Portuguese Political Parties in ARQUIVO.PT

João Afonso M. D. Andrade
up201905589@edu.fe.up.pt
M.EIC
Faculty of Engineering of the University of Porto
Porto, Portugal

Maria José V. S. Carneiro
up201907726@edu.fe.up.pt
M.EIC
Faculty of Engineering of the University of Porto
Porto, Portugal

Miguel Azevedo Lopes
up201704590@edu.fe.up.pt
M.EIC
Faculty of Engineering of the University of Porto
Porto, Portugal

Abstract

ARQUIVO.PT [1] is a research infrastructure that allows searching and accessing web pages archived since 1996, with the goal of preserving information published on the Web for research purposes.
The aim of this project is to collect, prepare and process data retrieved from ARQUIVO.PT [1] about all of the portuguese political parties currently holding parliamentary seats, throughout history, in order to develop an information search system of all of their websites.
Data collection and preparation, information retrieval, querying and evaluation were the tasks performed to achieve a complete and effective search system, that correctly identifies and answers specific information needs.

CCS Concepts

• **Information systems** → **Information retrieval**; **Structured text search**.

Keywords

datasets, information processing, information retrieval, full-text search

1 Introduction

ARQUIVO.PT [1] provides an extensive amount of data collected from the web throughout the years, about multiple different topics and hosts a contest each year that rewards the most innovative projects that analyses and explores their preserved data. Inspired by their mission of maintaining and archiving the information from the web that would inevitably be lost, we decided to explore the content published by all of the current political parties with parliamentary seats on their websites and create a search system that would allow to find each party's statements and evolution throughout history. Considering the misinformation regarding politics and the rise of fake news, having a search system that allows to clarify each political groups opinion and stance about specific issues will allow people to make more conscious decisions about who they elect and choose as their representative. That is our biggest motivation and what makes us passionate about this project.

2 Milestone 1 - Data Preparation

The data preparation step consists of preparing raw data and analysing it in order to better understand the dataset in hands and organize it for later stages of information processing and retrieval.

2.1 Languages and Tools

We developed the project in `Python` and used the extensive `Pandas` [7] data manipulation tool to clean, refine and analyse all data. Moreover, the graphs plotted were done using `Matplolib` [5] and textual analysis was done with the help of the `NLTK` [6] for natural language processing.

2.2 Data Collection

In order to retrieve every item of content published on each political party's website, we took advantage of the API provided by ARQUIVO.PT [1].

2.2.1 ARQUIVO.PT's API

Our initial expectation was to collect all the content that concerned the party's website via the API, but after thorough analysis of the API's specification we concluded there was no direct way to do this and so we started developing other strategies to achieve this goal. ARQUIVO.PT's API [10] is a search API that presents a maximum of 2000 results per request, as such, the first approach we considered, consisted of doing an empty query search restricted to the party's web domain and iterating through each page of results. However, after running our first test-run we concluded that the API had some functional issues (confirmed by issues raised by other people in the issues page at Github) that wouldn't allow us to go beyond the first page of results. As such, we moved on to another approach. Since we could only retrieve 2000 results at a time, we decided to do the same search, but instead of searching the whole archive we limited the results to 2000 entries per month. The final API call we used is presented bellow:

Endpoint: "https://arquivo.pt/textsearch"
Parameters: (Parameter Name | Parameter | Value Used)

- Query | q | empty string
- Site search | siteSearch | Party's website (eg: www.ps.pt)
- Start date | from | Start month being queried
- End date | to | End month being queried
- Maximum items per response | maxItems | 2000

The response provided by the API is a list of links to the archived webpages, as well as some information regarding the date of collection and format of the page (whether it's an HTML page, a link to a PDF file, an image, etc). Also included in each of the results was a link to a page that provided the extracted text from that page. This proved very useful as it saved us the trouble of having to extract the text from every page.

2.2.2 Dataset Size and Associated Constraints

We anticipated that we would have big datasets for each of the political parties, namely the ones who have existed for over 20 years. Partido Social Democrata, for example, has had a webpage for 26 years, which multiplied by 12 months at 2000 results per month could translate to a staggering 624 thousand webpages. This large amount of data raised a couple concerns: how we would store the data and how long it would take to retrieve it.

