Information Retrieval

2021-2022

GOVONE Morgane – JACQUET Gaëtan

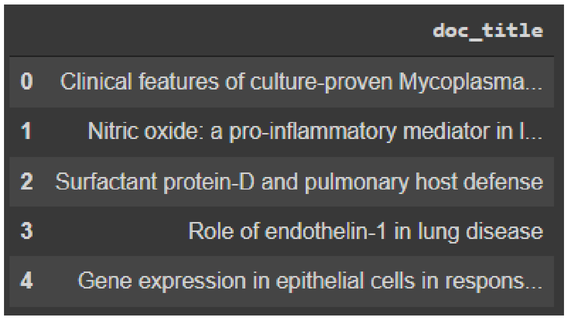
Project COVID-19 OPEN RESEARCH DATASET CHALLENGE (CORD-19)

Introduction

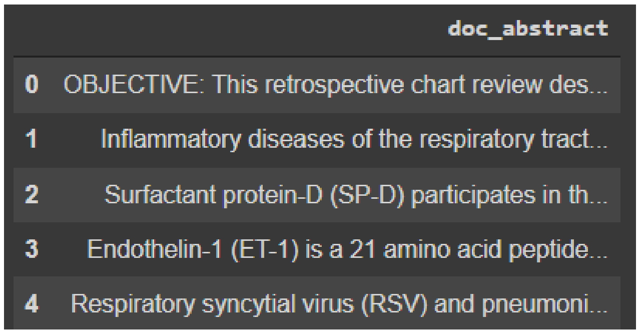
We chose the subject "COVID-19 OPEN RESEARCH DATASET CHALLENGE (CORD-19)". For this one, we have several tasks to do. The most important is to obtain the best answer to each question and ideally try to improve this research as much as possible.
Initially, we had a data set on the coronavirus. In order to carry out the tasks mentioned above, we first studied the available documents and then focused on the requests we have in three formats.
For this we have made a whole identical process on these, which we will be detailed in the following sections, so that they are readable and can be compared between each other.
Then we decided to run these experiments in order to get results.
These results will be a list of documents classified according to their rank, we can also characterize it according to their interests in relation to the requested request.
These results will be evaluated using several measurement systems. Our goal is to have the highest possible values
Finally, we will try to improve our model in order to increase the results of previous evaluations.

Test Collection - Analysis of Queries and Documents

In the elements we had to download we will first focus on the documents. What is important are the parts concerning titles and abstracts, which is why we will isolate them.
In the code provided in addition to this report, a list is obtained.
In order to better observe them for this report, we have displayed them as data frames :

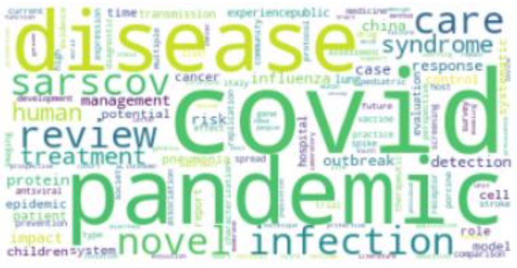


	doc_title
0	Clinical features of culture-proven Mycoplasma...
1	Nitric oxide: a pro-inflammatory mediator in l...
2	Surfactant protein-D and pulmonary host defense
3	Role of endothelin-1 in lung disease
4	Gene expression in epithelial cells in respons...



	doc_abstract
0	OBJECTIVE: This retrospective chart review des...
1	Inflammatory diseases of the respiratory tract...
2	Surfactant protein-D (SP-D) participates in th...
3	Endothelin-1 (ET-1) is a 21 amino acid peptide...
4	Respiratory syncytial virus (RSV) and pneumoni...

After the pre process (detailed in the next part) that we will apply to these two lists, we have created word clouds.
The sizes of each word are proportional to the frequencies of appearance of the latter.

