

INFORMATION RETRIEVAL

Approaches aimed at generating search engines

Marco Di Capua (878295)

Summary

[Introduction 3](#_heading=h.30j0zll)

[Collection description 3](#_heading=h.1fob9te)

[Data preprocessing 4](#_heading=h.2et92p0)

[Search engine 8](#_heading=h.tyjcwt)

[Results evaluation 9](#_heading=h.lnxbz9)

# 

# Introduction

This project’s scope is about implementing different models of information retrieval to gather documents from a collection given a set of queries. The need for information retrieval (IR) has risen due to the continuous increase of documents available on the web. Users need to retrieve these documents based on their needs, usually expressed through a query. With this project, I will experiment with different approaches to create a model of information retrieval that will be applied to different versions of the same document collection that has been pre-processed manually in different ways. I will also be using different weighting methods such as TF-IDF, BM25 in order to be able to create different experiments which will have different effectiveness. Finally, I will be comparing the results obtained by these different approaches and evaluate the performances of each.

# Collection description

The dataset that will be used for this project contains documents related to the field of computer science. The dataset contains a total of 480,968 documents and 55,277 queries. Both queries and documents are stored in a .jsonl file, with different data on each line.

The main information for each document that will be used in this project are:

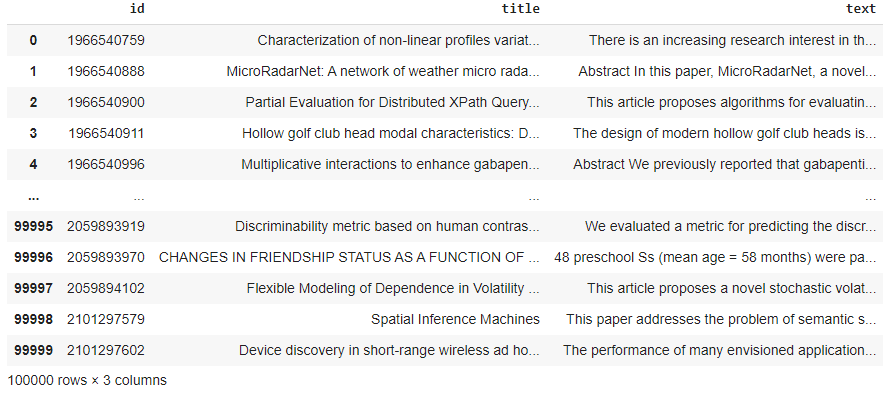
* **id**: unique numeric identifier of the document
* **title**: title of the document
* **text**: an abstract of the document

The main information for each query that will be used in this project are:

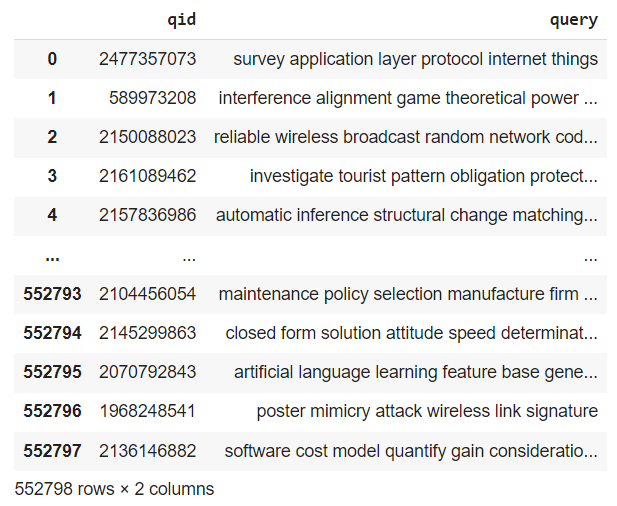
* **qid**: unique numeric identifier of the query
* **query**: string that represent the text of the query itself

Before conducting data analysis and preprocessing, it should be noted that I limited the dataset to only the first 100,000 documents to avoid RAM overload during resource-intensive processing tasks due to the large size of the document collection. The experiments that will be conducted will use both an integral version of the documents dataset that has not been pre-processed and the resized version that has been pre-processed. The following image shows an example of the information contained in each document and query that will be used in the experiment:

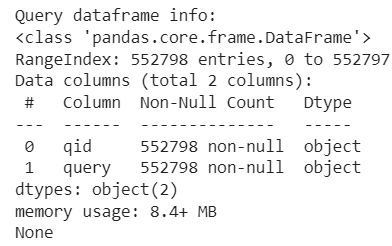
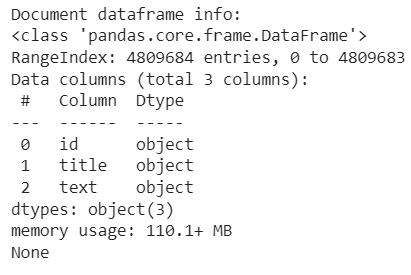
Document dataset:



Query dataset:



Taking a closer look at both of these dataframe we can say that all records contained in the dataset are not null:



# Data preprocessing

After a brief description of the content of both datasets, it is easy to say that some preprocessing tasks are needed before proceeding with the creation of the IR model. By looking at the previous representation of some lines of the document's dataframe, we can already tell that some tasks, such as removing punctuation marks that aren’t needed for the creation of the IR model and setting the text to lower case, are steps very much needed. All the following steps were mainly conducted using the NLTK library. The “cleaning” of the original “**title**” and “**text**” columns altered the columns’ content as follows:

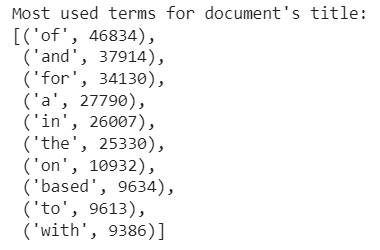
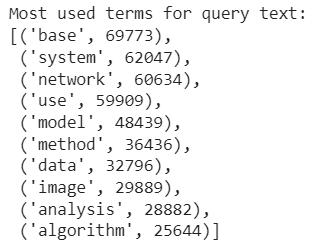
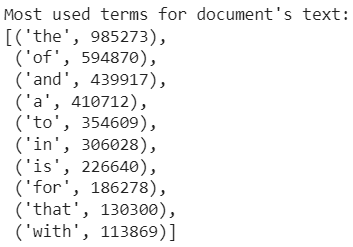
* All the contracted forms typical of English language were removed and brought back to their extended form
* All the text has been set to lowercase
* All the numbers and punctuations marks were removed
* All web page links were removed
* All characters that differ from a capital or lowercase letter have been removed

Since the dataset should contain texts regarding academic paper publications, tasks such as removing whitespaces, removing duplicate characters, removing emoji and spelling corrections were not deemed necessary, given the tendency of these kinds of documents not to have these types of errors.

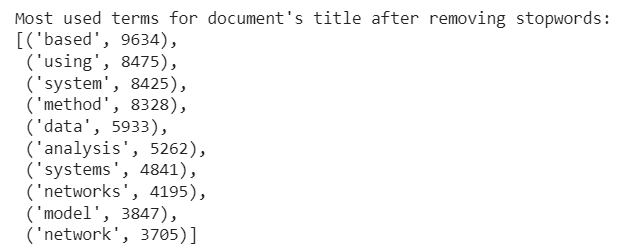
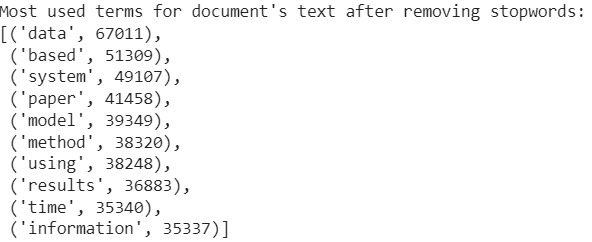
Moving further into the preprocessing task, the “cleaned” text has been tokenized, and some statistics have been extracted from these tokens in order to make some useful considerations in the future steps:

* The average number of tokens per document title is: 10.43528
* The average number of tokens per document text is: 146.1561
* The average number of tokens per query text is: 7.92927

