

ArXiv - Scientific Papers - Information & Search

BEATRIZ SANTOS, SÉRGIO ESTÊVÃO, and SÉRGIO DA GAMA

Additional Key Words and Phrases: Information Processing and Retrieval, M.EIC, dataset, arXiv, papers, statistics

1 INTRODUCTION

We decided to elaborate the project based on scientific papers because there would be vast amounts of textual information associated with them and they are easily categorized since we could use authors and fields of investigation to batch and select specific articles. Both of these properties are essential in creating an ideal information retrieval system.

2 DATA PREPARATION

The project's first milestone is preparing and characterizing the dataset. The dataset, *ArxivData.json*, was obtained through Kaggle. This dataset is a collection of scientific papers and their corresponding information from the website *arXiv* [1], maintained by Cornell Tech.

2.1 Data Pipeline

The first step was to aggregate all of the **authors** from an article into a list. After that, the normalization of the publication **date** was created. This is useful to search for a paper from a specific month or date. In addition, some unnecessary links, that the original data set kept, were removed and only the **link** that has all of the information about the article, such as the PDF file, was kept.

Lastly, each element of the dataset had a sub-element containing **tags** that characterize the area of the article. After some observation, the group noticed that not all of the information from that sub-element mattered to the project, dropping the unnecessary information. Besides that, in order to get the full name and normalize the tags, some **web-scraping** had to be done.

Starting the scrapping phase, we confirmed that the dataset had outdated tags from the websites. Fortunately, concatenating these outdated tags with the *arXiv* [1], it redirected to the web pages of its corresponding updated tag. To confirm we had the updated one, we scrapped the web pages of each tag to confirm we had the most recent ones.

After confirming we had all the tags, we could begin the process of extracting the additional information from the tags. By scrapping the page of each tag, we got its respective **field** and **subject**, and by scrapping the main website page we got the **area of study** of each tag.

After this process, the dataset was ready to be converted into a Pandas dataframe and the existence of null values was searched for, such as the removal of duplicates.

Figure 1, shows this process of pipeline formation.

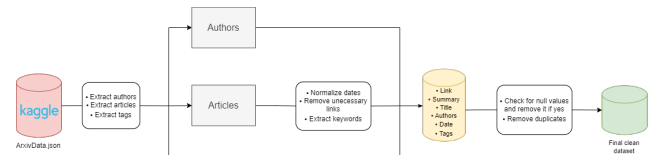


Fig. 1. Pipeline formation

Note: we used a cache system to save the corresponding information of each tag, reducing the number of requests to the website and consequently speeding up the refining process.

2.2 Data Characterization

Our dataset focuses on the scientific **articles**, which are then accompanied by their respective **authors**, as well as other useful information.

The dataset itself has **41 000 entries**. However, there are more authors than entries (articles), because an article can have more than one author, making up to **59 888 authors**, of which 10 are represented in Figure 2. The authors in said figure are the ones with more articles published, sorted in descending order.

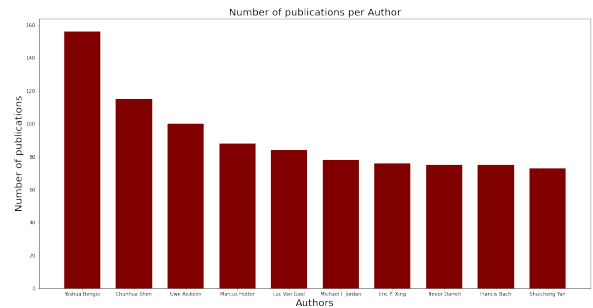


Fig. 2. The 10 authors with more articles

Yoshua Bengio appears to be the author with more articles published, having almost 160 publications. Therefore, a big gap exists between him and the second author, Chunchua Shen, which has about 120 articles, 40 less than Yoshua. After the second author, the number of articles from the rest of the authors start stabilizing with about 80 articles published.

Figure 2 only gave us a good understanding of each author. So, to get a better grasp on how authors appeared in our dataset, we created the histogram in Figure 3, which uncovered that most of the dataset entries had from 1 to 15 authors. Besides that, the most frequent amount is 2 to 5 authors per Article. As the zero column is empty, we can also conclude that all the articles have at least one author.

Authors' address: Beatriz Santos, up201905680@up.pt; Sérgio Estêvão, up201905680@up.pt; Sérgio da Gama, up201905680@up.pt.

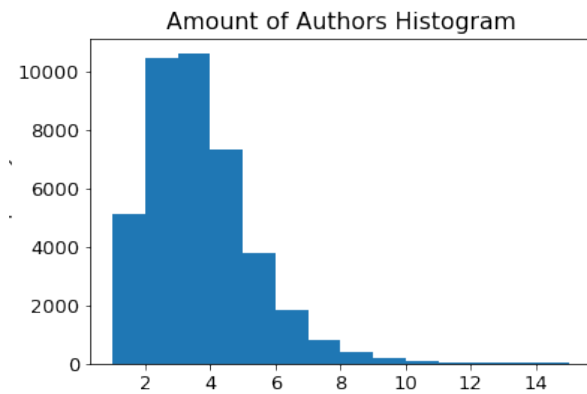


Fig. 3. Frequency of Authors per Article

Additionally, we also analyzed the frequency of subjects per article, represented in Figure 4. This allowed us to see that each Article is related to a relatively small amount of subjects, ranging in general, from 1 to 4 subjects per Article. It is also visible that all Articles have at least one subject.