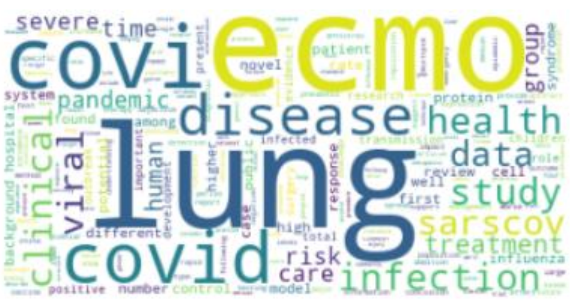


First of all the one associated with the titles:

We have chosen to consider all the words of the dataset in order to really see the most frequent.

The result is that the three main words are “covid”, “pandemic” and “disease”

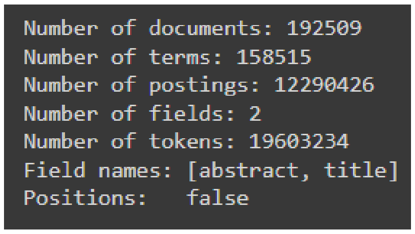
The word cloud that we get for the abstract is the following:



In the latter also we select the totality of possible words. The most common words are “ecmo”, “lung” and “covid”.

The word “covi” is present, this is due to the fact that we did not apply a correction function to our dataset. The function is well programmed but being too long at execution we preferred not to use it.

We can also look at the statistics related to these documents. And we obtain :



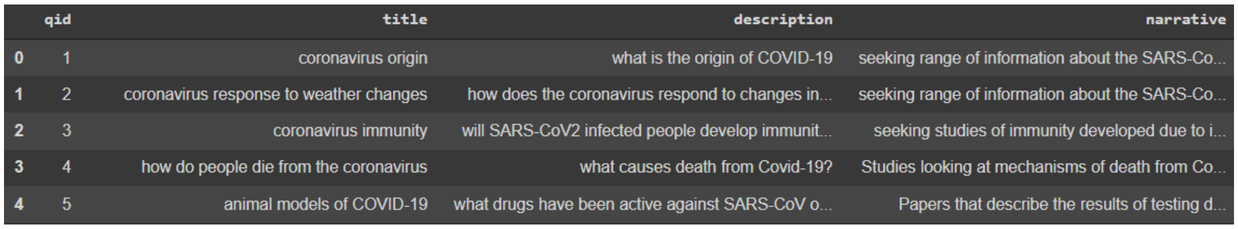
```
Number of documents: 192509
Number of terms: 158515
Number of postings: 12290426
Number of fields: 2
Number of tokens: 19603234
Field names: [abstract, title]
Positions:   false
```

After describing the data we had for the documents, we will now focus on the second important part which deals with queries.

In our dataset we can extract three types of requests :

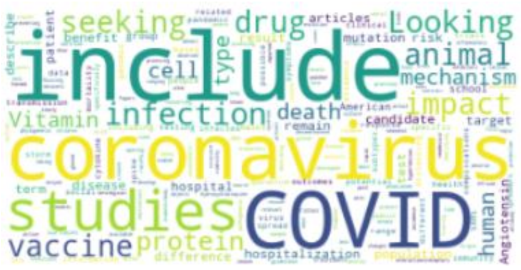
- Adhoc query : they are associated with the “title” column
- Descriptive query : they correspond to the 'description' column
- Narrative query : these are the ‘narrative’ column

We have 50 queries by type. We can in the following picture the first five.



	qid	title	description	narrative
0	1	coronavirus origin	what is the origin of COVID-19	seeking range of information about the SARS-Co...
1	2	coronavirus response to weather changes	how does the coronavirus respond to changes in...	seeking range of information about the SARS-Co...
2	3	coronavirus immunity	will SARS-CoV2 infected people develop immunit...	seeking studies of immunity developed due to i...
3	4	how do people die from the coronavirus	what causes death from Covid-19?	Studies looking at mechanisms of death from Co...
4	5	animal models of COVID-19	what drugs have been active against SARS-CoV o...	Papers that describe the results of testing d...

We have not applied a pre process to the queries for the moment because we will do it in the next part with a complete pre process. We still get a word cloud for all queries :



Of all possible words, the most common are “coronavirus”, “include” and “COVID”.

We also generated a word cloud for each type of request, we can find these in the colab attached to this report.

