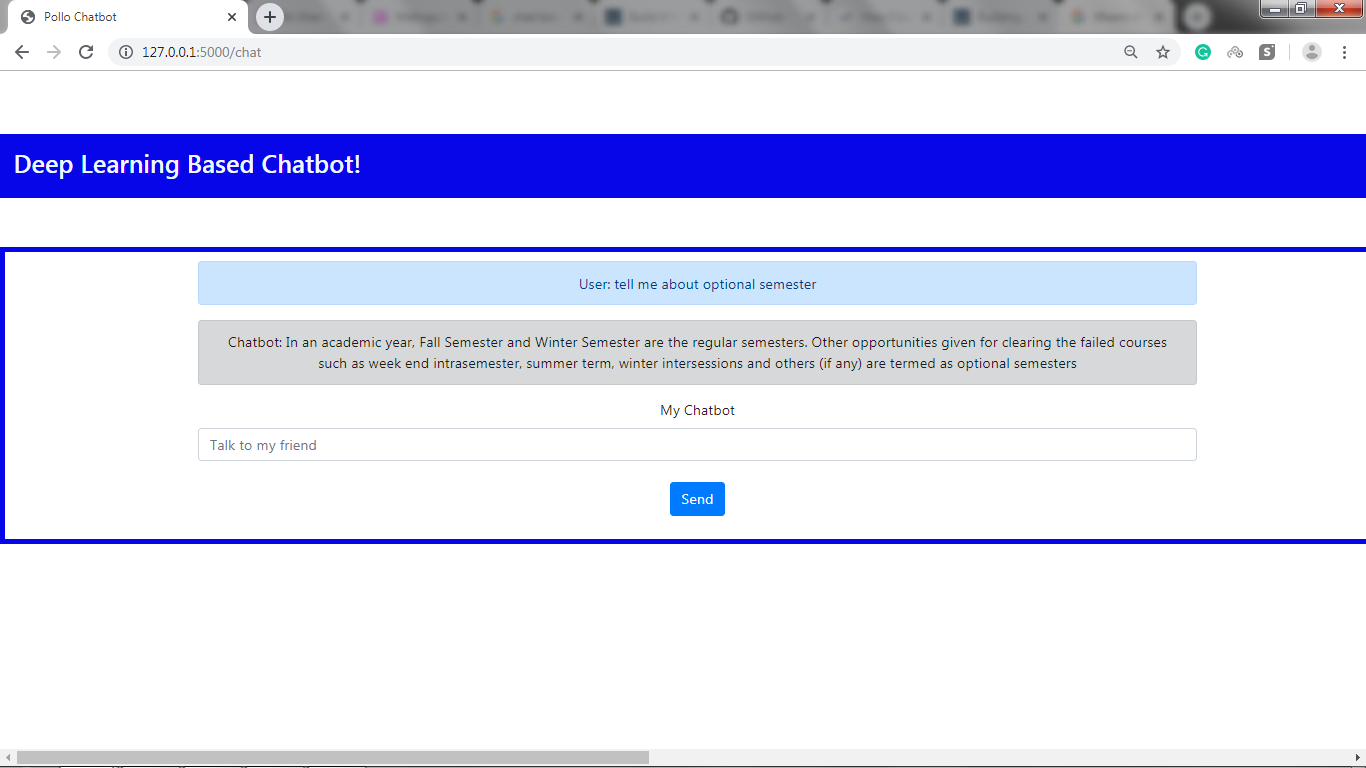
**RESULTS**

Using flask a webpage UI was created to converse with the AI. The user will give an input to which the chat will reply.









**CONCLUSION**

There are several things we can do provide a better user experience when chatting with our bot:

* + If the bot encounters something unfamiliar (i.e. if the highest class probability is below a certain threshold or if the nouns/verbs are not found in the training corpus), it should communicate its confusion. Use a combination of RegEx and Parts of Speech tagger to ask the user to explain what they are talking about. Hopefully, their next explanation will use terms that is familiar to the bot.
  + Evaluate what you want the bot to do and tweak class probabilities accordingly. If the purpose of your chat bot is to act as a stand in for communicating with your friends and you don’t want them to find out, you may want to bump the class probabilities of generic phrases a bit higher to avoid saying something risky.
  + Use all previously sent messages to predict future responses (e.g. use the entire conversation history as textual features) to incorporate the idea of memory.

**REFERENCES**

1. A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks. (2018). *2018 IEEE 15th International Conference on E-Business Engineering (ICEBE), e-Business Engineering (ICEBE), 2018 IEEE 15th International Conference on, ICEBE*, 54. <https://doi.org/10.1109/ICEBE.2018.00019>
2. Improvement of Chatbot in Trading System for SMEs by Using Deep Neural Network. (2019). *2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Cloud Computing and Big Data Analysis (ICCCBDA), 2019 IEEE 4th International Conference On*, 517. <https://doi.org/10.1109/ICCCBDA.2019.8725745>
3. BRITT, P. (2019). Chatbot Development: YOUR GUIDE TO GETTING IT RIGHT. *Speech Technology Magazine*, *24*(3), 15–18. Retrieved from <http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=138113875&site=eds-live>
4. <https://medium.com/@BhashkarKunal/conversational-ai-chatbot-using-deep-learning-how-bi-directional-lstm-machine-reading-38dc5cf5a5a3>
5. A Neural Network based Vietnamese Chatbot. (2018). *2018 International Conference on System Modeling & Advancement in Research Trends (SMART), System Modeling & Advancement in Research Trends (SMART), 2018 International Conference On*, 147. <https://doi.org/10.1109/SYSMART.2018.8746962>

