

# Report 2

SNumbers: u264332, u264443, u264202

Names: Levente Olivér Bódi, Riccardo Zamuner, Giada Izzo

Github repository: [https://github.com/Levilevi01/Info\\_Retrieval/tree/main](https://github.com/Levilevi01/Info_Retrieval/tree/main)

Repository TAG: **IRWA-2025-part-2**

## Introduction

In this second part of the IRWA project, we focused on implementing the core information retrieval pipeline: building an inverted index, performing TF-IDF ranking, and evaluating system performance using a set of defined metrics. We used the preprocessed dataset from Part 1, indexed it for retrieval, ran multiple test queries, and assessed the system's effectiveness using both predefined and custom queries. This report provides an overview of our approach, summarizes the key implementation decisions, and presents the obtained results alongside a discussion of their implications.

## Part 1: Indexing

After pre processing the dataset, we constructed an inverted index to enable efficient search. Each unique term was assigned a term ID and mapped to the list of documents containing it. We used conjunctive (AND) queries to ensure that retrieved documents contained all query terms.

We then defined five custom queries to evaluate our search engine. These queries were selected to cover a diverse range of product attributes and terms. The exact results are viewable in the notebook.

## Part 2: Evaluation

We implemented several standard evaluation metrics to measure retrieval effectiveness. These included Precision@K, Recall@K, Average Precision, F1-Score, Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG). We applied these metrics to both predefined and custom queries using relevance judgments from the given file and custom labeling, respectively.

Below are the computed evaluation results, rounded to three decimal places:

- Evaluation for Query 1:  
k=15: Precision@k=0.067, Recall@k=0.050, F1@k=0.057, AP@k=0.500,

NDCG@k=0.108

Evaluation for Query 2:

k=15: Precision@k=0.067, Recall@k=0.050, F1@k=0.057, AP@k=0.083,  
NDCG@k=0.046

MAP@15: 0.292, MRR = 0.292

- Evaluation for query: cotton tshirt 50 100 men blue
  - k=5: Precision@k=0.000, Recall@k=0.000, F1@k=0.000, AP@k=0.000,  
NDCG@k=0.000
  - k=10: Precision@k=0.000, Recall@k=0.000, F1@k=0.000, AP@k=0.000,  
NDCG@k=0.000

Evaluation for query: adidas red

k=5: Precision@k=0.600, Recall@k=1.000, F1@k=0.750, AP@k=1.000,  
NDCG@k=1.000

k=10: Precision@k=0.300, Recall@k=1.000, F1@k=0.462, AP@k=1.000,  
NDCG@k=1.000

Evaluation for query: denim jeans skinny

k=5: Precision@k=0.000, Recall@k=0.000, F1@k=0.000, AP@k=0.000,  
NDCG@k=0.000

k=10: Precision@k=0.000, Recall@k=0.000, F1@k=0.000, AP@k=0.000,  
NDCG@k=0.000

Evaluation for query: dress red

k=5: Precision@k=0.000, Recall@k=0.000, F1@k=0.000, AP@k=0.000,  
NDCG@k=0.000

k=10: Precision@k=0.300, Recall@k=0.083, F1@k=0.130, AP@k=0.216,  
NDCG@k=0.199

Evaluation for query: leather jacket

k=5: Precision@k=1.000, Recall@k=0.143, F1@k=0.250, AP@k=1.000,  
NDCG@k=1.000

k=10: Precision@k=1.000, Recall@k=0.286, F1@k=0.444, AP@k=1.000,  
NDCG@k=1.000

MAP@5: 0.400, MRR = 0.425

MAP@10: 0.443, MRR = 0.425

## Discussion and Analysis

### Evaluation metrics explanation and analysis

- Precision@k: fraction of the top-k results that are relevant. Uses only the top-k set and binary relevance judgments.
- Recall@k: fraction of all relevant documents that appear in the top-k. Uses top-k and the full set of known relevant docs.
- F1@k: harmonic mean of Precision@k and Recall@k. Summarizes the balance between precision and recall at a given k.
- Average Precision (AP@k): mean of precision values computed at ranks where relevant documents occur (truncated at k). Uses ordering information within top-k and rewards early placement of relevant docs.
- Mean Average Precision (MAP@k): mean of AP@k across queries. Used to assess the quality of the retrieval system.
- Reciprocal Rank (RR) and Mean Reciprocal Rank (MRR):  $RR = 1 / \text{rank of first relevant doc}$ ; MRR averages RR across queries. Uses only the first relevant hit.
- DCG@k / NDCG@k: discounted cumulative gain accounts for position (log discount); NDCG normalizes by ideal DCG so scores are comparable. Generally used with graded relevance (with binary relevance it simply accounts for discounted hits).

### Result interpretation

- High precision, low recall (small k): top results are clean but many relevant docs fall outside top-k. To fix this we can increase recall via broader matching or query expansion, raise k, or tune retrieval. (This is the case for Q5)
- Low precision, high recall: system retrieves many relevant items but top positions contain non many non relevant documents. To fix this improve ranking, incorporate quality/popularity signals. May also be the fact there are not many relevant documents to begin with (like Q2, where one should actually use  $k = \min(\text{chosen\_k}, \text{len}(\text{relevant\_documents}))$ )
- All metrics low: it can mean multiple things: preprocessing is not done well, tokenization may have mismatch, there could be indexing problems, or simply the retrieval model is too weak. (This is the problem for the other 3 queries, where the queries are pretty general and contain frequent terms)

## Retrieval problems and solution proposals

Right now the index and ranking focus almost entirely on token overlap in the text fields, but a lot of useful information in the product records is effectively ignored. Numeric fields like price, discount and rating are converted/cleaned but not used as ranked signals even though they can be powerful relevance cues for users (cheap vs premium, high rating, big discount), yet the current pipeline treats them as side data rather than features that should influence ranking. That means two items that match the query text equally well may be ordered arbitrarily even though one is clearly preferable by price or rating.

Another important issue is that all text fields are treated the same. Title, brand and description are concatenated or given equal treatment in scoring, so a match in a brand string carries the same weight as the same match in a long description. In practice users care much more about matches in short, high-precision fields (title, brand) than in noisy long fields. Not giving fields different weights dilutes strong signals and reduces ranking accuracy.

Finally, the ranking ignores word order and phrases. The system is effectively bag-of-words: it counts tokens but doesn't reward exact phrases or tokens that occur close together. Queries like "slim blue jeans" or "full sleeve sweatshirt" lose meaning when order and proximity aren't considered; documents that contain the words scattered across different parts of the page can be ranked equally to documents that contain the exact phrase, which harms perceived relevance.

In short: useful numeric signals aren't used as ranking features, field importance is flattened, and phrase/proximity information is lost. Some fixes proposals are the followings: treat numeric fields as features or filters, boost short/high-precision fields (title, brand) when scoring, and enable positional/phrase handling (N-grams instead of unigrams).

## Conclusion

Overall, this second part of the project demonstrated the complete implementation of an end-to-end information retrieval pipeline, from indexing to evaluation. The results obtained highlight both the effectiveness and limitations of our current approach. Future improvements could focus on refining the ranking function, incorporating field-based weighting, and experimenting with alternative retrieval models.