Intro to Computational Linguistics Final Project

MiRa Command Line Chatbot

William Kelly

3/11/2024

Abstract

This document is a reference on how to create a chatbot that you can interface through the command line. The goal of this project shifted a few times while I was creating it, which led to a very interesting design. Overall, I ended up creating a powerful command line chatbot that can have basic conversations, tell date/time, and leverage the power of Chat GPT 3.5 The goal of this project shifted a few times while I was creating it, which led to a very interesting design. Overall, I ended up creating a powerful command line chatbot that can tell date, time, tell jokes, and leverage the power of GPT 3.5 for complex queries.

1 Introduction

A chatbot is a computer program that is designed to simulate conversation with human users. In this new age of digital innovation and interaction, chatbots are becoming increasingly prominent for their ability to offer automated services, entertainment, and information-gathering purposes. As I have been learning more about computational linguistics, I have learned much more about NLP, LLM, and dialogue systems and their potential capabilities. As chatbots are now becoming more advanced with more and more features, a good way to wrap up multiple aspects of the course is by creating the MiRa chatbot (multifunctional Intelligent Response Assistant).

This project highlights the importance of blending rule-based systems with advanced AI models to create chatbots that are responsive and capable of analyzing complex requests. MiRa's development highlights the potential for chatbots to become integral parts of our lives, offering convenience, humor, and insight.

Data

MiRa is a Python command line chatbot designed to engage its users in conversation, responding to inputs with a variety of outputs. The basic conversation outputs are determined by a custom set of patterns and responses using the NLTK reflections library.

```
    # Your patterns and responses
patterns = [
    (r'hi|hello|hey|hi MiRa|MiRa', ['Hello!', 'Hey there!', 'Hi!']),
    (r'how are you|whats up', ['I am good, thank you.', 'I\'m doing well.', 'All good!', 'Nothing much']),
    (r'bye|goodbye', ['Goodbye!', 'Bye!', 'See you later.']),
    #(r'what|who|when', ['how', 'why', 'where']),
    (r'Tell me a joke|Make me laugh|haha', ['Did you hear the one about the mountain goats in the andes? It w
    'What do you get when you cross music and an automobile? Cartune.']),
    (r'no|so|ok|oh ok', ['If you said yes more, your life would be morei interesting', 'Fine then', 'Sassy']),
    (r'knock knock', ['Whos there']),
    (r'Thanks | I appreciate it | sure', ['No Problem']),
    (r'maybe', ['Son, theres no such thing as a maybe']),
    (r'what.*your name', ['My name is MiRa. (Multifunctional Intelligent Response Assistant)']),
    (r'what.*your age', ['I am truly ageless.']), #regex
    (r'what.*your purpose|what are you doing', ['My purpose right now is to entertain, if you would like me t
    (r'who.*you', ['I am a chatbot created by Trey to help you.']),
    (r'where.*you', ['I exist on the 1s and 0s.']),
```

Figure 1: Patterns Format

As a part of this custom patterns set, I included a date and time feature utilizing the Python date/time module. I also included jokes from Reddit libraries online.

Finally, I utilized the OpenAi to scan ChatGPT for complex inquiries. This feature is activated by the keywords 'GPT' or 'MiRa analyze".

```
# Set your OpenAI API key here
openai.api_key = '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'

def explain_code(code_snippet):
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": f"{code_snippet}"}
        ]
    )
    return response.choices[0].message['content'].strip()

```

Figure 2: OpenAi Api Request Format

2 Methods

2.1 Basic Chatbot Functionality

The original aim for this project was to create a funny command line chatbot, so I first decided to use the chatterbot library to accomplish this. There were originally many issues with downloading the correct versions of Python and chatterbot. After finally implementing the chatterbot library (with basic responses and jokes) I received outputs that were random and inaccurate, so I created my own.

I then decided to use the NLTK reflections library after reviewing the chatterbot library. This library essentially allowed me to create sample patterns and responses that match sample input to predetermined responses using regex commands.

2.2 Jokes

I created the jokes for my chatbot mostly out of a Reddit thread/simple Google search. I know I could've used my actual brain to create some witty jokes, but I wanted to keep it mostly PG. The jokes are triggered by certain words in the user input and cycles each time a joke is said.

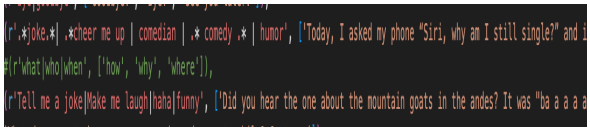

Figure 3: Joke request format

2.3 Date/Time

Here I started to think about other possibilities for the project. I figured the date and time would be a nice addition, even though it's already listed on your computer.

2.4 Enhanced Computing Requests

To access the internet and receive requests, I utilized the OpenAI API to get information from ChatGPT. This was a nice feature to add because I can access this much easier than going online. I can also edit how long the responses can be, using the tokens parameter in the code.

3. Results

The chatbot "MiRa" was fairly successful, I was able to complete my goal of creating a basic command line chatbot that is usable by everyone (once they download the packages). I succeeded in creating a library that can have basic conversations, tell jokes, and tell date/time, as well as utilize a language generation model like ChatGPT for complex inquiries.

4. Discussion

4.1 Places for Improvement

When discussing the chatbot there are a few things that I wish I would've done in retrospect. Instead of creating a library of custom responses from scratch (that I couldn't possibly build to be complex enough for all human convo) I wish I would've found a better patterns and responses library and added on from there.

Additionally, I wish I had added better jokes and a better sarcastic nature to the answers in general.

4.2 Future Ideas

I enjoyed my time working on this chatbot and I plan to continue working on this project. With this being said I have plenty of new ideas for what to add to the chatbot.

- Upgrade the pattern response to include more facets of human conversation.
- Schedule Reminders in Apple Calendar
- Create Calendar Dates
- Personalize computer (light/dark theme)
- Tell weather
- Take excerpts of code and convert them into different languages.
- Possible Multilingual Support
- ULTIMATLEY, text to speech, possibly utilizing Siri?

5. How to Operate Code:

The code is currently hosted in the QUEST server. I will also include a copy of it in my submission to you guys.

5.1 Operating on your PC:

The chatbot script operates on Python version 3.8+ (That's as far as I've checked back). In order to run the code you will need to run these commands:

Pip install nltk

Pip install requests

Pip install openai ***or*** *pip install openai == 0.28 (whichever works)*

5.2 Operating on Quest:

To operate on QUEST is slightly different and has a few more bugs. QUEST runs python 3.6, and the Openai package requires at least 3.7. The python on quest would need to be upgraded, but I wasn't exactly sure how to do that so I left it.

6. Acknowledgments

Thank you, Rob, Grace, and Adrian for being wonderful mentors in computational linguistics. I appreciate all the insight and I hope to see you around.

7. References

ChatterBot Documentation. (3/12/2024). Retrieved from https://chatterbot.readthedocs.io
Natural Language Toolkit (NLTK) Documentation. (3/12/2024.). Retrieved from https://nltk.org

NLTK API Documentation. (3/12/2024). Retrieved from https://nltk.org/api/nltk.html
OpenAI Documentation. (3/12/204). Retrieved from https://platform.openai.com/docs/overview