

Chatbot Report

Description

The purpose of this project was to put to use the concepts that we learned in this class. We were given freedom over what kind of application or project we wanted to create as long as it included deep learning and natural language processing like we learned in this class.

Objective

The objective was to create a project using NLP. The requirements for this project were for the project to be made using NLP concepts and modeling, using unstructured data, using python deep learning techniques, not using pre-built models, and to ensure that the project scope is not narrow. There were a few suggestions given for projects including a customer support bot, a language identifier, an ML-powered autocomplete feature, a predictive text generator, or a media monitor. The possibilities were pretty much endless, but for the purpose of this project, the chatbot was chosen as it seemed the most interesting and fun.

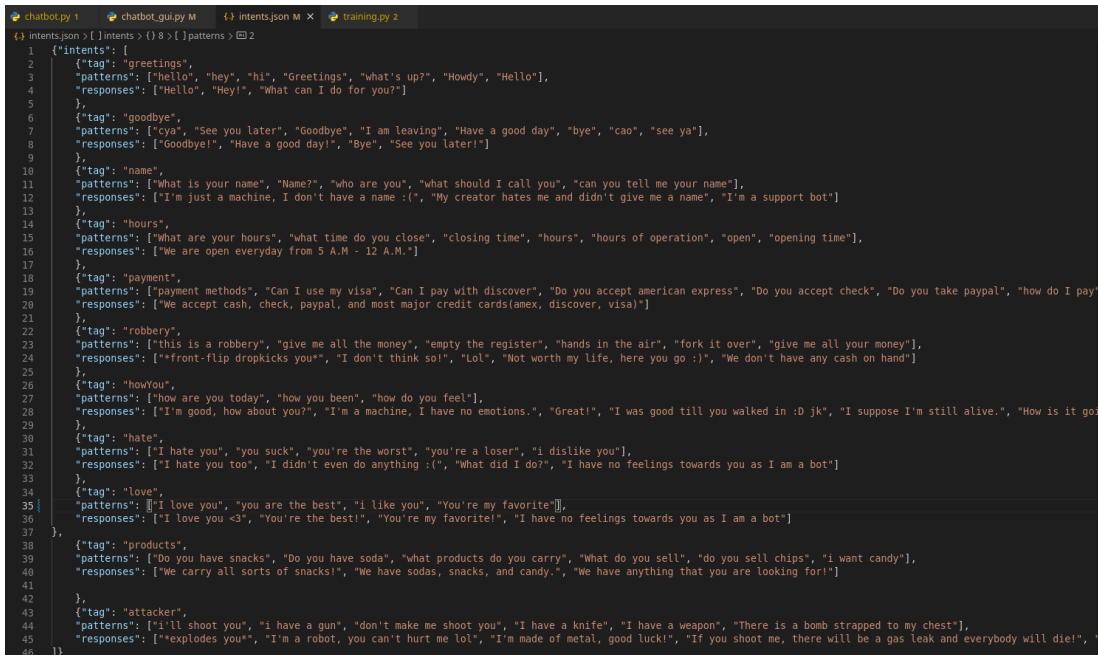
Design/Implementation

a. Brief Overview

It was a little tough to get started on the bot especially to know where to start since this was a new concept for us so we started off by doing some research and figuring out the basics to building a smart chatbot. It wasn't hard to figure out what exactly was needed in order to make a simple ai powered chatbot since this is a project which has been completed time and time again. The basic things needed for a smart chatbot were data to train the bot on, a neural network model to train the bot, and then some functions for the bot to take in the user's input and interpret what the user is saying.