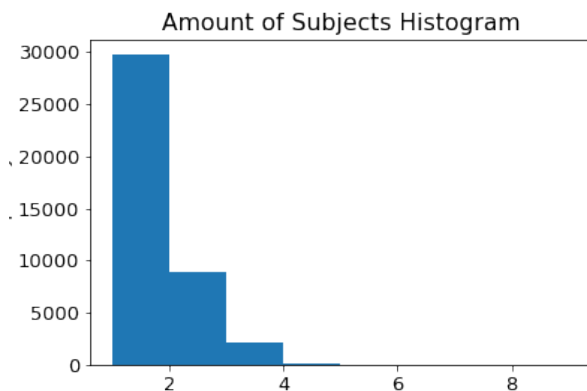


Fig. 4. Frequency of Subjects per Article

In order to place the data in time, we created Figure 5 which revealed that the oldest article in our dataset is from 1993 and the most recent from 2018. Nevertheless, the majority of the reports were published in 2018, almost having normally distributed data between those dates. This graph also reveals that the year with more articles published is by far 2017, with over 12 000 publications made.

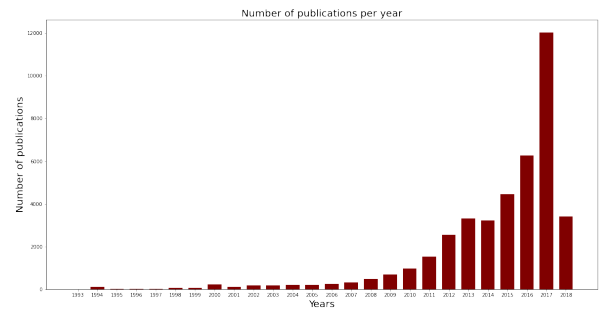


Fig. 5. Articles published by year

We decided it would be interesting to check the distribution of subjects in the papers and verify which ones were more widely common. Since the dataset has a total of **161 unique subjects**, 156 of them appear in less than 4000 articles ($>0,1\%$ representativity), so we group them in the "Others" category. In Figure 6 we can verify that Machine Learning, Computer Vision and Pattern Recognition, and Artificial Intelligence are the most common subjects, all subjects of Computer Science.

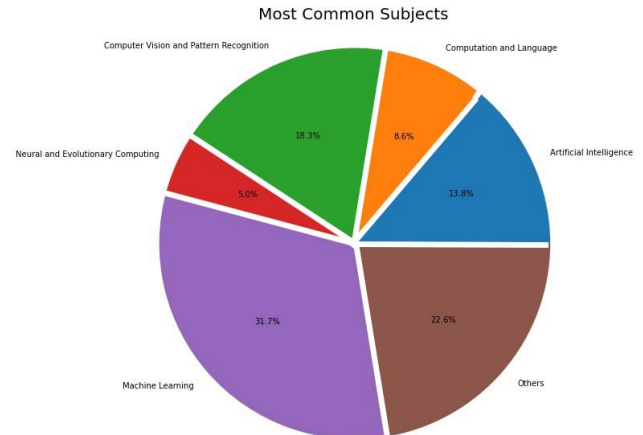


Fig. 6. Subjects of the papers

Furthermore, each article also has areas associated. In total, our dataset has **8 areas**, although, the most common areas are Computer Science, Statistics, Mathematics. The majority of the articles are about Computer Science, and here we can confirm that the dataset isn't balanced when it comes to areas, being more focused in Computer Science. This can be observed in Figure 7.

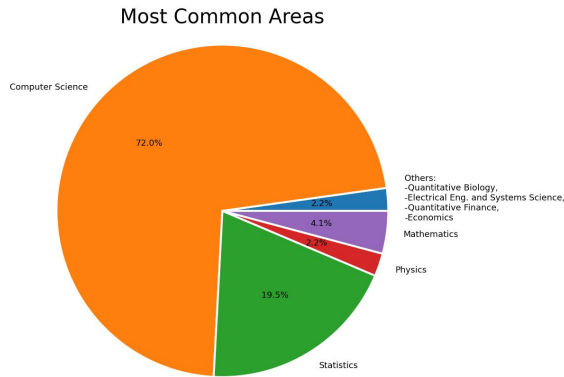


Fig. 7. Areas of the papers

Apart from areas and subjects, in our dataset, each area can have multiple fields. In this case, Physics is the only area that has multiple ones, in Figure 8 we can observe the distribution of Physics fields in the articles.

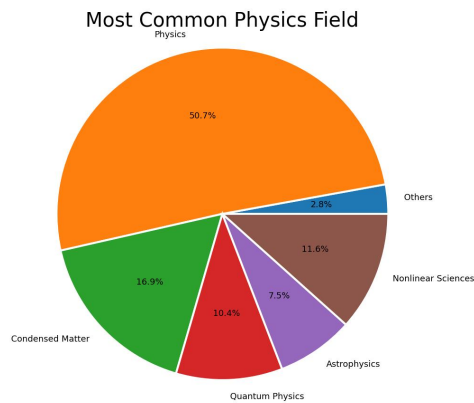


Fig. 8. Fields of physics

Analysing now the "summary" attribute, to find the most common vocabulary used in the papers we created a graph showing the top words in the dataset. This data shows that words like "model", "method" and "network" are widely used. This statistic confirms the dataset context, since the contents of these papers are scientific, academic, and mostly about Computer Science and these top words are commonly used in those fields. In the Figure 9, we can observe this facts.

