Paula Ceprián (u198630), Judit Viladecans (198724), Berta Noguera (u199893)

Final Project - Part 4

RAG, User Interface, and Web Analytics

GitHub: https://github.com/juditvribe/IRWA_2025_G14_FinalProject.git
TAG: IRWA-2025-part-4

In this part of the project we built a complete web interface for our search engine. We implemented the UI using Flask, following the structure required by the project statement. Our design choices focused on simplicity, clarity, and direct connection between the UI and the backend engine.

PART 1: User Interface

1.1. Search page

We created a main landing page (index.html) that contains a **central search box** where users type their query, a dropdown for selecting the **ranking algorithm** (TF-IDF, BM25, our own custom score, or Word2Vec), and a **button to run the search**. When users access the search page, we also initialize session data using Flask's secure cookies. We identify the user's browser and operating system through the *httpagentparser* library and immediately log this information in our analytics tables.

1.2. Search action

When the user clicks the button, the query and selected ranking method are sent via POST to the */search* route. The */search* route retrieves these values and forwards them to search_engine.search(). We also count the search event as part of the total click tracking and we also store the last query in the session so the system can also track analytics such as dwell time and number of returned results.

1.3. Search function in the engine

The search_engine.search() method takes the **query string**, the selected **ranking option** (the user's search algorithm choice), a **search ID** assigned by the analytics module, and the **product corpus**. Then the function receives the necessary arguments and abstracts the logic for calling the appropriate ranking method.

1.4. The search algorithms

Inside search_engine.search(), we trigger the appropriate algorithm based on the user's selection as said before. We optimized our ranking implementations for web use by **preloading the corpus into memory** at application start, **precomputing term statistics** and avoiding recomputation on each request, **cleaning textual fields during loading** so that algorithms run without repeated preprocessing and finaly, **structuring the ranking modules** so they work efficiently inside a web request.
These decisions make the engine faster, cleaner, and more suitable for an interactive user experience.

1.5. The results page

The */search* route returns **results.html**, which displays each document in ranked order. As specified in the statement, each result shows:

- Title
- Description
- Selling price and discount
- Average rating
- URL linking to the original website
- Any other attributes available in the corpus

We also display the number of total results and provide a link to each document's details page.