To address the storage concern, we decided to store each political party's data in a separate JSON file. As for the time it would take to retrieve all of the data, we initially considered a multi-threaded approach, but soon realized the bottleneck of the process was not the time it took our computers to process the requests but rather the request limit associated with the ARQUIVO.PT's API[10] (250 requests per minute). The API limits and the associated time constraint ended up being one of our biggest obstacles in the first delivery. Our calculations estimated that at most we could have almost 3 million webpages to collect (2 952 000 pages), at 250 requests per minute this would translate to 197 hours of runtime. For this reason, in this first delivery we decided to reduce the range of data by only collecting data of the 4 biggest parties (by number of representatives in the Assembleia da Republica), from the past 5 years.

In the next delivery we intend to include all the data from all the parties who are currently elected.

2.2.3 Political Parties Website's Domains - Changes Over Time

Before collecting the data through the process described earlier, we conducted research to determine when each of the political parties first created their websites and checked whether or not that website's domain name had changed over time. Table 1 describes our findings:

Political Party Name	Domain Name	Active Period
Partido Socialista (PS)	www.ps.pt	1999 - 2018
	ps.pt	2001 - current
Partido Social Democrata (PSD)	www.psd.pt	1996 - current
Chega (CH)	partidochega.pt	2019 - current
Iniciativa Liberal (IL)	iniciativaliberal.pt	2017 - current
Partido Comunista Português (PCP)	www.pcp.pt	1996 - current
	pcp.pt	2001 - current
Bloco de Esquerda (BE)	www.bloco-de-esquerda.pt	1999 - 2003
	www.bloco.org	2003 - current
	bloco.org	2005 - current
Livre (L)	livrept.net	2015 - 2018
	www.livrept.net	2015 - 2018
	partidolivre.pt	2018 - current
Pessoas–Animais –Natureza (PAN)	pan.com.pt	2013 - 2015
	www.pan.com.pt	2013 - current

Table 1: Political Parties Website's Domains Over Time

Apart from checking domain name changes, we also noticed some parties use both "www" prefixed domains and non-prefixed domains, and so we checked the date for the first occurrence of both www prefixed domains and non-prefixed domains in the ARQUIVO.PT's [1] archive. The content of all these domains was collected and later merged onto a single file (per party).

2.3 Data Preparation

In order to obtain a cleaned data set in the next stage of the project, we firstly need to prepare and analyse the original data set. Therefore all the data, especially the text one, should be normalized to fit better search criteria. We firstly began by examining the data collected to better grasp it's extent and content and find the best way to adopt in order to correctly organize and clean the data frame.

2.3.1 Data Pipeline

As we can see, the pipeline presented in figure 3 (Section 6 - Annex) represents the whole process of data collection as a sequence of events. It starts with the decision we made of selecting political parties to collect data from, using the API available from ARQUIVO.PT [1]. After we collected the data we proceeded with the cleaning and refinement of the data, obtaining refined and more precise data frames in the end for each political party. In order to further improve the quality of the data, we also did some data exploration in the end.

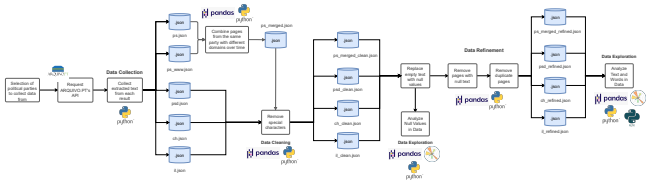


Figure 1: Pipeline Diagram

2.3.2 Data Cleaning

After we finished analysing and preparing the data frame we decided to improve the quality of our data in order to obtain the best results possible. In order to accomplish this goal we noticed that most of our text fields had multiple unrecognized characters and escape sequences that were unuseful for the better comprehending of the text provided, so we removed them.

2.3.3 Data Refinement

After we proceeded with the data cleaning we then decided to more thoroughly refine and treat some more specific aspects of our data frame. For this we removed all the data which had no text field (web pages that provided no information) and we also proceeded to removing any data with corresponding text fields (removing all the duplicates) since there would be no more extra information from those. In the end we ended up with a much more refined and cleaned data frame being much easier to comprehend it.

