AI-based Product Recommendation System

Title: *AI-based Product Recommendation System Using FAISS and Hugging Face Embeddings*

Author(s): Chandan Kumar , Shiv Kumar, Apoorv Kulshrestha, Khozema Goodluck

---

Table of Contents

---

1. Introduction

Objective

The goal of this project is to create an AI-based product recommendation system that can recommend products to users based on a text query. The system leverages FAISS (Facebook AI Similarity Search) for fast similarity search and Hugging Face models to generate embeddings for products.

This system is useful for e-commerce platforms, where users are interested in finding similar products based on their preferences. The recommendations will be generated from an existing dataset of products, including product details like product name, category, rating, price, and reviews.

Background

In the highly competitive world of e-commerce, providing relevant and accurate recommendations is a key factor in improving customer experience and driving sales. Traditional recommendation systems like collaborative filtering are limited because they don't account for semantic understanding of product features. Hence, we explore a deep learning-based approach using text embeddings to improve product recommendations.

Scope

The system focuses on text-based product recommendations using product names. The project scope includes:

- Using product names and associated metadata for semantic recommendations.
- Fast similarity search using FAISS.
- Embedding generation using pre-trained models from Hugging Face.

---

2. System Design Overview

Overall System Architecture

The recommendation system is divided into the following layers:

1. **Data Layer**:
    - Stores product details, including product names, categories, ratings, and reviews.
    - Data is preprocessed to remove inconsistencies (e.g., missing ratings) and prepare it for embedding generation.
2. **Embedding Layer**:
    - Product names are converted into dense vector embeddings using a Hugging Face model. These embeddings capture semantic similarities between product names.
3. **Similarity Search Layer**:
    - FAISS is used to store and index the product embeddings, enabling fast similarity search when a user inputs a query.
4. **Recommendation Layer**:
    - Based on user input (product name), the system performs similarity search within the FAISS index to find and recommend the most relevant products.

System Diagram

Here is a high-level system design:

```
Python
User Query → Embedding Layer (Hugging Face Model) → FAISS Index → Similarity Search → Recommended Products
```

---

3. Design and Implementation

Data Collection and Processing

We use an Amazon product dataset containing columns such as product_id, product_name, category, discounted_price, actual_price, discount_percentage, rating, rating_count, review_content, etc. Below is an example row from the dataset:

product_id	product_name	category	discounted_price	rating	review_content
101	Wireless Headphone	Electronics	$59.99	4.5	Good sound...

Key tasks include:

- Cleaning data (handling missing or invalid ratings, converting prices to float).
- Ensuring uniformity in product names (e.g., converting to lowercase, removing special characters).

Embedding Generation

We use a pre-trained model from Hugging Face (sentence-transformers/all-mpnet-base-v2) to generate embeddings for product names. Each product is transformed into a dense vector representing its semantic meaning.

```
Python
embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-mpnet-base-v2")
```

The embeddings capture relationships between products based on their names, allowing us to make more relevant recommendations.

FAISS Indexing

FAISS is used to store the embeddings and provide fast similarity search. Once the embeddings are generated, they are added to the FAISS index.

```python
vectorstore = FAISS.from_documents(docs, embeddings)
```

Recommendation Logic

When a user enters a query (e.g., "wireless headphones"), the system generates an embedding for the query and uses FAISS to find the top-N most similar products.

```python
def recommend_product(query, top_n=5):
    results = vectorstore.similarity_search(query, k=top_n)
    return results
```

The returned results include product details (name, price, rating, etc.) for easy display.

User Interface

We implemented the user interface using ipywidgets. A text input box is provided for users to enter a product name. The system then recommends products based on the similarity search.

```python
product_input = widgets.Text(description='Product Name:', placeholder='Enter a product name')
button = widgets.Button(description='Recommend')
output = widgets.Output()
```

```
def on_button_click(b):
    product_name = product_input.value
    recommended_products = recommend_product(product_name)
    display_recommendations(recommended_products)
```

---

4. Challenges Faced

1. **Data Quality**: Handling missing or incomplete data, especially ratings and reviews, was a major challenge. We had to implement default values (e.g., setting missing ratings to 0.0) to ensure consistent results.
2. **Embedding Selection**: Choosing the right model for generating embeddings was critical. Pre-trained models were tested for relevance in capturing product semantics. We eventually selected the sentence-transformers/all-mpnet-base-v2 model, which provided a good balance between performance and accuracy.
3. **FAISS Optimization**: FAISS works well for large datasets, but it required tuning for speed and accuracy trade-offs. Finding the right balance between the number of dimensions and search efficiency was crucial.

---

5. Lessons Learned

1. **Importance of Preprocessing**: Proper preprocessing of the data, especially handling missing values and converting data types (like price and rating), was critical to the success of the project.
2. **Model Selection**: Different embedding models behave differently with various datasets. Hugging Face provides multiple pre-trained models, but selecting the one that works best for your specific use case is important.
3. **Real-time Search with FAISS**: FAISS provides excellent performance for similarity search, especially with large-scale datasets, and understanding its indexing and search parameters greatly improved the project's performance.

---

6. Conclusion

The AI-based product recommendation system successfully recommends products based on the semantic meaning of the product names. By using FAISS for similarity search and Hugging Face embeddings for generating semantic vectors, we were able to create a scalable and efficient recommendation engine.

The system can be further enhanced by integrating additional product metadata (such as user behavior or product reviews) to provide even more personalized recommendations.

7. Future Work

1. **Improve Embeddings**: Incorporate additional metadata (e.g., product descriptions, reviews) into the embeddings to improve recommendation accuracy.
2. **User Behavior Tracking**: Implement user behavior tracking to recommend products based on past interactions.
3. **Performance Optimization**: Further optimization of FAISS parameters (e.g., indexing strategies) to enhance performance for even larger datasets.

8. References

- Hugging Face Models: https://huggingface.co/models
- FAISS Documentation: https://github.com/facebookresearch/faiss
- Pandas Library: https://pandas.pydata.org/