Università degli studi di Milano-Bicocca

Information Retrieval

Final Project Summary Report

Search Engine for Medical IR

Authors:
Elio Gargiulo - 869184
Cristian Piacente - 866020

January 27, 2025

Contents

1 Introduction **2**

1.1 The main approach 2

2 Analysis of the Data **2**

2.1 Documents and Queries 2

2.2 Relevance Judgements 3

3 Experiments and Evaluation **4**

3.1 Experiments 4

3.2 Results and Considerations 4

4 Improvements to the performance **5**

4.1 Query Expansion 5

5 Conclusions & Future Works **5**

1 Introduction

The goal of this project is to develop a search engine capable of retrieving relevant answers to user queries expressed in natural language. The queries are natural health-related questions harvested from the NutritionFacts.org website.

1.1 The main approach

The project has been developed on **Google Colab**. This document is structured as follows:

2. **Analysis of the Data**: The first part of the project is focused on importing the documents, queries and relevance judgements and analyzing their distributions, sizes and particular signs.

3. **Experiments and Evaluation**: The second part is focused on Retrieval Pipelines and the evaluation of different experiments using TF-IDF and BM25. PorterStemmer and SnowballStemmer have been used as Stemmers, while the indexing has been done on text, title and both title and text of the documents.

4. **Improvements to the performance**: The third part is focused on trying to improve the results obtained in the second part, using the Query Expansion techniques provided by PyTerrier Query Rewriting and Expansion. Besides PyTerrier's functions, the SpaCy library has been tested, which uses Word Embeddings for expanding the Queries.

5. **Conclusions & Future Work**: The last part is focused on collecting all the results and giving final considerations and future work about the project.

2 Analysis of the Data

Before proceeding with the EDA, the data need to be processed with these steps:

- **Tokenization**: text is split into tokens.
- **Normalization**: words are normalized by removing punctuation, converting to lowercase and excluding stop words.
- **Stemming**: words are reduced to their root form by removing suffixes, often resulting in non-dictionary words.
- **Lemmatization**: words are converted to their base form (lemma) based on their meaning, ensuring they are valid dictionary words.

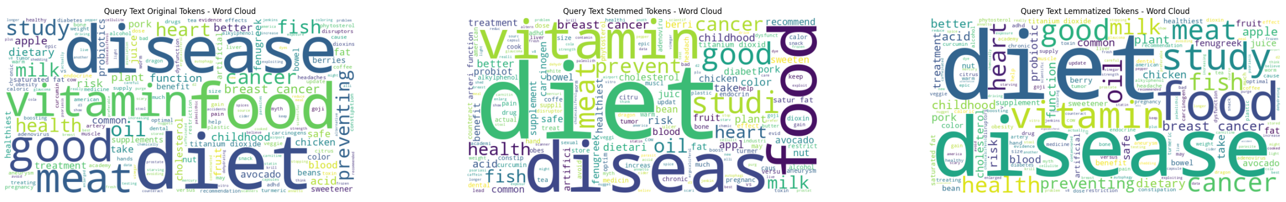
2.1 Documents and Queries

For both **Documents (abstracts and titles)** and **Queries (text)**, the analysis begins by inspecting the tokens, where the impact of text processing is observed, along with the term count for each step (after tokenization, stemming, and lemmatization). A clear way to visualize the changes at each processing step is by comparing **Vocabulary Sizes**. For instance, in document text (abstract), the original typically contains more unique tokens, while stemming reduces the count by collapsing words to their stems.

Table 1: Vocabulary size and differences between original, stemmed, and lemmatized texts.

Type (Abstract)	Vocabulary Size	vs Original	vs Stemmed	vs Lemmatization
Original	24283	0	7018	2266
Stemmed	17265	-7018	0	-4752
Lemmatized	22017	-2266	4752	0

The analysis then focuses on the **length and distributions** of each processed text, presenting plots like histograms (indicating how many documents contain a specific number of terms) and word clouds, as well as inspecting some outliers by looking at the upper and lower tails of the distributions. For both **Document Text** and **Query Text**, the histogram plots exhibit a normal distribution trend with a long tail on the right, whereas for **Document Titles**, the distribution is more balanced. **Word clouds**, in contrast, highlight the most frequent words in the text. For example, queries yield the following results:



(a) Queries - Original (b) Queries - Stemmed (c) Queries - Lemmatized

Figure 1: Word Clouds for Queries Text

The processing visibly alters words in each word cloud (e.g., 'diseas' from Stemming), but their frequencies remain consistent, as the same stems remain the most frequent across all three word clouds. **Outliers** are mostly very long or short texts, both regarding documents and queries, where, for example, long document titles are similar to an abstract (which is the text) or short titles consisting of just one word. Some curious results appear while analyzing the lower tail of the query texts, which show how the tokenization completely erases the query (empty list).