2.3.4 Conceptual Data Model

As we can see in the picture below, our conceptual model consists of a single main class:

- political_party: each political party has a name and an acronym that represents it and also a foundation year that specifies when the party was founded.
- webpage: each political party page has a date that corresponds to the date when it was published, a link to the corresponding web page, the contentLength, which indicates the length of the text obtained, the type of the file and finally the obtained text.

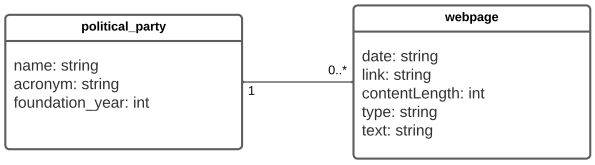


Figure 2: Domain Conceptual Diagram

2.4 Data Characterization

Data characterization was made to analyze information about the data collected and to provide a better understanding of each political parties web content. The exploration was initially performed using a Jupyter Notebook, but to ensure larger granularity in the makefile and an organized storage of the outputs we also performed it using python scripts.

All the characterizations done were applied to each party's dataset and can be obtained by running the correspondent scripts, but for the scope of this report we chose to only show, in the case of single party analysis, the results for IL, since we had all of the party data throughout it's genesis and it has a more representative size that CHEGA, for example.

2.4.1 Null Analysis

Globally, the data retrieved had no null values since it came directly from the API's responses and we only stored the successful ones. Considering that the goal of our information system is to search the webpage content of each political party, it was important to explore the empty text content present in each dataset. As we can see in table 5, CHEGA was the party with the highest percentage of pages with empty text at 68%, contrasting with PSD, which had the lowest, at 23%. We also wanted to explore the correlation between the webpage type and the empty text values, as that could explain why a lot of the information retrieved had no text. As we can observe in figure 3, almost all of the application/json webpages have empty text, and only a small amount of text/html webpages are empty, so only the ones that are of a textual type are worthy of performing text and word analysis.

	Missing Data	Percentage
PS	22922.0	51.590106
PSD	6994.0	22.885377
CH	2380.0	68.058336
IL	6378.0	47.650355

Table 2: Missing Data per Party Webpage

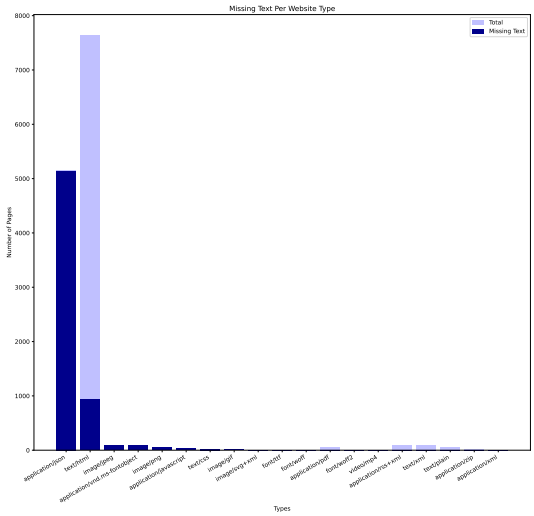


Figure 3: Missing Text In IL

2.4.2 Text Analysis

In order to grasp the amount of data collected for each party and compare the text between political groups, we analyzed the amount of pages and the text length for each one. This improved our understanding of the data and it's quality.

In table 3 we can see the average number of webpages retrieved for each party per year, and their minimum and maximum values. CHEGA is a good example of the ARQUIVO.PT [1] limitations because it is a relatively new party, with genesis in 2019, and since ARQUIVO.PT[1] lacks in updated information in recent years, searching and obtaining data about new parties is rather difficult. In contrast, older parties like PSD, have larger amounts of data.

	Avg Number of Pages	Min	Max
PS	2637.8	6.0	4967.0
PSD	3481.8	1.0	7203.0
CH	117.4	0.0	457.0
IL	1095.6	0.0	2628.0

Table 3: Pages per Party per Year

In Figure 4, we can see the total amount of pages per year by each party and the limitation highlighted before is clearly present

in all of them, as there is barely any data present in 2021. PS and PSD are the most active in each of their webpages, and CHEGA is the least active.