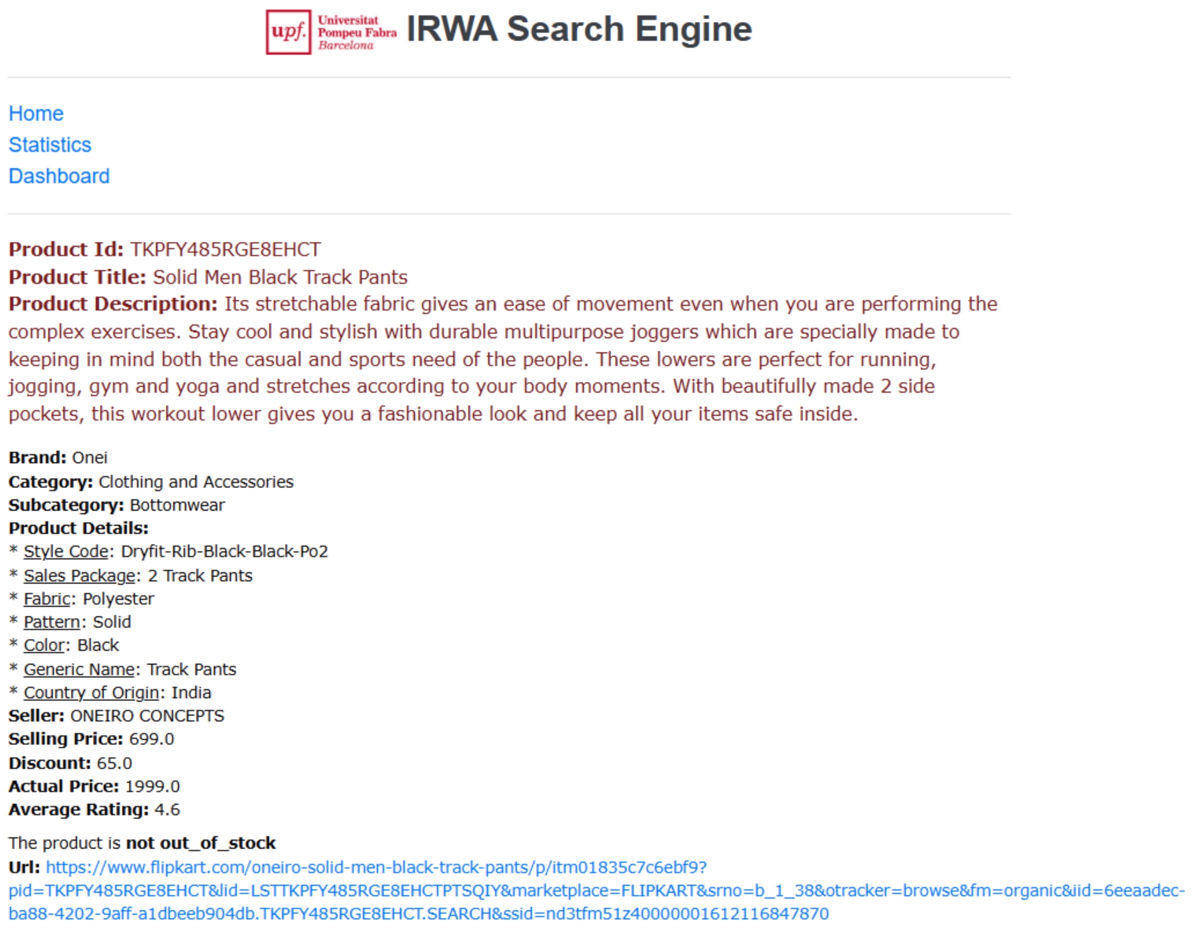
1.6. LLM summary on the results page

We integrate the RAG system by generating a summary of the retrieved results. After computing the ranking, the */search* route calls rag_generator.generate_response() using the query and the top results. The generated summary is inserted at the top of the results page. Adding results summarization using a third-party API to form a RAG system over our custom corpus.

It generates a short natural-language summary that appears above the ranked list. This allows users to receive an instant synthesis of the search results, making the system a simple end-to-end RAG search engine over the custom e-commerce corpus.

1.7. The document details page

The */doc_details* route provides a page (doc_details.html) that displays all available information for a clicked product: full description, pricing, rating, brand, categories, etc. Each visit to the details page is logged into analytics as a click event, incrementing both the global click count and the per-document click statistics.



Product Id: TKPFY485RGE8EHCT
Product Title: Solid Men Black Track Pants
Product Description: Its stretchable fabric gives an ease of movement even when you are performing the complex exercises. Stay cool and stylish with durable multipurpose joggers which are specially made to keeping in mind both the casual and sports need of the people. These lowers are perfect for running, jogging, gym and yoga and stretches according to your body moments. With beautifully made 2 side pockets, this workout lower gives you a fashionable look and keep all your items safe inside.

Brand: Onei
Category: Clothing and Accessories
Subcategory: Bottomwear
Product Details:
* Style Code: Dryfit-Rib-Black-Black-Po2
* Sales Package: 2 Track Pants
* Fabric: Polyester
* Pattern: Solid
* Color: Black
* Generic Name: Track Pants
* Country of Origin: India
Seller: ONEIRO CONCEPTS
Selling Price: 699.0
Discount: 65.0
Actual Price: 1999.0
Average Rating: 4.6

The product is **not out_of_stock**
Url: https://www.flipkart.com/oneiro-solid-men-black-track-pants/p/itm01835c7c6ebf9?pid=TKPFY485RGE8EHCT&lid=LSTTKPFY485RGE8EHCTPTSQIY&marketplace=FLIPKART&srno=b_1_38&otracker=browse&fm=organic&iid=6eeaadec-ba88-4202-9aff-a1dbeeb904db.TKPFY485RGE8EHCT.SEARCH&ssid=nd3tfm51z40000001612116847870

