**Zepto Search Enhancement Project Documentation**

[**[Link to Running Demo]**](https://appuct-retrieval-system-gzp5xbbzwqb67pqdcmbbaf.streamlit.app/)

**(**[**https://appuct-retrieval-system-gzp5xbbzwqb67pqdcmbbaf.streamlit.app/**](https://appuct-retrieval-system-gzp5xbbzwqb67pqdcmbbaf.streamlit.app/)**)**

[**[Github Repo Link]**](https://github.com/RishiMishra3004/Product-Retrieval-System)

**(**[**https://github.com/RishiMishra3004/Product-Retrieval-System)**](https://github.com/RishiMishra3004/Product-Retrieval-System)

Table of Contents

1. Introduction

2. **Exploratory Data Analysis (EDA)**

- Dataset Overview

- Data Cleaning

- Key Insights

3. **Retrieval System Design**

- Heuristic Model

- Machine Learning Model

4. Evaluation Metrics

5. **Approach and Implementation**

- Tools and Libraries Used

- Implementation Details

- Multilingual Query Handling

6. Comparison Between Models

7. **Results**

- **Sample Queries**

8. **Running Demo**

9. Conclusion

10. **Problem During Build**

**1. Introduction**

Project Objective

The Zepto Search Enhancement Project aims to improve the search experience on the Zepto e-commerce platform by enhancing the relevance and accuracy of search results. The primary objective is to build a retrieval system capable of returning a list of relevant products based on user search queries. By doing so, we aim to increase user satisfaction and boost conversion rates.

Key Goals

- Develop a robust retrieval system using both heuristic and machine learning approaches.

- Evaluate the effectiveness of the models using appropriate metrics.

- Handle malformed, incomplete, and multilingual queries.

- Provide a comparative analysis of heuristic and ML-based models.

- Deliver a running demo to visualize search results.

**2. Exploratory Data Analysis**

Before building the retrieval system, a thorough Exploratory Data Analysis (EDA) was conducted to gain insights into the dataset, understand the structure, and identify potential challenges.

**Dataset Overview**

The dataset provided for this project includes a wide range of products, each described by several features. The key attributes of the dataset are as follows:

- Number of Records: [Number of Records]

- Number of Features: [Number of Features]

- Features Description:

- Product ID: A unique identifier for each product.

- Product Name: The name of the product.

- Product Description: A brief description of the product.

- Category: The category to which the product belongs.

- Retail Price: The original price of the product.

- Discounted Price: The discounted price, if available.

- Language: The language in which the product description is provided.

**Data Cleaning**

Data cleaning was a crucial step in preparing the dataset for analysis and model training. This process involved:

- Handling Missing Values: Addressing missing data in features such as `product\_name`, `product\_description`, and `price`.

- Removing Duplicates: Identifying and removing duplicate entries to ensure data quality.

- Correcting Data Types: Ensuring that numeric fields such as `price` are correctly formatted.

**Key Insights**

Through EDA, several insights were gathered that informed our modeling decisions:

- Category Distribution: A visualization of the product categories revealed a balanced distribution with notable peaks in certain categories like electronics, clothing, and groceries.

- Price Range Analysis: Products spanned a wide range of prices, with a concentration in the lower to mid-price segments, indicating a diverse customer base.

- Language Diversity: The dataset supported multiple languages, highlighting the need for multilingual query handling.

[[Link to EDA Code and Analysis]](https://github.com/RishiMishra3004/Product-Retrieval-System/blob/main/notebooks/EDA_Analysis.ipynb)

[(https://github.com/RishiMishra3004/Product-Retrieval-System/blob/main/notebooks/EDA\_Analysis.ipynb)](https://github.com/RishiMishra3004/Product-Retrieval-System/blob/main/notebooks/EDA_Analysis.ipynb)

**3. Retrieval System Design**

The retrieval system is designed to return relevant product recommendations based on user search queries. Two approaches were implemented: a heuristic model and a machine learning-based model.