Search Engine - Basic Search

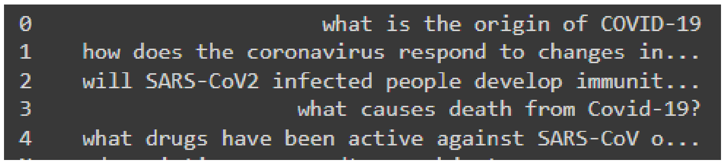
Now let's focus on our search engine. For this we decided to propose a simple search machine divided into two parts. On the one hand the queries and on the other the documents that we will then join together.

Start with the queries.
As we said before we have three categories of query which are "adhoc", "descriptive", and "narrative".
We apply exactly the same tasks for each of them.

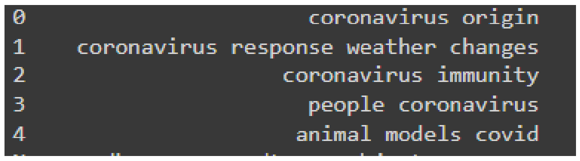
We start by creating a dataframe will be more convenient for us to choose the type of query we want as well as having a correct result display.
The first step was to perform a pre process function.
process_query(type_query) contains two functions necessary for this :

1. Pre_process(dataset) which contains the majority of the pre processing steps. For example, we remove punctuation, repetitions should not take place, as well as numbering. In addition, we put all the letters in lowercase.
2. Token_per_sent(dataset) which allows to tokenise the dataset. In the end we get all tokens but by query and not a list with all tokens present. In this step we remove the stop_words which is an important step because these words are not relevant for queries

We can see the result for the descriptive query :

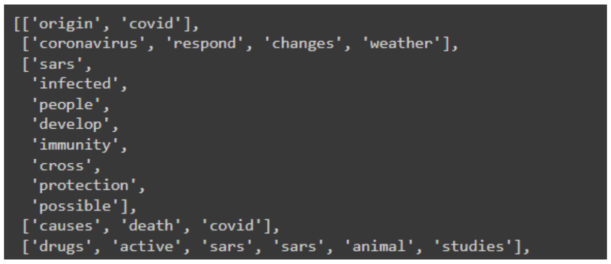


```
0                       what is the origin of COVID-19
1    how does the coronavirus respond to changes in...
2    will SARS-CoV2 infected people develop immunit...
3                       what causes death from Covid-19?
4    what drugs have been active against SARS-CoV o...
```



```
0                       coronavirus origin
1    coronavirus response weather changes
2                    coronavirus immunity
3                      people coronavirus
4                     animal models covid
```

Example of the query tokenized :



```
[['origin', 'covid'],
 ['coronavirus', 'respond', 'changes', 'weather'],
 ['sars',
  'infected',
  'people',
  'develop',
  'immunity',
  'cross',
  'protection',
  'possible'],
 ['causes', 'death', 'covid'],
 ['drugs', 'active', 'sars', 'sars', 'animal', 'studies'],
```

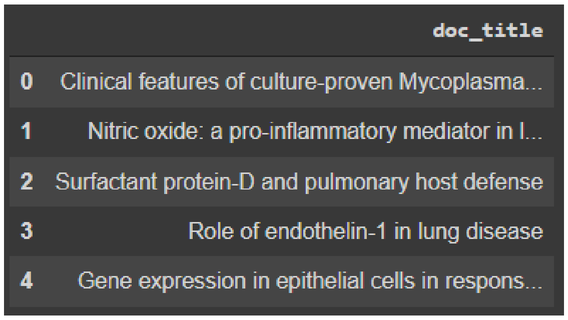
This example only covers the descriptive_query but we get the same results with our two other query types.

We apply the same pre process for the documents the only difference is the writing. In these cases we use only two functions:

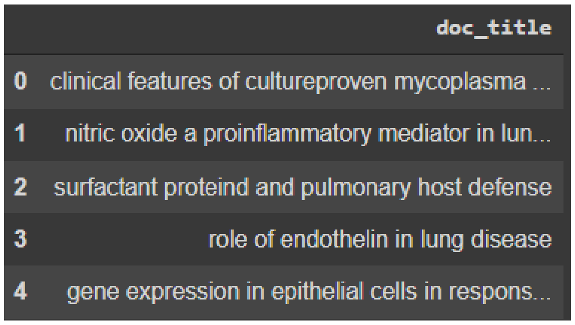
1. Pre_process(dataset) which is the same function defined previously
2. Word(dataset) : this function makes it possible to apply tokenization on our dataset and thus remove the words not relevant for our request (including stop words). Thanks to this function we get three lists:

- Words_list which is the list containing all the tokens of the documents without the stop words
- Token which is the list containing all tokens
- Token_list which is the list of tokens for each sentence of the dataset

After this process, we obtain the following result :



	doc_title
0	Clinical features of culture-proven Mycoplasma...
1	Nitric oxide: a pro-inflammatory mediator in l...
2	Surfactant protein-D and pulmonary host defense
3	Role of endothelin-1 in lung disease
4	Gene expression in epithelial cells in respons...

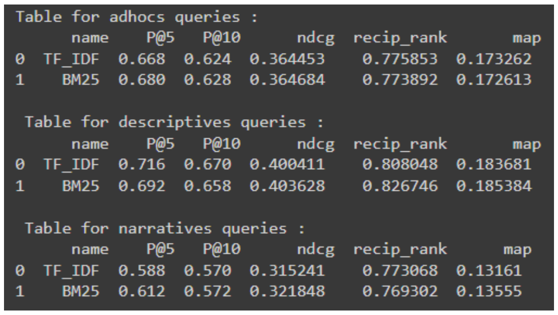


	doc_title
0	clinical features of cultureproven mycoplasma ...
1	nitric oxide a proinflammatory mediator in lun...
2	surfactant proteind and pulmonary host defense
3	role of endothelin in lung disease
4	gene expression in epithelial cells in respons...

After performing our pre-processing for documents and queries now we have to join them and evaluate if the documents we get match well and are relevant to our queries. Finally for the rest of this project in order to make a good evaluation, we will focus on the abstract part of our documents.

In order to perform this evaluation first extract important elements which are the qrels. To perform the experiment, we use the measurement given by the PyTerrier library.
The purpose of this step is to calculate some measurements such as P@5 (precision at 5), P@10 (precision at 10), NDCG (Normalize Discount Cumulative Gain), Reciprocal Rank and MAP (Average Precision).

First, we need to build a weighting model, for this we use tf_idf and BM25.
We need to implement a table with to the measures we have cited for each query and implement the evaluation. We get this table:



```
Table for adhocs queries :
     name    P@5   P@10      ndcg  recip_rank       map
0  TF_IDF  0.668  0.624  0.364453    0.775853  0.173262
1    BM25  0.680  0.628  0.364684    0.773892  0.172613

 Table for descriptives queries :
     name    P@5   P@10      ndcg  recip_rank       map
0  TF_IDF  0.716  0.670  0.400411    0.808048  0.183681
1    BM25  0.692  0.658  0.403628    0.826746  0.185384

 Table for narratives queries :
     name    P@5   P@10      ndcg  recip_rank      map
0  TF_IDF  0.588  0.570  0.315241    0.773068  0.13161
1    BM25  0.612  0.572  0.321848    0.769302  0.13555
```

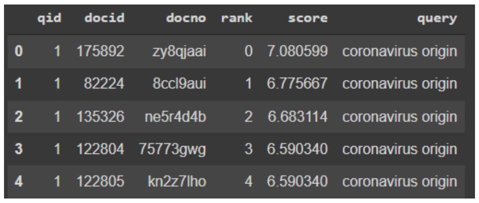
We can observe that between the 3 types of queries the results are similar. However, according to the measures, the results are not the same. We can see that the p@5, P@10 and recip_rank measurements are correct or high [0.6: 0.7]. On the other hand, we have the ndcg and map measures that are lower. We can say that they are less effective.

The results are slightly better for descriptive query than adhoc query, this can be understood and explained by the fact that the latter are more developed and this makes a real distinction between adhoc and descriptive.
For narrative queries we see a decline in the level of evaluation. This is due to the fact that these queries

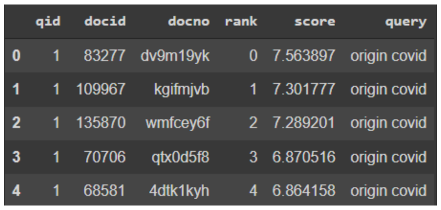
consist of a large amount of information which leads to a decrease because unnecessary information may be present. In the end the precision is less good than the two previous queries.

