Project Part 1 Report

Information Retrieval And Web Analysis 2025

Date: 25th October 2025
Group Member 1: Marc Bosch Manzano u215231
Group Member 2: Chris Matienzo Chilo u198726
Group Member 3: Àlex Roger Moya u199765

GitHub repository link

The GitHub repository with the code of the labs is the following: https://github.com/u215231/Information-Retrieval-and-Web-Analytics-Project.git

Here, you have to navigate to the folder where it is the Part 1 solution: `IRWA-2025-part-1`. The Python notebook with the solution's code is called `IRWA-2025-part-1-final-solution.ipynb`.

Introductory Section

We have started our code by initializing our dataframe df_fashion_products that contains our Fashion Products Dataset. In all modifications to this dataframe we have adopted the convention to append a suffix _XX, where XX is the modification number.

Furthermore, in this previous section, we have displayed the column names, the null count values, and data types of our fashion products dataset at the beginning of the notebook. We have also displayed the first rows and the number of rows and columns of the dataset to see in order to see how the dataset is structured.

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)**

Here, we have used defined a function to process our text data named `process_text`. This function is a slightly modified version of our function `build_terms` from our previous labs.

Our modifications to this function consisted on the following. First of all, we have made constant the `stemmer` object and `stop_words` set. We have included to the stop words "nan" and empty string "", since they are words the text process can include wrongly to the final string.

Another extra process we had done was removing all numbers (0, 1, ...) and special characters (%, @, ...) from the string line we are processing. Those characters can create single numbers as strings, or strings with dots or parenthesis. These wrong strings have to be omitted because they are not useful for index. Another reason to suppress them is because the user queries do not usually include numbers, let alone special characters. Numbers can be important to distinguish one product from another by model, but in this case we have omitted them too.

The only special character we have kept has been the hyphen (-), as it appears in the middle of some English words; without it, the meanings of these words would be altered.

Finally, another extra process has been removing repeated words for each fashion product. We achieve this by converting our text tokens to a set and reconverting them into a list. Since we are joining title and text section, repeated strings should introduce less efficiency to our index computation.

At the end of this section, we have processed `title` and `description` separately. Later, it will be joined in the section 3 of this part.

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.

We have created another dataframe version with these indicated columns.

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 pros and cons of each approach?

We have kept those fields separated and as well as an auxiliary field `document`. This field concatenates the proposed variables `category`, `sub_category`, `brand`, `product_details`, and `seller`. It also appends the `title` and `description` to improve query results, too. The reason why we have done this hybrid process is because all these five fields can be seen as categorical variables and as text variables.

Following this system, we have these advantages. In the one hand, keeping the fields separately allows having an organized view of categorical variables and to make queries based on each field. On the other hand, combining the fields into a single field can give equal weight to all the words and make possible indexing by word tokens.

On `product_details` variable, we had to do an extra process, since this variable contains a list of dictionaries for each instance. In the code, we have joined all of these dictionaries in a single one merged dictionary for each fashion product using the function `flatten_details`. We have created new columns for each key of those dictionaries, which are columns such as `Style Code`, `Closure`, `Color`, `Fabric`, etc. We have append the prefix `product_details_` to each of these columns.

Not all the rows have the same attributes in product details. Then, many rows will have not a number (NaN) values for attributes they do not have. In order to minimize the number of NaNs, we have selected the most significant details, those with less NaNs. The variable `top_product_details` manages how many features are selected

At the end, we have merged all these columns to a single string into a new column named `merged_product_details`, and we have dropped the original `product_details` column.

In short, we have split `product_details` into several columns by categories, and merged them into a text to a new column named `merged_product_details`.

Finally, we have merged the text of title, description, category, sub_category, brand, seller, and merged_product_details into a new column called `document`. Also, we have applied the `process_text` function to each row of `document`.

As a further analysis, we will index the `document` terms, and we will keep category, sub_category, brand, seller, and all the categories of product_details to another database to do queries that require more structure.

4. 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?

We do not index numeric fields as text. These five fields do not provide semantic information about the product, but rather quantitative information, useful for filtering or sorting results. During pre-processing, they are converted to numeric or boolean format, but are not included in the `document` field or the inverted index.

