Fashion Product Categorization using RAG

Introduction

- Fashion Product Categorization using RAG is a project aimed at developing a system that utilizes advanced natural language processing (NLP) techniques to categorize fashion products into specific sub-categories and retrieve their mandatory attributes. The system employs the Retrieval-Augmented Generation (RAG) model, which combines retrieval-based and generation-based approaches to provide accurate and contextually relevant responses.
- Fashion products often belong to various sub-categories such as shirts, t-shirts, sweatshirts, etc., each with its own set of mandatory attributes. Identifying the correct sub-category and its associated mandatory attributes is crucial for effective product management, inventory tracking, and customer satisfaction.
- By leveraging the RAG model, this project offers an intelligent solution for automating the categorization process and retrieving relevant product attributes based on a given taxonomy. Users can input the name of a fashion product, and the system will generate a response containing the corresponding sub-category and its mandatory attributes.
- The primary goals of this project include:
- Providing an intuitive user interface for querying fashion product information.
- Harnessing the power of NLP models like RAG to accurately categorize products and retrieve attributes.
- Enhancing the efficiency of fashion product management and classification processes

workflow

- Data Loading:

  The code starts by defining a class JSONLoader responsible for loading JSON data from a file specified by the file_path argument. It handles exceptions such as FileNotFoundError and JSONDecodeError.

  The JSON data contains information about product categories and their attributes.

- Vector Store Creation:

  After loading the JSON data, an instance of JSONLoader is created with the path to the JSON file.

  The RAGManager class is initialized with the JSONLoader instance. Inside the RAGManager, a Vector Store is created to store embeddings of each sub-category and its attributes.

  For each item in the JSON data, the sub-category and its mandatory attributes are extracted. The HuggingFaceEmbeddings class is used to generate embeddings for both the sub-category and its attributes.

  The embeddings are stored in the Vector Store along with the attributes.

- Recommendation Retrieval:

  The retrieve_with_rag method in the RAGManager class is responsible for retrieving recommendations based on a given product name.

  When a product name is provided, its embedding is generated using the embed_query method from HuggingFaceEmbeddings.

  Cosine similarity is calculated between the product embedding and the embeddings of all sub-categories in the Vector Store.

  The sub-category with the highest similarity score is selected as the recommendation. If the similarity score is below 0.5, indicating low similarity, no recommendation is made.

- Flask Application Setup:

  A Flask application is initialized with FlaskApp class, which takes an instance of RAGManager as an argument.

  Routes are configured for handling HTTP requests:

  The root route ("/") serves both GET and POST requests. When a POST request is received with a product name, the recommendation is retrieved using the RAGManager, and the result is rendered using Jinja templates.

  If the recommendation is found, it is rendered on the result.html page along with the mandatory attributes. If not, an error message is displayed on the error.html page.

- Main Function:

  In the main function, the file path to the JSON data is specified. The JSON data contains information about product categories and their attributes.

  An instance of JSONLoader is created to load the JSON data.

  An instance of RAGManager is initialized with the JSONLoader instance.

  Finally, an instance of FlaskApp is created with the RAGManager instance, and the Flask application is run.

  This workflow describes how the code loads data, creates a recommendation system using RAG methodology, sets up a Flask application to interact with users, and provides recommendations based on user input.

Result:

INPUT PAGE

Product Classification

Product Name:
shirts

Submit

RESULT PAGE

Result

Product Name: shirts

Sub-Category: Shirts

Mandatory Attributes:

- Brand: M
- Size Chart: M
- Gender: M
- Colour: M
- Size: M
- Fabric: M