Music Recommendation Chatbot

Ethan McCosky

INTRODUCTION

The purpose of this chatbot project is to create a chatbot to demonstrate NLP principles. The chatbot is named Orpheus, after the Ancient Greek mythical hero musician. The vision for the chatbot was to create a natural language interface for learning about a user's musical preferences. The completed version of the chatbot at this time is able to learn about a user's music tastes and recommend additional songs the user might enjoy. This is accomplished through connection to the Spotify API. Originally, I had wanted to expand on this further and construct a model of a detailed custom user's preferences that ask the user why they liked certain songs to determine taste drivers across varying taste clusters. However, due to the underestimated complexity of creating a chatbot from scratch in Python, those advanced features were moved out of the definition of MVP and are reserved for later development, most likely part of a different project focused on analysis and with a more visual interface.

NLP TECHNIQUES

The chatbot is built using several NLP techniques. These can best be described by explaining how the chatbot works. The corpus for the chatbot is an JSON file that describes user intents and has sample things a user may say to signify that intent. A sample of the file is shown below, the full data can be found in intents.json.

```
"intents": [
  {
   "tag": "greeting",
   "patterns": [
    "Hi there",
    "How are you",
    "Hey",
    "Good day",
    "hi",
    "hello",
    "yo"
   ],
   "responses": [
    "Hello, I am Orpheus, a Music reccomendation bot.",
    "What would you like to do today?",
    "Hi there, how can I help?"
   ],
   "context": "general"
  },

  ...

  {
   "tag": "recommend_music",
   "patterns": [
        "What music should I listen to?",
        "what music would you recommend?",
        "what songs do you recommend?",
        "what do you recommend?",
        "What would I like",
        "can you recommend a song to me",
        "can you recommend me music",
        "what else would I like",
        "I dont know what to listen to",
        "I want to find new music"
      ],
      "responses": [],
      "context": "hydrated"
    }
  ]
}
```

From the intents file, documents are built consisting of all of the pattern phrases. These documents are then fed into an sklearn TfidfVectorizer. Next, cosine similarity is computed between the user's query and each document in the corpus to find probabilities that the query falls into each intent. Earlier iterations of the chatbot design attempted to use a Keras Neural network, however, the simple bag of words approach to creating that training data proved to be unsuccessful at correctly predicting intent. The code for this can be found in the `training_keras.ipynb` notebook.

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
from sklearn.metrics.pairwise import cosine_similarity

documents_raw = [d[0] for d in documents]

query = "what can you do"
documents_raw.append(query)

tfidf_docs = tfidf_vectorizer.fit_transform(documents_raw)

arr = list(cosine_similarity(tfidf_docs[-1], tfidf_docs)[0][:-1])
```

For the query `"what can you do"` tested above, the resulting probabilities are

```
0.06942139602513236
0.07190573158174676
0.09652389999247564
0.5536496060195553
0.0
0.5536496060195553
0.2512911794704487
0.28464084359130293
```

```
0.37484630309320416
0.42479359410902484
```

You may notice that the query is equally probable to belong to the 4th and 6th intents, which if you look are built from the same patterns.

```
{
   "tag": "options",
   "patterns": [
    "How you could help me?",
    "What can you do?",
    "What help you provide?",
    "How you can be helpful?",
    "What support is offered",
    "What options do you have?",
    "help",
    "help me",
    "instructions",
    "huh"
   ],
   "responses": [
    "I can guide you through connecting to spotify, after which I'll be able to
learn your top artists or tracks."
   ],
   "context": "main"
  },

...