Moreover, we can see that in this case, we do not observe any significant difference between tf_idf and BM25.

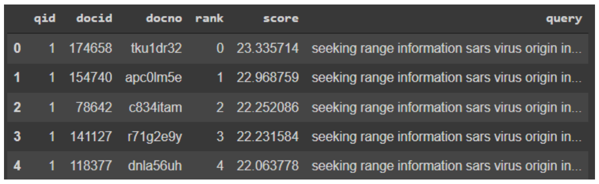
We also wanted to see the first 5 documents that are the most relevant for our three types of queries:



	qid	docid	docno	rank	score	query
0	1	175892	zy8qjaai	0	7.080599	coronavirus origin
1	1	82224	8ccl9aui	1	6.775667	coronavirus origin
2	1	135326	ne5r4d4b	2	6.683114	coronavirus origin
3	1	122804	75773gwg	3	6.590340	coronavirus origin
4	1	122805	kn2z7lho	4	6.590340	coronavirus origin



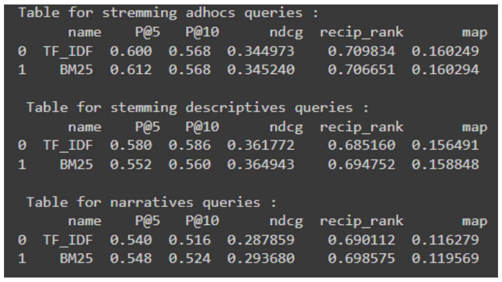
	qid	docid	docno	rank	score	query
0	1	83277	dv9m19yk	0	7.563897	origin covid
1	1	109967	kgifmjvb	1	7.301777	origin covid
2	1	135870	wmfcey6f	2	7.289201	origin covid
3	1	70706	qtx0d5f8	3	6.870516	origin covid
4	1	68581	4dtk1kyh	4	6.864158	origin covid



	qid	docid	docno	rank	score	query
0	1	174658	tku1dr32	0	23.335714	seeking range information sars virus origin in...
1	1	154740	apc0lm5e	1	22.968759	seeking range information sars virus origin in...
2	1	78642	c834itam	2	22.252086	seeking range information sars virus origin in...
3	1	141127	r71g2e9y	3	22.231584	seeking range information sars virus origin in...
4	1	118377	dnla56uh	4	22.063778	seeking range information sars virus origin in...

We observe that the proposed documents are not the same for adhoc, descriptive and narrative queries. Ideally this would have been the case but the adhoc was shorter than the descriptive or narrative, we can tell ourselves that with more information our query is more accurate and therefore we get a document that corresponds more.

We then wanted to see if the results could vary if the pre-process was modified. So we added the stemming step. This allows you to reduce each word to its initial root without worrying about the context or the meaning of sentences and words.

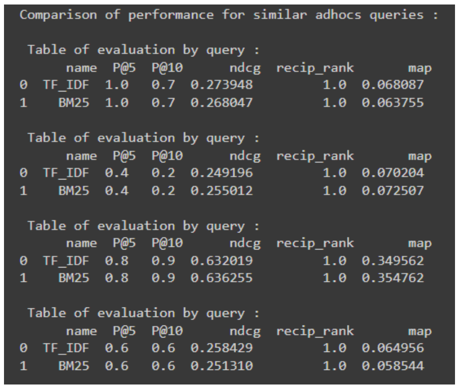


```
 Table for stremming adhocs queries :
      name    P@5   P@10      ndcg  recip_rank       map
0  TF_IDF  0.600  0.568  0.344973    0.709834  0.160249
1    BM25  0.612  0.568  0.345240    0.706651  0.160294

 Table for stemming descriptives queries :
      name    P@5   P@10      ndcg  recip_rank       map
0  TF_IDF  0.580  0.586  0.361772    0.685160  0.156491
1    BM25  0.552  0.560  0.364943    0.694752  0.158848

 Table for narratives queries :
      name    P@5   P@10      ndcg  recip_rank       map
0  TF_IDF  0.540  0.516  0.287859    0.690112  0.116279
1    BM25  0.548  0.524  0.293680    0.698575  0.119569
```

We can see that the evaluations have changed, they have all decreased. That can be explained by what we just said above. The words being reduced, the context not taken into account then the returned documents are not necessarily the most relevant in aggregate.