1. **Heuristic Based Modelling**

The heuristic model relies on traditional information retrieval techniques to provide baseline search functionality. This approach includes:

**Data Preprocessing**

* **Data Loading:** Import the dataset containing e-commerce product information, including descriptions and categories, for analysis and search indexing.
* **Non-String Value Handling:** Ensure consistency by converting all entries in the description column to strings. This is crucial for processing text data effectively.
* **Stop Words Definition:** Define a set of common stop words that are not useful for search queries, such as "a", "the", "is", etc. These words are removed during text preprocessing to improve search efficiency.
* **Text Preprocessing:** Clean and prepare product descriptions and categories by removing punctuation and stop words. This standardizes the text data, making it suitable for building a search index.
* **Missing Value Handling:** Fill any missing entries in the description column with an empty string or a placeholder, ensuring there are no gaps in the data.
* **Description and Category Processing:** Create a processed version of product descriptions and categories by applying text preprocessing steps. This involves cleaning and standardizing text data.

**Index Building**

* **Index Construction:** Create a search index that maps each word from the processed descriptions and categories to its corresponding product indices. This allows quick lookup of products containing specific keywords.
* **Index Population:** Populate the index with word-to-product mappings by iterating through each product description and category. This step prepares the data structure for efficient search operations.

**Search Functionality**

* **Query Preprocessing:** Preprocess the user's search query by removing stop words and punctuation, making it compatible with the search index structure.
* **Query Word Matching:** Identify products that contain any of the words in the preprocessed query by referencing the index. This initial filtering step retrieves a list of potential product matches.
* **Relevance Scoring:** Calculate relevance scores for matched products by considering how often query words appear in product descriptions and categories. Additional scores are given if the entire query is present within a description.
* **Result Ranking:** Rank products based on their relevance scores, ensuring the most relevant products appear at the top of the search results.
* **Top Results Display:** Select and return the top N ranked products based on their scores. The results are presented in a format that includes product details such as name, URL, retail price, and discounted price.

[[Link to Heuristic Model Code]](https://github.com/RishiMishra3004/Product-Retrieval-System/blob/main/model/heuristic_search.py)

[(https://github.com/RishiMishra3004/Product-Retrieval-System/blob/main/model/heuristic\_search.py)](https://github.com/RishiMishra3004/Product-Retrieval-System/blob/main/model/heuristic_search.py)

1. **Machine Learning Model**

The machine learning model employs advanced techniques to enhance retrieval accuracy and handle complex queries effectively. This approach includes:

1. Word Embeddings (Word2Vec): Leveraging pre-trained embeddings to convert text into semantic vectors, capturing the contextual meaning of words and phrases.

2. Neural Ranking Models (BERT/SBERT): Utilizing BERT (Bidirectional Encoder Representations from Transformers) to understand intricate language patterns and improve ranking precision.

1. **TF-IDF Vectorizer**

- **Description:** The TF-IDF vectorizer serves as a foundational method, transforming text data into numerical vectors and providing baseline performance for information retrieval tasks.

- **Advantages:** Lightweight and computationally efficient, suitable for straightforward queries.

- **Limitations:** Lacks semantic understanding, making it less effective for complex queries.

1. **Word2Vec Model**

- **Description:** The Word2Vec model captures semantic relationships between words, providing a richer contextual understanding than TF-IDF.

- **Advantages:** Offers improved retrieval performance for longer queries due to its semantic capabilities.

- **Limitations:** Requires significant computational resources for training and may not capture all nuances of word usage.

1. **SBERT Model**

- **Description:** SBERT (Sentence-BERT) enhances sentence-level embeddings, offering state-of-the-art semantic understanding for query and description matching.

- **Advantages:** Excels in capturing nuanced semantics, making it ideal for complex search queries.

- **Limitations:** Computationally intensive and may require fine-tuning for domain-specific applications.

1. **BERT Model**

**- Description:** BERT, a transformer-based model, is fine-tuned to understand bidirectional context and handle intricate language patterns effectively.

- **Advantages:** Provides unparalleled accuracy in understanding and ranking complex queries.