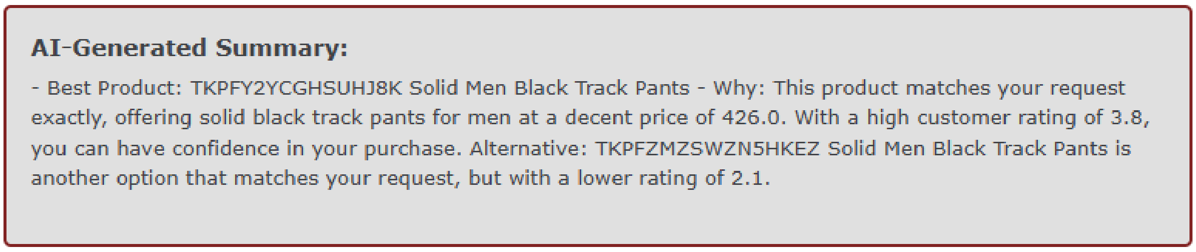
PART 2: RAG

The baseline RAG generator (as initially given) suffered from three important limitations. The first one is that the prompt was very generic and produced vague summaries disconnected from what the user actually wanted. It simply passed the query to the LLM without any consideration of the product structure or user intent. The second one was that it treated all products equally, ignoring metadata such as rating or discount. And the third limitation was that it did not handle cases where retrieved results were irrelevant or too weak to form a meaningful answer.

The improvements that we implemented were to **redesign the prompting strategy to be much more context-aware**. The updated prompt instructs the LLM to behave as an e-commerce assistant, so that it understands that it is summarizing e-commerce products in order to focus on the most relevant product attributes, avoids features not present in the data and generates structures and concise summaries. However, it increased the amount of text passed to the LLM.

Moreover, we **included additional metadata** when building the RAG context. Now the retrieved documents not only contain their title and description but also the rating and the price. This improvement helps the LLM make more useful comparisons and recommendations and making the summary more aligned with real consumer criteria. However, this improvement lead to higher token usage and slightly slower response times. The system also became more dependent on the quality and consistency of the product metadata: when some entries are incomplete or poorly structured, the LLM may still struggle to produce balanced summaries.
Finally, we introduced **simple logic for handling cases where the query yields no strong matches**. Instead of returning an uninformative or hallucinated summary, the system now instructs the LLM to acknowledge the lack of suitable products and suggest that the user reformulate the query.

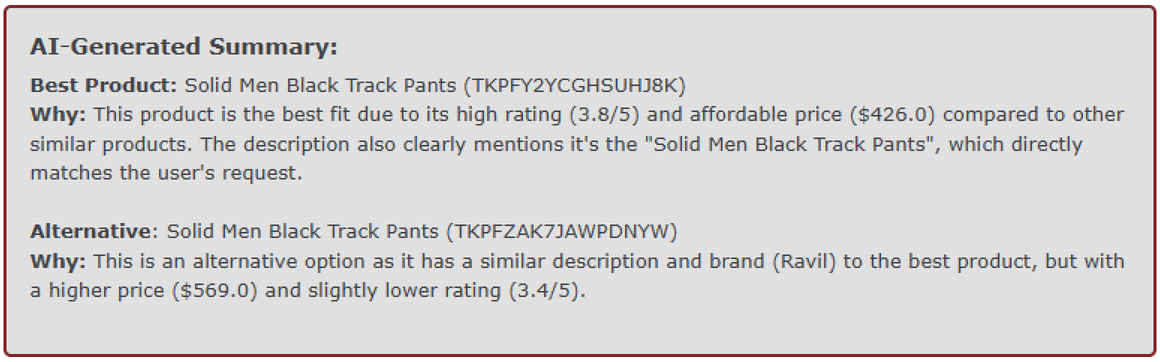
Here is an example of the improved implementation for the query "Solid Track Black Pants for Men" using BM25 as the chosen search method.
<u>Before</u>:



AI-Generated Summary:

- Best Product: TKPFY2YCGHSUHJ8K Solid Men Black Track Pants - Why: This product matches your request exactly, offering solid black track pants for men at a decent price of 426.0. With a high customer rating of 3.8, you can have confidence in your purchase. Alternative: TKPFZMZSWZN5HKEZ Solid Men Black Track Pants is another option that matches your request, but with a lower rating of 2.1.

