Zepto Retrieval System

Abhishek Rajesh Mistry
Motilal Nehru National Institute of Technology
Reg. Number: 20214504

Objective : To develop a solution to enhance the search experience for Zepto. The goal is to improve the relevance and accuracy of search results, ultimately leading to a better user experience and increased conversion rates.

Note: The following problem was solved on Kaggle.

Exploratory Data Analysis

I initially loaded the dataset using the Pandas module into a variable 'df'. I could view information regarding the dataset using the .info() (gives information regarding the dataset like number of columns, column labels, data types, memory usage, etc). Then with the help of the isnull() function in pandas, I could figure out how many null values each column had. I correspondingly removed all the tuples with null values.

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
import nltk
import numpy as np
```

+ Code + Markdown

Importing the dataset into a variable df

+ Code + Markdown

[8]:
```
df = pd.read_csv('/kaggle/input/flipkart-dataset/flipkart_com-ecommerce_sample.csv')
df.head()
```

[8]:

	uniq_id	crawl_timestamp	product_url	product_name	product_category_tree	pid	retail_price	discounted_price	image	is_FK_Advantage_product	description	product_rating
0	c2d766ca982eca8304150849735ffef9	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	SRTEH2FF9KEDEFGF	999.0	379.0	["http://img5a.flixcart.com/image/short/u/4/a/...	False	Key Features of Alisha Solid Women's Cycling S...	No rating available
1	7f7036a6d550aaa89d34c77bd39a5e48	2016-03-25 22:59:23 +0000	http://www.flipkart.com/fabhomedecor-fabric-do...	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	SBEEH3QGU7MFYJFY	32157.0	22646.0	["http://img6a.flixcart.com/image/sofa-bed/j/f...	False	FabHomeDecor Fabric Double Sofa Bed (Finish Co...	No rating available
2	f449ec65dcbc041b6ae5e6a32717d01b	2016-03-25 22:59:23 +0000	http://www.flipkart.com/aw-bellies/p/itmeh4grg...	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	SHOEH4GRSUBJGZXE	999.0	499.0	["http://img5a.flixcart.com/image/shoe/7/z/z/r...	False	Key Features of AW Bellies Sandals Wedges Heel...	No rating available
3	0973b37acd0c664e3de26e97e5571454	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	SRTEH2F6HUZMQ6SJ	699.0	267.0	["http://img5a.flixcart.com/image/short/6/2/h/...	False	Key Features of Alisha Solid Women's Cycling S...	No rating available

Information Regarding the Dataframe

```
[4]: print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20002 entries, 0 to 20001
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   uniq_id                  20000 non-null  object
 1   crawl_timestamp          20000 non-null  object
 2   product_url              20000 non-null  object
 3   product_name             20000 non-null  object
 4   product_category_tree    20000 non-null  object
 5   pid                      20000 non-null  object
 6   retail_price             19922 non-null  float64
 7   discounted_price         19922 non-null  float64
 8   image                    19997 non-null  object
 9   is_FK_Advantage_product  20000 non-null  object
 10  description              19998 non-null  object
 11  product_rating           20000 non-null  object
 12  overall_rating           20000 non-null  object
 13  brand                    14136 non-null  object
 14  product_specifications   19986 non-null  object
dtypes: float64(2), object(13)
memory usage: 2.3+ MB
None
```

Statistics about the 2 float columns

```
[5]: print(df.describe())
```

```
        retail_price  discounted_price
count   19922.000000      19922.000000
mean     2979.206104       1973.401767
std      9009.639341       7333.586040
min        35.000000         35.000000
25%       666.000000        350.000000
50%      1040.000000        550.000000
75%      1999.000000        999.000000
max    571230.000000     571230.000000
```

+ Code + Markdown

```
[6]: print(df.isnull().sum())
```

```
uniq_id                       2
crawl_timestamp               2
product_url                   2
product_name                  2
product_category_tree         2
pid                           2
retail_price                 80
discounted_price             80
image                         5
is_FK_Advantage_product       2
description                   4
product_rating                2
overall_rating                2
brand                      5866
product_specifications       16
dtype: int64
```

```
[7]: df_cleaned = df.dropna(subset=['discounted_price', 'retail_price', 'product_specifications', 'description'])
     print(df_cleaned.isnull().sum())
```