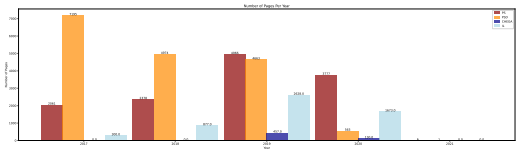


Figure 4: Pages per Year

The average webpage length allows us to understand some of the text quality in each party. For example, CHEGA is the party with the smallest amount of pages, but the second highest in text length , which shows a tendency in writing less but more extensive pages by this party. IL is the party with the smallest text length, contrasting with PSD, which is the highest.

	Avg Length	Min	Max
PS	4679.444214	1	735766
PSD	66950.572294	2	1185007
CH	19425.709830	39	153538
IL	4400.423930	2	418327

Table 4: Text Length per Party per Year

Textual analysis makes us differentiate the style of broadcasting information by each party, as well as recognize gaps in our data and their future implications in the long run.

2.4.3 Words Analysis

When characterizing a dataset that includes text it can be relevant to see the frequency of the use of each word in the text. In our particular context, we expected the most frequent words to point out a theme in the discourse of each party. To do this analysis we plotted a word cloud and a bar plot, as can be seen in Figures 5 and 6.

Figure 5: WordCloud IL

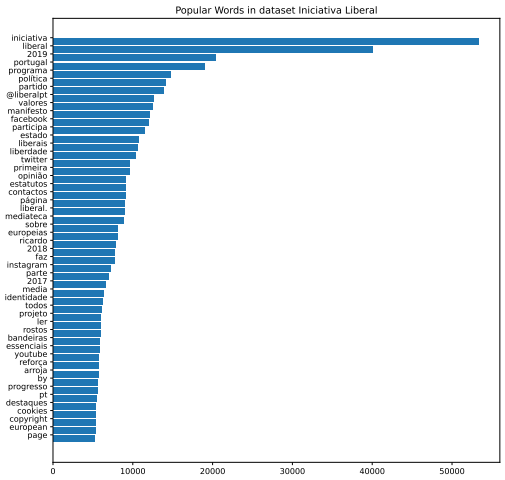


Figure 6: BarPlot IL

Still on the topic of word analysis, since these parties are in constant competition, we considered it would be interesting to analyse the number of times the other parties are mentioned in each party's website. The results can be seen in the example expressed by Figure 7.

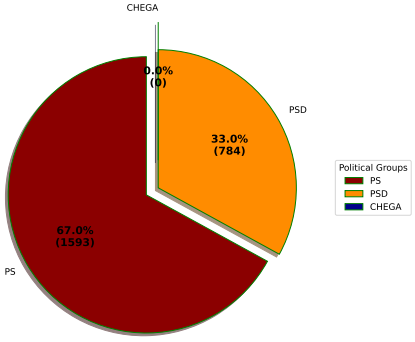


Figure 7: Mentions of Other Parties in IL

2.5 Prospective Search Tasks

In later stages of development, we intend that our search system is able to answer to some information needs like these:

- Qual foi o candidato do Bloco de Esquerda às Eleições Presidenciais de 2016?
- Grupo parlamentar do PSD em 2019
- Programa eleitoral da Iniciativa Liberal para as Legislativas de 2019

- Posição dos partidos sobre o aborto
- Posição do PS sobre a eutanásia

3 Milestone 2 - Information Retrieval

The information retrieval step consists of the implementation and use of a retrieval tool on the project datasets and its exploration with free-text queries. It involves the view of the datasets as collections of documents, the identification of a document model for indexing, and the design of queries to be executed on the indexed information. Furthermore, the results of the specific information needs of the user are evaluated by how relevant they were answering the need.

In the last milestone we only considered the 4 parties with the biggest number of parliamentary seats, for the last 5 years, due to API constraints, but for this milestone we were able to collect data from ARQUIVO.PT regarding all parties since their year of foundation. Because of that, the original refined dataset increased majorly in size, and that was a factor we took into account when collecting, indexing, querying and evaluating our information system.