However the first 5 documents are the same for adhoc and descriptive queries with or without stemming and change for narrative

After this analysis it seemed interesting to see if similar queries had the same evaluation or not. In this report we will present only the example of adhocs queries. We proceeded in two steps: first we compare similar queries without stemming and then with stemming.

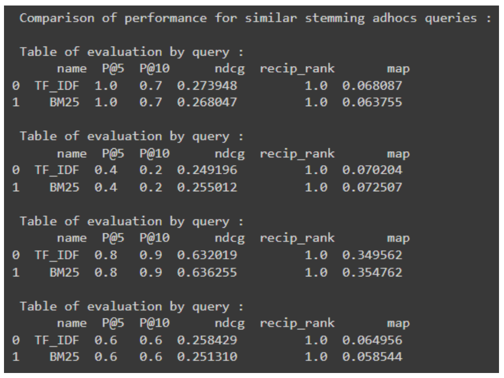


```
Comparison of performance for similar adhocs queries :

 Table of evaluation by query :
     name  P@5  P@10      ndcg  recip_rank       map
0  TF_IDF  1.0   0.7  0.273948         1.0  0.068087
1    BM25  1.0   0.7  0.268047         1.0  0.063755

 Table of evaluation by query :
     name  P@5  P@10      ndcg  recip_rank       map
0  TF_IDF  0.4   0.2  0.249196         1.0  0.070204
1    BM25  0.4   0.2  0.255012         1.0  0.072507

 Table of evaluation by query :
     name  P@5  P@10      ndcg  recip_rank       map
0  TF_IDF  0.8   0.9  0.632019         1.0  0.349562
1    BM25  0.8   0.9  0.636255         1.0  0.354762

 Table of evaluation by query :
     name  P@5  P@10      ndcg  recip_rank       map
0  TF_IDF  0.6   0.6  0.258429         1.0  0.064956
1    BM25  0.6   0.6  0.251310         1.0  0.058544
```

We thus obtain an evaluation for each query, that is to say the original (the 16th) and its 3 most similar.
We observe, for queries without stemming, that the results are almost identical for the ndcg, recip_rank and map measurements (except for one query). The results obtained for the precision measurements are really

different from one query to another.

In terms of the difference between tf_idf and BM25, we can see that BM25 is more efficient, its evaluations yield better results in the majority of cases.



```
 Comparison of performance for similar stemming adhocs queries :

 Table of evaluation by query :
      name  P@5  P@10      ndcg  recip_rank       map
0  TF_IDF  1.0   0.7  0.273948         1.0  0.068087
1    BM25  1.0   0.7  0.268047         1.0  0.063755

 Table of evaluation by query :
      name  P@5  P@10      ndcg  recip_rank       map
0  TF_IDF  0.4   0.2  0.249196         1.0  0.070204
1    BM25  0.4   0.2  0.255012         1.0  0.072507

 Table of evaluation by query :
      name  P@5  P@10      ndcg  recip_rank       map
0  TF_IDF  0.8   0.9  0.632019         1.0  0.349562
1    BM25  0.8   0.9  0.636255         1.0  0.354762

 Table of evaluation by query :
      name  P@5  P@10      ndcg  recip_rank       map
0  TF_IDF  0.6   0.6  0.258429         1.0  0.064956
1    BM25  0.6   0.6  0.251310         1.0  0.058544
```

We can see here the results obtained for the adhoc query with stemming. We see that the results are identical. We can therefore say that with or without stemming the evaluation of the performance of our search engine remains the same.