Part 2: Exploratory Data Analysis (EDA)

When working with data, it is important to have a better understanding of the content and some statistics. Provide an exploratory data analysis to describe the dataset you are working on in this project and explain the decisions made for the analysis. For example, word counting distribution; average sentence length; vocabulary size; ranking of products based on rating, price, or discount; top sellers and brands; out_of_stock distribution; word clouds for the most frequent words, and entity recognition.

Feel free to do the exploratory analysis and report your findings in the report.

We have centered our EDA on the following aspects:

- Numerical variables distribution

Here, we have analyzed how numerical variables are distributed. We have seen there are four numerical variables on the dataset: `selling_price`, `discount`, `actual_price`, and `average_rating`. All of them have different supports: selling_price and actual_price are in monetary units and range from hundreds to thousands, discount is in monetary units and ranges from zero to a hundred units, and average_rating is a score which ranges from one to five.

We have plotted the boxplots and histogram for each variable using our defined function `plot_numerical_variables_analysis`. In the one hand, the selling_price and actual_price have distributions which all the values are mainly in the left part of the graphics, like following a power law. On the other hand, the discount and average_rating follow a distribution somewhat similar to a Gaussian distribution, with medians of 53 and 3.8 respectively.

We have analyzed the correlation between variables, too. We have observed a strong positive correlation between actual_price and selling_price, a weak positive correlation between discount and actual price, and a negative correlation between selling_price and discount. The average rating is practically uncorrelated with the other variables. We have also plotted the plots between pairs of variables to graphically observe those correlations.

- Word counting distribution

Here, we have analyzed the word count of the fields `title`, `description`, and `document`.

In key categories such as title and description, observing their medians, most part of the records contain fewer than 8 words, respectively. This indicates that texts are concise and focused on describing the product. Nevertheless, there is a long tail in the distribution, with cases between 50–80 words and a few outliers exceeding 200 words in the case of description. In the document word count, the number of median words is greater, which value is 27 words, and the maximum number of words is no more than 225 words.

We can see the distribution of word count for title is centered on a median of 6 words. The description variable follows a power law, with the majority of products with descriptions of less than 10 unique words. The document word count also follows a power law but centered on a higher value of 20 or 30 words.

- Rankings

Here, we have analyzed `average_rating` and `selling_price` variables. We have defined a function called `ranking` which returns a dataframe with the top N ranked products.

In average_rating, most ratings are high and show some variability for both male and female items. In selling_price, the cheapest products (near 110 currency units) and those with the largest discounts (near 65%) are mainly targeted at a female audience, with brands such as Ina Gro standing out.

- Top brands, sellers, and words

In this section, we have defined the following three functions. The first, `get_top_values`, which produces a series with the top value conts for the values in any series. The second, `get_top_words`, which is focused on textual features, it gives the counts for all words in the column. Finally, `plot_top`, plots the top of any series of values with counts.

In our top-values plots, we have seen there are a dominant groups, both among `brand` and `seller` fields, that capture a large share of the market. The most popular sellers are RetailNet and SandSmarketing, and the most popular brands, ECKO Unl and Free Authori. This suggests a concentrated supply structure, where a few parties account for most of the products.

In our top-word plots, we have plotted the most frequent words for `document` field. These top words are those we can see as a word cloud arrangement on the next section.

- Word cloud visualization

Here, we have created a function to visualize our top words, `plot_word_cloud`, using WordCloud library.

It shows wide lexical variability, with terms related to our field we are analyzing, fashion products. The most popular terms are cloth, cotton, wear, wash, and accessory. This terminological richness is especially useful for associating concepts and establishing semantic hierarchies in future searches.

Summary of the Exploration Data Analysis

In summary, the EDA results show a structure consistent with the data processing carried out in the previous section. We have seen the texts are short and explanatory, and there are terms more frequent than others. We have also get well-defined and diverse categories that will allow us to relate prices, discounts, and brands to enable a subsequent analysis for search and recommendation tasks.

Concluding section

In this Part 1 of our Final project, we have understood how to process and clean the data of the Fashion Products Dataset. This will facilitate a subsequent process of indexing terms in the dataset. Moreover, in our EDA we have observed graphically what is the behaviour that follows the different features of our data, focusing our analysis on the words in the title, descriptions, and details of each product.