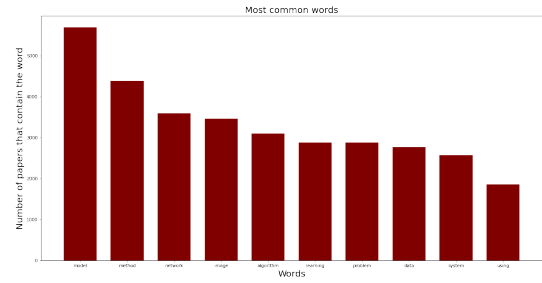


Fig. 9. Most common Words

Additionally, we also analyzed the frequency of words in the summary field per article, represented in Figure 10. This allowed us to see that most of the articles have between 100 and 200 words.

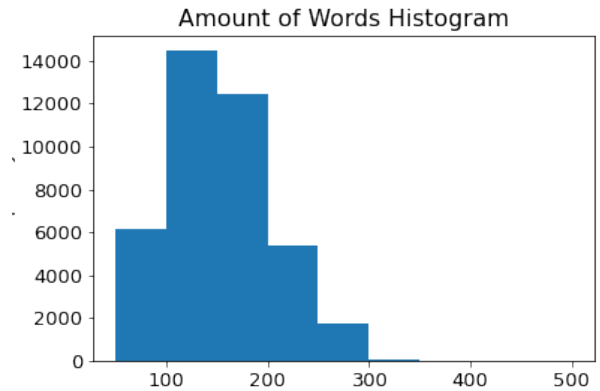


Fig. 10. Areas of the papers

2.3 Data Domain Conceptual Model

Our data conceptual model, represented in Figure 11, has one main table, **Article**, that has the attributes below:

- **Title:** Article's title
- **Date:** The publishing date
- **Link:** The link where the Article is available
- **Summary:** Summary of the Article's content

Each Article has one or more Authors, which are instances of the **Author** class that only has one attribute, the **Author Name**.

In addition to that, each Article also has one or more Areas, instances of the table **Area**. In turn, these last ones have one or more Fields, which are instances of the **Field** class. Finally, the fields have subjects related to them, which are instances of the **Subject** class. However, a subject can only belong to a field, and a field can only belong to an area.

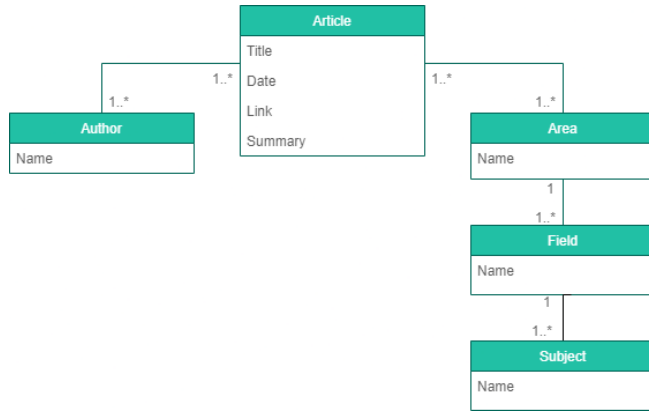


Fig. 11. Data Conceptual Model

```

"link":["http://arxiv.org/abs/1603.03827v1"],
"summary":"Recent approaches based on artificial neural networks
"title":"Sequential Short-Text Classification with Recurrent and
"authors":["Ji Young Lee",
"Franck Dernoncourt"],
"date":"2016-03-12T00:00:00Z",
"areas":["Computer Science",
"Statistics"],
"fields":["Computer Science",
"Statistics"],
"subjects":["Computation and Language",
"Artificial Intelligence",
"Machine Learning",
"Neural and Evolutionary Computing",
"Machine Learning"],
"id":"6097534b-cada-465f-b7bd-e4f0dedde6fb",
"_version_":1749492367336931328},
  
```

Fig. 12. Collection

2.4 Prospective Search Scenarios

In the future the user will be able to search for an article by its name, by its areas, by subjects and fields. Moreover, it will also be capable of searching by articles that contain some given words in their summary. Some search scenarios examples:

- Papers in the area of Physics about terminal velocity.
- Papers in the field of Statistics, about normal distributions.
- A Writer wants to write a biography about Francis Bach, but wants to focus on his algorithmic work from 2008 to 2018 mostly, and preferably from the year of 2015.
- A researcher in the area of Statistics wants to try some new approaches related to his case study in linguistics.
- A student that wants to get all the papers of related economics and computer science, in the year of 2017.

3 INFORMATION RETRIEVAL

Information retrieval in computing and information science is the process of obtaining information system resources that are relevant to an information need from a collection of those resources.

The most common retrieval task is when a user specifies what they want through a query that is matched against all the documents and matching results are ordered by how relevant the system deemed them.

3.1 Collection and Indexing Process

After some data preparation, in our collection of papers, a paper defines a document with its different characteristics the figure 27 below.

To index these documents we used *Solr* [2], an Information Retrieval system. After some study of the *Solr*'s API, we gathered the most relevant tokenizers and filters to be used in our schema in order to get a search with the best performance.