**APPENDIX**

from flask import Flask, render\_template, request

from flask import jsonify

import nltk

from nltk.stem.lancaster import LancasterStemmer

stemmer = LancasterStemmer()

import numpy

import tflearn

import tensorflow

import random

import json

import os

app = Flask(\_\_name\_\_)

with open("intents.json",encoding="utf8") as file:

    data = json.load(file)

words = []

labels = []

docs\_x = []

docs\_y = []

suggestion=[]

for intent in data["intents"]:

    for pattern in intent["patterns"]:

        wrds = nltk.word\_tokenize(pattern)

        words.extend(wrds)

        docs\_x.append(wrds)

        docs\_y.append(intent["tag"])

    if intent["tag"] not in labels:

        labels.append(intent["tag"])

words = [stemmer.stem(w.lower()) for w in words if w != "?"]

words = sorted(list(set(words)))

labels = sorted(labels)

training = []

output = []

out\_empty = [0 for \_ in range(len(labels))]

for x, doc in enumerate(docs\_x):

    bag = []

    wrds = [stemmer.stem(w.lower()) for w in doc]

    for w in words:

        if w in wrds:

            bag.append(1)

        else:

            bag.append(0)

    output\_row = out\_empty[:]

    output\_row[labels.index(docs\_y[x])] = 1

    training.append(bag)

    output.append(output\_row)

training = numpy.array(training)

output = numpy.array(output)

tensorflow.reset\_default\_graph()

net = tflearn.input\_data(shape=[None,len(training[0])])

net = tflearn.fully\_connected(net,8)

net = tflearn.fully\_connected(net,8)

net = tflearn.fully\_connected(net,len(output[0]),activation="softmax")

net = tflearn.regression(net)

model = tflearn.DNN(net)

model.fit(training,output,n\_epoch=25000,batch\_size=50,show\_metric=True)

model.save("chatbot.tflearn")

def bag\_of\_words(s, words):

    bag = [0 for \_ in range(len(words))]

    s\_words = nltk.word\_tokenize(s)

    s\_words = [stemmer.stem(word.lower()) for word in s\_words]

    for se in s\_words:

        for i, w in enumerate(words):

            if w == se:

                bag[i] = 1

    return numpy.array(bag)

@app.route('/')

def index():

    return render\_template('index.html')

@app.route('/chat',methods=["POST"])

def chat():

    #while True:

     ##  if inp.lower() == "quit":

       #     break

    user\_input = request.form['user\_input']

    results = model.predict([bag\_of\_words(user\_input, words)])[0]

    results\_index = numpy.argmax(results)

    tag = labels[results\_index]

    if results[results\_index] >0.6:

        for tg in data["intents"]:

            if tg['tag'] == tag:

                responses = tg['responses']

        resval=random.choice(responses)

    else:

        resval="Sorry I don't understand,try asking something else."

    for tg in data["intents"]:

        if tg['tag'] == tag:

            suggestion=(tg['patterns'])

            if resval =="Sorry I don't understand,try asking something else.":

                suggestion = "You can try:"+', '.join(tg['patterns'])

        return render\_template('index.html',user\_input=user\_input,bot\_response=resval)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

import nltk

from nltk.stem.lancaster import LancasterStemmer

stemmer = LancasterStemmer()

import numpy

import tflearn

import tensorflow

import random

import json

with open("intents.json") as file:

    data = json.load(file)

words = []

labels = []

docs\_x = []

docs\_y = []

for intent in data["intents"]:

    for pattern in intent["patterns"]:

        wrds = nltk.word\_tokenize(pattern)

        words.extend(wrds)

        docs\_x.append(wrds)

        docs\_y.append(intent["tag"])

    if intent["tag"] not in labels:

        labels.append(intent["tag"])

words = [stemmer.stem(w.lower()) for w in words if w != "?"]

words = sorted(list(set(words)))

labels = sorted(labels)

training = []

output = []

out\_empty = [0 for \_ in range(len(labels))]

for x, doc in enumerate(docs\_x):

    bag = []

    wrds = [stemmer.stem(w.lower()) for w in doc]

    for w in words:

        if w in wrds:

            bag.append(1)

        else:

            bag.append(0)

    output\_row = out\_empty[:]

    output\_row[labels.index(docs\_y[x])] = 1

    training.append(bag)

    output.append(output\_row)

training = numpy.array(training)

output = numpy.array(output)

tensorflow.reset\_default\_graph()

net = tflearn.input\_data(shape=[None,len(training[0])])

net = tflearn.fully\_connected(net,8)

net = tflearn.fully\_connected(net,8)

net = tflearn.fully\_connected(net,len(output[0]),activation="softmax")

net = tflearn.regression(net)

model = tflearn.DNN(net)

model.fit(training,output,n\_epoch=1250,batch\_size=8,show\_metric=True)

model.save("chatmodel.tflearn")

def bag\_of\_words(s, words):

    bag = [0 for \_ in range(len(words))]

    s\_words = nltk.word\_tokenize(s)

    s\_words = [stemmer.stem(word.lower()) for word in s\_words]

    for se in s\_words:

        for i, w in enumerate(words):

            if w == se:

                bag[i] = 1

    return numpy.array(bag)