<u>After</u>:



AI-Generated Summary:

Best Product: Solid Men Black Track Pants (TKPFY2YCGHSUHJ8K)
Why: This product is the best fit due to its high rating (3.8/5) and affordable price ($426.0) compared to other similar products. The description also clearly mentions it's the "Solid Men Black Track Pants", which directly matches the user's request.

Alternative: Solid Men Black Track Pants (TKPFZAK7JAWPDNYW)
Why: This is an alternative option as it has a similar description and brand (Ravil) to the best product, but with a higher price ($569.0) and slightly lower rating (3.4/5).

This example clearly shows the improvement made. In the baseline version, the output is generic, unstructured, and mixes product data in a single flat sentence, reflecting the weaknesses of the original prompt and the lack of explicit metadata guidance. After applying our improvements, the summary becomes organized into clear sections (“Best Product” and “Alternative”), each justified with concrete attributes such as rating, price, brand, and exact title match. This results from passing richer metadata into the context and using a refined prompt that demands concise, structured explanations as explained previously. To sum up, the improved version is easier to read, more trustworthy, and better aligned with the user’s intent, although it comes with the expected trade-offs of higher token usage and a reliance on consistent metadata quality.

PART 3: Web Analytics

For the web analytics part of the project, our goal was to design a system capable of tracking how users interact with the search engine, the interface, and the underlying data. Since persistence was not a requirement, we chose to store all analytics information in memory for simplicity and speed. The analytics module we created, **AnalyticsData**, acts as a central repository that collects information on requests, clicks, queries, sessions, and user context.

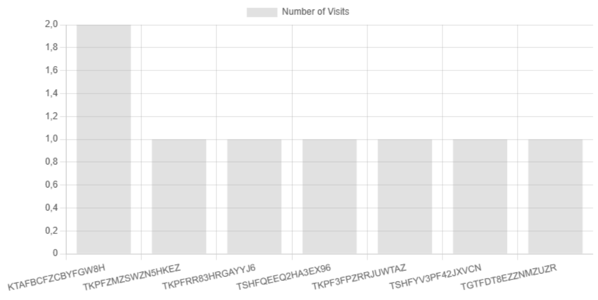
Each time a user accesses the application, the system records the browser type, the operating system, and the user’s IP address. This provides essential context for understanding the environment in which the search engine is being used. The Flask session mechanism allows us to maintain a simple session identifier, which we associate with a session start time. With this information, we can estimate the duration of a user session and count the total number of clicks that the user did. Every HTTP request increments a counter associated with the user's IP, which helps identify the most active visitors. When users submit queries, the analytics component stores the query text, counts how many times it has been searched, and analyzes the number of terms it contains. We also link each query to the list of documents returned by the search engine so we can later show which queries led to which documents. We also store each clock on individual product pages in order to determine which items generated the most interest.

To organized this information in a coherent way, we followed a structure inspired by a star schema. We maintained a set of “fact” structures such as **fact_clicks**, **fact_http_requests**, **fact_http_sessions**, **fact_queries**, and **fact_queries_to_docs**. Each structure captures a specific type of event or user action and is designed to allow fast retrieval from the dashboard.