In order to have our dataset with *Solr*'s structure, we had to change a little the structure of the areas, fields and subjects, having only one list associated with each field, as well as adding an id, since *Solr*'s id is not always the same.

Regarding indexing, not all fields make sense for a user to search for, such as the URL associated with the paper. Furthermore, we identified which fields would make more sense to emphasize in the search, for example, when a user searches for a paper, he will more easily search for keywords on the subject that he wants to get information on rather than the date of publication. Taking this into account, our scheme puts more emphasis on the title of the article, as well as the summary, being also indexed the authors, areas, fields, subjects and, finally, the date of publication, in this order. For the title's type, the one we put more emphasis on, on the document's index time, the following attributes were used by the order they appear:

- **Classic Tokenizer** (ClassicTokenizerFactory) -> splits the text field into tokens, treating whitespace and punctuation as delimiters.
- **Classic Filter** (ClassicFilterFactory) -> strips periods from acronyms and "s" from possessive.
- **Lower Case Filter** (LowerCaseFilterFactory) -> converts any uppercase letters in a token to the equivalent lowercase token.
- **ASCII Folding Filter** (ASCIIFoldingFilterFactory) -> converts alphabetic, numeric, and symbolic Unicode characters which are not in the Basic Latin Unicode block to their ASCII equivalents.
- **Porter Stem Filter** (PorterStemFilterFactory) -> a stemming algorithm.
- **Stop Filter** (StopFilterFactory) -> discards tokens that are on the given stop words list.
- **Phonetic Filter** (PhoneticFilterFactory) -> creates tokens using Double Metaphone algorithm.

- **Remove Duplicates Token Filter** (RemoveDuplicatesTokenFilterFactory) -> removes duplicate tokens in the stream.

On the queries' index time, the previous tokenizer and filters were used, adding the following one:

- **Synonym Graph Filter** (SynonymGraphFilterFactory) -> maps single or multi-token synonyms, producing a fully correct graph output.

For the summary type, on the document's index time, the following attributes were used by the order they appear:

- **Classic Tokenizer** (ClassicTokenizerFactory)
- **Classic Filter** (ClassicFilterFactory)
- **ASCII Folding Filter** (ASCIIFoldingFilterFactory)
- **Stop Filter** (StopFilterFactory)
- **Common Grams Filter** (CommonGramsFilterFactory) -> creates word shingles by combining common tokens such as stop words with regular tokens.
- **Beider-Morse Filter** (BeiderMorseFilterFactory) -> implements the Beider-Morse Phonetic Matching (BMPM) algorithm, which allows the identification of similar names, even if they are spelt differently or in different languages.

For the name type, used on the areas, files, subjects and authors' names, on the document's index time and on queries' index time, the following attributes were used by the order they appear:

- **Classic Tokenizer** (ClassicTokenizerFactory)
- **Classic Filter** (ClassicFilterFactory)
- **Lower Case Filter** (LowerCaseFilterFactory)
- **ASCII Folding Filter** (ASCIIFoldingFilterFactory)

For the date type, it was only used the Date Point Field, which is a field type that can represent any Date/Time with millisecond precision.

In order to have a term of comparison, two other schemas were made with fewer filters associated with the attributes that we will see in section 3.3.2 later on.

3.2 Retrieving Process

In this section, we will explain some proposed queries, while taking advantage of some *Solr*'s query boosters and features. In order to fully take advantage of *Solr*'s search capabilities we always used the *eDisMax* query parser, which allowed additional parameters, like *pf*, on top of the others provided by the standard parser. Additionally, we calculated the precision and recall for each normal and boosted query, of each information need, and compared them to understand the improvements given by the boosting.

3.2.1 Information Need 1 - A developer that is creating a Machine Learning model to predict cars' longevity based on their velocity and needs to know everything about this last topic.

In order to fulfil this information need, a perfect system would only retrieve papers that study the velocity itself. However, papers can reference velocity, without actually focusing on its study. Therefore, it was hard to fully satisfy this query, but one way to boost it was to increase the score of the documents that mentioned velocity in the summary, relative to appearing in the other fields. This is

based on the fact that if velocity appears in the summary, it is most likely that the paper explains some velocity-related topics. If it appeared in the title and not in the summary, most probably, the paper wasn't really focused on that topic, which is why we favoured the summary over the title, as well.

- **Query:** velocity

- **Parameters:**

- (1) **Normal:** qf: link summary title authors date area field subjects
- (2) **Boosted:** qf: link summary^10 title^2 authors date area field subjects

The results of this query revealed to be good both with and without the boost. Hence, the P@10 of the normal query was 1.0, and the recall 1.0, as well. The results of the boosted query were equal to this last one.

3.2.2 Information Need 2 - A Data analyst that is processing a dataset and needs to understand normal distributions.

With this information need, the system is intended to return documents that talked about normal distributions. Thus, being the query composed by the two terms, normal and distribution, we ensure that the query operation was *AND*, which implies the existence of both terms in the documents. As this is a major topic of the area of **Statistics**, the system should also prioritize documents related to said area. retrieved. In favour of accomplishing that, in the boosted query, we attributed a higher score to the documents that had the area of Statistics, through the use of the *bf* (boost query) parameter.

