Impala Chatbot

Name: Bangaru Mohnish

Reg.No: RA1711003011397

School: SRM IST

Abstract

Automated Customer service chatbot

The Success of an organization depends essentially on the customer interaction and the reviews given by them since it is highly difficult for an organization with a huge userbase to respond to the queries of each and every customer automation is the key to achieve this hence Impala can be incorporated to any organization by customizing the language data according to needs.

Libraries Used:

1.Keras

2.Numpy

3.json

4.random

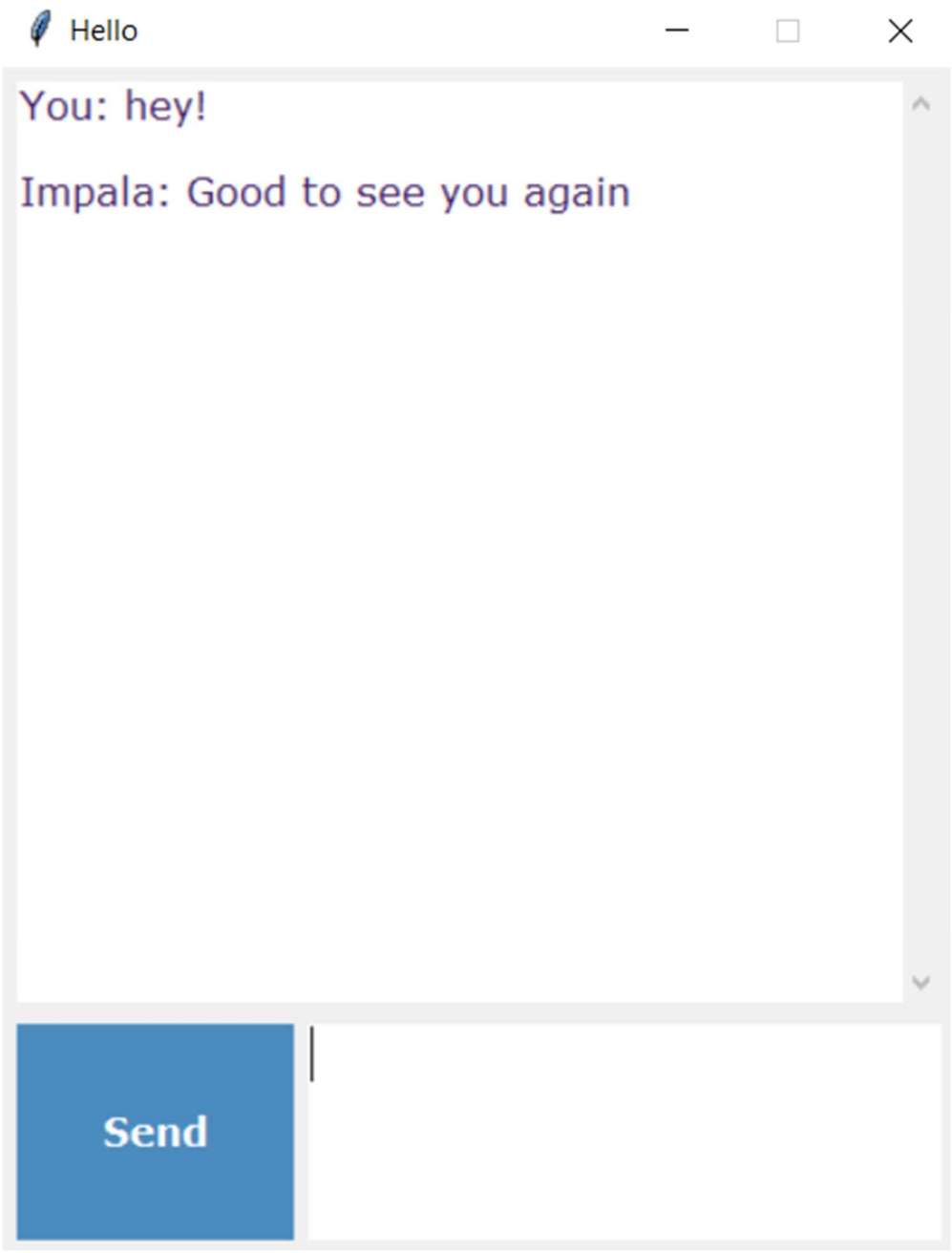
5.pickle

6.tkinter

Data Used:

1.intents.json(custom made)

Outputs:



Implementation:

The chatbot has been developed in two modules

1.Chatbot: This file contains the training data creation, model creation and training using Tensorflow.

2.ChatbotUI: This file contains the user interface for the chatbot developed using Tkinter

Conclusion:

This Project has helped me gain a better insight of the stock market functioning and an approach to solve the problems faced by investors using my technical skills

Appendix:

UI implementation:

```
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('D:\Academics\Projects\Chatbot\intents.json').read())
words = pickle.load(open('D:\Academics\Projects\Chatbot\words.pkl','rb'))
classes = pickle.load(open('D:\Academics\Projects\Chatbot\classes.pkl','rb'))


def clean_up_sentence(sentence):
    sentence_words = nltk.word_tokenize(sentence)
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_wor
ds]
    return sentence_words


def bow(sentence, words, show_details=True):
    sentence_words = clean_up_sentence(sentence)
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))

def predict_class(sentence, model):
    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]
    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(msg):
    ints = predict_class(msg, model)
    res = getResponse(ints, intents)
    return res

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, "Impala: " + res + '\n\n')

        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)


base = Tk()
base.title("Hello")
```

```
base.geometry("400x500")
base.resizable(width=FALSE, height=FALSE)

ChatLog = Text(base, bd=0, bg="white", height="8", width="30", font="Arial",)

ChatLog.config(state=DISABLED)

scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
ChatLog['yscrollcommand'] = scrollbar.set

SendButton = Button(base, font=("Verdana",12,'bold'), text="Send", width="10", he
ight="0",
                    bd=0, bg="#4B8BBE", activebackground="#3c9d9b",fg='#ffffff',
                    command= send )

EntryBox = Text(base, bd=0, bg="white",width="30", height="2", font="Arial")


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()
```

Model Implementation:

```
import nltk
nltk.download('punkt')
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle

import numpy as np
from keras import backend
from keras.models import Sequential, load_model
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random
word_data=[]
class_data = []
documents = []
ig_words = ['?', '!']
```

```
data_file = open('D:\Academics\Projects\Chatbot\intents.json').read()
intents = json.loads(data_file)


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

        # take each word and tokenize it
        w = nltk.word_tokenize(pattern)
        word_data.extend(w)
        # adding documents
        documents.append((w, intent['tag']))

        # adding classes to our class list
        if intent['tag'] not in class_data:
            class_data.append(intent['tag'])

words = [lemmatizer.lemmatize(w.lower()) for w in word_data if w not in ig_words]
words = sorted(list(set(word_data)))

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

print (len(documents), "documents")

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

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


pickle.dump(word_data,open('words.pkl','wb'))
pickle.dump(class_data,open('classes.pkl','wb'))

training = []
output_empty = [0] * len(class_data)

for doc in documents:
    bag = []
    pattern_words = doc[0]

    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words
]

    for w in word_data:
        bag.append(1) if w in pattern_words else bag.append(0)
```

```
    output_row = list(output_empty)
    output_row[class_data.index(doc[1])] = 1

    training.append([bag, output_row])
random.shuffle(training)
training = np.array(training)
train_x = list(training[:,0])
train_y = list(training[:,1])
print("Training data created")


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'))

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'
])

hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5,
verbose=1)
model.save('D:\Academics\Projects\Chatbot\chatbot_model.h5', hist)

print("model created")
```