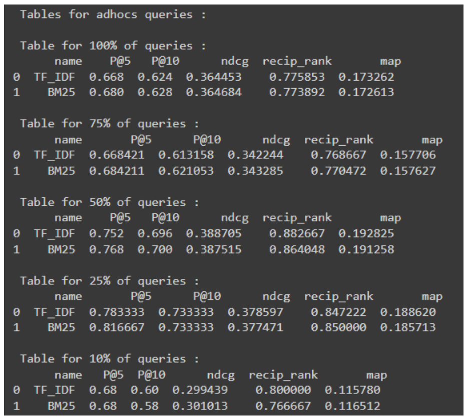
We can see the same situation for descriptive queries.

However, for narrative queries, values change whether the stemming is present or not. The presence of the stemming step reduces the evaluation, so our results are less good. This difference can perhaps be explained by the fact that these queries have a very large amount of information so the reduction of each word to its initial root can have a greater impact on the latter.

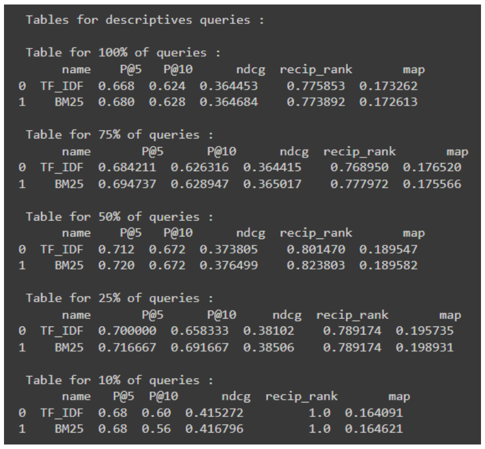
Search Engine - Advanced Search

After these rather satisfactory results we wondered if it was possible to improve our search engine or see how it evolves according to the changes. For this we had two ideas: reduce the number of queries and change the index defined at the beginning.

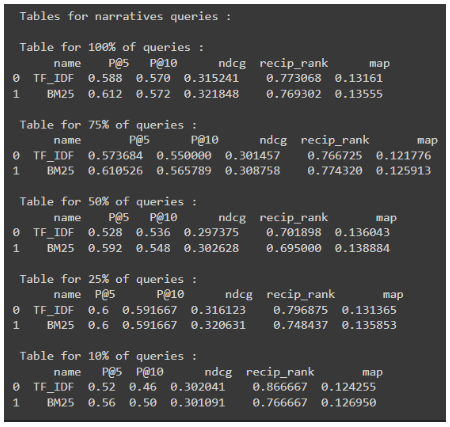
So, we have reduced our dataset of queries by several levels 75%, 50%, 25% and 10% and we get these results :



```
 Tables for adhocs queries :

 Table for 100% of queries :
     name    P@5   P@10      ndcg  recip_rank       map
0  TF_IDF  0.668  0.624  0.364453    0.775853  0.173262
1    BM25  0.680  0.628  0.364684    0.773892  0.172613

 Table for 75% of queries :
     name       P@5      P@10      ndcg  recip_rank       map
0  TF_IDF  0.668421  0.613158  0.342244    0.768667  0.157706
1    BM25  0.684211  0.621053  0.343285    0.770472  0.157627

 Table for 50% of queries :
     name    P@5   P@10      ndcg  recip_rank       map
0  TF_IDF  0.752  0.696  0.388705    0.882667  0.192825
1    BM25  0.768  0.700  0.387515    0.864048  0.191258

 Table for 25% of queries :
     name       P@5      P@10      ndcg  recip_rank       map
0  TF_IDF  0.783333  0.733333  0.378597    0.847222  0.188620
1    BM25  0.816667  0.733333  0.377471    0.850000  0.185713

 Table for 10% of queries :
     name   P@5  P@10      ndcg  recip_rank       map
0  TF_IDF  0.68  0.60  0.299439    0.800000  0.115780
1    BM25  0.68  0.58  0.301013    0.766667  0.116512
```



