Chatbot – Code Snippets for Backend

* **Importing the libraries**

import io

import random

import string # to process standard python strings

import warnings

import numpy as np

import numpy.linalg as LA

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

import warnings

warnings.filterwarnings('ignore')

import nltk

from nltk.stem import WordNetLemmatizer

from nltk.corpus import stopwords

* **Reading the data from training set – ChatBot.txt**

with open('ChatBot.txt','r', encoding='utf8', errors ='ignore') as fin:

raw = fin.read().lower()

lines = []

with open('ChatBot.txt') as fp:

contents = fp.read()

for entry in contents.split('|'):

lines.append(entry)

sent\_tokens = nltk.sent\_tokenize(raw)# converts to list of sentences

#print(sent\_tokens)

#print(len(sent\_tokens))

word\_tokens = nltk.word\_tokenize(raw)

* **Pre-Processing the raw text**

lemmer = WordNetLemmatizer()

def LemTokens(tokens):

return [lemmer.lemmatize(token) for token in tokens]

remove\_punct\_dict = dict((ord(punct), None) for punct in string.punctuation)

def LemNormalize(text):

return LemTokens(nltk.word\_tokenize(text.lower().translate(remove\_punct\_dict)))

* **Generating chatbot response**

def practice(user\_response):

robo\_response = ''

train\_set = sent\_tokens

test\_set = nltk.sent\_tokenize(user\_response)

stopWords = stopwords.words('english')

max = 0

#vectorizer = CountVectorizer(stop\_words = stopWords)

TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop\_words='english')

trainVectorizerArray = TfidfVec.fit\_transform(train\_set).toarray()

testVectorizerArray = TfidfVec.transform(test\_set).toarray()

cx = lambda a, b : round(np.inner(a, b)/(LA.norm(a)\*LA.norm(b)), 3)

count = 0

finalindex = -1

for index in range(len(trainVectorizerArray)):

for testV in testVectorizerArray:

cosine = cx(trainVectorizerArray[index], testV)

#print("cosine : "+str(cosine)+" : "+str(train\_set[index]))

if cosine > max:

max = cosine

finalindex = index

#print("max cosine :"+str(max))

#print("index : "+train\_set[finalindex])

if(max==0):

robo\_response=robo\_response+"I am sorry! I don't understand you"

return robo\_response

else:

#print(len(train\_set))

#print(len(lines))

#print(str(finalindex))

robo\_response = lines[finalindex]

return robo\_response

**Flask:**

@app.route('/')

def sessions():

return render\_template('ChatBotUI.html')

def messageReceived(methods=['GET', 'POST']):

print('message was received!!!')

* **Start the Conversation**

GREETING\_INPUTS = ("hello", "hi", "greetings", "sup", "what's up","hey",)

GREETING\_RESPONSES = ["hi", "hey", "hi there", "hello", "Hi there. I am glad! You are talking to me"]

@socketio.on('my event')

def handle\_my\_custom\_event(json, methods=['GET', 'POST']):

print('received my event: ' + str(json))

mes = str(json).split(":",1)[1]

mes = mes.strip()

user\_response = mes[1:len(mes)-2].strip()

hello = 'reply'

choice = ''

if(user\_response!='bye'):

if(user\_response=='thanks' or user\_response=='thank you' ):

choice = "You are welcome.."

else:

for word in user\_response.split():

if word.lower() in GREETING\_INPUTS:

choice = random.choice(GREETING\_RESPONSES)

break;

else:

choice = practice(user\_response)

#practice(user\_response)

#sent\_tokens.remove(user\_response)

else:

choice = "Bye! take care.."

socketio.emit('my response', {hello: choice})

* **Run the app**

if \_\_name\_\_ == '\_\_main\_\_':

socketio.run(app, debug=True)