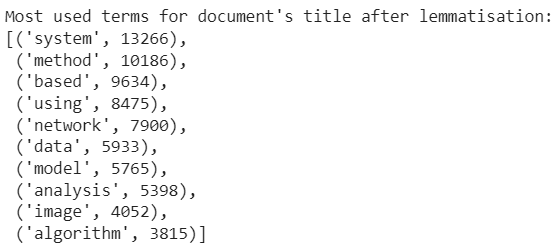
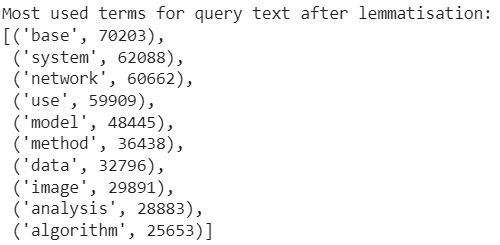
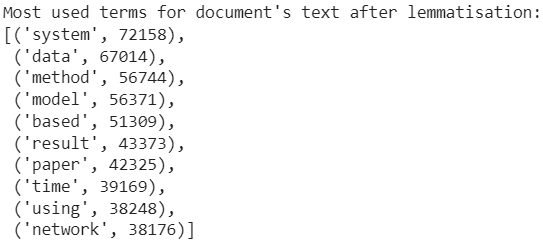
More information can be obtained from the representation of the most used terms for the above-mentioned texts:

The information revealed by the representation of the most used terms denotes the need of further processing of the data. Both the document title and text still contain some words that aren’t useful for the creation of an IR model, such as articles, conjunctions and prepositions. In order to remove these words, it has been applied a **stopwords** removal step with the following results:

The results obtained by removing the stopwords are quite satisfactory, but the most used terms still contain some words with the same meaning but different plurals, such as “***system”*** and “***systems”*** or “***network”*** and “***networks”***. To bring these words back to an inflected form so that they can be analyzed as a single word, a **lemmatization** step was applied using a WordNet lemmatizer. The following results were obtained:

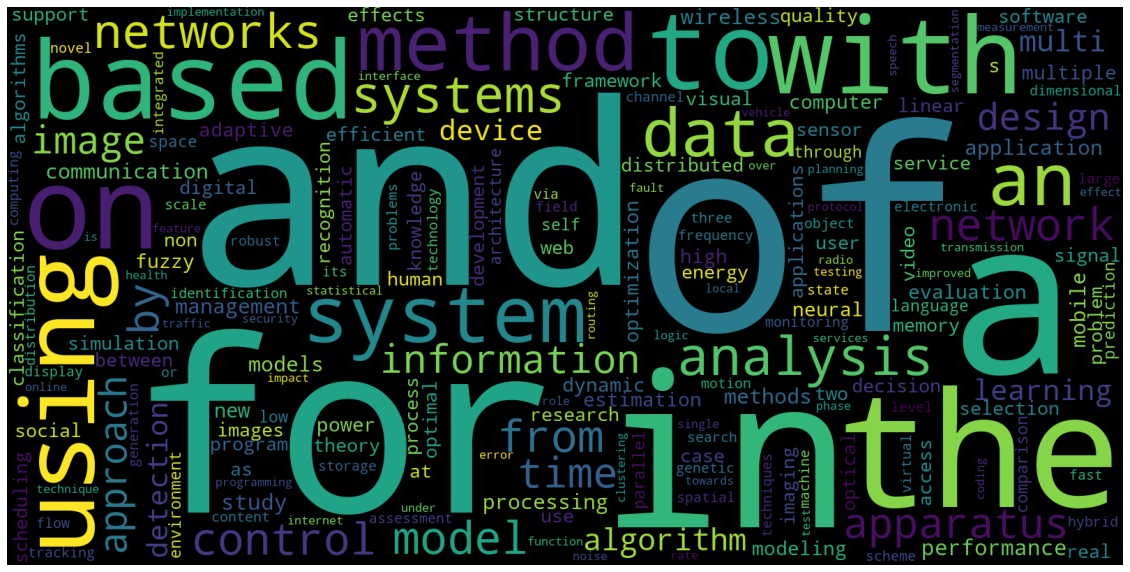
 

After all the steps of preprocessing the average number of tokens has also been changed as follows:

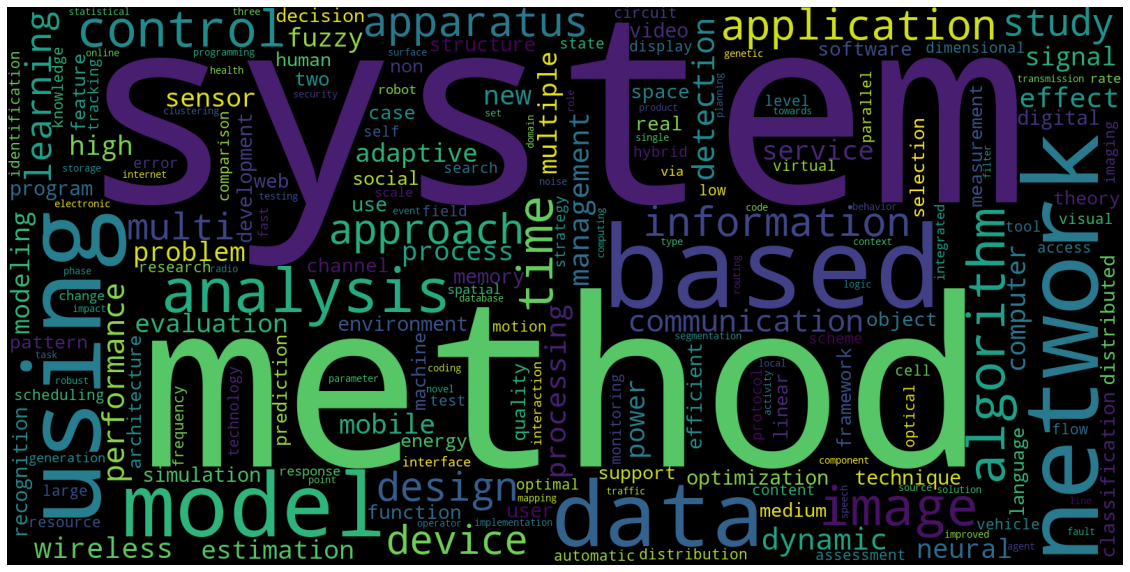
* The average number of tokens per document title is: 7.80516
* The average number of tokens per document text is: 89.50897
* The average number of tokens per query text is: 7.927476

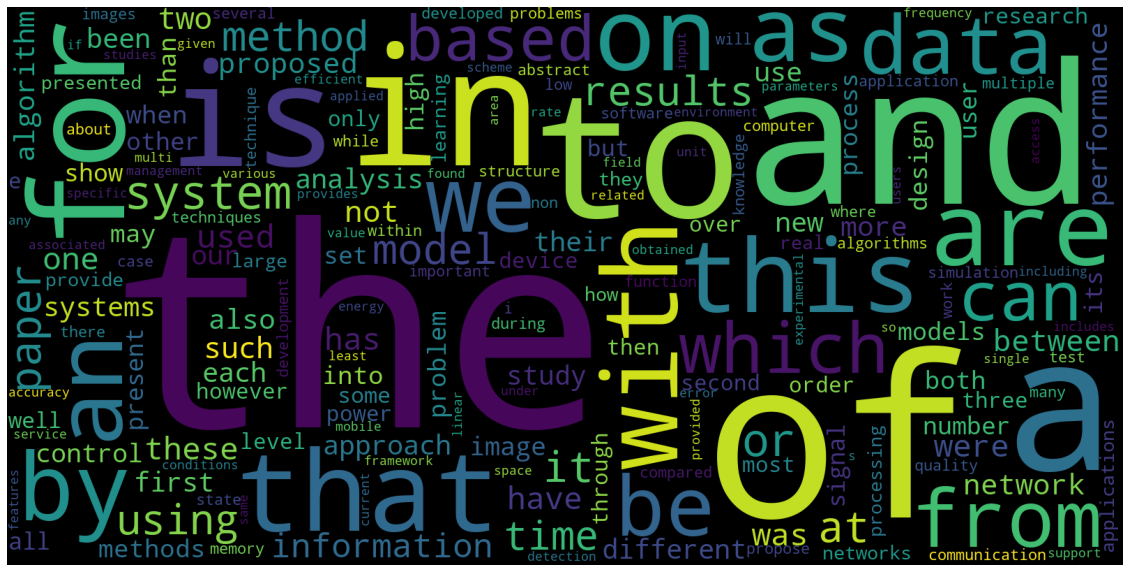
The average number of tokens has significantly decreased for both titles and texts of the documents which means that the preprocessing steps were much needed. On the other hand, the average number of tokens per query remained relatively unchanged due to an already adequate state of the query text in the original form.

