

Chatbot Report



Liam McMains - ljm170130 - CS 4395.001

Installation

- Download the zip file containing all of the code for the project.
- Unzip the zip file so that all the files are no longer compressed.
- Make sure that the `install_dependencies.sh` file has execute permissions. An easy way to make sure is to run `chmod 777 install_dependencies.sh`.
- Run the `install_dependencies.sh` to install all the dependencies required for the project. You can run it with the following command: `./install_dependencies.sh`. Make sure you are in the project directory when you run this.
- Make sure that you have the `en_core_web_sm` pack downloaded for `spacy`. You can download it using the command `python -m spacy download en_core_web_sm`. If this doesn't work, try replacing `python` with `python3`.
- Finally, you should be able to run the chatbot using the command `python3 index.py`.
- If you run into any issues trying to get the chatbot to run, feel free to send me an email at ljm170130@utdallas.edu and I will try and help you out.

Overview

This chatbot is written to respond to questions having to do with the popular "car soccer" game, Rocket League. The bot will respond to the following questions and statements:

- Any type of greeting
- Requests for functionality: Eg. "What does this chatbot do?"
- Declarations of names: Eg. "My name is Liam McMains"
 - It will remember this information when you ask it "What do you know about me?"
- Requests for learned information: Eg. "What do you know about me?" or "What have you learned?"
- An explanation of Rocket League: Eg. "What is Rocket League?" or "Tell me about Rocket League."
- An explanation of multiplayer support for Rocket League: Eg. "How many people can play?" or "Does it support local multiplayer?"
- A description of professional Rocket League players: Eg. "Who plays Rocket League?" or "Who is the best at Rocket League?"
- An explanation of the game rules: Eg. "How does Rocket League work?" or "How do you play?"
- The platforms that Rocket League supports: Eg. "Which consoles have Rocket League?" or "What does it run on?"
- Who created the game: Eg. "What team developed Rocket League?" or "Who made Rocket League?"
- The different game modes in Rocket League: Eg. "What are the different types of games in Rocket League?"
- Descriptions of the different game modes in Rocket League including Rumble, Snow Day, Hoops, and Dropshot: Eg. "What is Dropshot?"
- A description of the eSports scene in Rocket League: Eg. "Are there tournaments for Rocket League?"
- Finally, a description of where you can buy the game: Eg. "Where can I get Rocket League?" or "Where can I download it?"

System Description

The system consists of the following parts:

- Question Identification
- User Model
- Named Entity Recognition
- Term Frequency Distributions
- Sentiment Analysis

Question Identification

When having a conversation with a chatbot, the main goal that the chatbot needs to be able to accomplish is the ability to identify what type of question you are asking. For example, if you ask the question "what do you do," the chatbot needs to identify that you are making a request for the description of the chatbots functionality, not that you are asking about a specific game-mode in Rocket League. To achieve this functionality, I used a machine learning model produced by tensorflow that gives a probability that a given question or statement provided by the user falls into a specific class. For example, if you were to just say "Hello" to the chatbot, it would create the following prediction:

```
[9.9988902e-01 6.3158797e-07 4.5216271e-05 4.2323947e-07 1.7247861e-07
2.2036399e-07 9.7137569e-09 1.3640992e-07 8.4063039e-08 5.4520176e-05
7.3654093e-07 2.0169638e-07 1.9059505e-06 8.0423085e-08 2.3444922e-07
2.8986599e-06 2.0024522e-06 1.4527112e-06]
```

As you can see, the first element, which corresponds to the `greeting` class, is the largest value in the vector. This means that the user most likely initiated a conversation that would merit a response in the greeting class. Therefore, the bot could reply with "Hello!" or "Hi, what would you like to know about Rocket League?"

The piece of code responsible for predicting the class of the users input looks like this:

```
def predict(sentence, model):
    bag = bag_of_words(sentence)
    prediction = model.predict(np.array([bag]))[0]
    return classes[numpy.argmax(prediction)]
```

The bag of words is a vectorized representation of the users input, the prediction is the vector shown above that contains the probability of each class, and the return value is the actual string that represents the predicted class, such as "greeting."

