IRWA-2025-part-2
Github repository: https://github.com/laiaatomas/IRWA_FinalProject
Laia Tomàs U198723 NIA252294
Quim Ribas U198742 NIA251754

FINAL PROJECT: REPORT PART 2

PART 1: INDEXING

1. Build the Inverted Index

In this first part, we need to create the inverted index. Following the statement, we would compute the index as follows:

$$term_i : [pid_1, pid_2, pid_3 \ldots]$$

where each entry is composed of two elements, a term (e.g. the word in the collection) and a list which contains all products in which this term is present. However, for more clarity, we followed the following structure:

$$term_i : \{pid_1: tf_{i,1},\ pid_2: tf_{i,2},\ pid_3: tf_{i,3}\}$$

where $tf_{i,p}$ is the frequency of the given word i in the product p. In the notebook we included the implementation of both indices, even though we used the latter one. For the search function, because we are using conjunctive queries, we changed the function of Practice 1 to return the list of products containing all terms, and if some term is not present, then the list will be empty.

2. Test Queries

For this section, we are asked to propose five queries. Following the suggestion, we define the queries based on the following metric. We use term frequencies (tf) to rank all the terms in the collection and obtain the most popular words. The tf metric will help us simulate realistic user behavior (broad topic, many hits). Moreover, after testing many queries with different word combinations, we decided to include two queries which don't give the expected results (query 4 and 5), in order to account for the limitations of our system, which we outline later. Lastly, query 3 is included to show the results of a fairly specific query, as "animal" was not among the top common words.

Overall, this selection of queries allows us to evaluate the system across a range of scenarios, from broad, high-frequency terms to more challenging cases. By including queries that don't recommend such relevant products, we can better understand the system's strengths and weaknesses, and these insights will guide the analysis and discussion in the following sections. The queries are shown in Figure 1.

```
query1 = "women blue casual tshirt"
query2 = "cotton fit casual"
query3 = "animal print tshirt"
query4 = "round neck black dress"
query5 = "solid shirt pack of 3"
```

Figure 1. Test queries

Before the ranked results, we checked the implementation of the search function with 2 queries, and verified that results included all the terms from each query.

3. Ranking

In the implementation of the TF-IDF algorithm, we adapted the functions from the Practice 1 to our index structure to provide the ranked products. Afterwards, we printed the top 5 results for each of the queries defined above, along with the product title and description. The snippet in Figure 2 shows the results for query 1.

```
Results for query: women blue casual tshirt

======================
Top 5 results out of 301 for the searched query:

product_id= TSHEUJ4VFYTMZTXZ
- product_title: Solid Women Round or Crew Blue T-Shirt
- product_description: Axmann Light Blue Round Neck Casual Summer Wear Women T-shirt BY Axmann. Made with premium fabric for the most ultimate comfort. Range of vibrant colored casual t-shirts

product_id= TSHFEY8VRGCMRSGG
- product_title: Typography Women Round Neck Blue T-Shirt
- product_description: Blue Chest print Kintted Cotton Casual T-shirts, has a round neck , Half Sleeve

product_id= TSHEU7DTUYMHMGW6
- product_title: Striped Women Polo Neck Blue T-Shirt
- product_description: Axmann Blue Striped Casual Wear Women Summer Polo T-shirt BY Axmann. Made with premium fabric for the most ultimate comfort. Range of vibrant colored casual t-shirts

product_id= TSHEU7DSSQGESYNH
- product_title: Printed Women Round Neck Blue T-Shirt
- product_description: Axmann Blue Round Neck Casual Summer Wear Women T-shirt BY Axmann. Made with premium fabric for the most ultimate comfort. Range of vibrant colored casual t-shirt

product_id= TSHEU7DTRZDFUEZB
- product_title: Printed Women Round Neck Blue T-Shirt
- product_description: Axmann Blue Round Neck Printed Casual Wear Women Summer T-shirt BY Axmann. Made with premium fabric for the most ultimate comfort. Range of vibrant colored casual t-shirt
```

Figure 2. Results for query 1.

PART 2: EVALUATION

1. Metric implementation

In order to calculate all of the evaluation metrics, we define a function to compute each of them and later join them in a main function called *print_results_at_k()*. For each of the following metrics we created a function to implement its formula:

- Precision@K (P@K)
- Recall@K (R@K)
- Average Precision@K (P@K)
- F1-Score@K
- Mean Average Precision (MAP)
- Mean Reciprocal Rank (MRR)
- Normalized Discounted Cumulative Gain (NDCG)

We also have created some auxiliary functions to increase the readability of our code, such as *reciprocal_rank()* or *dcg_at_p()*.

Answering the question of the bonus point, we used np.argsort(-y_score). This is slightly faster than np.argsort(y_score)[::-1] because it avoids creating a reversed copy; negating the array lets us sort in descending order directly.

Moreover, we checked the validity of our functions comparing the results of *average_precision_at_k()* and normalized_discounted_cumulative_gain() to the ones from the function from *sklearn.metrics.*

2. Evaluation results

For this section, we need to evaluate the search results for the two stated queries based on the relevance judgments provided in validation_labels.csv. The evaluation is computed using the metrics explained above.
The queries are:

- Query 1 (Q1): *women full sleeve sweatshirt cotton*
- Query 2 (Q2): *men slim jeans blue*

For this task, we joined the products from the validation dataframe and ones from the predicted scores by “pid”, and then applied the evaluation functions. The numeric results for both queries, using k=10, are shown in Figure 3.

```
------------------------------------------------
Results for query_id: 1

Precision@10: 0.9

Recall@10: 0.818

Average precision@10: 1
```

```
F1-Score@10: 0.857

NDCG@10: 0.936
-------------------------------------------------
Results for query_id: 2

Precision@10: 0.75

Recall@10: 1

Average precision@10: 0.948

F1-Score@10: 0.857

NDCG@10: 0.984
-------------------------------------------------
MAP@10: 0.974

MRR@10: 1
```

Figure 3. Evaluation metrics for query 1 and 2 with k=10.

