Product Recommendation System Based on Text Embeddings

1. Objective

The goal of this system is to recommend similar products based on their descriptions. By converting product descriptions into embeddings (vector representations), we can compute similarity between products and suggest the most relevant items.

2. Solution Approach

Overview

- **Data Input**: The dataset consists of product names and their descriptions.
- **Text Embeddings**: We use a pre-trained language model to generate embeddings for product descriptions, converting textual data into numerical vectors.
- **Similarity Calculation**: Using cosine similarity, we compute the similarity between a selected product and other products in the dataset.
- **Recommendations**: For a given product, the system returns the most similar products based on the calculated similarity scores.

Process Flow

1. **User Input**: The user provides a product name.
2. **Data Retrieval**: The system retrieves the corresponding product description.
3. **Embedding Generation**: The description is converted into an embedding using a pre-trained model.
4. **Similarity Calculation**: Cosine similarity is computed between the target product and all other products.
5. **Output**: The system outputs the top N similar products.

3. Libraries Required

To run this solution, the following Python libraries are required:

- `pandas`: For data handling and manipulation.
- `sentence-transformers`: For generating text embeddings.
- `scikit-learn`: For calculating cosine similarity.

Install the necessary libraries by running:

```
bash
Copy code
pip install pandas sentence-transformers scikit-learn
```

4. Dataset

The dataset should be in CSV format with at least two columns:

- **name**: The product name.
- **description**: The product description.

Example:

name	description
Product X	This is a high-quality product...
Product Y	A premium product with advanced...

5. Code Implementation

Below is the complete Python code to implement the product recommendation system:

```
import pandas as pd
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity

# Load dataset (adjust the file path as necessary)
df = pd.read_csv('/mnt/data/dataset.csv')

# Initialize the pre-trained model from Hugging Face
model = SentenceTransformer('sentence-transformers/all-MiniLM-L6-v2')

# Generate embeddings for product descriptions
df['embeddings'] = df['description'].apply(lambda x: model.encode(x))

# Function to recommend similar products based on description
def recommend_similar_products(product_name, df, top_n=5):
    # Find the embedding of the given product name
    target_product = df[df['name'] == product_name]

    if target_product.empty:
        return f"Product '{product_name}' not found."

    # Get the embedding of the target product
    target_embedding = target_product['embeddings'].values[0].reshape(1, -
1)

    # Calculate cosine similarity between the target product and all others
    df['similarity'] = df['embeddings'].apply(lambda x:
cosine_similarity(target_embedding, x.reshape(1, -1)).item())

    # Sort by similarity, exclude the target product itself
    similar_products = df[df['name'] !=
product_name].sort_values(by='similarity', ascending=False)

    # Return top N similar products with their similarity scores
    return similar_products[['name', 'similarity']].head(top_n)

# Example usage: Recommend similar products to "Product X"
similar_products = recommend_similar_products('Product X', df)
print(similar_products)
```

Explanation of Key Components:

1. **Model Initialization**: The SentenceTransformer (`all-MiniLM-L6-v2`) model from Hugging Face is used to convert product descriptions into embeddings.
2. **Embedding Generation**: The `model.encode()` function generates a 384-dimensional embedding (vector) for each product description.
3. **Cosine Similarity**: The `cosine_similarity` function computes the similarity between two embeddings. A higher cosine similarity value means the products are more similar.
4. **Recommendation Function**: The `recommend_similar_products` function takes a product name as input and returns the top N similar products based on their cosine similarity scores.

Function Parameters:

- **product_name**: The name of the product for which similar products are needed.
- **df**: The DataFrame containing product data (names, descriptions, embeddings).
- **top_n**: The number of similar products to return (default is 5).

6. Example Usage

To recommend products similar to "Product X":

```
similar_products = recommend_similar_products('Product X', df)
print(similar_products)
```

Output:

```
plaintext
Copy code
          name  similarity
2    Product Y      0.8745
7    Product Z      0.8521
9    Product W      0.8457
4    Product V      0.8329
11   Product T      0.8210
```

7. Results

The function will return the names of the most similar products along with their similarity scores, enabling the user to understand which products are most relevant to the selected product.

8. Customization

- **Top N Similar Products**: You can modify the `top_n` parameter to return more or fewer products.
- **Filter by Additional Criteria**: You can add additional filters (e.g., price range, category) to fine-tune the recommendations.

9. Performance Optimization

For larger datasets, you can use libraries like `FAISS` (Facebook AI Similarity Search) to speed up similarity searches by indexing the embeddings.

10. Conclusion

This system provides an efficient way to recommend similar products based on their descriptions using state-of-the-art language models. By leveraging cosine similarity on embedding vectors, the system can accurately find and rank similar products, offering a robust solution for product recommendation tasks.