FETCH TAKE HOME - WRITE UP
OFFER SEARCH TOOL

My Approach to the Problem:

1. **Initial Data Analysis:**
    - I began with three primary datasets: brand_category.csv, categories.csv, and offer_retailer.csv. A comprehensive analysis of this data was essential to understand the relationships between offers, brands, categories, and retailers.
2. **Initial Attempt - Entity Recognition:**
    - My initial idea was to utilize entity recognition to accurately identify brands and categories from user input. The rationale was to use off-the-shelf models for this purpose.
    - However, I quickly realized that the generic models weren't yielding satisfactory results for this specific dataset. The inherent challenge lay in the uniqueness of our data – consisting mainly of brand and category names – which made direct fine-tuning of the models problematic.
    - To navigate this, I attempted to fabricate some data and use it for fine-tuning purposes. Despite my efforts, the results remained subpar. This led me to pivot from the entity recognition approach altogether.
3. **Embedding Generation with Transformer Models:**
    - Recognizing the potential of embeddings to capture semantic relationships, I decided to generate embeddings for the textual data in the datasets. This approach was chosen over traditional text search methods for its ability to capture deeper semantic meanings and its superior performance in similarity-based tasks. For example, when I searched for ‘deodorant’, I got results of body-spray.
    - In the demo.ipynb notebook, I employed the Hugging Face model, a transformer-based architecture, to create these embeddings. Every unique brand, category, and offer description was passed through the model to obtain its corresponding embedding.
    - To know if an offer is directly relevant to a query (brand, category), I added the columns offer, brand and corresponding category to get a new column. And that column was used to generate embeddings for offers. So now, if there are any brands in the offers, they would directly show up.
4. **Embedding-based Search:**
    - With embeddings in hand, the search logic was then built around the concept of similarity in the embedding space. For any user query, I generate embeddings in real-time and then compare them with the pre-generated embeddings for offers, brands, and categories to fetch the most relevant results.

5. **Search Logic Refinement:**
    - To ensure optimal relevance in search results, I implemented a logic that considers the nature of the user's query. If the query closely matches a brand, relevant offers associated with that brand are prioritized. The same logic applies to categories and retailers.
    - **Result Prioritization:**
        i. Offers: If the user's query closely matches an offer (based on score thresholds), return the top matching offers.
        ii. Brands: If the query aligns more with a brand, fetch offers associated with that brand. If no brand-specific offers, delve into brand-category relationships for relevant results.
        iii. Categories: If the query is category-centric, return offers from the top-matching category or its associated parent category if direct matches aren't available.
    - **Result Presentation:**
        i. Display results sorted by their similarity scores, ensuring the most relevant appear first. Each result is accompanied by its similarity score to the user's query.
6. **Deployment & Scalability:**
    - For deployment, I containerized the application using Docker, ensuring consistency across different environments. I chose AWS ECS with Fargate for cloud deployment, leveraging its serverless capabilities. The integration of the Application Load Balancer (ALB) ensures efficient handling of user traffic, distributing it across multiple instances of the application when needed.

Assumptions & Trade-offs:

- Most of my conclusions were based on intensive testing using various embedding models, NER models, and similarity thresholds.
- Relying on the Hugging Face model for embedding generation was a balance between computational intensity and the superior quality of embeddings.

Link to GitHub repository:

https://github.com/preethampathi2305/fetch-take-home/tree/main

Link to Application:

http://fetch-balancer-916446164.us-east-1.elb.amazonaws.com:8501/