DATA 515 Project Report

Introduction

A bot uses artificial intelligence software that is designed to perform a series of tasks on its own without the help of a human being. The bot that has been designed for this project is utilized for text analysis that reads a user input and returns a formulated response. The chatbot is also capable of simulating a conversation with a person and can be utilized in many different applications.

The main purpose behind the chatbot is to make many routine things and daily tasks become automated which saves time and convenience. The chatbot created here "Good-Bot" utilizes Natural Language Processing (NLP) which is a field within data science that describes the interactions between computers and human languages. Although the structure is simple, the ability to intercommunicate on a broad range of topics or cover a specific discipline has not been compromised.

Functional Specification

The intended users for Good-Bot are a mixture of students, researchers, product managers, and business leaders who are in the need of an automated process in analyzing text documents quickly and with precision.

The use case for Good-Bot in the hands of business leaders are functionalities that are focused on automating website support and reducing customer service costs. Without the automated interaction and using humans, that would result in a higher labor expenditure. Reducing the human labor costs is where Good-Bot comes into play in answering customers questions while providing adequate 24/7 support.

The use case for Good-Bot in the hands of product managers are functionalities that are focused on automating the collection of customer feedback. Customer service/user feedback is an essential component in the product/software development cycle and using a bot to collect user feedback on a product would make the development cycle move more efficiently.

The use case for Good-Bot in the hands of researchers and students would be the assistance of finding any search terms or specific information pertaining to a text document and scholarly articles to aid in research endeavors.

Component Specification

The functionality for Good-Bot consists of a single module, Good-Bot.py, which contains different public functions that reads the users response, generates an index sorting algorithm, and generates a response to the users' question.

Software Component: Good-Bot

The Good-Bot module is dependent on the random, nltk, sklearn and newspaper libraries. It prompts the user to input a URL to a text document where the module analyzes the text and returns a response through count vectorization and cosine similarity scores.

Functions:

- Greeting_response(text): A function to return a random greeting to a user's greeting. Automatically converts every user input into lower case to prevent any errors with valid user responses. List of potential greetings is used as a training data, a randomized greeting from the bot greetings list is returned.
- Index_sort(list_var): A function that creates an index and sort the sentences within the text document.
- Bot_response(user_input): Response function that uses cosine similarities from predefined texts to generate a response. Cosine similarity metric measures the cosine of the angle between two n-dimensional vectors projected in a multi-dimensional space. It is one of the metrics to measure the text-similarity between two documents irrespective of their size in Natural language Processing. Matches the text and assigns it a score on its similarity to other texts.

Design Decisions

The design goals of Good-Bot are simplicity and flexibility that can satisfy many of the use-cases for having a chatbot that provides 24/7 support in generating timely responses to user's questions and satisfy a key portion of the software development lifecycle requirement of customer feedback.

- Procedural vs functional
  There are preferences in style, however the coding paradigm that is most appropriate for an AI chatbot would be a functional style. Procedural programming is chaotic whereas functional is organized. The downside with procedural is that everything is done in a specific order and the execution of that routine may have side effects which include implementing solutions in a linear fashion. The reason why the functional style is used is that it is recursive, which include the order of evaluation is usually undefined and most importantly it emphasizes a divide and conquer approach versus doing everything at once. Each function of the Good-Bot module has a distinct role in reading the users input followed by indexing the sentences and then using count vectorization with cosine similarities to return an answer to the user is a methodical way of execution.
- Separation of concerns
  All the code for each task of the module is separated out, if any changes were to be made to one task, making direct changes would require less code and be easier to identify what part of the code would need to be changed. Any changes that need to be

made is less likely to break the code in unrelated features, if all features were mixed as seen in procedural programming, changes to a task unintentional is more likely to happen.

Comparison

ChatBot-for-Sentiment-Analysis/ChatBot using AI.py at master · tripathi-abhishek/ChatBot-for-Sentiment-Analysis (github.com)

Chatbot for sentiment analysis is a package that uses NLP, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.

The structure of the code is simpler in the respect that it is easier to read and follow in comparison to the Good-Bot package. However, the execution of the sentiment analysis chatbot relies heavily on if, else and elif statements combined with procedural programming.

The usage between the two packages are different where the sentiment analysis package is pigeonholed into a more casual conversational use case whereas good-bot is more flexible in being able to be used by researchers, business leaders and product managers for specific tasks in customer service and feedback.

The absence of separation of concerns in the sentiment analysis may cause problems in future works on the code where there is no structure and based off sequential steps. Separation of concerns is important where in Good-bot, there needs to be a separation of different responsibilities into distinct roles to enable any code trouble shooting and bug fixes without affecting the entire module.

Extensibility

Good-Bot allows for future work and collaboration to be done to easily add features (extending the system) to different use-cases. The newspaper library easily makes it possible to extend to different languages (non-English) which results in an expanded userbase that include non-English speaking countries. For future work, there are opportunities to integrate a JSON API training data to expand the functionality in terms of deep learning applications. Further improvements could also be made in implementing abstract classes to preserve privacy and non-accessibility to prevent data leakage.