**Project Documentation: Image and Text Search for Fashion Products**

**Github link:-** [**https://github.com/NiteshAnand190/Image-Recommendation-Matcher**](https://github.com/NiteshAnand190/Image-Recommendation-Matcher)

**Deploy Streamlit link:-https://image-recommendation-matcher.streamlit.app/**

**Deploy Render link:-** [**https://image-recommendation-matcher.onrender.com**](https://image-recommendation-matcher.onrender.com)

**Kaggle-dataset:-https://www.kaggle.com/datasets/paramaggarwal/fashion-product-images-small**

**Overview**

This project aims to implement a search functionality for fashion products using both text and image queries. It involves processing fashion product data, extracting image and textual embeddings, and using machine learning techniques to compute similarity between images and text. The project is built using Python 3.11 and utilizes multiple libraries like TensorFlow, scikit-learn, pandas, and Streamlit.

**Steps Breakdown**

1. **Data Extraction and Preparation (1st step.py)**:
   * The project begins by fetching product data from a CSV file stored on Google Drive. This data includes product attributes such as gender, category, and product name.
   * Images related to the products are also downloaded from Google Drive, where each image is identified by a unique ID.
   * The images are downloaded and saved locally in a folder called new\_images, and a subset of the data is saved in a new CSV file (new\_style.csv).
2. **Image and Text Embedding Extraction (2nd step.py)**:
   * Image embeddings are generated by processing the images using a pre-trained ResNet50 model. The images are resized, preprocessed, and passed through the model to extract feature embeddings.
   * The textual data is processed using the TF-IDF Vectorizer, which converts categorical data like gender, subcategory, and product name into numerical embeddings.
   * Both the image and text embeddings are normalized and saved to disk for future use.
3. **Similarity Search (3rd step.py)**:
   * The project allows users to search for products based on either a text query or an image query.
   * Dimensionality reduction is performed using PCA to reduce the image and text embeddings to a manageable size.
   * Cosine similarity is then used to find the most similar products based on the query, either text or image.
   * Results are displayed as a list of similar products, with the corresponding product ID and image shown.
4. **User Interface (app.py)**:
   * A simple user interface is created using Streamlit. The UI allows users to either input a text query or upload an image to search for similar products.
   * For a text query, users can input a description (e.g., "blue shirt"), and the system will return similar products based on the textual embeddings.
   * For an image query, users can upload an image, and the system will find similar products based on the image's feature embeddings.

**Technologies Used:**

* **TensorFlow**: For image preprocessing and feature extraction using the ResNet50 model.
* **scikit-learn**: For dimensionality reduction (PCA) and TF-IDF vectorization.
* **pandas**: For data manipulation and handling the product dataset.
* **Streamlit**: For creating a simple, interactive user interface.
* **NumPy**: For handling array-based operations.

**Challenges Faced and Solutions:**

* **Complexity**: This was a high-end project involving advanced machine learning techniques for image and text embeddings. The challenge of combining image-based and text-based search was daunting and required a deep understanding of machine learning concepts and model integration.
* **Learning Curve**: Since many of these concepts were new to me, I had to rely on Google, YouTube, and online resources to learn and implement them effectively. This added to the time required to complete the project.
* **Project Delay**: Due to the complexity of the project and the steep learning curve, there was a delay in its completion. The time was needed to understand and implement the various components, such as ResNet50, PCA, TF-IDF, and cosine similarity.

**Deployment Notes:**

* **Deployment Limitations**: While the core functionality of the project works as expected, the deployment may not always function as intended due to the high-end nature of the project and my limited knowledge in deployment. Issues may arise in the deployment setup or compatibility with certain systems.
* **Local Execution**: However, if you download the dataset and CSV file from Google Drive or Kaggle and run the project locally using Streamlit, it will work fine without deployment-related issues.

**Conclusion:**

Despite the challenges, I did my best to implement the functionality and reach the desired outcome. This project goes beyond my current knowledge, but through persistence and learning from available resources, I was able to make significant progress. The delay in the project was mainly due to the time spent acquiring the necessary skills and knowledge.

**Requirements (requirements.txt):**

streamlit>=1.25.0

numpy>=1.24.0

pandas>=1.5.0

scikit-learn>=1.2.0

tensorflow>=2.13.0,<2.19.0