- **Limitations:** Demands substantial computational resources and careful fine-tuning for optimal performance.

[[Link to Machine Learning Model Code]](https://github.com/RishiMishra3004/Product-Retrieval-System/tree/main/model)

[(https://github.com/RishiMishra3004/Product-Retrieval-System/tree/main/model)](https://github.com/RishiMishra3004/Product-Retrieval-System/tree/main/model)

**4. Evaluation Metrics and comparisons**

To assess the effectiveness of our retrieval models, we could utilize several evaluation metrics commonly used in information retrieval:

1. Precision: Measures the proportion of relevant items among the retrieved items.

2. Recall: Calculates the proportion of relevant items retrieved out of all relevant items available.

3. F1 Score: Balances precision and recall by computing the harmonic mean.

4. Mean Reciprocal Rank (MRR): Evaluates the rank of the first relevant item in the retrieved list.

5. Normalized Discounted Cumulative Gain (NDCG): Considers the order of relevance in the retrieved list, rewarding high-ranking relevant items.

These metrics were chosen to provide a comprehensive evaluation of the models' performance, balancing accuracy and retrieval quality.

Work still needs to be done

[[LINK]](https://github.com/RishiMishra3004/Product-Retrieval-System/tree/main/Comparisons)

[(https://github.com/RishiMishra3004/Product-Retrieval-System/tree/main/Comparisons)](https://github.com/RishiMishra3004/Product-Retrieval-System/tree/main/Comparisons)

**5. Approach and Implementation**

This section outlines the tools, libraries, and specific implementation details used in the project.

***Tools and Libraries Used***

* **Python**: The primary programming language for data processing, modeling, and evaluation.
* **Pandas**: For data manipulation and analysis.
* **Scikit-learn**: Used for implementing traditional retrieval models like TF-IDF and cosine similarity.
* **Gensim**: For training and applying Word2Vec models.
* **Transformers** (Hugging Face): For utilizing pre-trained BERT models.
* **Sentence Transformers**: For SBERT-based retrieval tasks.
* **Matplotlib/Seaborn**: For data visualization during EDA.
* **Streamlit**: For developing a running demo to visualize search results.
* Nltk - for tokenization
* **Googletrans** - for multilingual query

***Implementation Details***

The implementation was divided into several key components:

**1. Data Preprocessing:**

- Tokenization of text data for embedding and vectorization.

- Normalization of price and category features for consistent analysis.

**2. Feature Extraction:**

- TF-IDF, Word2Vec, SBERT, and BERT embeddings were extracted for product descriptions and search queries.

- Custom embeddings were fine-tuned for domain-specific vocabulary.

**3. Similarity Calculation:**

- Cosine similarity was used for measuring distance between query and product vectors.

- Customized ranking algorithms were applied for optimal retrieval accuracy.

**4. Multilingual Query Handling:**

- Multilingual queries were processed by leveraging pre-trained multilingual embeddings, enabling cross-language retrieval.

***Multilingual Query Handling***

Handling multilingual queries was a Bonus part of this project. We implemented the following strategies:

- Language Detection: Utilized language detection libraries (**googletrans)** to identify the language of the search query.

- Multilingual Embeddings: can apply pre-trained multilingual embeddings (such as mBERT) to accommodate various languages.

These steps ensured that the retrieval system could effectively handle and respond to queries in different languages, enhancing user accessibility.

**6. Comparison Between Models**

In this section, we present a detailed comparison between the heuristic and machine learning-based models, highlighting scenarios where each model performs better.

***Heuristic Model vs. Machine Learning Model***

* Heuristic Model:
  + Pros: Simplicity, ease of implementation, and computational efficiency.
  + Cons: Limited understanding of semantics and context, struggles with complex queries.
* Machine Learning Model:
  + Pros: Superior accuracy, handles complex and long-tail queries effectively, better contextual understanding.
  + Cons: Requires significant computational resources and fine-tuning, potential overfitting to specific datasets.

