Intelligent Document Search- Vineet Srivastava

A. **Link to GitHub-repo for code**: https://github.com/vineet1409/intelligent-search-tool.git
B. **Writeup to the approach**

Approach to the Problem:

1. **Data Preprocessing:**

- Multiple datasets containing information about brands, product categories, offers, and retailers were loaded.

- Essential data preprocessing tasks were performed such as renaming columns for uniformity, dropping unnecessary columns, and merging datasets. The primary intent behind merging was to create a unified table, streamlining the search and retrieval process based on user queries.

- Missing values, especially in the 'RETAILER' column, after merging of dataset, were addressed by filling them with a placeholder 'NA'.

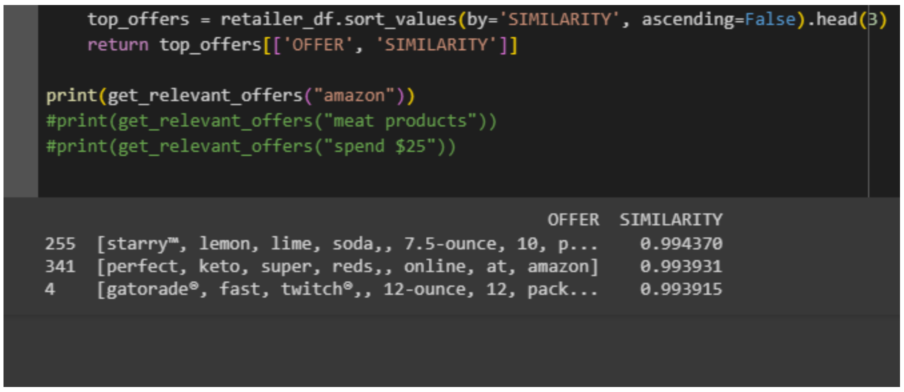
2. **Word2Vec Approach**:

- Utilized Gensim's Word2Vec to generate word embeddings from the various datasets. The model was trained with a window size of 5 and a vector size of 100, aiming to capture semantic meanings between words.

- For representation, the average vector of words in the text was computed to represent offers.

- Relevance was computed using cosine similarity between the query's vector and offers' vectors. The highest similarity scores were deemed the most relevant.

Results Word2Vec Model:



```
    top_offers = retailer_df.sort_values(by='SIMILARITY', ascending=False).head(3)
    return top_offers[['OFFER', 'SIMILARITY']]

print(get_relevant_offers("amazon"))
#print(get_relevant_offers("meat products"))
#print(get_relevant_offers("spend $25"))
```

```
                                                OFFER  SIMILARITY
255  [starry™, lemon, lime, soda,, 7.5-ounce, 10, p...    0.994370
341  [perfect, keto, super, reds,, online, at, amazon]    0.993931
4    [gatorade®, fast, twitch®,, 12-ounce, 12, pack...    0.993915
```

3. **Vector Database Creation with OpenAI***:*

- A vector database was conceived using the Chroma library and OpenAI embeddings. This method embeds textual information from the processed final dataset, storing it in a concise vector format.

- Textual data was split into appropriate chunks to comply with model constraints.

- Utilizing the ***OpenAI API***, embeddings were obtained for these text segments, which were then securely stored in the ***Chroma vector database***. ***The vector databases indexing algorithms helps in efficient retrieval of relevant contexts.***

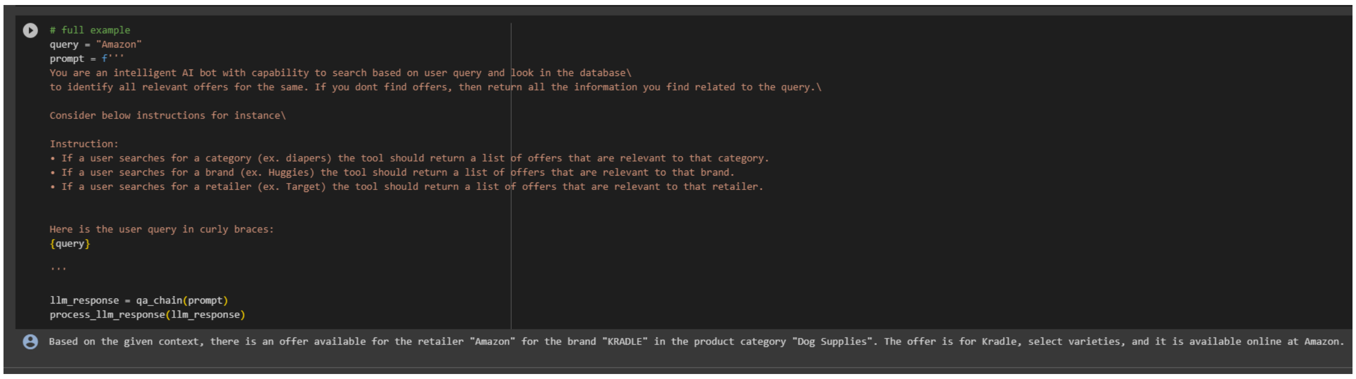
4. **Retrieval and Question-Answering**:

- The vector database was optimized for swift retrieval of relevant documents based on user inquiries. Users can probe a category, brand, or retailer, and the system swiftly fetches pertinent offers.

- Integration of the ***RetrievalQA chain***, underpinned by the OpenAI model, allowed for a seamless blend of vector database retrieval with the linguistic prowess of OpenAI.

- To bolster response accuracy and speed, a GPT3.5 Turbo from OpenAI was interwoven into the system.

<u>Results:</u>



```
# full example
query = "Amazon"
prompt = f'''
You are an intelligent AI bot with capability to search based on user query and look in the database\
to identify all relevant offers for the same. If you dont find offers, then return all the information you find related to the query.\

Consider below instructions for instance\

Instruction:
• If a user searches for a category (ex. diapers) the tool should return a list of offers that are relevant to that category.
• If a user searches for a brand (ex. Huggies) the tool should return a list of offers that are relevant to that brand.
• If a user searches for a retailer (ex. Target) the tool should return a list of offers that are relevant to that retailer.


Here is the user query in curly braces:
{query}

'''


llm_response = qa_chain(prompt)
process_llm_response(llm_response)
```

```
Based on the given context, there is an offer available for the retailer "Amazon" for the brand "KRADLE" in the product category "Dog Supplies". The offer is for Kradle, select varieties, and it is available online at Amazon.
```

5. ***Decision to Use Vector Database over Word2Vec:***

- While Word2Vec offers a robust understanding of semantic relationships, its reliance on averaging word vectors could potentially dilute the contextual richness of certain words.

- On the other hand, the vector database, empowered by OpenAI embeddings, provides a holistic approach, leveraging extensive datasets and understanding not only word semantics but broader contexts.

- Given the intricate nature of the problem and the necessity for nuanced responses, the vector database in tandem with OpenAI embeddings emerged as the superior choice.

Assumptions:

- The primary focus of user queries would center around categories, brands, or retailers and offers related to the same.

- In cases of missing retailer information, an 'NA' placeholder was used, under the assumption that certain brands might not have specific retailers linked to them.

Tradeoffs:

- Leveraging a vector database enhances retrieval speeds but might cause minor information loss during the embedding phase.

- Use of OpenAI API keys might be costly for large scale datasets, so open-source embedding models such hugging face embeddings or sentence transformers (sentence-transformers/all-MiniLM-L6-v2) could be used.

Conclusion:

The constructed solution meticulously processes datasets, transmutes them into an efficiently searchable vector format, and taps into OpenAI's capabilities for precise and swift retrieval and answer generation. This approach is designed not only to meet the technical requirements but also to provide an intuitive and enriching user experience, making it a standout solution.

C. Instructions on how to run your tool locally, if applicable

1. Firstly, run the "preprocessing" notebook under "notebooks" folder for processing and merging of datasets. The intent seems to be to create a unified dataset (df_merge_final) that provides a holistic view of each brand's product category, its associated offers, and the retailers providing those offers.

2. Then run the "VectorDB_creation_ChromaDB_OpenAI" notebook under "notebooks" folder to create a vector-database for the dataset.

3. Create a virtual environment locally and install all the requirements or dependencies by "pip install -r requirements.tx".

4. Run the python script to check if everything is set up properly "python main_py_file.py". The script uses vector databases for semantic search based on user's query. By default the query mentioned in script is "amazon". Be sure to add your openapi-key in the script.

5. Finally run the UI tool based on streamlit by "streamlit run streamlit-app_main_file.py".