- **Query:** normal distribution

- **Parameters:**

- (1) **Normal:** qf: link summary title authors date area field subjects
- (2) **Boosted:**
 - (a) qf: link summary^2 title^10 authors date area field subjects
 - (b) bq: area:Statistics^10

Contrary to the first information need queries, this ones presented a more notorious difference between the normal and the boosted one. The normal query P@10 was 0.5 and the recall was 0.625. On the other hand, the boosted query unveiled to have significantly better results, with a P@10 of 1.0 and a recall of 0.9375.

3.2.3 Information Need 3 - A Writer wants to write a biography about Francis Bach, but wants to focus on his algorithmic work from 2008 to 2018 mostly, preferably from the year 2015.

For this information need, the search system should first retrieve the papers that referenced algorithms, published, by *Francis Bach* in 2015, and then, the rest of the papers, published between 2008 and 2018. In order to accomplish that, the created query specifies what the author wanted, followed by the term algorithm. We noticed that just by using the filter (*fq* parameter), corresponding to the dates wanted, we wouldn't get the year 2015 papers first. Consequently, in the boosted query we created a boost function (*bf* parameter), which gives a higher score to the documents that have a 2015 date,

relative to the other dates. Finally, it was given a higher score to the documents that had the terms in the *summary* and in the *title*, through the query factor (*qf* parameter).

- **Query:** Francis Bach algorithm
- **Parameters:**
 - (1) **Normal:**
 - (a) fq: date:[2014-01-01T00:00:00Z TO 2018-01-01T00:00:00Z}
 - (b) qf: link summary title authors date areas fields subjects
 - (2) **Boosted:**
 - (a) fq: date:[2014-01-01T00:00:00Z TO 2018-01-01T00:00:00Z}
 - (b) qf: link summary^5 title authors^5 date areas fields subjects
 - (c) bf: if(and(gte(ms(date), ms(2015-01-01T00:00:00Z)), lt(ms(date), ms(2016-01-01T00:00:00Z))), 10, 0.1)

These queries resulted in similar precisions and recalls, having just a small 0.1 difference, from the normal to the boosted query precision. The recall of the two was 0.933.

3.2.4 Information Need 4 - A researcher in the area of Statistics wants to try some new approaches related to his case study in linguistics.

For this information need, the system should return only documents that presented with both terms, new and linguistics, as both are imperatives in order to search for the idea of papers related to linguistic innovations. Hence, it was used the *pf* parameter (proximity factor), to attribute a higher score to the documents that have the title with both terms close to each other.

- **Query:** areas:(statistics) new approaches linguistics
- **Parameters:**
 - (1) **Normal:** qf: link summary title authors date areas fields subjects
 - (2) **Boosted:**
 - (a) qf: link summary title^2 authors date areas fields subjects
 - (b) pf: summary^10

After executing both queries, we realized that both were having a low P@10 value of 0.5, and of 0.6, respectively in the normal and the boosted query. This was mostly due to the fact that there aren't many new, approaches, and linguistics terms close to each other and, therefore, the proximity factor in the summary didn't affect the first ten results as much as expected. However, the boosted query was able to achieve a recall of 1.0, compared to the 0.45 of the normal query. This happened because the query factor over the title in the boosted query actually made the difference and retrieve all the relevant results, it just wasn't able to retrieve all of them with a high score.

3.2.5 Information Need 5 - A student that wants to get all the papers related to economics and computer science, in the year 2017.

Finally, in order to fulfil this information needs the system should only return articles related to economics and computer science, published in the year 2017. With this in mind, we use a query factor (*qf*) parameter, to make sure that we gave a higher score to the documents with the terms appearing in the area field and subject

fields. Moreover, to guarantee that documents with the Computer Science field have a higher score, we added a boost query, *bf*, to the Computer Science field.

- **Query:** Computer Science economics
- **Parameters:**
 - (1) **Normal:**
 - (a) fq: date:[2017-01-01T00:00:00Z TO 2018-01-01T00:00:00Z}
 - (b) qf: link summary title authors date area field subjects
 - (2) **Boosted:**
 - (a) fq: date:[2017-01-01T00:00:00Z TO 2018-01-01T00:00:00Z}
 - (b) qf: qf: link summary title authors date areas^5 fields^5 subjects^5
 - (c) bq: field:Computer Science^5

Indeed, with the help of the boost query in the computer science field we were able to have three times the precision we had with the normal query, from 0.3 to 0.9. Finally, both the queries had a recall of 1.0.

3.3 Evaluation

To evaluate the performance of the information retrieval system we used all the queries of information needs and measured their precision, recall, and mean average precision, and visualized their precision-recall curve. Overall our information retrieval system had satisfactory results, the mean average precision in the boosted system was 90% and the recall was 97,4%, being significantly higher than the non-boosted counterpart, which had a mean average precision of 64% and a recall of 80,3%, these results mean the optimization worked as accordingly. In figures 13 and 14 we can see the precision and recall of each query in the both non-boosted and boosted systems, respectively.