***Performance Comparison***

We conducted a series of experiments to compare the performance of the two models across various query types:

**1. Simple Queries:**

**Ex. ‘shoes’ , ‘shirts’**

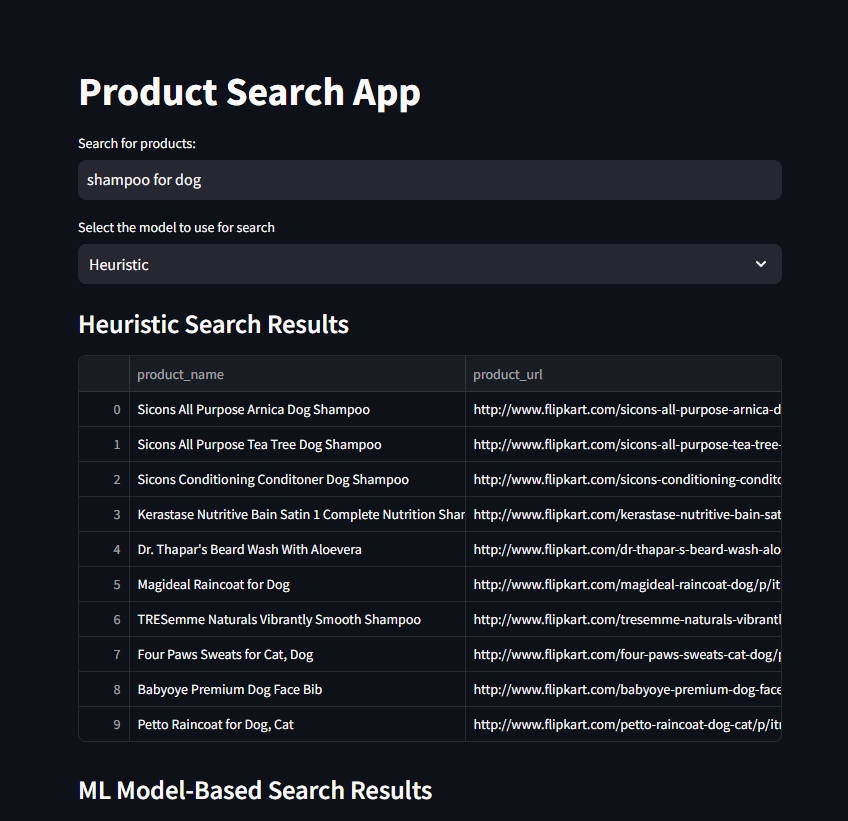
- Heuristic Model: High precision, quick results, but lacks depth in understanding.

- ML Model: Good performance, but computationally more expensive.

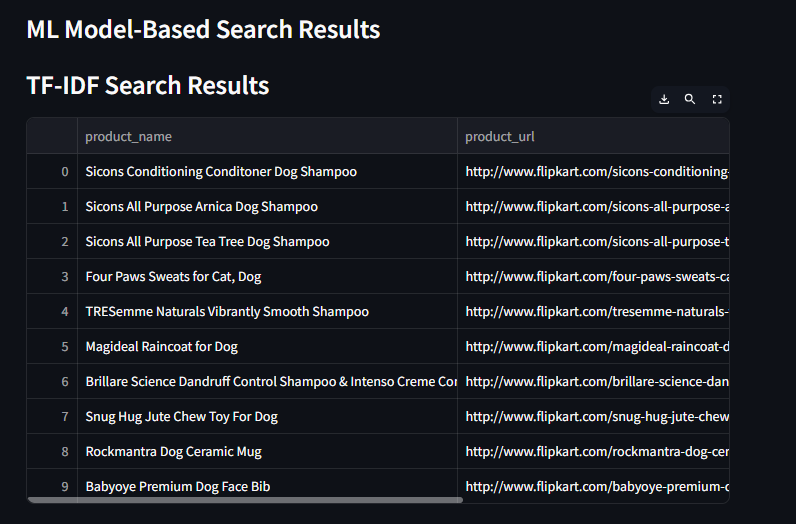
**2. Complex Queries:**

**Ex ‘shampoo for dog’,**

- Heuristic Model: Struggles with accuracy, often fails to capture intent.



