Jupyter Notebook Code with Documentation

```
Unset
# Product Recommendation System Using LangChain and FAISS

## Overview
This notebook demonstrates a product recommendation system using Amazon product
data. The system generates embeddings from product descriptions and reviews
using a transformer model, then stores these embeddings in a FAISS vector store
for efficient similarity search.

## Requirements
- Python 3.x
- Libraries:
  - langchain
  - langchain-community
  - sentence-transformers
  - faiss-cpu
  - ipywidgets
  - pandas
  - numpy
  - torch

### Installation
To install the required libraries, run the following commands:


!pip install langchain langchain-community sentence-transformers faiss-cpu
ipywidgets
```

Import Necessary Libraries

```python
import pandas as pd
import numpy as np
import torch
from transformers import AutoTokenizer, AutoModel
from sklearn.metrics.pairwise import cosine_similarity
import ipywidgets as widgets
from IPython.display import display, HTML
from langchain.vectorstores import FAISS
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.docstore.document import Document
```

Load the Dataset

```python
# Load the datasets
amazon_data = pd.read_csv('amazon.csv')

# Display the first few rows of the dataset to understand its structure
print("\nAmazon Data Overview:")
print(amazon_data.head())
```

Basic Data Analysis

```python
# Basic statistics to understand numerical columns
print("\nAmazon Data Statistics:")
print(amazon_data.describe())

# Check for missing values
print("\nMissing Values in Amazon Data:")
print(amazon_data.isnull().sum())
```

Data Preprocessing

```python
# Fill missing values with an empty string to avoid issues during tokenization
amazon_data.fillna('', inplace=True)

# Combine product attributes into a single text column for embedding generation
amazon_data['text'] = amazon_data['product_name'] + ' ' +
amazon_data['category'] + ' ' + amazon_data['about_product'] + ' ' +
amazon_data['review_content']

# Check the new 'text' column
print("\nAmazon Data Text Feature:")
print(amazon_data['text'].head())
```

Load Tokenizer and Model

```python
# Load the tokenizer and model from Hugging Face
tokenizer =
AutoTokenizer.from_pretrained('sentence-transformers/all-MiniLM-L6-v2')
model = AutoModel.from_pretrained('sentence-transformers/all-MiniLM-L6-v2')
```

Create Embeddings

Python

```
# Function to create embeddings for the text data
def get_embeddings(text):
    """Generate embeddings for the provided text using the transformer
model."""
    inputs = tokenizer(text, return_tensors="pt", padding=True,
truncation=True, max_length=128)
    with torch.no_grad():
        outputs = model(**inputs)
    return outputs.last_hidden_state.mean(dim=1).numpy()

# Apply embedding generation to the text column
amazon_data['embeddings'] = amazon_data['text'].apply(lambda x:
get_embeddings(x))

# Check the embeddings of the first product
print("\nAmazon Data Embeddings (first product):")
print(amazon_data['embeddings'].iloc[0])
```

Compute Cosine Similarity

Python

```
# Compute cosine similarity between products in the dataset
amazon_similarity =
cosine_similarity(np.vstack(amazon_data['embeddings'].values))
```

Prepare Documents for LangChain

```python
# Convert amazon_data into a format LangChain can use
docs = [
    Document(
        page_content=row['product_name'],
        metadata={
            'product_id': row['product_id'],
            'product_name': row['product_name'],
            'category': row['category'],
            'discounted_price': row['discounted_price'],
            'rating': float(row['rating'].replace('|', '')) if row['rating'] !=
'|' else 0.0,
            'rating_count': row['rating_count'],
            'about_product': row['about_product']
        }
    )
    for _, row in amazon_data.iterrows()
]

# Initialize the HuggingFace Embeddings model
embeddings = HuggingFaceEmbeddings()

# Use FAISS to store embeddings
vectorstore = FAISS.from_documents(docs, embeddings)
```

Recommendation Function

Python

```python
# Recommendation function that handles product name input
def recommend_product(query, top_n=5):
    """Recommend products based on a query string."""
    results = vectorstore.similarity_search(query, k=top_n)

    # Return product details
    recommended_products = []
    for result in results:
        recommended_products.append({
            'product_name': result.metadata['product_name'],
            'category': result.metadata['category'],
            'about_product': result.metadata['about_product'],
            'rating': result.metadata['rating'],
        })

    return recommended_products
```

Interactive User Interface

Python

```python
# Create text input widget for product name
product_input = widgets.Text(description='Product Name:', placeholder='Enter a
product name')

# Create a button to trigger the recommendation
button = widgets.Button(description='Recommend')

# Create an output widget to display the results
output = widgets.Output()

# Function to display recommendations when the button is clicked
def on_button_click(b):
    with output:
        # Clear the previous output
        output.clear_output()

        # Get user input
        product_name = product_input.value
```

```
        # Get recommended products using the recommendation function
        recommended_products = recommend_product(product_name)

        # Display the recommended products in a table format
        if recommended_products:
            df = pd.DataFrame(recommended_products)
            display(HTML(df.to_html(index=False)))
        else:
            print("No products found.")

# Link the button click event to the function
button.on_click(on_button_click)

# Display the widgets and output area
display(product_input, button, output)
```

Conclusion

This notebook provides a complete workflow for creating a product recommendation system using embeddings and FAISS. You can further enhance this project by adding features like advanced filtering options, user feedback mechanisms, or visualizations of the recommendation results.

Notes

- Ensure that the **amazon.csv** file is available in your working directory for the code to run successfully.
- The provided model (**sentence-transformers/all-MiniLM-L6-v2**) should be suitable for general text embeddings. Feel free to experiment with other models as needed.