CHATBOT CODE

pip install tensorflow, keras, pickle, nltk

Here are the 5 steps to create a chatbot in Python :

1. Import and load the data file
2. Preprocess data
3. Create training and testing data
4. Build the model
5. Predict the response

**Import and load the data file**

import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import json

import pickle

import numpy as np

from keras.models import Sequential

from keras.layers import Dense, Activation, Dropout

from keras.optimizers import SGD

import random

words=[]

classes = []

documents = []

ignore\_words = ['?', '!']

data\_file = open('intents.json').read()

intents = json.loads(data\_file)

**Preprocess data**

for intent **in** intents['intents']:

**for** pattern **in** intent['patterns']:

#tokenize each word

w = nltk.word\_tokenize(pattern)

words.extend(w)

#add documents in the corpus

documents.append((w, intent['tag']))

# add to our classes list

**if** intent['tag'] not **in** classes:

classes.append(intent['tag'])

# lemmatize, lower each word and remove duplicates

words = [lemmatizer.lemmatize(w.lower()) **for** w **in** words **if** w not **in** ignore\_words]

words = sorted(list(set(words)))

# sort classes

classes = sorted(list(set(classes)))

# documents = combination between patterns and intents

print (len(documents), "documents")

# classes = intents

print (len(classes), "classes", classes)

# words = all words, vocabulary

print (len(words), "unique lemmatized words", words)

pickle.dump(words,open('words.pkl','wb'))

pickle.dump(classes,open('classes.pkl','wb'))

**Create training and testing data**

# create our training data

training = []

# create an empty array for our output

output\_empty = [0] \* len(classes)

# training set, bag of words for each sentence

**for** doc **in** documents:

# initialize our bag of words

bag = []

# list of tokenized words for the pattern

pattern\_words = doc[0]

# lemmatize each word - create base word, in attempt to represent related words

pattern\_words = [lemmatizer.lemmatize(word.lower()) **for** word **in** pattern\_words]

# create our bag of words array with 1, if word match found in current pattern

**for** w **in** words:

bag.append(1) **if** w **in** pattern\_words **else** bag.append(0)

# output is a '0' for each tag and '1' for current tag (for each pattern)

output\_row = list(output\_empty)

output\_row[classes.index(doc[1])] = 1

training.append([bag, output\_row])

# shuffle our features and turn into np.array

random.shuffle(training)

training = np.array(training)

# create train and test lists. X - patterns, Y - intents

train\_x = list(training[:,0])

train\_y = list(training[:,1])

print("Training data created")

**Build the model**

# Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd output layer contains number of neurons

# equal to number of intents to predict output intent with softmax

model = Sequential()

model.add(Dense(128, input\_shape=(len(train\_x[0]),), activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(len(train\_y[0]), activation='softmax'))

# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results for this model

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=**True**)

model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])

#fitting and saving the model

hist = model.fit(np.array(train\_x), np.array(train\_y), epochs=200, batch\_size=5, verbose=1)

model.save('chatbot\_model.h5', hist)

print("model created")

**Predict the response (Graphical User Interface)**

import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import pickle

import numpy as np

from keras.models import load\_model

model = load\_model('chatbot\_model.h5')

import json

import random

intents = json.loads(open('intents.json').read())

words = pickle.load(open('words.pkl','rb'))

classes = pickle.load(open('classes.pkl','rb'))

def clean\_up\_sentence(sentence):

# tokenize the pattern - split words into array

sentence\_words = nltk.word\_tokenize(sentence)

# stem each word - create short form for word

sentence\_words = [lemmatizer.lemmatize(word.lower()) **for** word **in** sentence\_words]

**return** sentence\_words

# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence

**def** bow(sentence, words, show\_details=**True**):

# tokenize the pattern

sentence\_words = clean\_up\_sentence(sentence)

# bag of words - matrix of N words, vocabulary matrix

bag = [0]\*len(words)

**for** s **in** sentence\_words:

**for** i,w **in** enumerate(words):

**if** w == s:

# assign 1 if current word is in the vocabulary position

bag[i] = 1

**if** show\_details:

print ("found in bag: %s" % w)

**return**(np.array(bag))

**def** predict\_class(sentence, model):

# filter out predictions below a threshold

p = bow(sentence, words,show\_details=**False**)

res = model.predict(np.array([p]))[0]

ERROR\_THRESHOLD = 0.25

results = [[i,r] **for** i,r **in** enumerate(res) **if** r>ERROR\_THRESHOLD]

# sort by strength of probability

results.sort(key=lambda x: x[1], reverse=**True**)

return\_list = []

**for** r **in** results:

return\_list.append({"intent": classes[r[0]], "probability": str(r[1])})

**return** return\_list

def getResponse(ints, intents\_json):

tag = ints[0]['intent']

list\_of\_intents = intents\_json['intents']

**for** i **in** list\_of\_intents:

**if**(i['tag']== tag):

result = random.choice(i['responses'])

break

**return** result

**def** chatbot\_response(text):

ints = predict\_class(text, model)

res = getResponse(ints, intents)

**return** res

#Creating GUI with tkinter

import tkinter

from tkinter import \*

**def** send():

msg = EntryBox.get("1.0",'end-1c').strip()

EntryBox.delete("0.0",**END**)

**if** msg != '':

ChatLog.config(state=NORMAL)

ChatLog.insert(**END**, "You: " + msg + '\n\n')

ChatLog.config(foreground="#442265", font=("Verdana", 12 ))

res = chatbot\_response(msg)

ChatLog.insert(**END**, "Bot: " + res + '\n\n')

ChatLog.config(state=DISABLED)

ChatLog.yview(**END**)

base = Tk()

base.title("Hello")

base.geometry("400x500")

base.resizable(width=**FALSE**, height=**FALSE**)

#Create Chat window

ChatLog = Text(base, bd=0, bg="white", height="8", width="50", font="Arial",)

ChatLog.config(state=DISABLED)

#Bind scrollbar to Chat window

scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")

ChatLog['yscrollcommand'] = scrollbar.set

#Create Button to send message

SendButton = Button(base, font=("Verdana",12,'bold'), text="Send", width="12", height=5,

bd=0, bg="#32de97", activebackground="#3c9d9b",fg='#ffffff',

command= send )

#Create the box to enter message

EntryBox = Text(base, bd=0, bg="white",width="29", height="5", font="Arial")

#EntryBox.bind("<Return>", send)

#Place all components on the screen

scrollbar.place(x=376,y=6, height=386)

ChatLog.place(x=6,y=6, height=386, width=370)

EntryBox.place(x=128, y=401, height=90, width=265)

SendButton.place(x=6, y=401, height=90)

base.mainloop()

**Run the chatbot**

python train\_chatbot.py

python chatgui.py