IRWA Final Project – Part 2

Group 23

- Luca Franceschi
- Pau Peirats
- Júlia Othats-Dalès

Overview

This part of the project implements an information retrieval pipeline for fashion product search. We build and test a search engine that:

- indexes preprocessed product data,
- retrieves products based on keyword queries,
- ranks them using TF-IDF cosine similarity, and
- evaluates performance with standard IR metrics.

Review of Part 1

There was a error processing the "product_details" field in Part 1, now each product details has its own column. We have corrected this re-exported the cleaned dataset as `fashion_products_cleaned.csv`. From now on, we will use this corrected dataset.

We also retrieved the data cleaning used in Part 1 of the project. Each text field is lowercased, with punctuation removed, normalized using unicode, tokenized, stop words removed, stemmed, and short token words removed.

1. Indexing

1.1.

1. Each product row is tranformed into a document by concatenating selected text fields.

> Note: We decided to use only the `title` column after initially trying to use all available text columns (`title`, `description`, `brand`, `seller`, `category`, `subcategory`) but:
>
> - Including all fields degraded ranking quality, since most queries (from the assignment) are short title-based phrases.
> - Using only `title` improved relevance, but led to many identical TF-IDF scores, as titles are short.
> - To handle that, we preserved identical scores but decided to rank documents with equal score equally, instead of trying to order them randomly, ensuring reproducibility in evaluation.

2. We build a dictionary mapping terms --> list of product IDs that contain the term. Duplicates are avoided by tracking seen terms per document.

Sample of our inverted index built:

```
'cashmer': ['SWLFZC9YJ2GKCTHE', 'SWLFZC94PPHPGKZF', 'SWLFZC9GNNEK7TP2'],
'gener': ['SWLFZC9FYKU4EQ8V'],
'crepe': ['SWLFZC8VTKKHRTBH'],
```

3. We implement conjunctive (AND) queries, meaning all query items must appear in a returned document.

For instance, `conjunctive_search("men cotton shirt", inverted_index)` returns all products that contain *men*, *cotton*, and *shirt* in their title.

1.2.

Here we define 5 queries which we will later use to evaluate our search engine. We chose:

- 'men cotton shirt',
- 'women casual polo neck',
- 'men regular fit tshirt',
- 'zipper sweater',
- 'solid round neck cotton'

1.3.

In this part we implement the Tf-IDF algorithm seen in class to rank the documents by relevance scores.

We compute **Term Frequency** $$ tf_{t,d} = \frac{f_{t,d}}{\sum_t f_{t,d}} $$

Inverse Document Frequency $$ idf_t = \log \frac{N}{df_t} $$

Similarity Score $$ score(d, q) = \cos(\mathbf{v_d}, \mathbf{v_q}) $$

> Note: Again, note that we obtained multiple instances of documents with the same scores, so decided to rank those instances with the same rank. For example if we have score~d1=0.9, score~d2=0.9, score~d3=0.9, score~d4=0.7, score~d5=0.7, then rank number 1 would be given to all (d1, d2, d3) and rank 2 would be shared by documents (d4, d5).

2. Evaluation

2.1.

To assess search quality, we implemented and computed the following metrics:

- Precision@K
- Recall@K
- Average Precision (AP@K)
- F1-Score@K
- Mean Average Precision (MAP)
- Mean Reciprocal Rank (MRR)
- Normalized Discounted Cumulative Gran (NDCG@K)

These metrics are explained in more depth in Section 2.3. of the report.

2.2.

All metrics are computed for the two validation queries from `validation_labels.csv`.

1. women full sleeve sweatshirt cotton
2. men slim jeans blue.

The evaluation was performed at cutoff **K=20**, using the metrics defined in Section 2.1.

Query 1: *"women full sleeve sweatshirt cotton"*

Metric	Value
Precision@K	**0.8125**
Recall@K	**1.0000**
Average Precision@K	**0.1138**
F1-Score@K	**0.8966**
NDCG@K	**0.2759**
Reciprocal Rank (RR@K)	**0.5000**

This query achieved high precision and perfect recall, indicating that most relevant items were retrieved. However, a low Average Precision and NDCG suggest that relevant documents were not consistently ranked near the top, likely due to tied TF-IDF scores across multiple documents. Another indicative that something is not right is the low value for the average precision in comparison to the precision. As described more in depth in the notebook, this is an indicative that the amount of documents retrieved in the first 20 ranks are very few, but they are mostly correct (as explained in class, the reason why precision is not a good metric in IR systems).

Query 2: *"men slim jeans blue"*

Metric	Value
Precision@K	**0.5000**
Recall@K	**1.0000**
Average Precision@K	**0.1621**
F1-Score@K	**0.6667**
NDCG@K	**0.3215**
Reciprocal Rank (RR@K)	**1.0000**

Although precision is lower compared to Query 1, the first relevant document appears at the very top of the ranked list (RR = 1.0).
The model successfully retrieves all relevant results (Recall = 1.0) but continues to suffer from low ranking differentiation, as reflected by the moderate NDCG.

