Title:

Module - 1 - Report

Name:

Afolami Timothy Olawale

Submission Date : 28-4-2024

Objectives

Task 1: E-commerce Dataset Cleaning
Objective: Ensure the dataset is clean and ready for analysis and vectorization.
Key Actions: Remove duplicates, handle missing values, and standardize formats.

Task 2: Vector Database Creation
Objective: Set up a vector database using Pinecone to store product vectors.
Key Actions: Define the database schema and integrate with Pinecone.

Task 3: Similarity Metrics Selection
Objective: Choose and justify the similarity metrics used to compare product vectors.
Key Actions: Evaluate different metrics (e.g., cosine similarity, dot product) and select the best fit based on the dataset characteristics.

Endpoint 1: Product Recommendation Service
Functionality: Handle natural language queries to recommend products, including safeguards against bad queries and sensitive data exposure.
Input: Customer's natural language query.
Output: Product matches array and a natural language response within specified constraints.

Implementation Flow

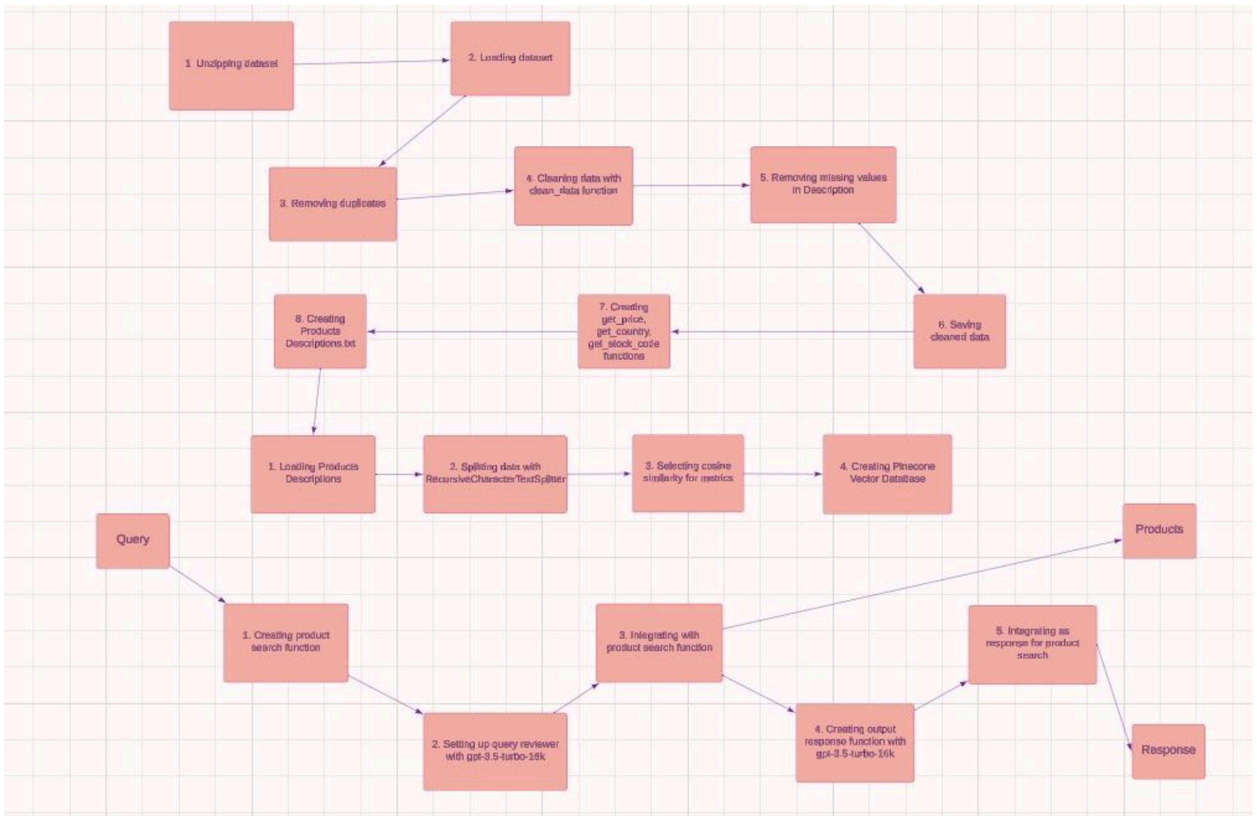
1. Data Exploration and Cleaning
    1. Unzipping dataset
    2. Loading dataset
    3. Removing duplicates
    4. Cleaning the data and making sure they are in the right structure using a clean_data function.
    5. Removing missing values in Description
    6. Saving cleaned data
    7. Creating functions like - get_price, get_country, and get_stock_code for data retrieval
    8. Creating a text file to store the Products Descriptions.

2. Creating Vector Database
    1. Loading the Products Descriptions file with Langchain TextLoader
    2. Splitting the data into chunks using RecursiveCharacterTextSplitter, with a chunk size of 50 and overlap of 10
    3. Based on research result, selecting cosine similarity for selection metrics
    4. Creating a Pinecone Vector Database, using openai embeddings, and Langchain PineconeVectorStore module.

3. Product Search Implementation
    1. Creating a product search function that returns similar items based on query.
    2. Setting up a query reviewer powered by gpt-3.5-turbo-16k to review query. Setting this with a crafted prompt.
    3. Integrating with the product search function.
    4. Creating an output response function powered by gpt-3.5-turbo-16k to create short comments on the product after query.
    5. Integrating this as a response for the product search.

4. Endpoint 1:
    1. Creating an html file for the front end
    2. Creating a flask endpoint that is sending out response and products based on the query
    3. Integrating this into the frontend
    4. User sends in a query, and the response is sent back to the user.

Diagrams:


1. Unzipping dataset
2. Loading dataset
3. Removing duplicates
4. Cleaning data with clean_data function
5. Removing missing values in Description
6. Saving cleaned data
7. Creating get_price, get_country, get_stock_code functions
8. Creating Products Descriptions.txt
1. Loading Products Descriptions
2. Splitting data with RecursiveCharacterTextSplitter
3. Selecting cosine similarity for metrics
4. Creating Pinecone Vector Database
Query
1. Creating product search function
2. Setting up query reviewer with gpt-3.5-turbo-16k
3. Integrating with product search function
Products
4. Creating output response function with gpt-3.5-turbo-16k
5. Integrating as response for product search
Response

Flow chart link: https://lucid.app/lucidchart/d535720e-3c61-4e6b-b202-d1c30713dab7/edit?viewport_loc=-3436%2C-935%2C4126%2C1903%2C0_0&invitationId=inv_fd504ddd-4061-45ce-9b09-60a69bcad776

Key Decisions:

1. Using cosine similarity instead of Product Search for similarity search in the database
2. Carefully crafting prompts that best handles the query and give good comments as response.

Conclusion:

By the end of this module, high level efficient functions and methods are available for intrgration with other modules.

Functions like clean_data, get_price and so on are stored in the utils.py file for easy implementation.

References:

Pandas
OpenAi
Langchain
Pinecone