3.1 Tool Selection

There are multiple information retrieval tools that would allow us to document, index and query our data, like `ElasticSearch`[2] or `Solr`[9]. We ended up choosing `Solr` as our retrieval tool since we were familiar with it and it provided features like full-text search, hit highlighting, faceted search, real-time indexing, dynamic clustering, database integration and rich document handling. Since it is more text-oriented and our data is heavily textual, for this project, `Solr` met our information retrieval needs better.

3.2 Collection and Indexing

In order to search and extract documents from the information system, we need to import data, index it and further query specific rules for each field.

3.2.1 Document Definition

At the end of the data preparation milestone, we had a clean and refined dataset for each party, since their inception date, totaling up to about 2.16GB of data. Despite, having already prepared the data in the first delivery, in this milestone we had to alter it slightly, in order to explore `Solr's` features to their full extent. The transformations performed include:

Alter the date format: conversion of the 'date' field from epoch to the ISO8601 format

Add title field: extraction of the title field from the webpage's link. We did this by considering that most pages include the title in between the last backlashes of their URL (eg: in a link like `https://arquivo.pt/wayback/20171221064558/https://iniciativaliberal.pt/coluna-liberal-acores-caminho-do-resgate-financeiro/` the title field would be `"coluna liberal acores caminho do resgate financeiro"`). We opted for this approach because we didn't collect the title field from the API. However, we intend to do that for the next milestone, in order to have more accurate titles.

Add party field: insertion of a new party field in each object, according to the party the webpage belongs to.

Adding new fields to those we previously had allowed us to boost results based on those particular fields, since both the title and the party can help narrow down important results.

All the documents were indexed in a single collection where the information necessities will be queried upon.

3.2.2 Indexing Process

`Solr` indexing configuration is specified in a schema file that assigns each field a custom type and a set of operations to be performed during indexing and querying. It is uploaded using the following endpoint:

```
curl -X POST -H 'Content-type:application/json' \
    --data-binary @/data/schema.json \
    http://localhost:8983/solr/parties/schema
```

3.2.2.1 Indexed Field Types

Custom field types were created for each schema field. These new fields encapsulate `Solr's` default fields, such as the `dateParameter` (`DatePointField`), `strParameter`(`StrField`) and `intParameter` (`IntPointField`). The `textParameter` makes use of the TextField class as well as tokenizers and filters for indexing and processing both the documents themselves and the queries:

```
"name":"textParameter",
"class":"solr.TextField",
"large": true,
"stored": true,
"multiValued": false,
"charFilters":{"class":"solr.HTMLStripCharFilterFactory
    "},
"indexAnalyzer":{
    "tokenizer":{
        "class":"solr.StandardTokenizerFactory"
    },
    "filters":[
        {"class":"solr.ASCIIFoldingFilterFactory", "
    preserveOriginal":true},
        {"class":"solr.LowerCaseFilterFactory"},
        {"class":"solr.PortugueseStemFilterFactory"}
    ]
},
"queryAnalyzer":{
    "tokenizer":{
        "class":"solr.StandardTokenizerFactory"
    },
    "filters":[
        {"class":"solr.ASCIIFoldingFilterFactory", "
    preserveOriginal":true},
        {"class":"solr.LowerCaseFilterFactory"},
        {"class":"solr.PortugueseStemFilterFactory"}
    ]
}
```

The `HTMLStripCharFilterFactory` is used to remove any HTML code and annotations that the extracted text still might have.

For the indexation, the standard tokenizer is used along with the `ASCIIFoldingFilterFactory` that converts alphanumeric and symbolic Unicode text to it's ASCII equivalents; the `LowerCaseFilterFactory` that converts any uppercase letters to lowercase; and the `PortugueseStemFilterFactory` that reduces words to the same root.

As for the querying, the standard tokenizer is also used, as well as all the filters applied to the indexation. For the next iteration, we might consider adding some synonyms and stopwords filtering.

