**Module 8: Portfolio Project**

**AI Use-Case With Solution**

Joshua Humphreys

Colorado State University Global

Course Code:CSC510

Dr. Pubali Banerjee

October 6, 2024

**Portfolio Project**

For my portfolio project, I built a simple AI bot to respond to the following use case: A user inputs a query in natural human language. The bot interprets that query and performs a search on a document library if necessary. The bot then interprets the text it finds in the document library and produces a summary in natural language. Importantly, the bot is required to exist on a local machine or network. In the following pages I will discuss the methods and libraries I used to create my bot.

Before digging into this project, I would like to note that this bot represents a very simple initial solution and that it was built with the intent to expand and grow in usability and complexity. Throughout my discussion I will provide an overview of the current state of the bot as well as plans for future developments.

The first challenge to overcome in creating my bot was to represent the document library in a way that is accessible to search and summarization functions. That is, the intent of the project was for the bot to access files stored in common format like pdf. For this project, I used the *extract\_text()* function from the PyPDF2 library to convert the contents of each document to a string. Each instance of my bot uses this function which is very inefficient. Future development of the bot would include creating a more permanent representation of the files.

Next came the task of determining if the user’s input was a search query or not. For this project, I used a simple function that checked for key words related to searching in the user’s input. This function is more of a placeholder than anything. The ideal state would be to use a classification algorithm to determine if the input is a search or not.

Once the input is defined as a search, it is passed to the search function. This function uses vectorization to represent both the searched documents and the search queries. Specifically, this function uses the *TfidfVectorizer()*, *fit\_transform(),* and *transform()* functions from the sklearn library. Once the query and searched text are vectorized, I used the *cosine\_similarity()* function to find the cosine similarity between the query and the documents. I then returned the document with the most similarity to the query. For future development, I would like to parse the documents texts into small chunks of text and use a search algorithm like a\* search with cosine similarity for its heuristic to efficiently search the chunks and return a group of similar chunks for summarization.

Once the most similar document is identified, the document’s text is passed to the summarize function. This function leverages the BART LLM to tokenize the text, then summarize the text. As I previously mentioned, future development would involve passing multiple smaller chunks of text to the summarize function to then be summarized. I would also like to pass the source of the text chunk so that bot returns answers with sources to the user. Finally, I would also like to add functionality using the BART LLM for a simple chat interaction for cases where the user does not submit a search query.

In conclusion, this bot is a very simple solution leveraging several concepts from the course to assist a user in gaining insight from a document library. This bot is far from being a deployable real-world solution but is a foundation from which to work toward that end.

**Figure 1.**

*Screenshot of bot running on local machine.*

*A screenshot of a computer program

Description automatically generated*

**References**

Geeksforgeeks. (2024, April 22). *Convert PDF to TXT File Using Python.* <https://www.geeksforgeeks.org/convert-pdf-to-txt-file-using-python/>

Payong, A. (2024, September 30). *BART Model for Text Summarization.* Digital Ocean. <https://www.digitalocean.com/community/tutorials/bart-model-for-text-summarization-part1>

Scikit learn. (2024). *Feature Extraction.* [*https://scikit-learn.org/stable/modules/feature\_extraction.html*](https://scikit-learn.org/stable/modules/feature_extraction.html)

Scikit learn. (2024). *Cosine\_similarity. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine\_similarity.html*