IRWA Final Project – Part 3

Group 23

- Luca Franceschi
- Pau Peirats
- Júlia Othats-Dalès

Overview

In Part 3 we add more ranking methods and simple filters. We will be:

- reusing the cleaned dataset from Part 1
- ranking with: TF-IDF + cosine, BM25, a custom mixed score
- adding a Word2Vec + cosine semantic rank
- allowing simple sorting (rating, price, discount)

Queries use the same preprocessing as before (lowercase, remove punctuation, normalize, remove stopwords, stem, drop very short tokens).

Data

- File: `fashion_products_cleaned.csv`
- Text fields used: title, description, brand, category, sub_category and seller.
- All those fields are cleaned already.
- Queries are also cleaned the same way.

1. Ranking Methods

1.1 TF-IDF + cosine similarity

In this method we have implemented thw same we had in Part 2. But now using more text fields. Ties keep happening, but fewer than using title only.

PROS

- Easy debugging and understanding, we control every step and can inspect TF and IDF values directly.
- Fast to compute for our dataset size.
- Good results for exact term matches.

CONS

- Ignores sysnonyms and semantic meaning. (Will be improved with Word2Vec later)
- No length norm besides L2, long descriptions may skew results.
- No term saturation control (unlike BM25). Same term repeated many times can keep increasing score.

1.2 BM25

In this method we have implemented BM25 ranking using the `rank_bm25` library with default parameters (k1≈1.5, b≈0.75). We expect it to perform better than TF-IDF due to its term saturation and length normalization features.

PROS

- Length normalization helps balance short vs long documents.
- Term-frequency saturation avoids giving extra importance to repeated terms.

CONS

- More complex to implement and understand than TF-IDF, we have used a library for it.
- Therefore less debug control over exact scoring details.
- Still lexical like TF-IDF, so synonyms are not handled.

1.3 Custom Hybrid Score

Our idea is to combine in our score textual relevance (like we do in TF-IDF and BM25) together with numerical relevance (higher product average rating can be more relevant for the user searching for that product, or relevance for the user could be inversely proportional to the price, or higher discount could be relevant, etc).

```
combined_score = 0.40 * text_score + 0.30 * rating_score + 0.20 * discount_score + 0.05 * availability_score + 0.05 * price_score
```

Where:

- `text_score`: TF-IDF or BM25 score normalized by max score in that result set
- `rating_score`: average_rating / 5
- `discount_score`: discount / 100
- `availability_score`: 1 if in stock else 0
- `price_score`: higher for lower prices (we use a log transformation on `selling_price`)

PROS

- Mixes textual relevance with business/user value signals.
- More aligned with retail business goals.
- Flexible tuning of weights for different user preferences.
- Works as a layer on top of existing text ranking.

CONS

- Weights are subjective and hand-tuned, don't learn from user behavior.
- Very relevant text matches can be pushed down by low ratings/price.

2. Word2Vec + cosine similarity

In this method we have implemented a simple Word2Vec-based ranking. We expect it to help with semantic meaning, returning relevant items even if the query is not an exact lexical match. This is a way more realistic approach than the prevois methods.

Steps of the Word2Vec ranking:

- Train Word2Vec (vector_size=100, window=5, min_count=1, seed=42) on tokenized corpus.
- Doc vector = average of word vectors in that doc.
- Query vector = average of its tokens.
- Score = cosine similarity.

PROS

- Captures semantic meaning beyond exact term matches.
- This allows matching queries and products that use different but related words.
- Once word vectors are trained, ranking is fast (just vector averages and cosine).

CONS

- Depends strongly on training data, a small dataset may not capture all word meanings well.
- More complex to maintain and debug than previous methods. (Because of Word2Vec training)

Filtering and Sorting

After ranking we can:

- take top K
- sort by highest rating, lowest price, highest discount
- combine multiple sort keys if needed

Useful to refine results after relevance scoring.

Comparison between methods

TF-IDF vs BM25

BM25 usually ranks stronger matches higher for short queries thankst to length normalization and term frequency saturation.

TF-IDF is simpler and easier to debug since we are implementing it oursevles.

On our dataset both have similar performance, but BM25 is slightly better, at least for top-20 with our test queries.

Hybrid

With the hybrid score we basically took the lexical result (TF-IDF or BM25) and re-ranked it using product metadata: rating, discount, availability and price. So the text score still decides which products enter the top results, but then the hybrid score pushes up well-rated, in-stock and discounted items, and pushes down expensive or low-rated ones.

Compared to pure TF-IDF/BM25, this is more realistic scenario because users usually care not only about matching the query, but also about quality and price.

The downside is that sometimes a very good textual match with bad rating or no discount goes down in the list. So pure BM25 is better if we only care about strict text relevance, while the hybrid ranking is better if we care about “best deal for this query”.

Word2Vec

Word2Vec differs from the other methods because we do not rely only on exact words. We train embeddings on our corpus and then rank by cosine between the average query vector and the average product vector. This lets us retrieve products that are semantically related, even if they do not share all query terms exactly.

Compared to TF-IDF/BM25, Word2Vec helps when the wording changes or when the query is very short and vague. However, it can bring in items that are “topically similar” but not really what the user wanted. Also, training quality is limited by the size of our dataset.

Ways to improve our results

- Field weighting (2*title, 1.5 * description, ...).
- Tune BM25 (k1, b).
- Try Doc2Vec or Sentence2Vex.

Summary and Conclusions

In this third lab, we implemented and compared four ranking options:

- TF-IDF + cosine (lexical baseline)
- BM25 (lexical with length and saturation control)
- Custom hybrid (text + rating + discount + availability + price)
- Word2Vec + cosine (simple semantic averaging)

On our dataset, BM25 slightly improves early ordering compared to TF-IDF, so it is a better default lexical ranker.

The hybrid score adds user and business value by mixing text relevance with rating, discount, availability and price. This makes the ranking more useful for a retail setting, even if we sometimes sacrifice strict text relevance.

Word2Vec brings semantic information into the ranker and helps when query wording does not match the product text exactly. On its own it is not as reliable as BM25 for precise matching, but it is very useful as a complementary layer.

Overall, a reasonable final setup for this project would be:

1. Use BM25 (or TF-IDF) to get a candidate set of products.
2. Re-rank them with the hybrid score to reflect user value.
3. Maybe add Word2Vec on top as a semantic helper (for similar items or for queries with few lexical matches).

This combination wpuld give a more robust and realistic ranking pipeline than any single method alone.