3.2.2.2 Schema Fields

All the data fields were analyzed to check which ones should be indexed, and we decided the `link` and `contentLength` shouldn't be indexed since they wouldn't be the subject of any queries. All the fields were stored.

Field	Type	Indexed
text	textParameter	Yes
title	textParameter	Yes
link	strParameter	No
type	strParameter	Yes
party	strParameter	Yes
contentLength	intParameter	No
date	dateParameter	Yes

Table 5: Schema's fields, types and indexation

The `title` and `text` fields are indexed and queried according to the definition of the `textParameter` custom type referenced in section 3.2.2.1. The `link`, `type` and `party` were defined as string since they are single string values. The `date` was defined as date so we can filter our results by time periods.

3.3 Retrieval

In order to check and evaluate the performance of the system, 5 different information needs were created based on the prospective search tasks presented in section 2.5. For each information need, we associated a query that took advantage of some `Solr's` features in order to refine and improve search results. Those same results were compared and evaluated for each query in section 3.4.

While analyzing `Solr's` available query parsers, we ended up choosing the `Extended Dismax Query Parser`[4] in detriment of the `Standard Query Parser`[8] and the `Dismax Query Parser`[3], since it incorporates and extends all the features from both of the previous parsers. It has improved proximity and boost functions, smaller partial escaping and allows for the specification of which fields to query and the definition of their respective weight.

3.3.1 Boosts

Boosts represent the process of giving higher relevance to a set of documents over others and `Solr` has different ways to implement them, using `Term` and `Field` boosts, as well as some `Independent` Boosts like the `Query` and `Function` ones. We used `Term Boosts` to prioritize query terms that had greater importance over the others, like in Q3. `Field Boosts` were used in every query for the `title` field, since titles that contain the query search arguments are highly likely to represent the information we're looking for.

3.3.2 Proximity Search

A proximity search looks for terms that are within a specific distance from one another, in any direction, and considers the results a match if they fall under a specific number of movements required to fulfill the search task. In our case, it was only relevant to guarantee a proximity search in cases where a phrase match is applied, so we used the `'slop'` value to define the proximity distance instead of just specifying it with the tilde (~) operator. Results that contain the search terms within a phrase will be boosted, up to a certain `'slop'` value. In Q2, results where the query terms are together at a maximum distance of 100, are prioritized over ones where the query terms are scattered.

3.3.3 Wildcards and Fuzziness

Wildcard searches use single (?) and multiple (*) character operators to accept results with words that have slightly different variations. Fuzzy searches accept similar words to the term without necessarily being an exact match. Since both searches are based on term stemming and we already apply a stemming filter in our schema, doing these searches is irrelevant. Moreover, if not applied correctly and taking into consideration every variation of each word, wrongful results might be frequent since words with completely different meanings are being matched.

3.3.4 Queries

From the available parameters from the Solr query parser `Estended DisMax` we used the following to formulate our queries:

(1) q - represents the main query on which we are performing the search.
(2) q.op - represents the operator used in the query expressions
(3) qf - list of fields whom importance gets determined by a boost factor associated with each one of them.
(4) fq - defines a query that is used to restrict the superset of documents that can be returned, it is mainly used to represent date periods or specific political parties.
(5) ps - represents the phrase 'slop', which is the distance between the terms of the query while still considering it a phrase match.

The information needs presented in section 2.5 were translated to the following queries:

(1) **Grupo parlamentar do PSD em 2019**
q.op=AND
query=grupo parlamentar
qf=title^5 text
fq=party:PSD
fq=date:[2019-01-01T00:00:00Z TO 2019-12-31T23:59:59Z]

(2) **Qual foi o candidato do Bloco de Esquerda às Eleições Presidenciais de 2016?**
q.op=AND
query=candidato presidencial
qf=title^5 text
ps=100
fq=party:BLOCO
fq=date:[2015-01-01T00:00:00Z TO 2016-02-01T23:59:59Z]

(3) **Programa eleitoral da Iniciativa Liberal para as Legislativas de 2019**
q.op = AND
query = programa^3 eleitoral legislativas^3

qf=title^5 text
ps = 30
fq = party:IL
fq = date:[2019-01-01T00:00:00Z TO 2019-12-31T23:59:59Z]

(4) **Posição dos partidos sobre o aborto**
q.op = AND
query = aborto
qf=title^10 text

(5) **Posição do PS sobre a eutanásia**
q.op = AND
query = eutanásia
qf = title^10 text
fq = party:PS

3.4 Evaluation

In order to evaluate recall measures we needed to know the complete set of relevant search results among all documents for each query, and since we have a very large dataset (242 890 documents), we would've had to analyze each document for every query, which isn't feasible.