Table 2: Shortest Queries considered as Outliers, where pre-processing ignores the words

Query Text	Query ID	Original	Stemmed	Lemmatization
African-American	PLAIN-499	[]	[]	[]
IGF-1	PLAIN-1398	[]	[]	[]

2.2 Relevance Judgements

The analysis on **relevance judgements** focuses on observing the **distributions and particular cases**, i.e. minimum and maximum number of documents retrieved. The distributions show that there are two distinct labels (relevance scores) which represent the **levels of relevance** (thresholds), with 1 being relevant (95.3% of the labels distribution) and 2 possibly being highly relevant. The **QRels distribution** gives an idea of min/max number of documents retrieved by looking at the upper and lower tails, like the outliers analyzed previously.

Table 3: Examples of query-docs pairs that have few/many relevance judgments

Query ID	Number of Relevant Documents	Doc ID (Example)
PLAIN-681	1	MED-5017
PLAIN-660	475	MED-2509, MED-1374...

3 Experiments and Evaluation

3.1 Experiments

The goal of the experiments is to evaluate the retrieval performance of **BM25** and **TF-IDF** models using different fields for indexing: **Only Document Titles**, **Only Document Texts**, and **Both Titles and Texts**.
A configuration consists of one or more fields, one stemmer and one model (BM25 or TF-IDF). Two stemmers were considered, which are **PorterStemmer** and **SnowballStemmer** (the last one is an improved version with also multiple languages support).
The experiments were conducted using PyTerrier, with the following evaluation metrics: **P@10**, **R@10**, **MAP** (Mean Average Precision), and **NDCG** (Normalized Discounted Cumulative Gain). The retrieval models were used to rank documents by their relevance to the provided queries. Each model was evaluated with respect to the different configurations to analyze performance trends. In particular, in this task **high precision is needed** because it ensures that out of the retrieved documents there are mostly relevant ones, which is a critical aspect in the medical domain for avoiding misinformation.

3.2 Results and Considerations

The following table is a summary of every configuration.
Only SnowballStemmer is considered because the metrics are the same as PorterStemmer.

Table 4: Performance comparison using SnowballStemmer.

Indexing Type	**Model**	**P@10**	**R@10**	**MAP**	**NDCG**
Only Documents Titles	BM25	0.169	0.109	0.099	0.203
	TF-IDF	0.168	0.109	0.098	0.203
Only Documents Text	BM25	0.226	0.145	0.146	0.295
	TF-IDF	0.227	0.146	0.146	0.295
Both Documents Titles and Text	BM25	0.232	0.150	0.149	0.298
	TF-IDF	0.231	0.148	0.148	0.298

Using **both titles and texts** generally resulted in better metrics compared to considering document titles or texts alone.
Additionally, an in-depth analysis was performed to analyze the queries. **P@10** was calculated for each query and the results show that we have a variability from 1.0 (e.g. "Infectobesity Adenovirus 36 and Childhood Obesity", "beans") to 0.0 (e.g. "lard").
The exact query texts in the ranking depend on the configuration.
Furthermore, the query-document retrieved relevant pairs that actually aren't relevant (i.e. not in the given qrels) and the opposite case were analyzed for comparison, with the first case presenting a lot more results in all configurations.

4 Improvements to the performance

4.1 Query Expansion

To improve the performance shown in the previous part, the **Query expansion and rewriting** technique was performed. Four different approaches have been tested:

- **RM3 Relevance Model**
- **Bo1 Divergence**: Rewrites the query based on the occurrences of terms.
- **Kullback Leibler Divergence**
- **Word Embeddings with SpaCy**: SpaCy uses pre-trained word embeddings to convert query terms into vector representations. This project uses the normal version as a quick test, but it would be better to use "SciSpaCy", since the dataset contains highly specific terms. Therefore, it will probably perform worse than the other approaches.

Table 5: Comparison of BM25 and TF-IDF performance across various query expansion techniques.

Query Expansion Technique	Model	P@10	R@10	MAP	NDCG
Original Queries	BM25	0.232	0.150	0.148	0.298
	TF-IDF	0.231	0.148	0.148	0.298
RM3 Expansion	BM25	0.253	0.169	0.173	0.375
	TF-IDF	0.251	0.168	0.173	0.376
Bo1 Divergence Expansion	BM25	0.251	0.165	0.174	0.373
	TF-IDF	0.251	0.165	0.173	0.373
KL Divergence Expansion	BM25	0.250	0.164	0.174	0.373
	TF-IDF	0.250	0.165	0.173	0.374
spaCy Query Expansion	BM25	0.201	0.134	0.130	0.294
	TF-IDF	0.200	0.140	0.129	0.294

5 Conclusions & Future Works

In conclusion, the results obtained show little difference but constant improvements starting from the baseline to the query expansion models, except for spaCy. The average metrics are not really high but there is still room for improvements in possible future work, for example:

- Neural Re-ranking using BERT or using Pseudo-Relevance Feedback.
- Query expansion using other techniques such as LLMs or using SciSpaCy.
- Using different Models for the retrieval pipelines.