IRWA Project Part 1: Preprocessing & EDA Report

Authors: Lucas Andreu, Pau Chaves, Pol Bonet, Joan Company

Course: Information Retrieval and Web Analytics (IRWA)

Date: October 24, 2025

1. Project Summary

This report describes the preprocessing steps and exploratory data analysis (EDA) done on a dataset of 28,080 fashion products. The main goal was to prepare the data for a later information retrieval system and to find useful insights that could help design the search engine, especially for feature engineering and ranking methods.

2. Introduction

2.1 Project Context

Information retrieval systems need careful preprocessing of textual data in order to enable effective search and ranking. This project implements a complete pipeline for processing fashion e-commerce product data, preparing it for subsequent ranking and retrieval tasks in Parts 2 and 3.

2.2 Objectives

1. **Text Preprocessing:** Clean and normalize product titles and descriptions.
2. **Feature Extraction:** Parse and structure non-text fields (prices, ratings, categories).
3. **Exploratory Analysis:** Understand dataset characteristics and distributions.
4. **Corpus Preparation:** Create a searchable index-ready dataset.

3. Dataset Overview

3.1 Data Source

The dataset contains fashion product information scraped from e-commerce platforms, including: product titles and descriptions, pricing information (actual price, selling price, discounts), ratings and reviews metadata, categorical information (brand, category, sub-category), product details and specifications and seller information.

3.2 Data Structure

Input file: `fashion_products_dataset.json`
Format: JSON array of product objects
Key fields:

- `title`: Product name/title
- `pid`: Product identifier
- `description`: Detailed product description
- `brand`, `category`, `sub_category`: Categorical attributes
- `actual_price`, `selling_price`, `discount`: Pricing information
- `average_rating`: Customer rating
- `product_details`: Structured product specifications
- `url`: Product page link

4. Data Preprocessing & Feature Engineering

A multi-step pipeline was developed to handle both text and structured data, producing a clean and enriched dataset suitable for indexing.

4.1 Text Processing Pipeline

A standard IR preprocessing pipeline was implemented using NLTK due to its robustness and modularity. The chosen sequence prioritizes efficiency and recall.

Pipeline: `Normalization` → `Tokenization` → `Stopword Removal` → `Stemming`

- **Normalization (`normalize_basic`):** An aggressive cleaning strategy was employed to reduce vocabulary size. This included lowercasing, Unicode accent folding (`unidecode`), and removal of URLs, currency symbols, and all numeric digits. Punctuation and special characters were replaced with spaces to preserve token boundaries (e.g., "red-shirt" → "red shirt").
- **Tokenization (`word_tokenize`):** NLTK's `word_tokenize` was selected over a simple `split()` for its superior handling of linguistic structures like contractions.
- **Stopword Removal:** Standard English stopwords from NLTK were removed to reduce index size and filter out low-signal terms.
- **Stemming (`PorterStemmer`):** The Porter stemmer was chosen for its balance of speed and aggressive stemming, which is effective for improving recall in IR systems. Lemmatization was considered but deemed unnecessarily complex and computationally expensive for this stage.

4.2 Handling of Metadata Fields

For fields like `category`, `sub_category`, `brand`, `product_details`, and `seller`, a hybrid approach was chosen to balance recall and precision.

Chosen Strategy: Hybrid Approach

1. **Index as Separate Fields:** Each field is normalized (e.g., `brand_norm`) and kept separate. This is essential for **faceted search**, allowing users to filter results precisely (e.g., show only `brand: "Nike"`).
2. **Merge into a Single Text Field:** All these fields are also combined into the `metadata_clean` text field, which is then tokenized and stemmed.

Justification: This approach provides the best of both worlds. The merged field maximizes **recall** for broad user queries (e.g., a search for "Nike running shoes" can match "Nike" from the brand field and "running shoes" from the sub-category). The separate, normalized fields provide the **precision** needed for filtering and faceting, which is a critical feature in e-commerce search.

Alternative 1: Merged Field Only

- **Pros:** Simple to implement, great for recall.
- **Cons:** Loses field-specific context, making faceted search impossible and preventing field-weighted scoring (e.g., boosting matches in `brand` over `seller`).

