Universitat Pompeu Fabra Barcelona

Pau Reig Vaqué - u215101
Marc Rodríguez Vitolo - u215105
Clara Serra Borràs - u204635

REPORT
Part 2: Indexing and Evaluation

Part 1: Indexing

To start the project, we have copied part 1 of the project in order to reuse some information that we have already completed. To build the inverted index, we have started from the cleaned dataset stored in the variable df. This dataframe contains both the original product information and the new transformed features created during the preprocessing stage in part 1.

For indexing purposes, we have used the column cleaned_title_description_extra_fields, which contains a list of cleaned and stemmed tokens derived from several textual attributes, specifically those that we have considered of much importance in part 1. We have made this design choice to create a comprehensive text representation for each product to ensure that the indexing and subsequent retrieval processes could capture a broader range of textual cues about each product.

The inverted index was implemented using a dictionary structure where each term is associated with the list of product identifiers in which it appears.

One assumption we have made during this stage was that each token in the cleaned_title_description_extra_fieldscleaned_title_description_extra_fields column is equally important for indexing.

Finally, we have verified the correctness of the inverted index by inspecting random terms and confirming that their document lists matched the expected results based on manual checks.

After building the inverted index, we have implemented a ranking mechanism, like in practice 2, based on the TF-IDF weighting scheme to improve retrieval quality and provide more relevant results. The create_index_tfidf() function extends the basic index by computing the term frequency, document frequency, and the inverse document frequency values for each token.

For each document, we have calculated normalized term frequencies, dividing each term's raw frequency by the document norm to ensure that longer documents do not unfairly dominate the ranking due to a higher number of tokens. The IDF value for each term was then computed as the logarithm of the ratio between the total number of documents and the number of documents containing the term, highlighting rare and distinctive words.

Using these, the rank_documents() function computes the TF-IDF vector representation for each candidate document, all the documents that contain all query terms, and for the query itself. We then calculate the cosine similarity between the query vector and each document vector, ranking documents in descending order of their similarity scores. This approach rewards documents that contain the query terms multiple times and penalizes common terms that appear in many documents.

Pau Reig Vaqué - u215101
Marc Rodríguez Vitolo - u215105
Clara Serra Borràs - u204635

To test the retrieval and ranking system, we proposed five test queries that represent realistic user searches and cover diverse product categories. We have selected the following queries by analyzing the most frequent and most distinctive terms based on df, idf and tf-idf values. The selected queries are:

- cotton t-shirt men
- women winter jacket
- sports shoes running
- formal shirt slim fit
- casual footwear black

These queries were chosen to reflect common search patterns such as gender-specific clothing, material-based filtering, product types, etc. For each query, we have returned a ranked list of product identifiers corresponding to the most relevant items according to the TF-IDF cosine similarity score.

Finally, we have implemented an interactive query interface that allows users to manually enter any search query and retrieve the top-ranked results. This functionality is useful for validating our ranking approach and for exploring how well the search engine performs across different types of queries.

Universitat Pompeu Fabra Barcelona

Pau Reig Vaqué - u215101
Marc Rodríguez Vitolo - u215105
Clara Serra Borràs - u204635

Part 2: Evaluation

To evaluate the performance of our retrieval and ranking system, we implemented some evaluation metrics. These functions are designed to assess how well the ranked lists produced by our search algorithm align with the ground truth relevance labels provided in the validation set.

Precision @ K measures the proportion of relevant items among the top K retrieved results, while Recall @ K evaluates how many of the relevant products are retrieved. The Average precision @ K metric averages the precision values at the ranks of relevant products, rewarding systems that rank relevant results higher. The F1-score @ K combines both precision and recall to provide a balanced indicator of performance. The Mean Average Precision (MAP) aggregates the average precision values across all queries, giving an overall performance score.

We also implemented Mean Reciprocal Rank, which measures the position of the first relevant result, and, finally, Normalized Discounted Cumulative Gain, which evaluates ranking quality by giving higher weight to relevant items that appear earlier in the list.

Although some metrics can be implemented with non-binary relevance, we have implemented all metrics with the assumption of binary relevance (1 for relevant, 0 for non-relevant). Together, these measures provide a complete evaluation framework to compare different retrieval algorithms and verify the effectiveness of our ranking system.

All the metrics functions assume to have as inputs the whole or partial ranking returned by our search engine, and the validation dataframe containing ONLY the ones from that query. Therefore, inside the function we only filter to get the relevant documents pids without looking at the rest of attributes of the validation dataframe.

Therefore, in the next section we have applied the evaluation metrics to assess the performance of our TF-IDF–based retrieval system using two representative queries from the validation dataset, “women full sleeve sweatshirt cotton” and “men slim jeans blue”.