3. Establish ground truth for queries

For this last part of the practice, we were asked to assign a binary relevance to the products for the queries defined in Part 1. For simplicity, we selected the top 20 products to manually assign their relevance. For the first 3 queries, all the recommended products were relevant to their corresponding query, as the output item correctly reflected the query meaning.

For query 4 (“round neck black dress”), none of the top 20 ranked products were relevant. Each item was either a t-shirt or sweater, but never a dress. After some investigation, we found that the term “dress” appeared only in product descriptions, where it was used in expressions such as “dress up” or “dressing,” rather than referring to the actual clothing item.

For query 5 (“solid shirt pack of 3”), even though the items were all packs of “solid shirts”, all of the retrieved items were packs of 2, instead of 3. For this reason, we labeled all results as non-relevant.

The numeric results for our queries are shown in Figure 4.

```
-------------------------------------------------
Results for query_id: women full sleeve sweatshirt cotton

Precision@10: 1

Recall@10: 0.5
```

```
Average precision@10: 1

F1-Score@10: 0.667

NDCG@10: 1
--------------------------------------------------
Results for query_id: men slim jeans blue

Precision@10: 1

Recall@10: 0.5

Average precision@10: 1

F1-Score@10: 0.667

NDCG@10: 1
--------------------------------------------------
Results for query_id: animal print tshirt

Precision@10: 1

Recall@10: 1

Average precision@10: 1

F1-Score@10: 1

NDCG@10: 1
--------------------------------------------------
Results for query_id: round neck black dress

Precision@10: 0

Recall@10: 0

Average precision@10: 0

F1-Score@10: 0

NDCG@10: 0
--------------------------------------------------
Results for query_id: solid shirt pack of 3
```

```
Precision@10: 0

Recall@10: 0

Average precision@10: 0

F1-Score@10: 0

NDCG@10: 0
--------------------------------------------------
MAP@10: 0.6

MRR@10: 0.6
```

Figure 4. Evaluation metrics for our five queries.

Evaluation metrics

Each of the implemented metrics provides different insights into the performance of our retrieval system:

- Precision@K (P@K) measures the proportion of relevant items among the top-K retrieved products. It focuses on the quality of the returned results. In our case, using the queries from the validation file, high precision values indicate that the top results are mostly relevant when the system retrieves items correctly.

- Recall@K (R@K) captures how many of the relevant items in the whole collection were actually retrieved in the top-K. Lower recall values, using our defined queries (0.5 for some queries) suggest that, although retrieved results were correct, the system missed other relevant items further down the ranking.

- Average Precision (AP) combines both precision and ranking order by giving higher importance to relevant items retrieved earlier. It provides a smoother evaluation of ranking quality, which helps identify whether the system consistently retrieves relevant documents at top positions.

- F1-Score is the harmonic mean of Precision and Recall, offering a balanced view of both measures. It's useful to understand how well the system maintains a compromise between completeness and accuracy.

- Mean Average Precision (MAP) aggregates the AP values across queries, reflecting overall ranking performance. Our MAP@10 of 0.6 indicates that while the system performs well for simple, well-represented queries, its global consistency across all cases is limited. However, when using the queries from the validation set, we reached a MAP of 0.974.

- Mean Reciprocal Rank (MRR) measures how early the first relevant result appears. A perfect score (1.0) for some queries confirms that relevant results are often ranked first.

- Normalized Discounted Cumulative Gain (NDCG) considers both relevance and ranking position, rewarding systems that rank highly relevant results near the top. Perfect NDCG values for relevant queries (1.0) show that, when the system succeeds, it ranks results optimally.

In summary, these metrics together show that our system performs strongly for queries well represented in the index but struggles when query terms are missing or used in unexpected contexts.

System limitations and possible improvements
While the system performs adequately for simple, high-frequency queries, several limitations became evident:

Strict Conjunctive Query Matching

- Problem: If any query term is missing from the index, the system returns no results (e.g., queries containing “nike”), even though some other words of the query might be in the index and be a relevant product.

- Possible Solution: Implement disjunctive (OR) matching, allowing products that contain most of the query terms to be retrieved. Another option is to use a ranking threshold rather than a hard intersection.

Lack of Semantic Understanding

- Problem: The system fails to distinguish word meanings and contexts (e.g., “dress up” vs. “dress” as a noun).

- Possible Solution: Integrate word embeddings (e.g., Word2Vec) or transformer-based models to capture semantic similarity between query and product terms.

Limited Field Awareness

- Problem: All text fields (title and description) are treated equally, even though title usually carries more meaning. Moreover, important contextual information is available in other fields such as category, sub_category, brand, product_details, and seller, which are excluded from the index.

- Possible Solution: Extend the indexing process to include these additional fields and apply field-specific weighting (boosting) to improve ranking quality.

Multi-word Query Handling

- Problem: Multi-term expressions like “pack of 3” are not interpreted as a single concept, and numbers are kept out of the index.

- Possible Solution: Use n-grams during indexing to preserve such expressions.

TF-IDF ranking only

- Problem: TF-IDF doesn’t handle document length normalization perfectly. Many products have either very long descriptions, or an empty description, which can lead to distorted scores.

- Possible solution: Replace or compare with BM25, which often yields better retrieval quality.

By addressing these limitations, the system could achieve better recall, semantic accuracy, and robustness across diverse query types.