  {
   "tag": "options_2",
   "patterns": [
    "How you could help me?",
    "What can you do?",
    "What help you provide?",
    "How you can be helpful?",
    "What support is offered",
    "What options do you have?",
    "help",
    "help me",
    "instructions",
    "huh"
   ],
   "responses": [
    "Now that you've authenticated with Spotify, I can tell you about your
listening preferences including top tracks and top artists. I can also reccomend
songs you might like!"
   ],
   "context": "hydrated"
  },
```

The difference between these two intents is the context. The first intent has the context `main` and is only valid before the user has connected to spotify, whereas the second intent has the context `hydrated` and is only valid after the user preferences have been hydrated from Spotify. (see User model below) Intents with the `general` context are valid at all times.

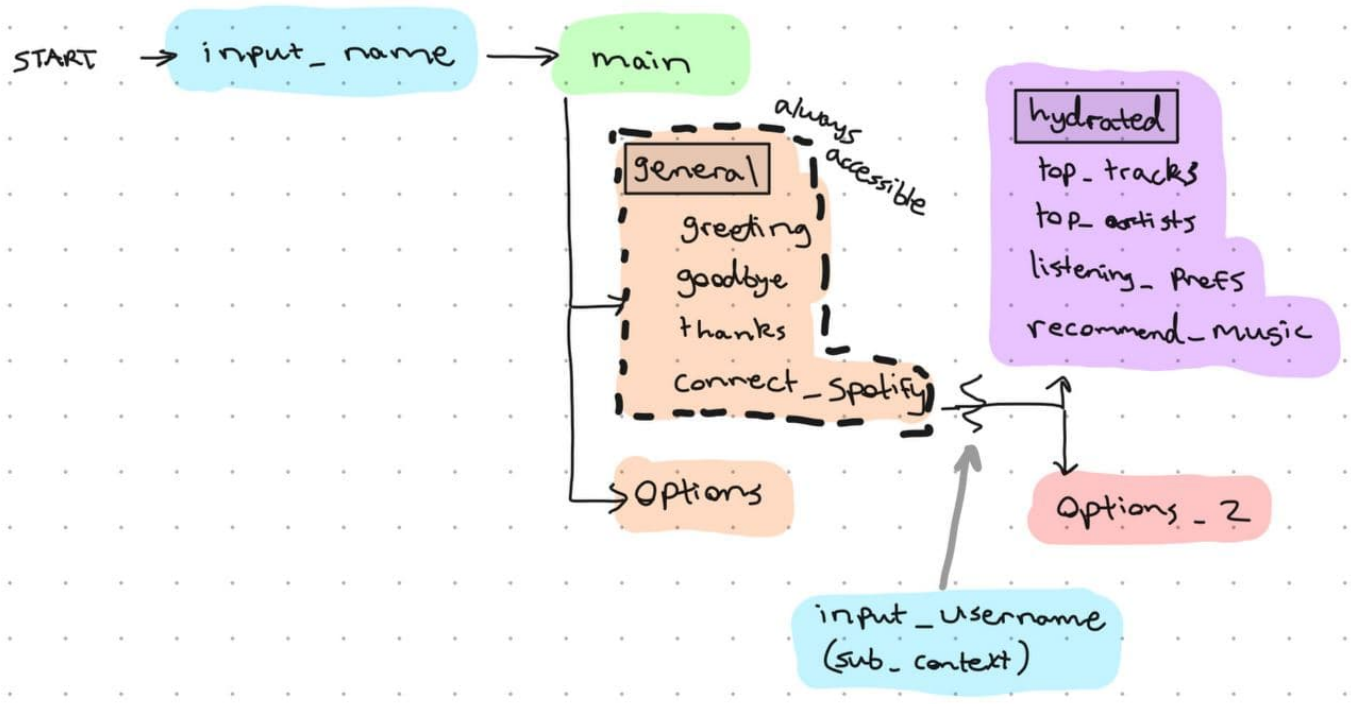
```
class User:

    name: str
    spotify_username: str
    genre_prefs: typing.List[bool]
    artist_likes: typing.List[str]
    song_likes: typing.List[str]

    # Ideally these would be part of a track/artist object but for now are
separate
    # because they are only used for recommendation seeding
    artist_ids: typing.List[str]
    song_ids: typing.List[str]
```

PROGRAM FLOW

Contexts are used to control the flow of the program and limit the available intents to classify the user query into based on the chatbot state. The diagram below represents a simplified version of the program flow model.



SAMPLE INTERACTION

The following is sample interaction with the chatbot and shows working usage of the chatbot.