```
uniq_id                       0
crawl_timestamp               0
product_url                   0
product_name                  0
product_category_tree         0
pid                           0
retail_price                  0
discounted_price              0
image                         3
is_FK_Advantage_product       0
description                   0
product_rating                0
overall_rating                0
brand                      5850
product_specifications        0
dtype: int64
```

Since the ‘Brand’ attribute had a large number of null values, I assumed it would not have much effect on the training model, hence removing all 5850 tuples where 'brand' was null was undesirable since it would affect the training, and the brand name would also be mentioned in the product name. Hence we removed the null values in ‘discounted_price’, ‘retail_price’, ‘product_specifications’and ‘description’ and that fixed the problem of the presence of null values in all the attributes I thought were relevant to training my model.

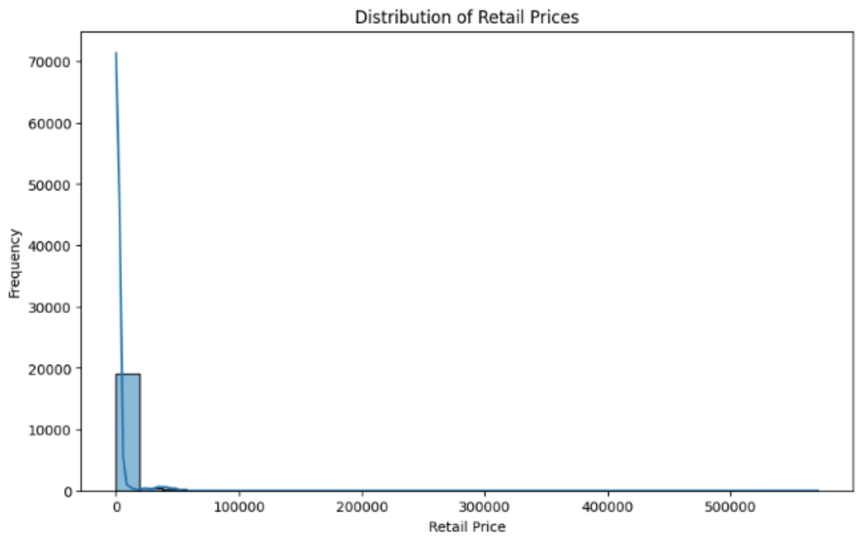
I then plotted distributions of Retail Prices and the Top Product Categories to understand more about the data I’m working with, what are the more frequent values, etc.

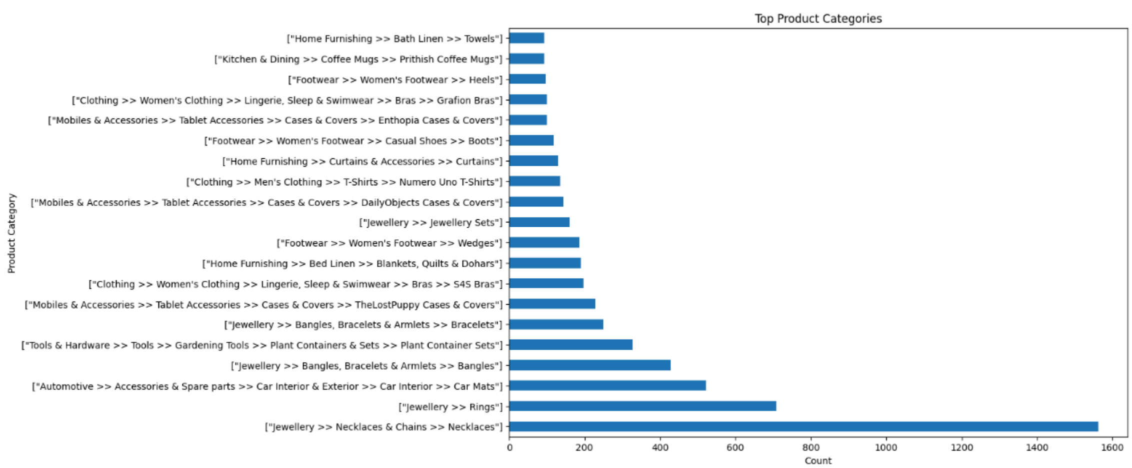
[19]:

```
plt.figure(figsize=(10, 6))
sns.histplot(df_cleaned['retail_price'], bins=30, kde=True)
plt.title('Distribution of Retail Prices')
plt.xlabel('Retail Price')
plt.ylabel('Frequency')
plt.show()
print('\n\n')

plt.figure(figsize=(12,8))
df_cleaned['product_category_tree'].value_counts().head(20).plot(kind = 'barh')
plt.title('Top Product Categories')
plt.xlabel('Count')
plt.ylabel('Product Category')
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version.
  with pd.option_context('mode.use_inf_as_na', True):
```





Retrieval System

Initially, I had to preprocess the textual data I was going to be working with. Preprocessing the textual data is important as it converts the raw text into a more structured format, that we can effectively implement in our ML models.

I initially made every letter in each word a lowercase letter, removed special characters and tokens, and tokenized each word. Lemmatization is the process of converting words into their root form. Due to System Constraints, I was unable to perform lemmatization, due to the large amount of textual data, but it is an important step in the preprocessing stage. We can use libraries like spaCy and nltk to implement it.

```python
import re
import nltk

# Download necessary NLTK data (skip wordnet if not lemmatizing)
nltk.download('punkt')

def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove special characters and numbers
    text = re.sub(r'\W+', ' ', text)
    # Tokenize the text
    words = nltk.word_tokenize(text)
    # Join the words back into a single string
    return ' '.join(words)

# Apply the preprocessing function to the combined text
df_cleaned['combined_text'] = df_cleaned['product_name'].fillna('') + ' ' + \
                      df_cleaned['product_specifications'].fillna('') + ' ' + \
                      df_cleaned['description'].fillna('') + ' ' + \
                      df_cleaned['product_category_tree'].fillna('')

df_cleaned['combined_text'] = df_cleaned['combined_text'].apply(preprocess_text)

df_cleaned.head()  # Check the first few rows
```

I used the Okapi BM25 Model for the Heuristic Approach. BM25 takes into account both term frequency (TF) and document length normalization to determine the relevance of a document to a given query. It follows the probabilistic retrieval framework, which assumes that relevant and non-relevant documents follow different statistical distributions.

```python
from rank_bm25 import BM25Okapi

tokenized_corpus = [doc.split() for doc in df_cleaned['combined_text']]
bm25 = BM25Okapi(tokenized_corpus)

def bm25_retrieve(query, top_k=10):
    processed_query = preprocess_text(query)
    tokenized_query = processed_query.split()
    scores = bm25.get_scores(tokenized_query)

    df_scores = pd.DataFrame({'score': scores, 'product_name': df_cleaned['product_name'],
                          'product_category_tree': df_cleaned['product_category_tree']})
    top_results = df_scores.sort_values(by='score', ascending=False).head(top_k)
    return top_results
```

For my ML approach, I used a SentenceTransformer or SBERT. It is based on the Transformer architecture where it transforms sentences into embeddings (which in this case are numerical representations of the sentences such that the ML model can easily understand them). With these embeddings, I was able to calculate the cosine similarity between each embedding to determine its similarity. Hence, we can now capture the semantic meaning of each sentence, making it easier to compare, search, and analyze the data.

```python
from sentence_transformers import SentenceTransformer, util

model = SentenceTransformer('all-MiniLM-L6-v2')

def sbert_retrieve(query, top_k=10):
    query_embedding = model.encode(query, convert_to_tensor=True)
    corpus_embeddings = model.encode(df_cleaned['combined_text'].tolist(), convert_to_tensor=True)

    #calculating cosine similarities
    cos_scores = util.pytorch_cos_sim(query_embedding, corpus_embeddings)[0]
    top_results = pd.DataFrame({
        'score': cos_scores.cpu().detach().numpy(),
        'product_name': df_cleaned['product_name'],
        'product_category_tree': df_cleaned['product_category_tree']
    })

    top_results = top_results.sort_values(by='score', ascending=False).head(top_k)
    return top_results
```

For model metrics, I developed a function to calculate the precision, recall, and F1 Score.

- Precision: measures how many predicted positive instances are actually positive
- Recall: measures how many of the actual positive instances were correctly defined by the model
- F1 Score: It is the harmonic mean between precision and recall.