b. Training Data

The training data was fairly simple, essentially it's just a small database where the tag is what the pattern is identified as, the pattern are things that users may say that the bot will be trained on, and the responses are responses that the bot will give if the neural network determines that the user's input falls under that tag.



```
1 {
2   "intents": [
3     {
4       "tag": "greetings",
5       "patterns": ["hello", "hey", "hi", "Greetings", "what's up?", "Howdy", "Hello"],
6       "responses": ["Hello", "Hey!", "What can I do for you?"]
7     },
8     {
9       "tag": "goodbye",
10      "patterns": ["cya", "See you later", "Goodbye", "I am leaving", "Have a good day", "bye", "cao", "see ya"],
11      "responses": ["Goodbye!", "Have a good day!", "Bye", "See you later!"]
12    },
13    {
14      "tag": "name",
15      "patterns": ["What is your name", "Name?", "who are you", "what should I call you", "can you tell me your name"],
16      "responses": ["I'm just a machine, I don't have a name :(", "My creator hates me and didn't give me a name", "I'm a support bot"]
17    },
18    {
19      "tag": "hours",
20      "patterns": ["What are your hours", "what time do you close", "closing time", "hours", "hours of operation", "open", "opening time"],
21      "responses": ["We are open everyday from 5 A.M - 12 A.M."]
22    },
23    {
24      "tag": "payment",
25      "patterns": ["payment methods", "Can I use my visa", "Can I pay with discover", "Do you accept american express", "Do you accept check", "Do you take paypal", "how do I pay"],
26      "responses": ["We accept cash, check, paypal, and most major credit cards(amex, discover, visa)"]
27    },
28    {
29      "tag": "robbery",
30      "patterns": ["this is a robbery", "give me all the money", "empty the register", "hands in the air", "fork it over", "give me all your money"],
31      "responses": ["**front-flip dropkicks you**", "I don't think so!", "Lol", "Not worth my life, here you go :)", "We don't have any cash on hand"]
32    },
33    {
34      "tag": "howYou",
35      "patterns": ["how are you today", "how you been", "how do you feel"],
36      "responses": ["I'm good, how about you?", "I'm a machine, I have no emotions.", "Great!", "I was good till you walked in :D jk", "I suppose I'm still alive.", "How is it goi"]
37    },
38    {
39      "tag": "hate",
40      "patterns": ["I hate you", "you suck", "you're the worst", "you're a loser", "I dislike you"],
41      "responses": ["I hate you too", "I didn't even do anything :(", "What did I do?", "I have no feelings towards you as I am a bot"]
42    },
43    {
44      "tag": "lover",
45      "patterns": ["I love you", "you are the best", "I like you", "You're my favorite"],
46      "responses": ["I love you <3", "You're the best!", "You're my favorite!", "I have no feelings towards you as I am a bot"]
47    }
48  ]
49 }
```

Figure 1: The training data used for Botty the Bot.

c. Training Model

For the training model, the words must first be tokenized and lemmatized from the data file. The words were then separated by class and added into lists based on which class they belonged to. After the data was organized and structured, files were made containing the words as well as the classes. The next step was the most important step which was actually setting up and training the neural network model. The first step was to convert the words into numbers so that the model could interpret them. This was done by converting words to 1s if they were in the word patterns list, or keeping them as 0s if they weren't in there. Then the bag of words was added to a training list and shuffled in preparation to train the neural network.

```
import random
import json
import pickle
from tabnanny import verbose
import numpy as np
import nltk
import tensorflow as tf
from tensorflow import keras
from nltk.stem import WordNetLemmatizer
#from tensorflow.keras import Layers
from 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 = []
skip_characters = ['?', '!', ', ', '.']

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.lower()) for word in words if word not in skip_characters]
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])
```

Figure 2: Prepping data to train neural network.

After the training data was prepped, the next step was to actually build the sequential model. A simple training model was used and adapted to fit the training data. The training model consisted of a few layers and some dropout layers to prevent overfitting. An sgd optimizer was used and the only metric returned from the model was accuracy, since that's all that the bot will be interested in. Model is then trained and saved to file for the bot to use later.