To solve this issue, initially, we created a sample dataset by sampling 200 random documents from each party's dataset, adding up to a total of 1600 documents. However, due to the nature of our dataset (which is very sparse in terms of content), querying a random amount of results for each party wasn't fruitful and in most cases didn't produce any relevant results for the context of the queries. Each party's dataset has drastically different amounts of documents available (more recent parties like CHEGA or IL have a lot less data than older parties like PCP or PSD) and we had no guarantees that specific subjects like 'aborto' or 'grupo parlamentar' would even be present in a random and small sample.

Our second approach was then to do some simple queries over the whole dataset in order to manually select some relevant results that fulfilled the information needs for each query. We incorporated those results in our randomly selected sample (200 documents for each party), so that we could guarantee that our sample had at least some relevant results (true positives), that can then be used to evaluate our retrieval process. This also aids manual evaluation since we only have to judge the relevancy of at most 1600 documents instead of the whole dataset and we already know we should expect the results to at least match the ones we manually inserted.

We evaluated the results using multiple performance metrics over two different systems:

(1) Non-indexed Sample
(2) Indexed Sample

We also used two different sets of queries in each system, simple queries - without boosts and proximity searches - and complex queries (section 3.3.4) - with boosts and proximity searches.

3.4.1 Performance Metrics

In order to evaluate the performance, we calculated the following performance metrics:

Precision (P): fraction of relevant documents among the retrieved ones
Recall (R): fraction of relevant documents that were retrieved
F-Measure (F): measure that combines precision and recall
Average Precision (AvP): measure of quality across recall levels for a single query.
Mean Average Precision (MAP): average value of AvP that measures the quality of the system.

We also retrieved the precision values for the 10 first results, since it is the typical number of results in common search engines.

3.4.1.1 Non-Indexed System

Looking at table 6, right away, we can detect the flaws of a non-indexed system by looking at Q2's performance metrics. The results are all 0 because the system couldn't retrieve any results that matched the search query exactly. This highlights the importance of stemming since the reason for the lack of results was that Bloco de Esquerda had a female candidate running in the presidential elections and therefore, mentions in their webpages regarding that information need will contain the word "candidata" instead of "candidato".

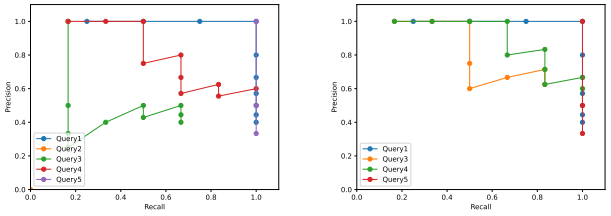
Queries	Precision	P@10	AvP	F Measure	Recall
Q1	0.057	0.4	1	0.108	1
Q2	0	0	0	0	0
Q3	0.384	0.4	0.6	0.526	0.833
Q4	0.065	0.6	0.837	0.123	1
Q5	0.333	0.1	1	0.5	1

Table 6: Performance Metrics for Simple Queries using a Non-indexed system

Although we are still working with a non-indexed system, we can see a significant improvement in results just by using more complex search queries with boosts and proximity search, as we can see on Table 7.