To compute the metrics for these two queries we filtered the ranking to only contain the documents that appear in the validation dataset. We maintain the order of the data returned by the ranking function, so we can evaluate how it worked.

Part 2.3.a : Expert judges

For the last subsection, we had to act as expert judges, establishing the ground truth for each document in the top 10 for each of the five queries defined in Part 1. We established the ground truth using the title and description information, and decided that perhaps some documents that were somewhat relevant should not be considered as such, since, considering that all documents in the ranking had to contain all the words in the query, it was difficult to determine that any document was truly irrelevant. We did this evaluation process with an auxiliary function that helped us create a CSV, called ground_truth.csv, containing

the same information as validation_labels.csv, to proceed in the same way as in the previous subsection.

Part 2.3.b : Metrics on our queries

With the relevance labels, we could calculate the metrics for our queries to evaluate how the search engine performed. The theoretical explanation is described before, but now we want to analyse and extract conclusions from the metrics for each query.

The first fourth queries have Precision@5 ≥ 0.8, meaning that in the top 5 results, 4 out of 5 documents (on average) were relevant, which implies that our system retrieves mostly relevant documents at the top positions. The last one has 0.6 which makes the system less consistent for this query, only 3 out of 5 results are relevant.

Regarding Recall, the values range from 0.37 to 0.67. This means our model retrieves about half of all relevant documents in the top 5. The system is precise but not fully exhaustive, it ranks the best items high but misses some others further down.

Regarding the Average Precision for the first four queries, the values are 1.0, which means that relevant documents are ranked near the top. The last query also has a high AP@5 (0.7), indicating an almost perfect ranking. This metric provides more information than precision, as it takes the ordering of relevant documents into account.

Regarding F1@5, the values range from 0.462 to 0.769. As in the other metrics, the first queries return better results, now they show balanced performance, indicating that the system maintains a good balance between retrieving relevant documents and ranking them highly. The last query has a lower F1@5 (0.462), reflecting its lower precision and recall, and highlighting the inconsistency in performance of our model for certain queries.

Regarding NDCG@5, the values range from 0.460 to 0.775. The first four queries have relatively high NDCG@5 scores (0.649–0.775), demonstrating that the system successfully ranks the most relevant items near the top. Query 5 again underperforms with an NDCG@5 of 0.460, indicating that relevant documents for this query appear lower in the ranking.

Regarding MAP@10, the value of 0.913 shows that relevant documents across all queries are ranked near the top, confirming that the performance and effectiveness of the search engine are quite good.

The last metric, MRR@10 has the value of 1.000, which indicates that the first relevant result for every query appears at the top position.

With all the metrics analyzed, we can conclude that the search system performs very well for most queries. However, more specific or unusual queries, like the last one, tend to retrieve fewer relevant results, which could be influenced by the limited size of our dataset. Additionally, our strict definition of relevance may have penalized some results that could still be useful, if we had been a bit more lenient in judging relevance, the system's performance could appear even stronger.

Universitat Pompeu Fabra Barcelona

Pau Reig Vaqué - u215101
Marc Rodríguez Vitolo - u215105
Clara Serra Borràs - u204635

Part 2.3.c : What problems does our search engine have and how can it be improved

The current search system relies on a tf-idf ranking algorithm, which only compares exact token matches. Because of this, documents that contain synonyms and related terms may be missed and ignored completely in the ranking. To solve this we could add something that searches for synonyms and give a certain weight to documents that might contain these words.

We also got problems with fields that are categorical and right now are used in the text. A clear example would be the color. Right now the color is included in the text and if the document contains the same color that appears in the query it might get a better ranking than others; but if it differs, that document should be excluded completely, and right now it can still be retrieved if other words match the ones in the query. We could solve this problem by having a structure inside the query, where the color could be specified separately and use it to filter results depending on that. We could also maintain the queries but identify if a term of the query is specifying the color and then proceed in the same way explained in the other possible solution.

When searching for products, users may formulate queries that are short and ambiguous, missing necessary information to get relevant results. We should try applying query expansion strategies or autocompleting suggestions to get more complete queries. Also parsing queries to certain structured attributes to implement field-weighted TF-IDF and allowing the user to fill out fields specifying those exact attributes (not forcing them to do it but giving them the opportunity to complete them).

Finally, with the current ranking system we are not using a lot of information that could be used to get more relevant results. For example the price, average ratings and other fields that might be very useful to decide which documents are relevant for the user making a certain query. All these features should be taken into account, giving more weight to the products that are popular and better rated.

GitHub URL: https://github.com/mmarcrv/irwa-search-engine-G005
TAG: IRWA-2025-part-2
AI Usage: We used ChatGPT and Gemini to review our draft explanation of pre-processing decisions and to help us make some prints or calculations.