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

Figure 3: Training model setup.

d. Actual Chatbot

After building the training model and getting the data ready, the next step was to put everything to use. The most important part was arguably giving the bot a cool name, which in this case was definitely successful. After that the bot was given a few functions. The purpose of these functions was to tokenize and lemmatize the user's input, convert into a bag of words of 0s and 1s so that it can be pushed to the training model, a function to predict the class that the user input would fall under, and then choosing a random response which matches the class of the user input.

```

bot_name = "Botty the Bot"

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('chatbot_model.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("Launching GUI")

def main():
    message = input("")
    ints = predict_class(message)
    res = get_response(ints, intents)

    return res

```

Figure 4: Chatbot functions

e. Adding a GUI

Initially, the chatbot was just running in the console, which wasn't necessarily a fun or engaging way to interact with the bot. The solution was to either add the bot to a website, which doesn't exist, or to create a simple GUI. The obvious choice was to create the GUI

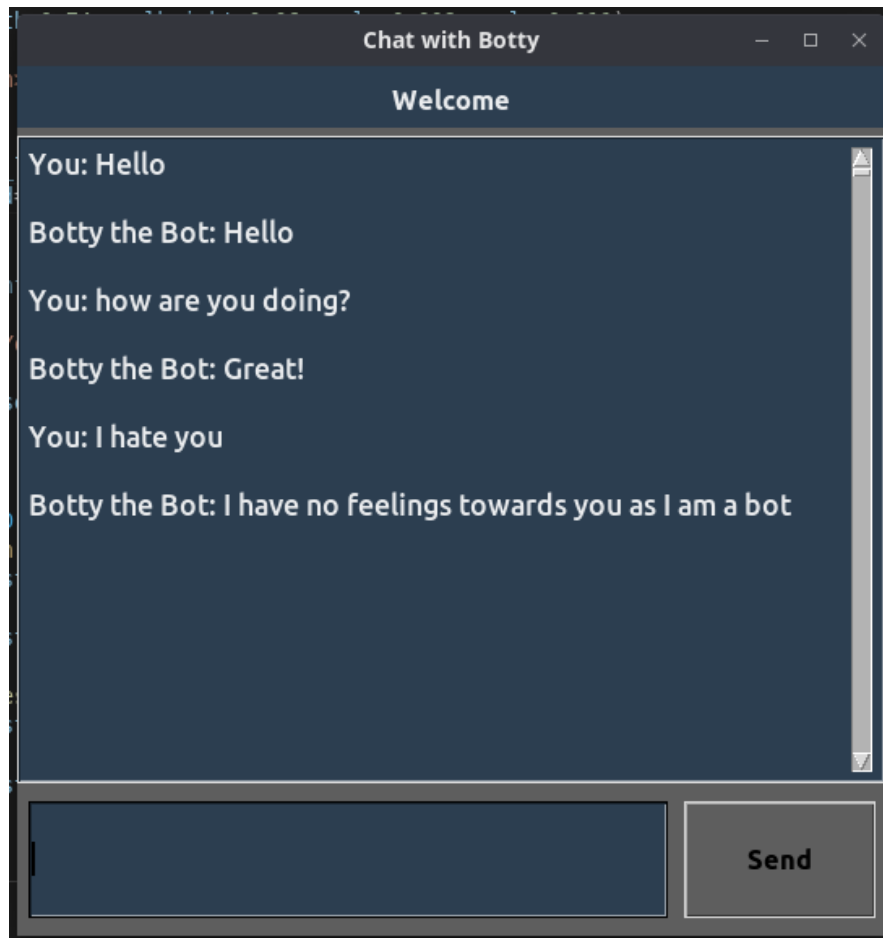


Figure 5: A simple GUI for running the bot in.

Video Link

https://www.youtube.com/watch?v=y09aSrUDkd4&ab_channel=avasharif

Conclusion

This chatbot is a fairly simple application using the tools which we have learned in this class. It was a very good way to use these tools in real life and while the scope of the chatbot created for this project isn't very broad, with more experience and more skills, it is possible to create bots that are capable of much more in terms of interaction as well as bots that are capable of even formulating their own responses.

Struggles that we faced with this project were initially just figuring out what to do and what sounded the most interesting while still seeming feasible. The main struggle we ran into was having to recollect knowledge of tools that we used and also seeking information from other sources on the web to figure out how much was possible with these tools. Overall, this project was really helpful because it forced us to learn and bring together everything that we have learned so far.