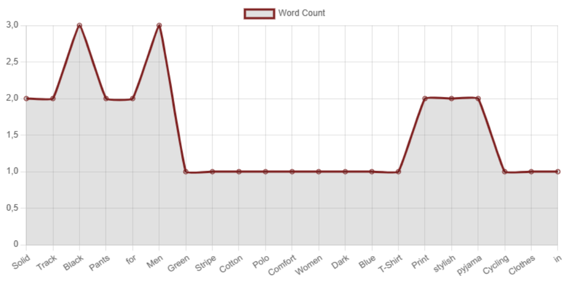
The **analytics dashboard**, implemented in the */stats* and */dashboard* routes, presents a collection of indicators that summarize the system’s usage. It displays the top five most clicked documents, the top five most frequent queries, and the average number of terms per query both globally and for the top subset. Moreover, it also shows the list of search algorithms chosen by users, the term frequency across all queries, and the documents associated with each search. Additional information such as the user’s browser, operating system, session time, and request patterns helps us understand how people interact with the system.

Here you can visualize the different graphs provided in the dashboard:

Ranking of Visited Documents



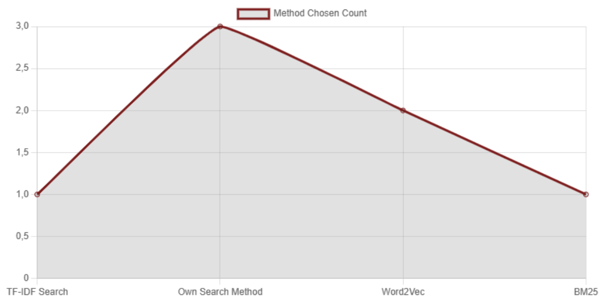
Term Frequency



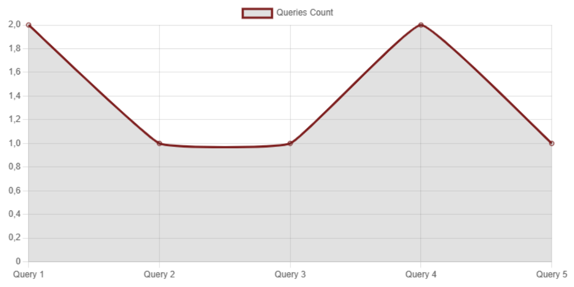
Ranking of Search Options

The search options to choose from are the following:

1. TF-IDF Search
2. Own Search
3. Word2Vec
4. BM25



Ranking of Queries



The queries displayed above are the following:

Query **#1**: "Solid Track Black Pants for Men" has been searched 2 times.