```
[19:29:14] D:\..\..\..\chatbot venv37
❯ & "d:/DEV/Class/CS4395 NLP/chatbot/venv37/Scripts/python.exe"
"d:/DEV/Class/CS4395 NLP/chatbot/bot_new.py"

[ORPHEUS] Hello, I am Orpheus, a Music reccomendation bot. What is your name?
:: Ethan
[ORPHEUS] Welcome back, ethan!
:: Hello bot!
[ORPHEUS] What would you like to do today?
:: What can you do
[ORPHEUS] I can guide you through connecting to spotify, after which I'll be able
to learn your top artists or tracks.
:: ok, connect to spotify
[ORPHEUS] Connecting to Spotify and populating data from account emccoskymusic
[ORPHEUS] connected
:: what can you do now
[ORPHEUS] Now that you've authenticated with Spotify, I can tell you about your
listening preferences including top tracks and top artists. I can also reccomend
songs you might like!
:: what songs do i like?
[ORPHEUS] Your top tracks are:

1. Title: Icarus (feat. Ari Lennox) Artist: Bas
2. Title: Cancel Me Artist: Dominic Fike
3. Title: Front Desk Artist: Bas
4. Title: Futura Free Artist: Frank Ocean
5. Title: Thank You Artist: Florist
6. Title: trix Artist: whiterosemoxie
7. Title: Self Control Artist: Frank Ocean
8. Title: Clouds Never Get Old Artist: Bas
9. Title: The Dreamer (feat. Talib Kweli & Timan Family Choir) Artist: Anderson
.Paak
10. Title: Purge Artist: Bas

:: thanks!
[ORPHEUS] My pleasure
:: what are my top artists?
[ORPHEUS] Your top artists are:

1. Tom Rosenthal
2. Drake
3. Florist
4. Mac Miller
5. Flatsound
6. Kendrick Lamar
7. Kanye West
8. Iron & Wine
9. Tyler, The Creator
10. Bon Iver

:: awesome!
[ORPHEUS] My pleasure
:: can you recommend me some more music?
```

```
[ORPHEUS]
Here's some songs you might like:
Title: Throw the Fear Artist: Tom Rosenthal
Title: Ocean Girl Artist: L'aupaire
Title: Double Up (feat. Belly and Dom Kennedy) Artist: Nipsey Hussle
Title: About The Weather Artist: Tom Rosenthal
Title: For You To Be Here Artist: Tom Rosenthal
:: thanks!
[ORPHEUS] Happy to help!
:: goodbye!
[ORPHEUS] Bye! Come back again soon.

[19:31:07] D:\..\..\..\chatbot venv37
❯
```

USAGE

To use the bot, simply run `bot_new.py`. The needed model information has already been pickled and is included. If you wanted to reanalyze the data, you could do so by running all the cells in `training_sklearn.ipynb`

A few notes: The Spotify API and python library `Spotipy` is very weird about authentication and sometimes incorrectly caches authorization and refresh tokens. Upon first run, a browser window should open, asking you to login to Spotify and grant the application access to your account data. (Don't worry, it is not being collected in any manner) Sometimes this will cause a program error, but a rerun of the program should connect successfully. (with the same name so the bot recognizes you)

EVALUATION AND CONCLUSIONS

Throughout the course of this project many difficulties were encountered, leading to the conclusion that building a chatbot is quite difficult and complex. The chatbot is quite limited in functionality in its current version, however would not be particularly difficult to scale to support more intents. Future exansion of this could include more usage of the Spotify API or even connection to additional datasources such as Wikipedia for artist biographical information. Critical reviews of music could also be fed in to give the bot a knowledge base upon which to discuess music with the user. Some of my biggest takeaways from the project is that python is a difficult language to build complex things in and I frequently found myself painstakingly debugging type errors.

Overall, the chatbot accomplishes the goal of providing a natural language based interface to learn user music preferences and acts as a simple interface for the Spotify API.