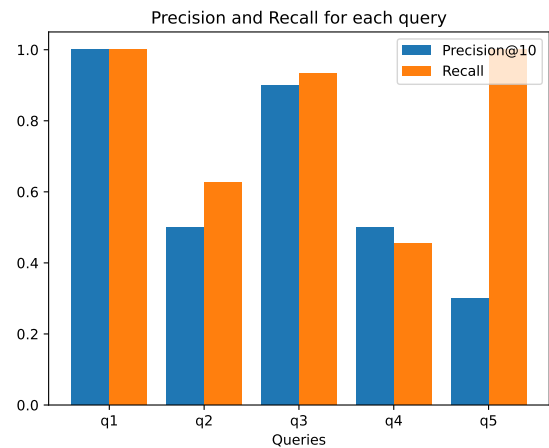


Fig. 13. Precision and Recall in the non-boosted system

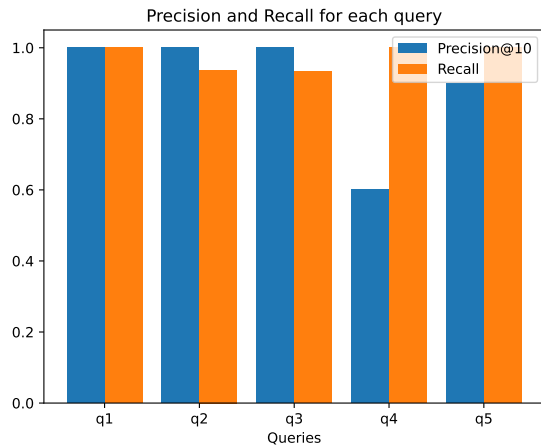


Fig. 14. Precision and Recall in boosted system

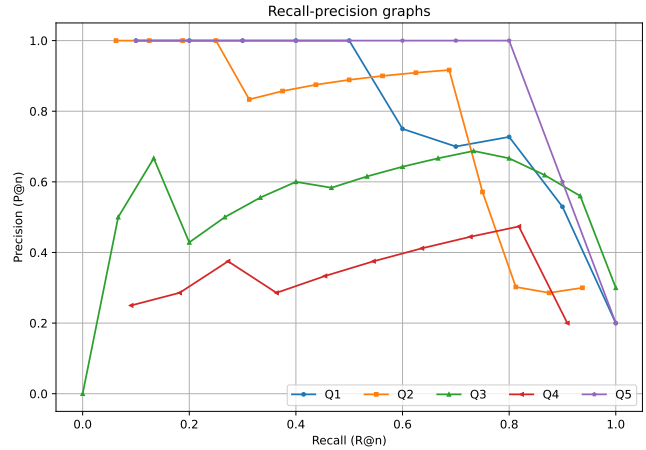


Fig. 15. Precision-Recall curves in Schema 1

3.3.1 Schemas. To decide the best schema to use in the information retrieval system we compared three different schemas using the corresponding progression of the precision-recall curves of each query and choose the schema with the best results. The three schemas differed from one another in the number of filters: Schema 1, which had basic filters, Schema 2, which incremented on these filters of Schema 1, and Schema 3, which incremented on the filters of Schema 2. Schema 3 ended up being the one used in this project.

With this data we were able to conclude that Schema 3 gave the overall best results, having the highest recall and precision for each result batch size. Schema 1, as expected, gave the worst results, since it only had basic indexing filters. Schema 2, although having precision and recall values close to Schema 3, it failed to get any relevant results in query 4, making it noticeably worst than Schema 3. With this information, we can confirm that well-selected filters impact positively the result of queries. Schema 3 ended up being the one used in this project. In figures 15, 16, and 17 we can visualize for each schema the precision-recall curves of the queries in the boosted system.

