Oomck Chatbot Project Plan

19th February, 2021

For COSC 310

By Owen Murovec, Opey Adeyemi, Mathew de Vin,
Carson Ricca, and Keyvan Khademi

Project Description

In this project, we are creating a chatbot that will allow a user to interact with the agent and maintain a conversation about the Fast and the Furious franchise. The agent will be able to maintain a conversation with the user for approximately 30 turns. The agent will play the role of a friend who has a vast knowledge of the Fast and the Furious. The user will have the role of another friend who also enjoys the series. The repository for our project can be found at https://github.com/oomck/oomck-bot.

SDLC & Rationale

An incremental development lifecycle is proposed for this project. More specifically, agile practices will be employed. An agile approach is preferred by the small and well-knit team of developers, as well as the scope of the work involving rapidly shifting requirements as the team becomes more acquainted with the tasks at hand. The chatbot is a small application with a specific purpose that is suited for agile development. Despite an agile approach being the primary choice, components of integration and configuration will be used. This is due to the prevalence of pre-existing, reliable, and robust systems that accomplish parts of what Oomck Chatbot aims to achieve.

SDLC Phases

SCRUM is an iterative development method that focuses on managing iterative development practices. Scrum has three main phases that we will implement in our project.

1. The initial phase is an outline planning phase in which the general objectives are established for the project, and the software architecture is designed.
    a. Creating a Project Plan:
        i. Brief Project Description.
        ii. Chosen SDLC and Rationale.
        iii. A List of the Phases of the SDLC.
        iv. A WBS.
        v. Create a Gantt Chart.
    b. Search for Dataset:
        i. Find Possible Datasets.
        ii. Explore Possibility of Dataset for NLP, Pattern Matching, API Implementation.
        iii. Determine Dataset(s) to Use.
    c. DevOps Setup:
        i. Set-Up Automated Unit Testing.
        ii. Containerize ElasticSearch Using Docker.
        iii. Transition Jira TODOs on Successful Merge.
        iv. Create Setup Script.
2. The second phase is a series of sprint cycles, each cycle implements an increment of the system.
    a. Create Command Line Interface:
        i. Read Input from Command-Line.
    b. User Input Parsing:
        i. Retrieve User Input.

        ii. Create a Dictionary of Useless Words.
        iii. Eliminate Useless Words and Return Key Words.
        iv. Categorize Key Words.
        v. Fix Extraneous Cases.
    c. Initialize ElasticSearch with Data (Script):
        i. Clone the Data Repo.
        ii. Filter Data and Delete Ones that are Not Needed.
        iii. Upload Data to ElasticSearch Container.
3. The final phase wraps up the project, required documentation is completed, and lessons learned from the project are assessed.
    a. Finish Project Plan:
        i. Create a List of Limitations of our ChatBot.
        ii. Include Sample Output.
    b. Project Presentation:
        i. Create a Video Showcasing the Project's Strengths and Weaknesses (3-4 Minutes).
        ii. Give a Brief Overview of Who Did What (1 Minute).
        iii. Show PM Graphs on Github (1 Minute).
        iv. Showcase the Jira Project.

Project Objectives

To develop a conversational agent capable of retaining dialogue with a user for 30 turns back and forth (a turn is defined as a prompt-response pair). This agent is to have a topic upon which it centers its responses as well as a distinct tone through which it responds.

Requirements

1. Capability to keep a 30 turn dialogue.
    a. A turn is defined as a prompt-response pair.
2. Focus conversation around the topic of the Fast and the Furious.
3. Distinctly have a conversational tone.

Project Scope

Technology and techniques used in this:

- Python.
- Docker.
- GitHub.
- ElasticSearch.
- Jira.
- Discord.

Activity Dependencies

Work Breakdown Structure

- Current Issues:
    - Create a Project Plan - Carson/Opey.
    - DevOps Setup - Owen.
    - User Input Parsing - Mathew.
    - Search for Dataset - Carson.
    - Class-based code structure - Keyvan.
    - Overall project file structure - Keyvan.
    - Create Command Line Interface - Mathew/Owen.
    - Update README - Carson/Opey.
    - Initialize ElasticSearch with Data - Owen/Carson.

Overall Schedule

The Agile process our project team is following inherently conflicts with the notion of a pre-defined schedule. We did, however, procedurally generate GANTT charts as issues were created and completed. Our underlying project structure was that of Scrum. In order to achieve this, we met every Wednesday at 11 am in order to refocus on our goals as a team and keep things on track.

Limitations

As a result of the method we chose, the competence of Oomck-Bot's responses are highly dependent on the indexed data it uses. Even with strong data, depending on the context (i.e. if API-driven data was used), Oomck-Bot would quite often give nonsensical responses to user prompts. This leads to conversations that don't seem connected, despite being algorithmically sound. The current dataset remedies this problem, with the tradeoff that it is quite small, leading to a generally limited range of dialogue .

The current implementation of Oomck-Bot lacks a GUI. We currently use a basic command-line interface to interact with the user, which is terribly unintuitive. Going forward, Oomack-Bot will feature a web-based GUI to allow for easy chat history viewing and streamlined back-and-forth communication with users.

Our bot is also only able to interpret a single line of input from users and output a single line of dialogue in response. Our command-line interface limits users such that they must wait for Oomck-Bot to respond before inputting a new prompt. However, in real conversations, users may want to add additional information to their prior message and may expect a similarly formatted response.

Sample Output