Aggregate Metrics

Metric	Mean Value
MAP	**0.1380**
MRR	**0.7500**

The system retrieves relevant items for both validation queries with **perfect recall**, but **ranking quality remains limited**.
Low MAP and NDCG confirm that while relevant documents are found, they are not always ranked optimally. These findings align with the analysis in Section 2.3, where short product titles and frequent TF-IDF score ties reduce ranking precision.

Attempted Probabilistic jittering (unsuccessfull)

As we have said before, during evaluation, we noticed that many documents shared identical TF-IDF scores, resulting in large tie groups and consequently poor ranking metrics (especially Average Precision and NDCG). To address this, we experimented with a probabilistic tie-breaking approach: for each query, we added a very small random noise to each document's score and recomputed the evaluation metrics across multiple random seeds.

In practice, this was implemented by sampling 10,000 different random seeds and selecting the one that produced the highest Precision@K for each query: `retrieved['score_randomized'] = retrieved['score'] + np.random.normal(0, 1e-7, retrieved['score'].size)`

The idea was to simulate a realistic ranking where equally-scored documents would be randomly ordered, avoiding artificial ties while keeping score differences negligible.

However, this approach failed to improve the overall results:

- Precision and recall remained inconsistent across seeds.
- Random noise introduced artificial ranking differences not supported by the actual TF-IDF similarity values.

As a result, we discarded this method and instead adopted a deterministic ranking using:
`retrieved['rank'] = retrieved['score'].rank(method='dense', ascending=False)`

This ensured reproducibility and consistent evaluation across runs, even though identical scores remained tied. Given the limited number of relevance labels and the high frequency of equal TF-IDF scores, the deterministic ranking provided a fairer and more interpretable evaluation baseline. This approach, however, poses significant changes in the way metrics are computed and might not be as theoretically backed as standard formulae.

2.3.

a. Relevance labels

To evaluate retrieval performance, we acted as expert judges and manually created binary relevance labels for each query–document pair.

For each of the five test queries defined in Section 1.2, documents were labeled as:

- **1** → relevant (document is clearly related to the query intent)
- **0** → not relevant

The resulting file `validation_labels2.csv` contains these annotations and is used as the ground truth for metric computation.

b. Metric Interpretation and Comparison

Each metric captures a different perspective of retrieval quality:

Metric	Interpretation
Precision@K (P@K)	Measures how many of the top-K retrieved documents are relevant. High precision means few false positives.
Recall@K (R@K)	Measures how many of all relevant documents were retrieved within the top-K. High recall means few false negatives.
Average Precision (AP@K)	Averages the precision values at ranks where relevant documents appear. Sensitive to ranking order.
F1-Score@K	Harmonic mean of precision and recall; balances the two aspects.
Mean Average Precision (MAP)	Mean of AP scores over all queries; overall measure of ranking quality.
Mean Reciprocal Rank (MRR)	Emphasizes how early the first relevant document appears.
Normalized Discounted Cumulative Gain (NDCG@K)	Considers the position of relevant items, rewarding higher-ranked relevant results.

From our experiments, **precision** values were generally high, but **average precision** and **NDCG** were relatively low.
This pattern occurs because many documents share identical TF-IDF scores, resulting in tied ranks.
While precision treats all retrieved relevant documents equally, average precision penalizes ties by rewarding only those that appear earlier in ranking.
Similarly, NDCG decreases when multiple relevant documents have equal (non-distinct) positions.

c. System Analysis and Limitations

Our current search system demonstrates the basic mechanics of indexing and TF-IDF ranking but also reveals several limitations:

1. **Score Ties due to Short Texts**
   Product titles are often extremely short (3–6 tokens). TF-IDF scores become identical for many items, leading to unreliable ranking.
   → **Possible improvement:** incorporate additional fields (e.g., *description*, *brand*) with lower weight, or use BM25 to handle term saturation more effectively.

2. **Limited Vocabulary Coverage**
   Restricting the index to a single column (`title`) ignores potentially useful contextual terms.
   → **Improvement:** build a multi-field index, possibly weighted (e.g., title × 2, description × 1).

3. **Binary Relevance Only**
   Current evaluation assumes relevance $\in$ {0, 1}. However, some products may be *somewhat* relevant.
   → **Improvement:** introduce graded relevance levels (e.g., {0, 1, 2}) to better exploit NDCG. However with our current ranking approach might not even be possible.

4. **No Query Expansion or Normalization**
   Queries are taken literally; synonyms like “tee” and “t-shirt” are treated differently.
   → **Improvement:** apply synonym expansion, stemming consistency, or word embeddings to improve recall.

5. **Exact-Match Conjunctive Search**
   The initial conjunctive retrieval requires all terms to appear, which can be too strict.
   → **Improvement:** move toward vector-space retrieval (TF-IDF cosine similarity) or semantic search.

Summary

Overall, our TF-IDF-based system retrieves relevant products with high precision but suffers from low discrimination among equally-scored documents.
Metrics such as MAP and NDCG reflect this limitation, emphasizing that ranking quality—not just retrieval—is crucial in information-retrieval systems.
Future improvements will focus on weighted multi-field indexing, BM25 ranking, and semantic query expansion to enhance robustness and user relevance.