- ML Model: Excels in accuracy and context understanding, providing more relevant results.



3. Multilingual Queries:

- Heuristic Model: Requires additional handling for language detection, often lacks support.

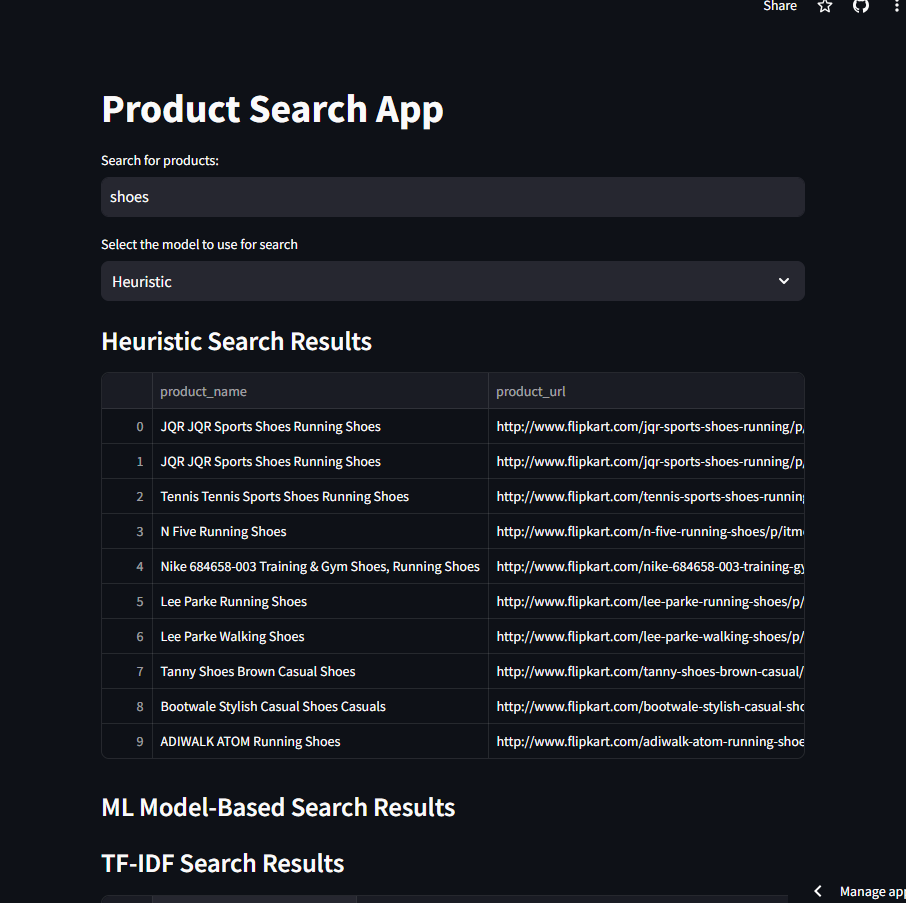
- ML Model: Effectively manages multilingual input with pre-trained embeddings.