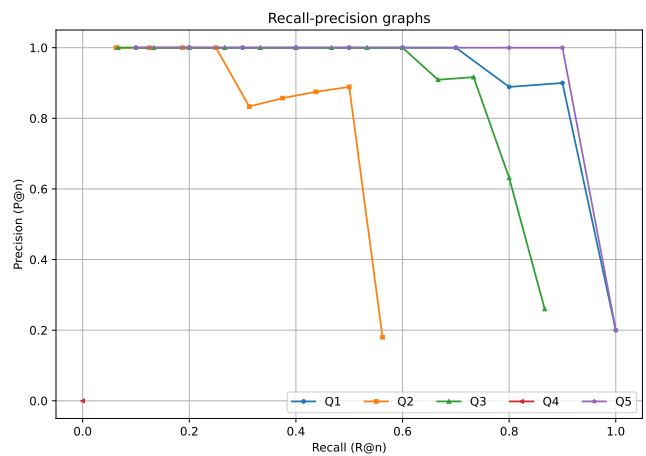


Fig. 16. Precision-Recall curves in Schema 2

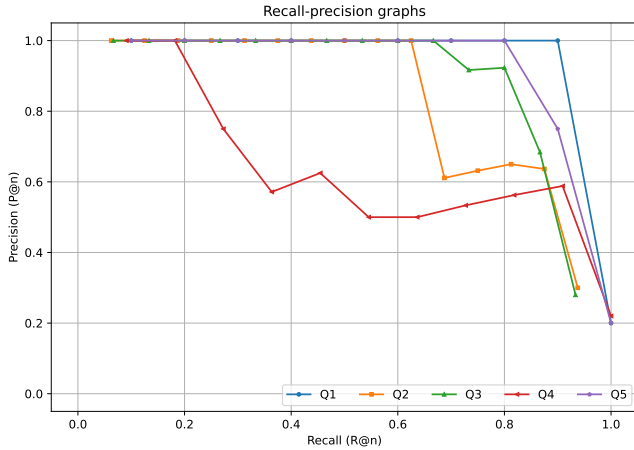


Fig. 17. Precision-Recall curves in Schema 3

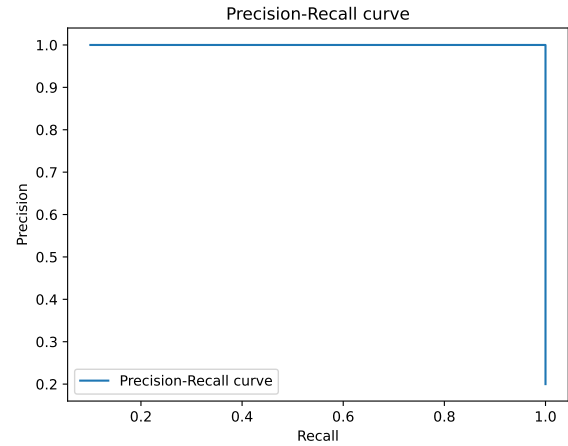


Fig. 19. Precision-Recall curves for Boosted Query 1

3.3.2 Queries. Query 1 in the non-boosted system had a precision of 1.0, this value was maintained until it reached a recall of 1.0, where it dropped, this happened because after reaching this value, the information retrieval system only finds articles that are not considered relevant. In the boosted system, this query maintains these values, as observed in figures 18 and 19.

Queries 2, 3, 4, and 5 with the boosted system had their initial precision values significantly increased, being queries 3 and 4 the ones with the most visible improvements. Query 4 stands out since the boost system doubled its recall. In the figures from 20 to 26, it is possible to visualize the improvement of the precision and recall displayed in the precision-recall curves for queries 2, 3, 4, and 5. These results indicate the boosted system in each query surpassed its non-boosted counterpart, giving and prioritizing altogether more relevant search results.