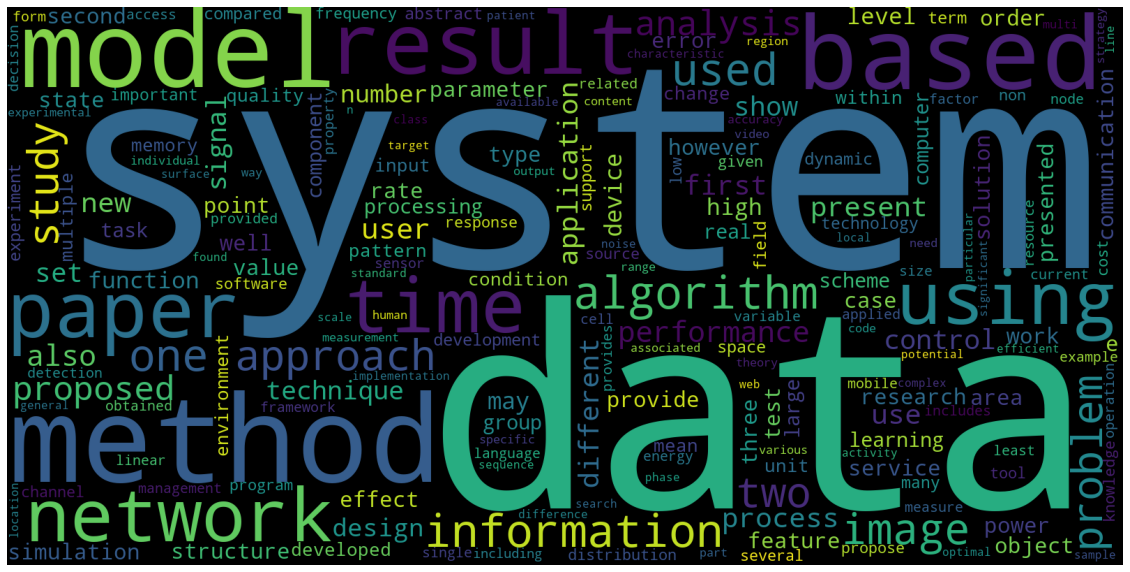
To conclude the preprocessing phase, some **WordCloud** graphics were produced to show the difference between unprocessed text and pre-processed text.

WordCloud of non-processed document’s title: 

WordCloud of pre-processed document’s title:



WordCloud of non-processed document’s text: 

WordCloud of pre-processed document’s text: 

# Search engine

The substantial preprocessing phase prepared the data to be used efficiently for the creation of an IR model. The library used for this step is PyTerrier.

In order to run a PyTerrier experiment two new dataframe have been created to match the experiment’s requirements: **df\_wnlem** and **dfquery\_wnlem**.