Alternative 2: Separate Fields Only

- **Pros:** High precision, enables powerful field-specific queries and weighting.
- **Cons:** Lower recall for simple, unstructured queries. A search for "Nike shoes" would require a complex query (`(title:"Nike shoes" OR (brand:"Nike" AND category:"shoes"))`), which is less user-friendly.

4.3 Handling of Numeric and Boolean Fields

Fields like `out_of_stock`, `selling_price`, `discount`, `average_rating` should **not** be indexed as textual terms. In- dexing them as text would treat their values as simple strings, losing their inherent numeric and ordinal properties. For example, the text tokens “100” and “500” have no mathematical relationship, making range queries (`price < 500`) and sorting impossible.

Chosen Strategy: Parse as Native Types

- These fields were parsed into their native data types (float, integer, boolean).
- In a real search system, these values would be stored in a **forward index** (or a columnar store), directly associated with each document ID.

Justification: This approach enables critical search functionalities:

- **Range Filtering:** Efficiently filter products within a specific price range (e.g., `selling_price` between 100 and 500).
- **Sorting:** Sort results by `average_rating` (descending) or `selling_price` (ascending).
- **Ranking Function:** Use these numeric values directly as features in the ranking model. For example, the score can be boosted based on a high `average_rating` or `discount_pct`.
- **Boolean Filtering:** Efficiently include or exclude items based on flags like `out_of_stock`.

5. Exploratory Data Analysis (EDA) & Key Insights

The EDA focused on understanding data distributions and identifying characteristics relevant to search system design. All generated plots and data tables are located in `data/eda_outputs/`.

5.1 Corpus Statistics

Metric	Value	Insight
Total Products	28,080	A substantial corpus for analysis.
Vocabulary Size	6,650	A manageable number of unique terms after preprocessing.
Median Title Length	6 words	Titles are concise; search must leverage other fields.
Median Desc. Length	9 words	Descriptions are also short, reinforcing the need for multi-field search.
Median Price	545	The catalog is dominated by budget-to-mid-range items.
Median Rating	3.8 / 5	Most products are reasonably well-rated.
Median Discount	53%	The market is highly promotional; discount is a key feature.
Out of Stock Ratio	5.85%	High inventory availability; OOS items are a minority.

5.2 Distribution Analysis & Search Implications

- **Price Distribution (`hist_prices.png`):** The distribution is heavily right-skewed, with most items concentrated below 1500. This validates the use of price buckets (`low`, `mid`, `high`) for faceted search and suggests that price is a strong signal for user intent.

- **Discount Distribution (`hist_discounts.png`):** A significant peak between 40-70% indicates that high dis- counts are standard. This makes `discount_pct` a critical feature for a “best deals” ranking strategy.
- **Rating Distribution (`hist_ratings.png`):** Ratings are clustered at 3.5-4.5 stars. The relative scarcity of low- rated items suggests survivorship bias. Since 8% of items have no rating, using it as a hard filter is inadvisable; it is better suited as a boosting signal in the ranking function.
- **Text Length (`hist_title_word_count.png`):** The conciseness of titles and descriptions implies that a search system cannot rely on these fields alone. The `metadata_clean` field is crucial for capturing user queries related to brand, color, or material.

5.3 Data Quality & Missingness

- **Critical Fields:** Price data is 99.99% complete, making it a reliable feature.
- **Sparse Fields:** `average_rating` (8.1% missing) and `brand` (7.2% missing) require robust handling. The system should not fail if a filter is applied to a field with missing data.

6. Conclusions & Next Steps

The preprocessing pipeline successfully transformed the raw dataset into a structured and searchable corpus. The EDA revealed key characteristics of the data that directly inform the design of the ranking and retrieval models for Part 2.

Key Takeaways for System Design:

1. **Multi-field search is mandatory** due to the conciseness of individual text fields. A weighted combination of `title_clean`, `description_clean`, and `metadata_clean` is recommended.
2. **Ranking signals should include** both text relevance (e.g., BM25) and business metrics. The EDA identifies `discount_pct`, `average_rating_num`, and `selling_price_num` as high-value features.
3. **Faceted search** on `brand_norm`, `category_norm`, and the engineered `price_bucket` and `rating_bucket` will provide a powerful user experience.

Appendix: Generated Files

- **Enriched Dataset:** `data/fashion_products_dataset_enriched.json`
- **EDA Outputs:** All 11 visualizations, 3 data tables, and 2 summary files are located in `data/eda_outputs/`.