Astrid Alins, Miquel Bisbe, Cinta Carot

Information Retrieval and Web Analytics

The aim of this project is to build an effective search and retrieval system for an e-commerce fashion products dataset. The project is structured into **four main parts**, each addressing a key stage of data preparation, exploration, and retrieval evaluation.

PART 1: Data preparation

1. As a first step, you must pre-process the documents. In particular, for the text fields (title, description) you should: Removing stop words, Tokenization, Removing punctuation marks, Stemming and... anything else you think it's needed (bonus point)

The first thing that we did was create a google collaboratory document that later would be sent to our github repository.

At the document, we began with the uploading of the json document provided. For the preprocessing of the textual fields (title and description), we based our approach on the *build_terms(line)* function from Practice 1.

The pipeline of our 'clean_line(line)' function performs the following steps:

- Lowercasing: Converts all text to lowercase to avoid mismatches due to capitalization.
- Punctuation and Number Removal: Removes punctuation, numbers, and special characters, producing a clean and uniform text.
- Whitespace Normalization: Replaces multiple spaces with a single space
- Tokenization: Splits the cleaned text into individual tokens using a split() method.
- Stop Words Removal: Removes common English stop words using NLTK's standard list
- Stemming: Applies the *Porter Stemmer* to reduce each token to its root form

This function allows us to generate cleaned token lists (title_clean, description_clean, product_details_clean) while keeping all original metadata intact (pid, brand, category, seller, prices, etc.) for future queries and result presentation.

2. Take into account that for future queries, the final output must return (when present) the following information for each of the selected documents: pid, title, description, brand, category, sub_category, product_details, seller, out_of_stock, selling_price, discount, actual_price, average_rating, url

Before querying the dataset, a preprocessing function was implemented to ensure that all relevant product information was properly cleaned and structured.
The function preprocess_document(doc) takes each document (product entry) and returns a standardized dictionary containing the required fields:

pid, title, description, brand, category, sub_category, product_details, seller, out_of_stock, selling_price, discount, actual_price, average_rating, url.

Textual fields such as *title* and *description* were cleaned using the clean_line() function to remove unnecessary symbols, spaces, and formatting inconsistencies.
All other fields were preserved as provided in the original data to maintain their integrity for later analysis.
Finally, a sample of five preprocessed documents was printed to verify that the output structure matched the expected format.

3. Decide how to handle the fields category, sub_category, brand, product_details, and seller during pre-processing. Should they be merged into a single text field, indexed as separate fields in the inverted index or any other alternative? Justify your choice, considering how their distinctiveness may affect retrieval effectiveness. What are the pros and cons of each approach?

Merging into a Single Text Field: Concatenate the values of these fields into the main text content of the document (e.g., add them to the description or create a new all_text field).

- Pros:
    - Simplifies the indexing process, as you only need one inverted index for the entire text.
    - Allows for keyword searches across all fields simultaneously.
- Cons:
    - Loses the distinctiveness of each field. A search for "Nike" would match "Nike" in the brand, but also if it appears in a product detail or description, which might not be the desired result.
    - Difficult to perform field-specific searches (e.g., "only show me products where the brand is Nike").
    - Might introduce noise into the index, making it harder to retrieve relevant documents based on specific attributes.

Indexing as Separate Fields (Faceted Search): Create separate indices or designate specific fields in your index for each of these fields (category_index, brand_index, etc.). This is often used in conjunction with the main text index.

- Pros:
    - Preserves the distinctiveness of each field, allowing for precise field-specific searches (e.g., brand:"Nike").
    - Enables faceted search, where users can filter results based on categories, brands, sellers, etc., which greatly enhances user experience and retrieval effectiveness in e-commerce.
    - Reduces noise in the main text index.
- Cons:
    - More complex indexing and querying process compared to a single text field.
    - Requires more storage space for multiple indices.

Combining with Textual Analysis (Hybrid Approach): Index these fields separately for faceted search and field-specific queries, while also potentially including them in the main text field (or a separate "searchable text" field) after some processing. For product_details, you might parse the key-value pairs and index both the keys and values, possibly with weighting.

- Pros:
    - Leverages the benefits of both approaches: enables faceted search and precise field-specific queries while still allowing general keyword search that can hit terms in these fields.
    - Provides more flexibility in querying.
- Cons:
    - Most complex to implement and manage.
    - Requires careful consideration of how to weight terms from different fields in the main text index.

We decided to index **brand**, **category**, **sub_category**, and **seller** as separate fields, since they contain structured and highly distinctive categorical information. In contrast, **product_details** will be merged with the main textual fields (**title** and **description**), as it provides descriptive terms that are useful for keyword-based retrieval.

This hybrid approach allows better control over field weighting and improves retrieval effectiveness by combining precise filtering (on categorical fields) with richer semantic matching (on textual content). The main drawback is a slightly higher implementation complexity, but the gain in ranking accuracy and interpretability justifies this choice.