```python
from sklearn.metrics import precision_score, recall_score, f1_score

def evaluate_retrieval_system(retrieve_func, queries, true_labels, top_k=10):
    precision_scores = []
    recall_scores = []
    f1_scores = []

    for query, true_label in zip(queries, true_labels):
        retrieved_items = retrieve_func(query, top_k)
        retrieved_labels = retrieved_items['product_name'].values

        # Convert true_label to a binary array indicating relevance
        relevant_retrieved = [1 if label in true_label else 0 for label in retrieved_labels]

        # Adjust the length of the true labels array to match retrieved items
        true_relevance = [1 if i < len(true_label) else 0 for i in range(top_k)]

        precision = precision_score(relevant_retrieved, true_relevance, average='macro')
        recall = recall_score(relevant_retrieved, true_relevance, average='macro')
        f1 = f1_score(relevant_retrieved, true_relevance, average='macro')

        precision_scores.append(precision)
        recall_scores.append(recall)
        f1_scores.append(f1)

    return np.mean(precision_scores), np.mean(recall_scores), np.mean(f1_scores)
```

```
[22]:
#test queries to run
sample_queries = [
    "Alisha Shorts",
    "Diamond Ring",
    "Laptop"
]
#expected labels to appear
true_labels = [
    ["Alisha Solid Women's Cycling Shorts"],
    ["Avsar Sachi Gold Diamond 18 K Ring", "Karatcraft Dulce Gold Diamond 18 K Ring", "His & Her Gold Diamond 18K Yellow Gold 18 K Ring", "Karatcraft Twinoh Gold Diamond 18 K Ring", "Karatcraft Wadell Gold Diamond 18 K Ring", "A
    ["Lenovo", "HP", "Asus", "Dell"]
]

precision_bm25, recall_bm25, f1_bm25 = evaluate_retrieval_system(bm25_retrieve, sample_queries, true_labels)
precision_sbert, recall_sbert, f1_sbert = evaluate_retrieval_system(sbert_retrieve, sample_queries, true_labels)

print("BM25")
print("Precision:", precision_bm25)
print("Recall:", recall_bm25)
print("F1 Score:", f1_bm25)

print("SBERT")
print("Precision:", precision_sbert)
print("Recall:", recall_sbert)
print("F1 Score:", f1_sbert)
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Batches: 100% 1/1 [00:00<00:00, 56.87it/s]
Batches: 100% 623/623 [00:18<00:00, 63.22it/s]
Batches: 100% 1/1 [00:00<00:00, 60.15it/s]
Batches: 100% 623/623 [00:18<00:00, 62.74it/s]
Batches: 100% 1/1 [00:00<00:00, 43.63it/s]
Batches: 100% 623/623 [00:18<00:00, 64.26it/s]
BM25
Precision: 0.5582010582010581
Recall: 0.44722222222222224
F1 Score: 0.42536630036630035
SBERT
Precision: 0.5661375661375662
Recall: 0.4531746031746032
F1 Score: 0.40851648351648345
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Although the scores here look low, I looked up each query and noticed that the searches were more accurate with jewelry and clothes since there was more data related to these items than laptops. There were largely laptop accessories that popped up in the search query hence the low precision. Along with the above reasoning, the inability to lemmatize the textual data may have been a factor.