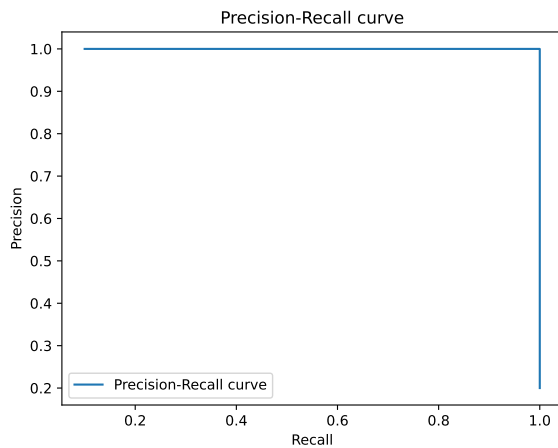


Fig. 18. Precision-Recall curve for Query 1

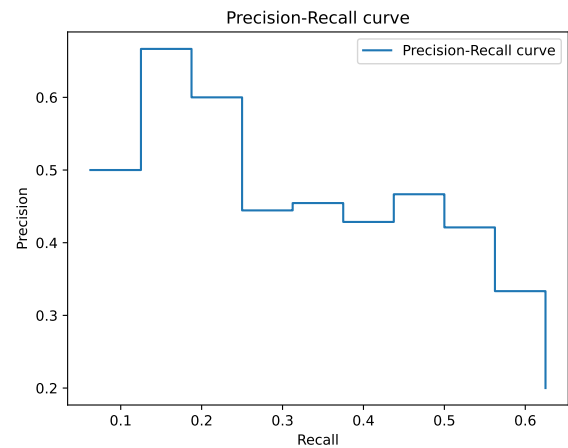


Fig. 20. Precision-Recall curve for Query 2

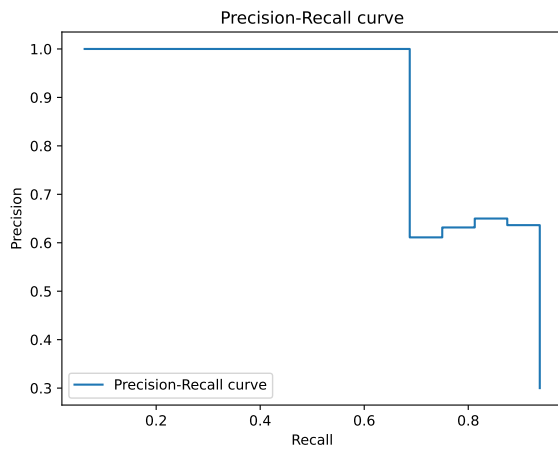


Fig. 21. Precision-Recall curves for Boosted Query 2

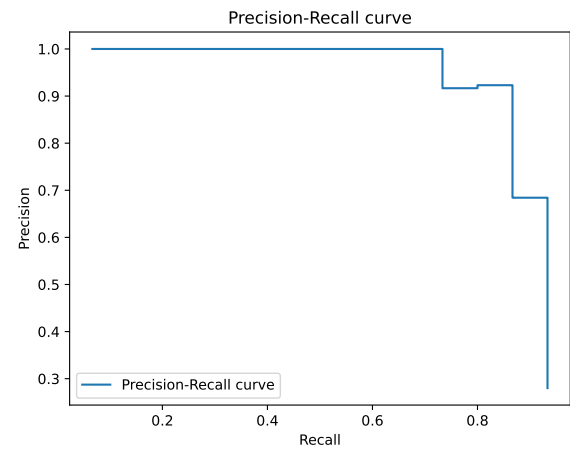


Fig. 23. Precision-Recall curves for Boosted Query 3

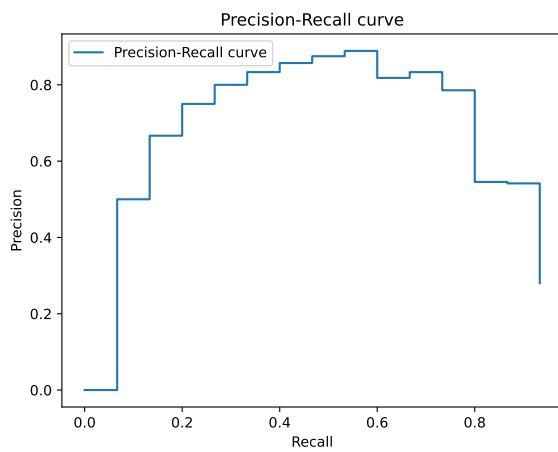


Fig. 22. Precision-Recall curve for Query 3

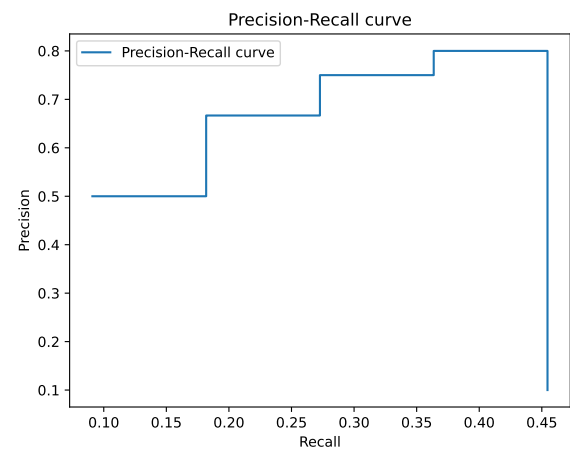


Fig. 24. Precision-Recall curve for Query 4

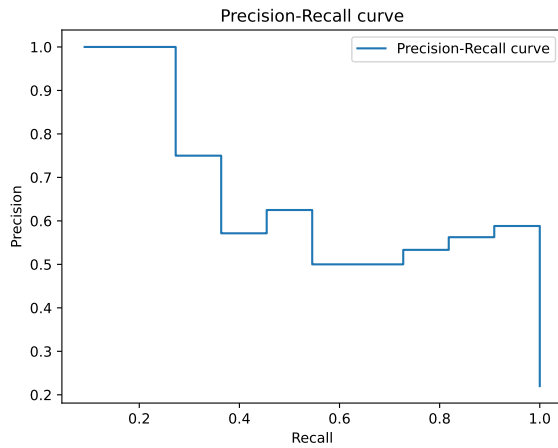


Fig. 25. Precision-Recall curves for Boosted Query 4

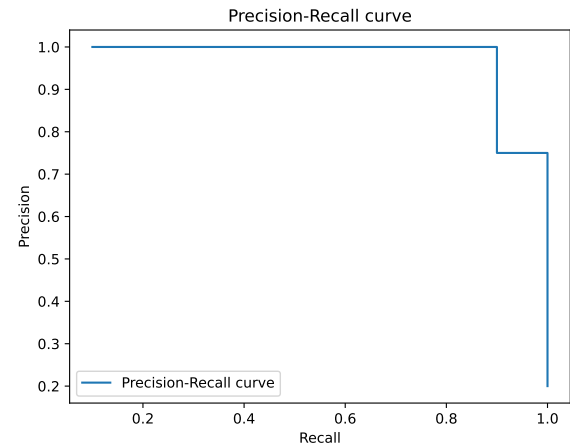


Fig. 27. Precision-Recall curves for Boosted Query 5

4 CONCLUSION

In this project, we acquired, prepared, and characterized a scientific articles dataset. After that initial data preparation, we started creating a schema to take advantage of *Solr* searching tool. We then experimented with multiple filters and tokenizers until we reached the optimal schema on our context. Finally, we tested this last one with 5 queries representing 5 different information needs and created a boosted version of each of them to test the results obtained with the precision and recall metrics.

REFERENCES

- [1] Ginsparg, P. (2022). Research-sharing platform. ARXIV, available at <https://arxiv.org/>. last accessed on october 2022.
- [2] Solr (2022). Solr reference guide. Solr, available at https://solr.apache.org/guide/6_6/index.html. last accessed on november 2022.

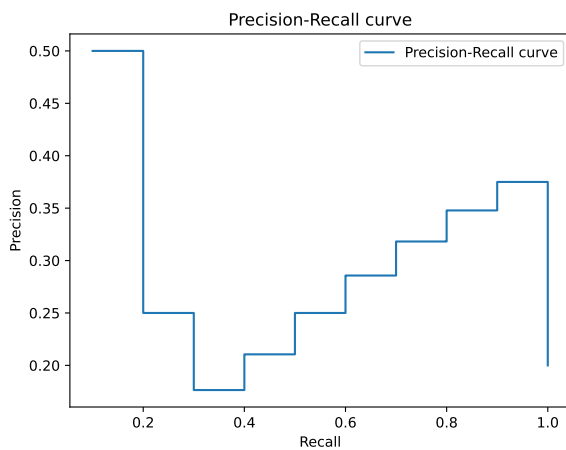


Fig. 26. Precision-Recall curve for Query 5