

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.

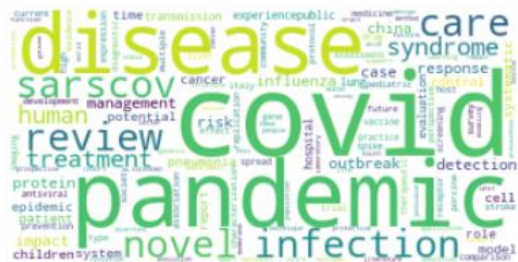
In order to better observe them for this report, we have displayed them as data frames :

	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.

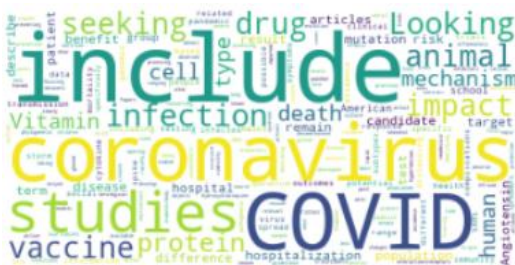
In our dataset we can extract three types of requests :

- 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 I...
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.

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 (accuracy at 5), P@10 (accuracy 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. Then, we create a database

```
print(pt_descriptives_queries)
qid      query
0 1      origin covid
1 2      coronavirus respond changes weather
2 3      sars infected people develop immunity cross pr...
3 4      causes death covid
4 5      drugs active sars sars animal studies
```

for each type of request containing the “qid” and pre-process requests.

The most important step is the following.

```
Table for adhoc queries :
name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.002226  0.000040  0.000040
1 BM25 0.0  0.0  0.002306  0.000049  0.000049

Table for descriptives queries :
name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.002661  0.000110  0.000110
1 BM25 0.0  0.0  0.002637  0.000105  0.000105

Table for narratives queries :
name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.0  0.0  0.0
1 BM25 0.0  0.0  0.0  0.0  0.0
```

We need to implement a table with to the measures we have cited for each query and implement the evaluation. We get this table:

We didn't expect those kinds of results. We thought we were getting higher values.

We have tried to change several parameters or our way of doing things but unfortunately we do not understand why we have such a low evaluation.

We therefore wanted to try to modify our pre process in order to see if our evaluation would change. For this we added the stemming step in order to have only the root of the words. So we do all the same commands again with this new dataset.

```

Table for stemming adhocs queries :
  name  P@5  P@10  ndcg  recip_rank  map
0  TF_IDF  0.0  0.0  0.002226  0.000040  0.000040
1   BM25  0.0  0.0  0.002306  0.000049  0.000049

Table for stemming descriptives queries :
  name  P@5  P@10  ndcg  recip_rank  map
0  TF_IDF  0.0  0.0  0.002425  0.000066  0.000066
1   BM25  0.0  0.0  0.002413  0.000064  0.000064

Table for narratives queries :
  name  P@5  P@10  ndcg  recip_rank  map
0  TF_IDF  0.0  0.0  0.0      0.0  0.0
1   BM25  0.0  0.0  0.0      0.0  0.0

```

This is the table we get. We see that the value we have decreased.

This is a result that can be understandable because we reduce each word to its root without worrying about the context and thus it can add ambiguity and therefore make the results less significant.

But with our value problems we cannot make a real comparison between different measures or between different datasets after a first modified process.

We also wanted to see if queries with similar performance could change the evaluation. For this we use the `measure_for_similar_queries(index_query: int, type_query: int, qdataset: pd.DataFrame, allqdataset: pd.DataFrame)`. We run this command for our query type.

We obtain the value 0 everywhere. We think it's because of the same problem as our last tables but we didn't find the reason.

