EXP NO: 4b	INFORMATIONRETRIEVAL USING SPACY
DATE: 12/8/25	

Aim:

Toimplement an information retrieval system that searches and ranks Amazon product reviews using SpaCy's NLP capabilities and TF-IDF vectorization.

Program:

Step 1: Import Required Libraries

```
import pandas as pd
import spacy
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

Step 2: Download and Load Dataset

```
import kagglehub
import os

# Download Amazon Fine Food Reviews dataset
path = kagglehub.dataset_download("snap/amazon-fine-food-reviews")
# Load the CSV file
df = pd.read_csv(os.path.join(path, "Reviews.csv"))
# Keep only review text and drop nulls, limit to 1000 for speed
df = df[['Text']].dropna()[:1000]
reviews = df['Text']
```

Step 3: Load SpaCy Model

```
nlp = spacy.load("en_core_web_sm")
```

Step 4: Define Preprocessing Function

```
def preprocess(text):




    text = text.lower()
    doc = nlp(text)
    tokens = [
        token.lemma_ for token in doc
        if token.is_alpha and not token.is_stop
    ]
    return " ".join(tokens)
```

Step 5: Apply Preprocessing

```
cleaned_reviews = reviews.apply(preprocess)
```

Step 6: Vectorize Reviews using TF-IDF

```
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(cleaned_reviews)
```

Step 7: Preprocess and Vectorize Queries

```
def  preprocess_query(query):  return
preprocess(query)


def vectorize_query(query):
    query_clean = preprocess_query(query)
    return vectorizer.transform([query_clean])
```

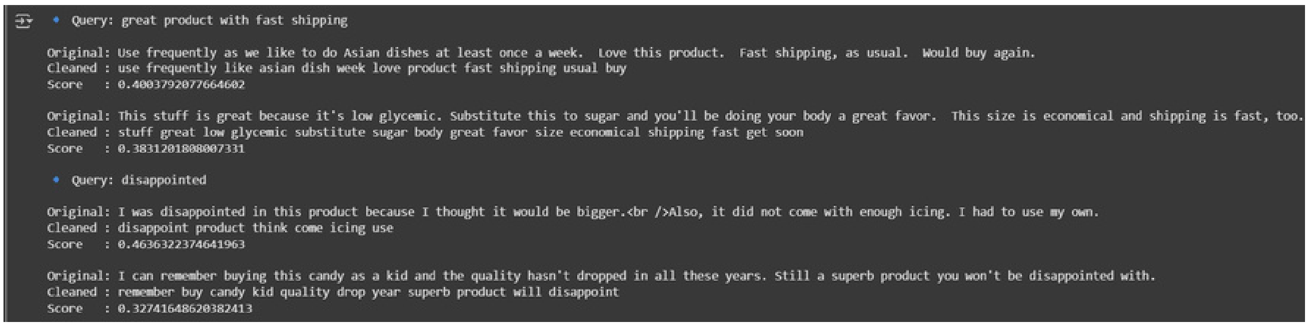
Step 8: Define Search Function

```
def search(query, top_k=2):

    query_vec = vectorize_query(query)
    similarity = cosine_similarity(query_vec, X).flatten()
    indices = similarity.argsort()[-top_k:][::-1] # top k results
    results = []
    for i in indices:
        results.append({
            "original": reviews.iloc[i],
            "cleaned": cleaned_reviews.iloc[i],
            "score": similarity[i]
        })
    return results
```

Step 9: Test Queries

```
# Query 1
print(" Query: great product with fast shipping")
for res in search("great product with fast shipping"):
    print("\nOriginal:", res["original"])
    print("Cleaned :", res["cleaned"])
    print("Score   :", res["score"])

# Query 2
print("\n Query: disappointed")
for res in search("disappointed"):
    print("\nOriginal:", res["original"])
    print("Cleaned :", res["cleaned"])
    print("Score   :", res["score"])
```

Output:



```
 • Query: great product with fast shipping

Original: Use frequently as we like to do Asian dishes at least once a week.  Love this product.  Fast shipping, as usual.  Would buy again.
Cleaned : use frequently like asian dish week love product fast shipping usual buy
Score   : 0.4003792077664602

Original: This stuff is great because it's low glycemic. Substitute this to sugar and you'll be doing your body a great favor.  This size is economical and shipping is fast, too.
Cleaned : stuff great low glycemic substitute sugar body great favor size economical shipping fast get soon
Score   : 0.3831201808007331

 • Query: disappointed

Original: I was disappointed in this product because I thought it would be bigger.<br />Also, it did not come with enough icing. I had to use my own.
Cleaned : disappoint product think come icing use
Score   : 0.4636322374641963

Original: I can remember buying this candy as a kid and the quality hasn't dropped in all these years. Still a superb product you won't be disappointed with.
Cleaned : remember buy candy kid quality drop year superb product will disappoint
Score   : 0.32741648620382413
```

Result:

The system successfully returned the most relevant reviews for user queries by lemmatizing text, removing stopwords, and computing cosine similarity, demonstrating effective document retrieval.