Comparing the two models, the SBERT Model performed slightly better than the Okapa BM25 based on the metrics, but while viewing the query results for an example (‘Alisha Shorts’), the results look very similar.

```
BM25 Results:
          score                             product_name  \
3      25.745915            Alisha Solid Women's Cycling Shorts
15     25.722912            Alisha Solid Women's Cycling Shorts
13     25.722912            Alisha Solid Women's Cycling Shorts
0      25.722912            Alisha Solid Women's Cycling Shorts
6      25.699954            Alisha Solid Women's Cycling Shorts
9      25.699954            Alisha Solid Women's Cycling Shorts
10422  10.240545             Broche Printed Boy's Sports Shorts
967    10.121007  Mynte Solid Women's Cycling Shorts, Gym Shorts...
965    10.121007  Mynte Solid Women's Cycling Shorts, Gym Shorts...
963    10.121007  Mynte Solid Women's Cycling Shorts, Gym Shorts...

                                  product_category_tree
3      ["Clothing >> Women's Clothing >> Lingerie, Sl...
15     ["Clothing >> Women's Clothing >> Lingerie, Sl...
13     ["Clothing >> Women's Clothing >> Lingerie, Sl...
0      ["Clothing >> Women's Clothing >> Lingerie, Sl...
6      ["Clothing >> Women's Clothing >> Lingerie, Sl...
9      ["Clothing >> Women's Clothing >> Lingerie, Sl...
10422  ["Clothing >> Kids' Clothing >> Boys Wear >> S...
967    ["Clothing >> Women's Clothing >> Sports & Gym...
965    ["Clothing >> Women's Clothing >> Sports & Gym...
963    ["Clothing >> Women's Clothing >> Sports & Gym...
Batches: 100% 1/1 [00:00<00:00, 45.06it/s]
Batches: 100% 623/623 [00:18<00:00, 63.07it/s]
SBERT Results:
          score                          product_name  \
13     0.608075         Alisha Solid Women's Cycling Shorts
15     0.605690         Alisha Solid Women's Cycling Shorts
0      0.598269         Alisha Solid Women's Cycling Shorts
3      0.597688         Alisha Solid Women's Cycling Shorts
9      0.597549         Alisha Solid Women's Cycling Shorts
6      0.587012         Alisha Solid Women's Cycling Shorts
834    0.517165     Alibi Casual Short Sleeve Solid Women's Top
808    0.504952       Alibi Casual Sleeveless Solid Women's Top
15134  0.497913                 UFO Printed Girl's Basic Shorts
8019   0.496960   Amirich Printed Women's Multicolor Basic Shorts

                                  product_category_tree
13     ["Clothing >> Women's Clothing >> Lingerie, Sl...
15     ["Clothing >> Women's Clothing >> Lingerie, Sl...
0      ["Clothing >> Women's Clothing >> Lingerie, Sl...
3      ["Clothing >> Women's Clothing >> Lingerie, Sl...
9      ["Clothing >> Women's Clothing >> Lingerie, Sl...
6      ["Clothing >> Women's Clothing >> Lingerie, Sl...
834    ["Clothing >> Women's Clothing >> Western Wear...
808    ["Clothing >> Women's Clothing >> Western Wear...
15134  ["Clothing >> Kids' Clothing >> Boys Wear >> S...
8019   ["Clothing >> Women's Clothing >> Lingerie, Sl...
```

+ Code + Markdown

And finally, I made a small demo using the streamlit module to implement a search query. I was only able to implement the BM25 model and was unable to implement the SBERT Model due to system constraints. Following are demos with different sets of queries:

Search Retrieval Demo

Enter your search query:

gold ring

Search

Top search results:

	product_name
1,717	Karatcraft Emmima Gold Diamond 18 K Ring
1,351	Malabar Gold and Diamonds HBDAAAABLZLD Gold Emerald 22 K Ring
1,334	Malabar Gold and Diamonds HBDAAAABLZJM Gold Ruby 22 K Ring
9,488	His & Her Gold Diamond 18K Yellow Gold 18 K Ring
13,358	BlueStone Priyala Ring Yellow Gold Diamond, Ruby 18 K Ring
376	Nishtaa Yellow Gold 22 K Ring
1,927	Jpearls Yellow Gold Yellow Gold 18 K Ring
367	Avsar Sachi Gold Diamond 18 K Ring
435	Nishtaa Yellow Gold Cubic Zirconia 22 K Ring
12,989	Fullcutdiamond FCDR3111R White Gold Diamond 18K White Gold 18 K Ring