Despite these problems, we wanted to extend this project.

```

Comparison of performance for similar narratives queries :

Table of evaluation by query :
  name  P@5  P@10  ndcg  recip_rank  map
0  TF_IDF  0.0  0.0  0.0      0.0  0.0
1   BM25  0.0  0.0  0.0      0.0  0.0

Table of evaluation by query :
  name  P@5  P@10  ndcg  recip_rank  map
0  TF_IDF  0.0  0.0  0.0      0.0  0.0
1   BM25  0.0  0.0  0.0      0.0  0.0

Table of evaluation by query :
  name  P@5  P@10  ndcg  recip_rank  map
0  TF_IDF  0.0  0.0  0.0      0.0  0.0
1   BM25  0.0  0.0  0.0      0.0  0.0

Table of evaluation by query :
  name  P@5  P@10  ndcg  recip_rank  map
0  TF_IDF  0.0  0.0  0.0      0.0  0.0
1   BM25  0.0  0.0  0.0      0.0  0.0

Comparison of performance for similar stemming narrativs queries :

Table of evaluation by query :
  name  P@5  P@10  ndcg  recip_rank  map
0  TF_IDF  0.0  0.0  0.0      0.0  0.0
1   BM25  0.0  0.0  0.0      0.0  0.0

Table of evaluation by query :
  name  P@5  P@10  ndcg  recip_rank  map
0  TF_IDF  0.0  0.0  0.0      0.0  0.0
1   BM25  0.0  0.0  0.0      0.0  0.0

Table of evaluation by query :
  name  P@5  P@10  ndcg  recip_rank  map
0  TF_IDF  0.0  0.0  0.0      0.0  0.0
1   BM25  0.0  0.0  0.0      0.0  0.0

Table of evaluation by query :
  name  P@5  P@10  ndcg  recip_rank  map
0  TF_IDF  0.0  0.0  0.0      0.0  0.0
1   BM25  0.0  0.0  0.0      0.0  0.0

```

Search Engine - Advanced Search

In this extension we thought about reducing the number of queries for each of our three types.

We start by taking only 75% of the queries then 50% and 25% to finish 10% of the queries.

We obtain these results :

```
Table for adhoc queries :
  name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.002929  0.000052  0.000052
1 BM25   0.0  0.0  0.003034  0.000065  0.000065
```

```
Table for adhoc queries :
  name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.004451  0.000079  0.000079
1 BM25   0.0  0.0  0.004612  0.000098  0.000098
```

```
Table for adhoc queries :
  name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.0      0.0  0.0
1 BM25   0.0  0.0  0.0      0.0  0.0
```

```
Table for adhoc queries :
  name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.0      0.0  0.0
1 BM25   0.0  0.0  0.0      0.0  0.0
```

```
Table for descriptives queries :
  name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.002929  0.000052  0.000052
1 BM25   0.0  0.0  0.003034  0.000065  0.000065
```

```
Table for descriptives queries :
  name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.0      0.0  0.0
1 BM25   0.0  0.0  0.0      0.0  0.0
```

```
Table for descriptives queries :
  name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.0      0.0  0.0
1 BM25   0.0  0.0  0.0      0.0  0.0
```

```
Table for descriptives queries :
  name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.0      0.0  0.0
1 BM25   0.0  0.0  0.0      0.0  0.0
```

```
Table for narratives queries :
  name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.0      0.0  0.0
1 BM25   0.0  0.0  0.0      0.0  0.0
```

```
Table for narratives queries :
  name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.0      0.0  0.0
1 BM25   0.0  0.0  0.0      0.0  0.0
```

```
Table for narratives queries :
  name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.0      0.0  0.0
1 BM25   0.0  0.0  0.0      0.0  0.0
```

```
Table for narratives queries :
  name  P@5  P@10  ndcg  recip_rank  map
0 TF_IDF 0.0  0.0  0.0      0.0  0.0
1 BM25   0.0  0.0  0.0      0.0  0.0
```

We can see that depending on the number that query, our evaluation varies.

But we still can't have a concrete conclusion or comparison because of our value problem