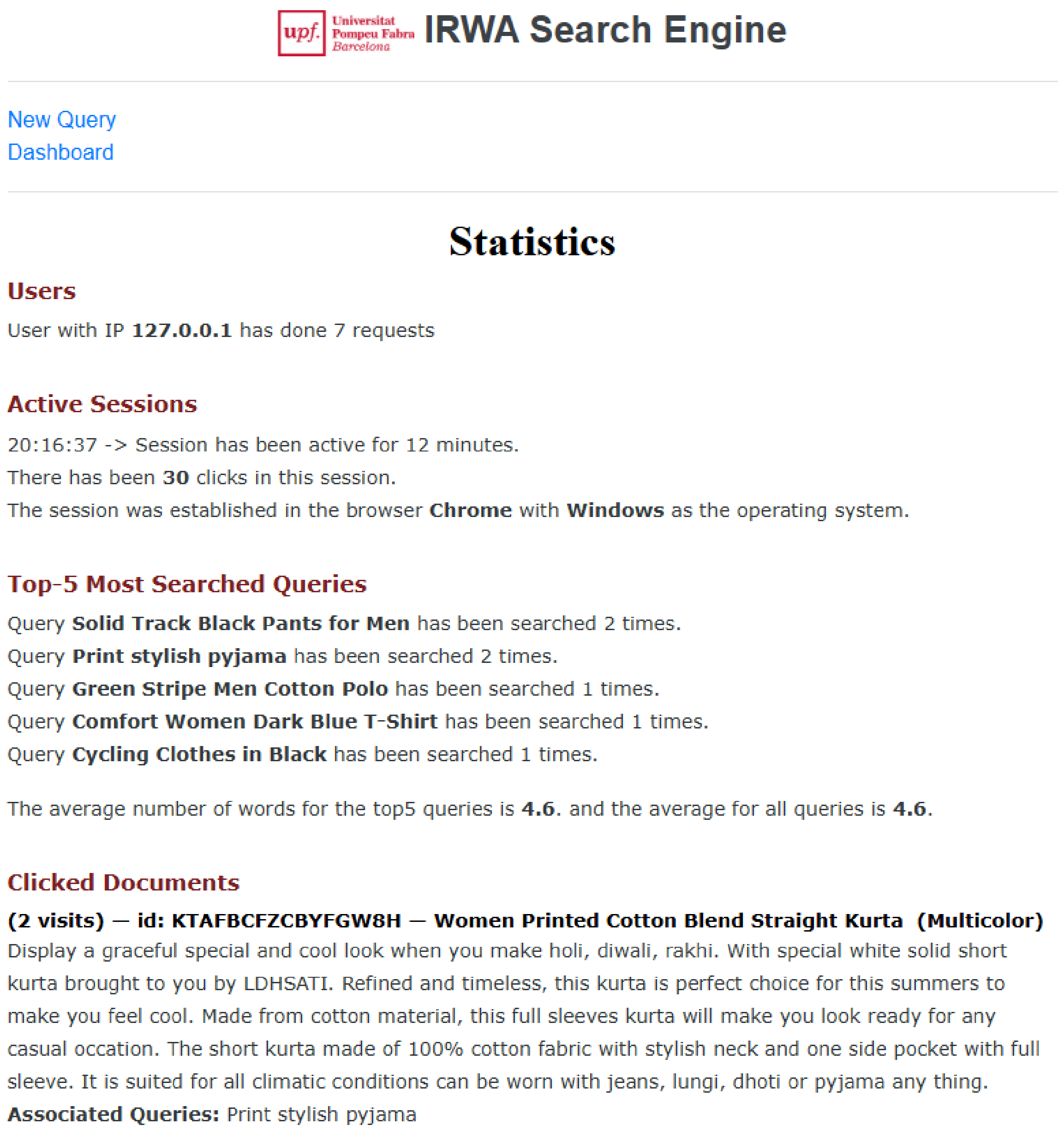
Query **#2**: "Green Stripe Men Cotton Polo" has been searched 1 times.

Query **#3**: "Comfort Women Dark Blue T-Shirt" has been searched 1 times.

Query **#4**: "Print stylish pyjama" has been searched 2 times.

Query **#5**: "Cycling Clothes in Black" has been searched 1 times.

Moreover, we provide an image of the statistics, so that you can visualize the different information retrieved from the users, the active sessions, the searched queries and the visited documents.



Statistics

Users

User with IP **127.0.0.1** has done 7 requests

Active Sessions

20:16:37 -> Session has been active for 12 minutes.
There has been **30** clicks in this session.
The session was established in the browser **Chrome** with **Windows** as the operating system.

Top-5 Most Searched Queries

Query **Solid Track Black Pants for Men** has been searched 2 times.
Query **Print stylish pyjama** has been searched 2 times.
Query **Green Stripe Men Cotton Polo** has been searched 1 times.
Query **Comfort Women Dark Blue T-Shirt** has been searched 1 times.
Query **Cycling Clothes in Black** has been searched 1 times.

The average number of words for the top5 queries is **4.6**. and the average for all queries is **4.6**.

Clicked Documents

(2 visits) — id: KTAFBCFZCBYFGW8H — Women Printed Cotton Blend Straight Kurta (Multicolor)
Display a graceful special and cool look when you make holi, diwali, rakhi. With special white solid short kurta brought to you by LDHSATI. Refined and timeless, this kurta is perfect choice for this summers to make you feel cool. Made from cotton material, this full sleeves kurta will make you look ready for any casual occation. The short kurta made of 100% cotton fabric with stylish neck and one side pocket with full sleeve. It is suited for all climatic conditions can be worn with jeans, lungi, dhoti or pyjama any thing.
Associated Queries: Print stylish pyjama