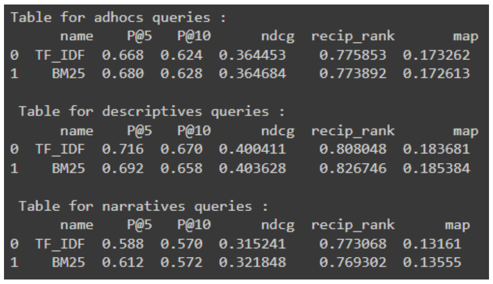
```
 Tables for descriptives queries :

 Table for 100% of queries :
     name    P@5   P@10      ndcg  recip_rank       map
0  TF_IDF  0.668  0.624  0.364453    0.775853  0.173262
1    BM25  0.680  0.628  0.364684    0.773892  0.172613

 Table for 75% of queries :
     name       P@5      P@10      ndcg  recip_rank       map
0  TF_IDF  0.684211  0.626316  0.364415    0.768950  0.176520
1    BM25  0.694737  0.628947  0.365017    0.777972  0.175566

 Table for 50% of queries :
     name    P@5   P@10      ndcg  recip_rank       map
0  TF_IDF  0.712  0.672  0.373805    0.801470  0.189547
1    BM25  0.720  0.672  0.376499    0.823803  0.189582

 Table for 25% of queries :
     name       P@5      P@10     ndcg  recip_rank       map
0  TF_IDF  0.700000  0.658333  0.38102    0.789174  0.195735
1    BM25  0.716667  0.691667  0.38506    0.789174  0.198931

 Table for 10% of queries :
     name   P@5  P@10      ndcg  recip_rank       map
0  TF_IDF  0.68  0.60  0.415272         1.0  0.164091
1    BM25  0.68  0.56  0.416796         1.0  0.164621
```



```
 Tables for narratives queries :

 Table for 100% of queries :
     name    P@5   P@10      ndcg  recip_rank      map
0  TF_IDF  0.588  0.570  0.315241    0.773068  0.13161
1    BM25  0.612  0.572  0.321848    0.769302  0.13555

 Table for 75% of queries :
     name       P@5      P@10      ndcg  recip_rank       map
0  TF_IDF  0.573684  0.550000  0.301457    0.766725  0.121776
1    BM25  0.610526  0.565789  0.308758    0.774320  0.125913

 Table for 50% of queries :
     name    P@5   P@10      ndcg  recip_rank       map
0  TF_IDF  0.528  0.536  0.297375    0.701898  0.136043
1    BM25  0.592  0.548  0.302628    0.695000  0.138884

 Table for 25% of queries :
     name  P@5      P@10      ndcg  recip_rank       map
0  TF_IDF  0.6  0.591667  0.316123    0.796875  0.131365
1    BM25  0.6  0.591667  0.320631    0.748437  0.135853

 Table for 10% of queries :
     name   P@5  P@10      ndcg  recip_rank       map
0  TF_IDF  0.52  0.46  0.302041    0.866667  0.124255
1    BM25  0.56  0.50  0.301091    0.766667  0.126950
```

We can see that for each type of queries the values do not change or remain relatively constant. This does not mean that no matter the number of queries our search engine remains as efficient. Having a single query and evaluating the documents it receives is not precise enough. We cannot base an analysis on a very small sample, it is necessary to have a certain number of data in order to make an analysis fairly accurate and of a good level.

Our second deepening is to change our index by adding «title» in addition to «abstract».



```
Table for adhocs queries :
     name    P@5   P@10      ndcg  recip_rank       map
0  TF_IDF  0.668  0.624  0.364453    0.775853  0.173262
1    BM25  0.680  0.628  0.364684    0.773892  0.172613

 Table for descriptives queries :
     name    P@5   P@10      ndcg  recip_rank       map
0  TF_IDF  0.716  0.670  0.400411    0.808048  0.183681
1    BM25  0.692  0.658  0.403628    0.826746  0.185384

 Table for narratives queries :
     name    P@5   P@10      ndcg  recip_rank      map
0  TF_IDF  0.588  0.570  0.315241    0.773068  0.13161
1    BM25  0.612  0.572  0.321848    0.769302  0.13555
```

We find that these evaluations are completely identical to those we get with the previous index whether for queries with stemming or without.

We can therefore think that in this case and with this modification, the change of index does not necessarily have an impact on the result returned by our search engine.

Conclusion

In order to conclude this project we can say that the pre process stage is important and that the choice of the stages that we execute at this time is essential. The latter can change the efficiency of the search machine.

In our case according to the metrics on which we focus we can say that our search engine is partially relevant. The results we get are correct but can be improved if we combine our machine with other information.