4. <u>Consider the fields out_of_stock, selling_price, discount, actual_price, and average_rating. Decide how these should be handled during pre-processing to use in further search. Should they be indexed as textual terms?</u>

Indexing these fields as textual terms would not be the most effective approach. This is because:

- **Numerical/Boolean Nature**: These fields are primarily numerical (selling_price, discount, actual_price, average_rating) or boolean (out_of_stock). Textual indexing is designed for free text search, not for numerical comparisons or filtering based on true/false values.
- **Loss of Structure and Meaning**: Treating these as text would lose their inherent structure and meaning. You wouldn't be able to easily perform range queries (e.g., products under $50), sort by price, or filter by availability.
- **Ineffective for Filtering/Sorting**: Users typically want to filter search results based on price range, availability, or sort by rating. Textual indexing doesn't support these operations efficiently or accurately.

To be handled, these fields should be indexed as their native data types and used for filtering, sorting, and potentially range queries:

- **<u>out_of_stock</u>**: This is a boolean field. It should be indexed as a boolean (True/False). This allows users to easily filter for items that are in stock.

- **selling_price**, **actual_price**: These are numerical values. They should be converted to a numerical data type (e.g., float or integer) and indexed as such. This enables:
    - Range Queries: Finding products within a specific price range (e.g., $10 - $50).
    - Sorting: Sorting results by price (lowest to highest or highest to lowest).
    - Filtering: Filtering by maximum or minimum price. Note: You might need to clean the price strings (remove currency symbols, commas) before converting them to numbers.
- **discount**: This is likely a percentage string (e.g., "69% off"). You could extract the numerical percentage and index it as a number to enable sorting by discount amount or filtering for products with a minimum discount.
- **average_rating**: This is a numerical value (often a float). It should be indexed as a number to allow:
    - Filtering: Showing products above a certain rating (e.g., 4.0 and above).
    - Sorting: Sorting results by average rating (highest to lowest).

In summary, these fields are best handled as structured data (boolean or numerical) for effective filtering, sorting, and range queries, rather than being indexed as unstructured text. This aligns with how users typically interact with these types of attributes on e-commerce websites.

PART 2: Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA)

The dataset contains information on more than 28,000 clothing products listed online. The exploratory data analysis (EDA) was conducted to understand the distribution of prices, ratings, text characteristics, and brand/seller dominance.

1. Text and Vocabulary Analysis

Product titles contain between 3 and 15 words, which is typical for short, descriptive names. Descriptions, on the other hand, vary widely — from fewer than 10 to more than 100 tokens. This exponential decay in length indicates heterogeneity in how sellers describe their products: some provide only basic details, while others include extensive information.

The dataset contains **4,536 unique words**, with an **average of 15.3 unique tokens per product description**.
The plot showing the most frequent words reveals that terms such as *shirt*, *round*, *neck*, *solid*, *women*, and *cotton dominate*, reflecting the focus on casual apparel. Words like *fit*, *casual*, *made*, and *comfort* emphasize the importance of style and usability in product listings. About **8.9% of descriptions mention a color**, suggesting that while color is relevant, it is not systematically included across listings.

2. Price and Discount Distributions

The plot of the **selling price distribution** shows that most products are priced between ₹0 and ₹1,000, roughly following a normal distribution centered around ₹500, with a long right tail representing a few premium products.
Similarly, the **actual price distribution** peaks between ₹0 and ₹2,000, again showing a right-skewed shape.

Discount rates are mainly concentrated between **40% and 60%**, with common peaks at **30%, 50%, and 60%**, which correspond to typical marketing thresholds used by online retailers. Overall, this suggests a **mid-range pricing strategy with significant discounting**, consistent with the competitive nature of large e-commerce platforms.

3. Product Ratings

The **average customer rating** is **3.63 out of 5**, with most products rated between 3 and 4 stars.
The plot of top-rated products indicates that **T-shirts and casual wear** dominate this group, suggesting that simple, familiar items with clear descriptions tend to receive better feedback. This could be because these products involve fewer uncertainties regarding fit or quality compared to more specialized garments.

4. Brands and Sellers

Plots showing brand and seller frequencies reveal a clear concentration pattern: a few brands — such as **ECKO Unltd., Free Authority, and ARBO** — dominate the dataset, each offering hundreds of products.Similarly, sellers like **RetailNet** and **SandSMarketing** stand out with over a thousand listings each.This indicates that the marketplace is shaped by a **small number of major contributors** and a **long tail of smaller sellers**, a structure typical of online retail ecosystems.

5. Stock Availability

The distribution plot for the *out_of_stock* field shows that **most products are in stock**, confirming that the dataset primarily represents **active listings** rather than discontinued or archived items.
This ensures that the analysis reflects the **current market offering**, not outdated or inactive products.

6. Summary Statistics

Variable	Mean	Std	Min	25%	50%	75%	Max
Selling Price	₹705.6	549.7	99	390	545	820	7,999
Actual Price	₹1,455.5	939.9	150	849	1,199	1,799	12,999

Discount (%)	50.3	16.9	1	40	53	63	87
Average Rating	3.63	0.66	1	3.2	3.8	4.1	5

These results confirm that most products are **low-cost, highly discounted, and moderately rated**, with a few premium items stretching the price distribution’s right tail.