User Model

The user model for this particular chatbot is quite simple. I am representing the user model as a plain dictionary. The dictionary includes a few key/values for the different aspects of a user object including the name, the users likes, and the users dislikes. Also, the entire history of the user's conversations is stored as well. These are used later when you ask the chatbot what information it has learned from you.

Here is the representation of the user that gets stored:

```
{
  "name": "",
  "likes": [],
  "dislikes": [],
  "all_chat_messages": []
}
```

As the user interacts with the chatbot, these particular fields will start to be filled in. If the chatbot notices you've said something very positive or negative, it will add those to your likes and dislikes. If the chatbot notices that you've said a name, it will also add the name to your name property. Finally, the chatbot keeps track of all the messages you've sent to perform Term Frequency over it later.

Named Entity Recognition

There are multiple places where Named Entity Recognition is used in the project. The first use is when you make a statement such as "My name is Liam McMains." The chatbot will recognize that you are making a `name-statement` class and will attempt to run Named Entity Recognition on your message. If it is able to locate a Named Entity with the tag of `PERSON` then it knows that you have supplied a name. This name is then taken and stored on your user object which is subsequently saved for later use. Then, when you say "What information do you have on me," the bot is able to recall your name with the previously stated name.

Secondly, when you make a statement with high sentiment polarity, Named Entity Recognition again reads through your message to try and identify what you are talking about. For example, if you said the statement, "I really hate George Lucas," the chatbot would recognize that you are talking about a person, George Lucas, and add him to your list of dislikes.

Term Frequency Distributions

The chatbot also will supply you with term frequency distributions on your chat to give you the most frequently said phrases or ngrams. This is achieved using the `FreqDist` class in the `nltk` library. I create these distributions for ngrams of size 2 and 3 because they are the easiest to generate. Next, if you request your information once again using the request "What do you know about me?" the chatbot will tell you some of your most frequently stated phrases. Here are a few of the responses that I received using TFD:

```
AI: Some of your most 2 word common phrases are: I love, have you, What have, you learned, rocket league
AI: Some of your most 3 word common phrases are: What have you, have you learned, My name is, Rocket league is, What do you
```

Unsurprisingly, `Rocket League` and `Rocket League is` show up as common phrases using TFD.

Sentiment Analysis

The last NLP technique that I used in this chatbot is sentiment analysis. Sentiment analysis is triggered for each message to check the polarity of your message. If your message is strongly positive or strongly negative, then the chatbot will check to see what it is you are being strongly negative or positive about. If it notices that you say something like "I love artichokes," it will recognize that statement as have a sentiment polarity above the positive threshold and add artichokes to your list of liked items, which you can then recall like usual using the "What have you learned?" request. The sentiment analysis is currently using `TextBlob`. This library isn't perfectly and can be easily fooled by saying something like "I don't really hate artichokes" which it assumes is negative, but is actually neutral.

Sample Conversations

```
AI: Hello! I am a ChatBot created for Human Language Technologies. Ask me anything about the video game Rocket League!
AI: You can ask what Rocket League is, how the game is played, who plays it professionally, which platforms support Rocket League, e

You: Hello
AI: Hi, what would you like to know about Rocket League?

You: What do you do
AI: I am a ChatBot that likes to talk about the video game Rocket League. Ask me anything about Rocket League!

You: What is rocket league
```

AI: Rocket League is a vehicular soccer video game developed and published by Psyonix. The game was first released for Microsoft Windows.

You: How do you play

AI: Described as 'soccer, but with rocket-powered cars', Rocket League has up to eight players assigned to each of the two teams, using rocket-powered cars.

You: How many friends can I play with

AI: Rocket League supports both local and online multiplayer for up to 8 people in a single game. It's a great game to play with your friends.

AI: You liked something a lot in there!

AI: I think I've detected that you like "friends"

You: Can I buy it on Steam

AI: On PC, you can download Rocket League on Steam. Otherwise, be sure to check out your console's store and I'm sure you can pick it up.

You: What game modes are there

AI: Rocket League has many different modes you can try out. The classics include 1v1, 2v2, 3v3, and 4v4 chaos mode. Some of the more recent additions include Dropshot and Hooligan.