Search Retrieval Demo

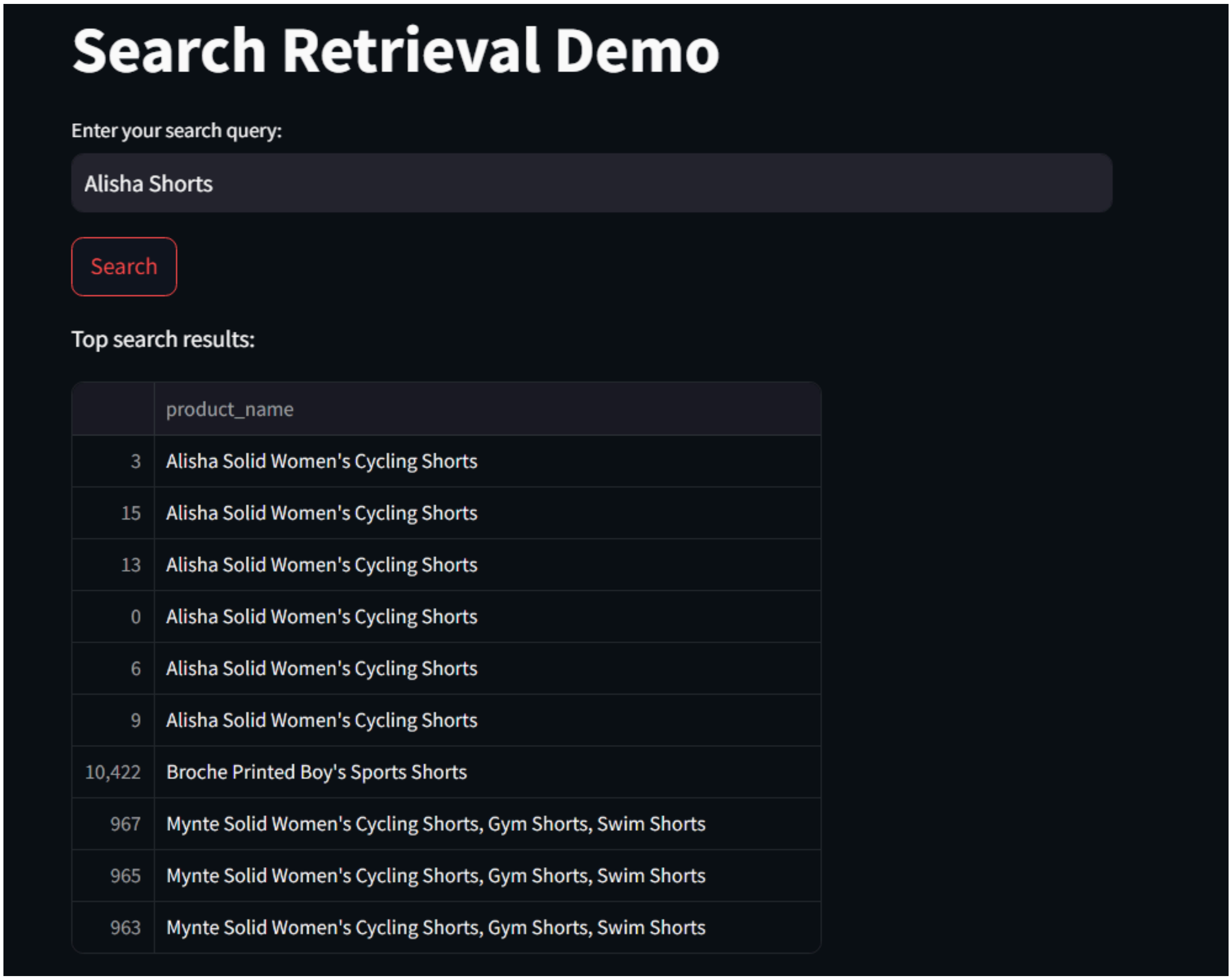
Enter your search query:

laptop

Search

Top search results:

	product_name
19,393	Navigator LpBck 15 L Laptop Backpack
1,947	TRENDIEZ 15.6 inch Laptop Backpack
19,229	Good Win Takssport 20 L Laptop Backpack
19,348	PRINT SHAPES psychic Laptop Skin with Mouse pad Combo Set
19,307	PRINT SHAPES Graphic Dancer Laptop Skin with Mouse pad Combo Set
19,356	PRINT SHAPES Peacock Feather Laptop Skin with Mouse pad Combo Set
19,187	PRINT SHAPES Sony headphone Laptop Skin with Mouse pad Combo Set
19,331	PRINT SHAPES mountain wolf Laptop Skin with Mouse pad Combo Set
19,294	PRINT SHAPES minion superhero Laptop Skin with Mouse pad Combo Set
19,219	PRINT SHAPES think positively Laptop Skin with Mouse pad Combo Set



Conclusion

I successfully implemented a retrieval system using both Okapi’s BM25 model and a SentenceTransformer (SBERT). According to the metric scores, the SBERT model performed slightly better than the BM25 model.

DDespite the encouraging results from both models, there is still room for improvement. With access to a larger dataset and a more powerful processing unit, the system's performance could be enhanced further. These current constraints limited my ability to apply advanced preprocessing techniques like lemmatization, which could potentially improve the accuracy and effectiveness of the retrieval system.