Chatbot based Song Recommender System

By

Aabshar Shaikh (20200812016)

Pratibha Desai (20200812025)

Anjali Kumari (20200812029)

CODE:

- intents.json file :

```
{
"intents": [
  {"tag": "greeting",
    "patterns": ["Hi", "Is anyone there?", "Hello", "Whats up",
"Hey"],
    "responses": ["Hello!", "Good to see you again!", "Hi
there, how can I help?", "Hey"],
    "context_set": ""
  },

  {"tag": "goodbye",
    "patterns": ["cya", "See you later", "Goodbye", "I am
Leaving", "bye", "see ya"],
    "responses": ["Sad to see you go :(", "Talk to you later",
"Goodbye!", "Bye,Have a Good day", "Bye!"],
    "context_set": ""
  },

  {"tag": "thanks",
    "patterns": ["Thanks", "Thank you", "That's helpful",
"Thanks a million", "Thanks a lot"],
    "responses": ["Happy to help!", "Any time!", "My
pleasure"],
    "context_set": ""
  },

  {"tag": "happy",
    "patterns": ["I am happy", "very happy", "Happy", "I’ve
never been this happy before", "I am extremely happy"],
   "responses": ["SONG RECOMMENDED: Dynamite By
BTS", "SONG RECOMMENDED: ME! By Taylor Swift &
Brendon Uri", "SONG RECOMMENDED: Shower By
Becky G", "SONG RECOMMENDED: Butter By BTS",
"SONG RECOMMENDED: Watermelon Sugar By Harry
Styles"],
   "context_set": ""
  },

  {"tag": "sad",
   "patterns": ["I am sad", "I’m sad", "sad", "My heart is too
full of sadness", "I sensed some deep sadness"],
   "responses": ["SONG RECOMMENDED: Someone You
Loved By Lewis Capaldi", "SONG RECOMMENDED: Lose
You To Love Me By Selena Gomez", "SONG
RECOMMENDED: See You Again By Wiz Khalifa ft.
Charlie Puth", "SONG RECOMMENDED: Lovely By Billie
Eilish & Khalid", "SONG RECOMMENDED: Spring Day
By BTS"],
   "context_set": ""
  },

 {"tag": "relax",
   "patterns": ["I am feeling relax", "relax", "clam", "I just
want to relax", "A bit of music will help me relax"],
   "responses": ["SONG RECOMMENDED: Willow By
Taylor Swift", "SONG RECOMMENDED: Drivers License
By Olivia Rodrigo", "SONG RECOMMENDED: pov By
Ariana Grande", "SONG RECOMMENDED: At My Worst
By Pink Sweat$", "SONG RECOMMENDED: Memories By
Maroon 5"],
   "context_set": ""
  },
  {"tag": "feelingdown",
    "patterns": ["I am feeling down", "feelingdown", "feeling
down", "i am not feeling good", "not feeling good"],
    "responses": ["SONG RECOMMENDED: Fix You By
Coldplay", "SONG RECOMMENDED: Lovely By Billie
Eilish & Khalid", "SONG RECOMMENDED: Scientist By
Coldplay"],
    "context_set": ""
  },

  {"tag": "loneliness",
    "patterns": ["I am feeling loneliness", "lonely", " I am
lonely"],
    "responses": ["SONG RECOMMENDED: Dancing on my
own By Calum Scott", "SONG RECOMMENDED: Wake Up
Alone By The Chainsmokers", "SONG RECOMMENDED:
Scared to Be Lonely By Martin Garrix & Dua Lipa", "SONG
RECOMMENDED: Tired of Being Alone By Al Green",
"SONG RECOMMENDED: Loneliness By Birdy"],
    "context_set": ""
  },

  {"tag": "angry",
    "patterns": ["I am feeling angry", "angry", " I am angry
now", "very angry"],
    "responses": ["SONG RECOMMENDED: So What By
Pink", "SONG RECOMMENDED: Without Me By Halsey",
"SONG RECOMMENDED: Sorry Not Sorry By Demi
Lovato", "SONG RECOMMENDED: Look What You Made
Me Do By Taylor Swift", "SONG RECOMMENDED: Bad
Blood By Taylor Swift"],
    "context_set": ""
  },
  {"tag": "surprise",
    "patterns": ["surprise", "I received a big surprise", "Things still surprise me", "I am total surprise"],
    "responses": ["SONG RECOMMENDED: Surprise Surprise By Billy Talent", "SONG RECOMMENDED: Surprise me By Yuvan Shankar Raja", "SONG RECOMMENDED: Surprise me By Mahalia", "SONG RECOMMENDED: No Surprise by Daughtry", "SONG RECOMMENDED: A wonderful Surprise By The Downtown Fiction"],
    "context_set": ""
  }
]
}
```

- <u>training.py :</u>

```
import random
import json
import pickle

import hist as hist
import numpy as np

import nltk
from nltk.stem import WordNetLemmatizer

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import SGD

lemmatizer = WordNetLemmatizer()

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

words = []
classes = []
documents = []
ignore_letters = ['?', '!', '.', ',']

for intent in intents['intents']:
    for pattern in intent['patterns']:
        word_list = nltk.word_tokenize(pattern)
        words.extend(word_list)
        documents.append((word_list, intent['tag']))
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

words = [lemmatizer.lemmatize(word) for word in words if
word not in ignore_letters]
words = sorted(set(words))

classes = sorted(set(classes))

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

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

for document in documents:
    bag = []
    word_patterns = document[0]
    word_patterns = [lemmatizer.lemmatize(word.lower()) for
word in word_patterns]
    for word in words:
        bag.append(1) if word in word_patterns else
bag.append(0)

        output_row = list(output_empty)
        output_row[classes.index(document[1])] = 1
        training.append([bag, output_row])

random.shuffle(training)
training = np.array(training)

train_x = list(training[:, 0])
train_y = list(training[:, 1])

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('chatbotmodel.h5', hist)
print("Done")
```

- chatbot.py :

```
import random
import json
import pickle
import numpy as np

import nltk
from nltk.stem import WordNetLemmatizer

from tensorflow.keras.models import load_model

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

words = pickle.load(open('words.pkl', 'rb'))
classes = pickle.load(open('classes.pkl', 'rb'))
model = load_model('chatbotmodel.h5')


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

def bag_of_words(sentence):
    sentence_words = clean_up_sentence(sentence)
    bag = [0] * len(words)
    for w in sentence_words:
        for i, word in enumerate(words):
            if word == w:
                bag[i] = 1
    return np.array(bag)

def predict_class(sentence):
    bow = bag_of_words(sentence)
    res = model.predict(np.array([bow]))[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 get_response(intents_list, intents_json):
    tag = intents_list[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

print("-----Hello! How Can I Help You...-----")

while True:
    message = input("")
    ints = predict_class(message)
    res = get_response(ints, intents)
    print(res)
```