The columns contained in the df\_wnlem dataframe are:

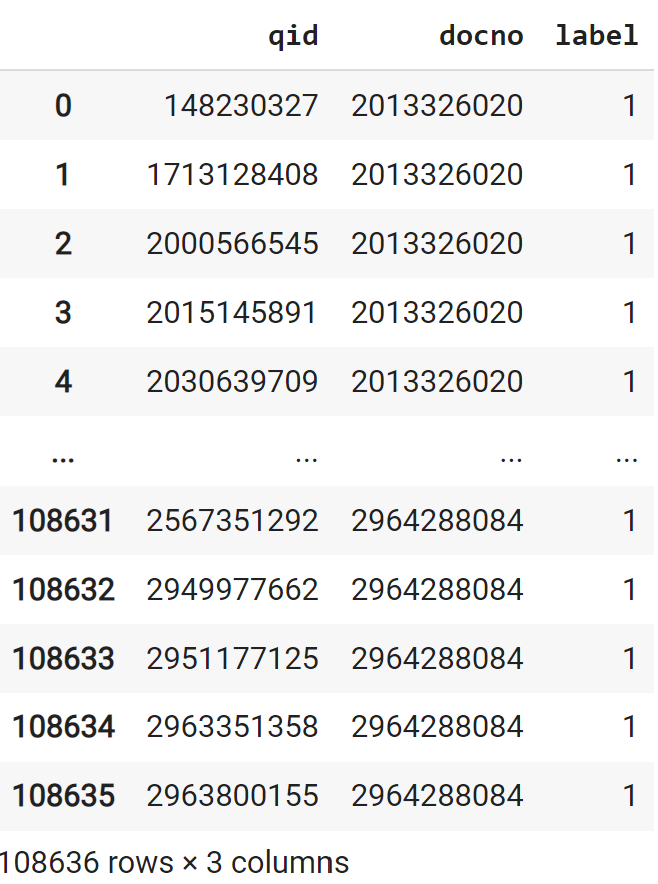
* **docno**: unique identifier of the document
* **title**: title of the document
* **text**: an abstract of the document

The columns contained in the dfquery\_wnlem dataframe are:

* **qid**: unique identifier of the query
* **query**: text of the query itself

The last dataframe needed in order to proceed with the experiment is the **qrels** dataframe. Given inside the *“test”* folder as a .json file, the qrels dataframe contains 108636 rows and 3 columns containing:

* **qid**: unique identifier of the query
* **docno**: unique identifier of the relevant document
* **label**: relevance score of the document (always set as 1)



Before conducting the experiments there were created three indexes: one that will be using the titles of the resized documents dataset, another that will be using the text of the resized documents dataset and the last one that will be using the texts of the entire documents collection: **index\_wnlem\_title** , **index\_wnlem\_text** and **index\_full\_text.** The first two indexes will use a pre-processed version of the document’s title and text without stopwords and after lemmatization.

Two different search models have been created for each of these indexes:

* Term frequency–inverse document frequency (**TF-IDF**) model
* Okapi **BM25** model

A total of 6 experiments have been conducted using the before-mentioned indexes of the document’s title and text, all the models before mentioned, the qrels given with the collection and the pre-processed query dataframe. All the experiments have been evaluated with the same metrics in order to be able to compare the results. These metrics are:

* **Average Precision**
* **Precision @ 10**
* **Precision @ 20**

# Results evaluation

Results obtained from the experiments that used both the title and text of the resized version of the original dataframe were unsatisfactory. The experiments results outcome are as follows:

Using the documents title index:

* **TF-IDF**: AP:0.0, P@10: 0.0, P@20: 0.0
* **BM25**: AP:0.0, P@10: 0.0, P@20: 0.0

Using the documents text index:

* **TF-IDF**: AP:0.0, P@10: 0.0, P@20: 0.0
* **BM25**: AP:0.0, P@10: 0.0, P@20: 0.0

It's clear that something has gone wrong during the setup of parameters that were given to the experiment, probably something related to the creation of the TF-IDF and BM25 models.

Both the experiments done using the documents full collection didn’t produce any outcome at all, due to the size of the collection itself and to the lack of resources available causing a RAM overload after several hours of processing.

The unsuccessful results of the experiments conducted were a clear sign that working on an information retrieval search engine is particularly difficult, not only because of the programming skills needed but also for the resources needed to elaborate such a huge collection of data.