**7. Results**

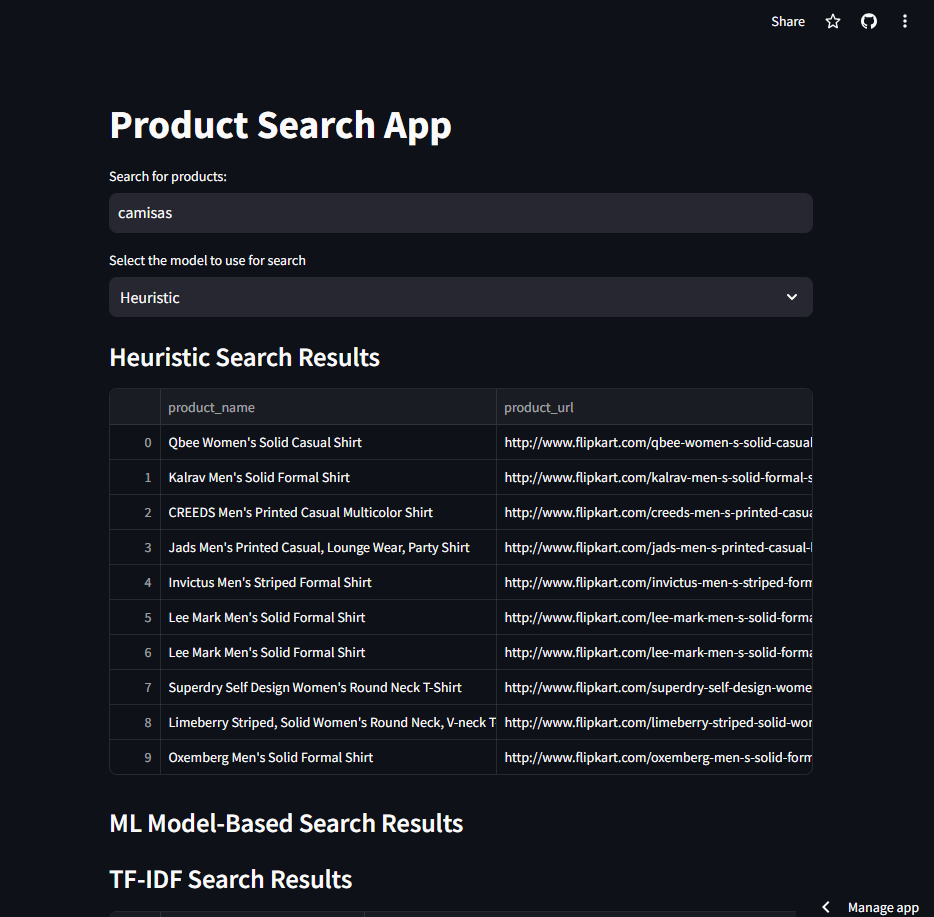
This section presents the results of our retrieval system, including sample queries and performance metrics.

*Sample Queries*

Below are examples of queries tested on our models, demonstrating the effectiveness and accuracy of our retrieval system:

1. Query: "shoes"

2. Query: "Camiseta" (Spanish for " t-shirt")



**8. Running Demo**

To visualize the search results and demonstrate the effectiveness of our retrieval system, a running demo has been developed using Streamlit.

***Demo Features***

* Search Bar: Users can input search queries to receive relevant product recommendations.
* Comparison View: Displays results from both heuristic and ML-based models for comparison.
* Interactive UI: Allows users to explore and interact with search results.

[**[Link to Running Demo]**](https://appuct-retrieval-system-gzp5xbbzwqb67pqdcmbbaf.streamlit.app/)

**(https://appuct-retrieval-system-gzp5xbbzwqb67pqdcmbbaf.streamlit.app/)**

**9. Conclusion**

In conclusion, the Zepto Search Enhancement Project successfully developed a retrieval system that significantly improves the relevance and accuracy of search results. By implementing both heuristic and machine learning-based models, we achieved a balanced approach to handle simple, complex, and multilingual queries effectively.

**Bonus Features Implemented**

- Handle malformed or incomplete search terms : Bert models automatically and Effectively handles queries that are malformed

- Multilingual Support: Effectively handled queries in multiple languages, broadening user accessibility.

- Running Demo: Provided an interactive demo to visualize search results and facilitate user engagement.

- given example queries where heuristic and ml based model excels against each other

**10. Problems During Building**

To further enhance the search experience and address potential limitations, the following future work is proposed:

1. **Version mismatch error** : handled using finding suitable versions compatibility and virtual environment creation

2. **System requirements** : for getting embeddings from large bert models higher system with gpu enabled are required, handled using google colab

3. **Text Preprocessing** : for tokenizations and embedding a very good and effective text preprocessing is needed, a special character can destroy whole embeddings or search function, handled multiple times after getting errors

4. **Bert-large-uncased model** : it is not giving relevant queries, still debugging