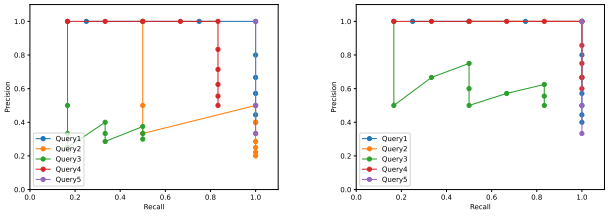
Queries	Precision	P@10	AvP	F Measure	Recall
Q1	0.2	0.4	1	0.333	1
Q2	0	0	0	0	0
Q3	0.625	0.5	0.876	0.714	0.833
Q4	0.065	0.6	0.916	0.123	1
Q5	0.333	0.1	1	0.5	1

Table 7: Performance Metrics for Complex Queries using a Non-Indexed system



(a) simple queries (b) complex queries

Figure 8: Precision Recall Curves for All Queries using a Non-indexed System



(a) simple queries (b) complex queries

Figure 9: Precision Recall Curves for All Queries using an Indexed System

3.4.1.2 Indexed System

While analyzing table 10, as expected the indexed system outperforms the non-indexed system in every category. Although the precision values might seem low at a glance, it should be noted that this value can be attributed to the low amount of relevant results present in the sample dataset. If a dataset only contains 1 relevant document, as is the case for Q5, even if the document is present in the first 10 search results, precision will always be low (given there are also non-relevent results, as is common in real-life scenarios). A more relevant indicator of performance is the Average Precision, which is considerably high for every search query. Recall is also high in both simple and complex queries, which is a good indicator.

Queries	Precision	P@10	AvP	F Measure	Recall
Q1	0.057	0.4	1	0.108	1
Q2	0.125	0.2	0.75	0.222	1
Q3	0.25	0.3	0.591	0.384	0.833
Q4	0.065	0.5	1	0.122	1
Q5	0.333	0.1	1	0.5	1

Table 8: Performance Metrics for Simple Queries using an Indexed system

Although we achieved overall good results, one query stands out for having a lower recall value. In query 3 we couldn't achieve a recall value of 1 regardless of how many tweaks we made to the query. This happened because although the webpage's content is relevant, it is mostly made up of images, which don't provide any information that our search system can use.

Queries	Precision	P@10	AvP	F Measure	Recall
Q1	0.2	0.4	1	0.333	1
Q2	0.666	0.2	1	0.8	1
Q3	0.454	0.5	0.722	0.588	0.833
Q4	0.065	0.6	1	0.122	1
Q5	0.333	0.1	1	0.5	1

Table 9: Performance Metrics for Complex Queries using an Indexed system

3.4.1.3 Mean Average Precision

As a final indicator of performance, we calculated the Mean Average Precision, to evaluate our system's performance as a whole. We concluded that our systems retrieved better results when both the documents and queries were indexed, and that, using complex queries with boosts and other artifacts can severely improve the system's performance.

System	MAP
Non-Indexed Simple Queries	0.687
Non-Indexed Complex Queries	0.759
Indexed Simple Queries	0.868
Indexed Complex Queries	0.944

Table 10: MAP Metrics

4 Conclusion

We conclude the second milestone with an indexed information system of the refined data regarding all the parties present in the parliament since each of their genesis. In the future, we plan to improve our system by adding more filters and query analyzers that allow us to more accurately answer the user's information needs.

References

[1] [n. d.]. Arquivo.pt - pesquise páginas do passado! https://arquivo.pt/
[2] [n. d.]. Busca Gratuita E Aberta: Os criadores do Elasticsearch, do elk e do Kibana. https://www.elastic.co/pt/
[3] [n. d.]. The DISMAX query parser. https://solr.apache.org/guide/7_6/the-dismax-query-parser.html
[4] [n. d.]. The extended DisMax (edismax) query parser. https://solr.apache.org/guide/7_6/the-extended-dismax-query-parser.html
[5] [n. d.]. Matplotlib: Visualization with Python. https://matplotlib.org/
[6] [n. d.]. NLTK: Natural Language Toolkit. https://www.nltk.org/
[7] [n. d.]. Pandas. https://pandas.pydata.org/
[8] [n. d.]. The standard query parser. https://solr.apache.org/guide/6_6/the-standard-query-parser.html
[9] [n. d.]. Welcome to Apache Solr. https://solr.apache.org/
[10] Arquivo. [n. d.]. Arquivo.pt API · ARQUIVO/PWA-Technologies Wiki. https://github.com/arquivo/pwa-technologies/wiki/Arquivo.pt-API