You: What is snow day

AI: Snow Day is a hockey variant of Rocket League. Instead of playing with a ball, you play with a puck that slides around on the ice.

AI: Hello! I am a ChatBot created for Human Language Technologies. Ask me anything about the video game Rocket League!

AI: You can ask what Rocket League is, how the game is played, who plays it professionally, which platforms support Rocket League, etc.

You: My name is George Lucas

AI: It's great to meet you, George Lucas.

You: I really hate Mark Hamill

AI: You disliked something a lot in there!

AI: I think I've detected that you dislike "Mark Hamill"

You: I really love Darth Vader

AI: You liked something a lot in there!

AI: I think I've detected that you like "Darth Vader"

You: What have you learned

AI: I know your name is George Lucas.

AI: I've learned that you really like Darth Vader

AI: I've learned that you really dislike Mark Hamill

AI: Some of your most 2 word common phrases are: do you, What is, What have, have you, you learned

AI: Some of your most 3 word common phrases are: What have you, have you learned, My name is, Hello What do, What do you

Sample Knowledge Base

The knowledge base contains a list of classes, the questions that correspond to that class, as well as the responses that the class can produce. Here is an example of the `functionality`` class. This class is used to respond to the question "What do you do," just like you would use the `--help` flag if you were running a command line program:

```
{
  "class": "functionality",
  "patterns": [
    "What do you do?",
    "How do you work?",
    "What does this program do?",
    "What can you tell me?",
    "What's up?",
    "What functionality do you provide?",
    "What can do you?"
  ],
  "responses": [
    "I am a ChatBot that likes to talk about the video game Rocket League. Ask me anything about Rocket League!",
    "You can ask what Rocket League is, how the game is played, who plays it professionally, which platforms support Rocket League"
  ]
}
```

The `class` is simply an ID for this specific type of question. The `patterns` property is a list of possible permutations of the question that could cause these responses to be used. These aren't strict word for word patterns but rather use the tensorflow model to get a probability that the user input matches these patterns. The `responses` list contains all possible responses to the given `class`.

Some `class`es do not contain static responses but rather run custom code. These can be seen in the `index.py` file here:

```
non_baked_responses = [
  {
    "class": "information",
    "code": respond_to_information_query
  },
  {
    "class": "name-statement",
    "code": respond_to_name_statement
  }
]
```

Sample User Models

After the last example conversation, the user model would look something like this:

```
{
  'name': 'Darth Vader',
  'likes': ['Darth Vader'],
  'dislikes': ['Mark Hamill'],
  'all_chat_messages': [
    'Hello',
    'What do you do',
    'What is rocket league',
    'How do you play',
    'How many friends can I play with',
    'Can I buy it on Steam',
    'What game modes are there',
    'What is snow day',
    'What have you learned',
    'My name is George Lucas',
    'I really hate Mark Hamill',
    'I really love Darth Vader',
    'What have you learned'
  ]
}
```

Evaluation

Strengths

- The chatbot is quite good at responding to different permutations of different questions. For example, the chatbot will respond correctly whether or not you ask it "What do you do," "What functionality do you have," or "How do you work?" It isn't too hard coded to not be flexible.
- If you stay in the realm of Rocket League, the conversations seem quite fluid and natural.
- The chatbot is quite quick in its responses because all of the answers to its questions are either stored locally or require minimal processing to respond. It doesn't require a web request every time the users asks a question.
- You can add functionality to the chatbot quite easily by adding new classes, responses, and patterns and then retraining the model to account for these new types of questions.

Weaknesses

- The chatbot can become confused if you ask it a question outside of its knowledge realm. For example, if you ask it "Who is Babe Ruth," it will talk about famous Rocket League players rather than baseball players. Clearly it is not a generalized intelligence.
- The sentiment analysis can be easily fooled by more complex sentences. "I don't really love broccoli" will make the bot think that you love broccoli, rather than just being a neutral sentence. A better sentiment analysis library should be used here.
- Sometimes if you tell it what you like or dislike, the bot gets confused and thinks that you are giving it your name. It will then store your like or dislike as your name, instead of just storing your like or dislike.