European Parliament Data Information Retrieval System

Catarina Figueiredo
up201606334@fe.up.pt
University of Porto
Faculty of Engineering
Porto, Portugal

João Bernardo Sousa
up201606649@fe.up.pt
University of Porto
Faculty of Engineering
Porto, Portugal

Mariana Dias
up201606486@fe.up.pt
University of Porto
Faculty of Engineering
Porto, Portugal

Tiago Ribeiro
up201605619@fe.up.pt
University of Porto
Faculty of Engineering
Porto, Portugal

ABSTRACT

In this article we refined, analysed and queried data collections related to European Parliament data with the goal of making its data more accessible to the general public through building retrieval tools that satisfy information needs. Every step in the process was addressed from dataset gathering, data preparation, natural-language processing and data storage in a relational database. We indexed a collection of two documents using the Apache Solr tool, which allowed us to evaluate the results. Two information needs were used in three types of systems that were dependent on indexing and weighting operation. The best result obtained (MAP=0.55) was with the systems that performed only indexing or both indexing and field weighting. An ontology based on existing ontologies was created and populated using the Protégé tool. Several retrieval tasks were answered using SPARQL queries with satisfactory results. Information retrieval and semantic web tools were analysed and compared as a conclusion.

KEYWORDS

Data Preparation, European Parliament, Information Retrieval, Ontology, Semantic Web

ACM Reference Format:
Catarina Figueiredo, João Bernardo Sousa, Mariana Dias, and Tiago Ribeiro. 2021. European Parliament Data Information Retrieval System. In *DAPI*. FEUP, Porto, Portugal, 11 pages.

1 INTRODUCTION

Over the years the volume of information has been growing and become accessible to all via the World Wide Web. The organization and structuring of information can bring many benefits to users, such as the dissemination of knowledge and relevant data. However, the emergent information has steadily become more unorganized and sparse making its research an arduous task.

Likewise the information provided by the European Parliament is very sparse, making it hard to analyse and search relevant information. For example, it is impossible to understand the trajectory of each European Parliament member. How did their work, voting behaviour and topics of interest change over time? Can we understand how political groups and individual politicians vote depending on the topic being voted, who submitted the proposal and other relevant factors?

For this reason, we decided to organize the data and make it available in a new interface with more advanced search parameters and new ways of visualizing possible patterns and other useful information.

The processes of collecting, preparing, characterizing, storing, indexing and querying data related to the European Parliament are described in the following sections. In Section 2, the data extraction, preparation and storage are detailed, while in Section 3 documents are indexed and queried using the searching tool Apache Solr. Then in Section 4, existing ontologies relating to the domain are identified and explored resulting in the creation of an ontology of our own that was populated and queried using the Protégé tool and SPARQL query language. Finally, Section 5 reflects a final consideration over the work done.

2 DATASET PREPARATION

This section describes the process related to data extraction and preparation for further usage, including the tools that were used, as can be seen in Figure 1.

2.1 Data Collection

For this project we used two sources: **Parltrack** [1] and the **European Parliament website** [2].

Regarding the European Parliament website, we employed scraping techniques, such as HTML parsing, to extract relevant text data from reports, e.g. its text, rapporteur(s) and committee, if any.

As for Parltrack, it is an European initiative that aggregates information from various official EU sources and releases it in JSON format. It provides a huge amount of data so we decided to focus our efforts in a subset of the dumps [3] provided, namely **Members of the European Parliament (MEPs)** and **MEP Plenary Votes**.

The MEPs dump contains information on all the current and previous members of the European Parliament since 2004, including their names, age, country, political groups affiliation, national party affiliation, committees they were or are part of and their social media info, while the MEP Plenary Votes dump contains information on the votes cast by MEPs in the plenary (in favor, against or abstention) and information on what is being voted.

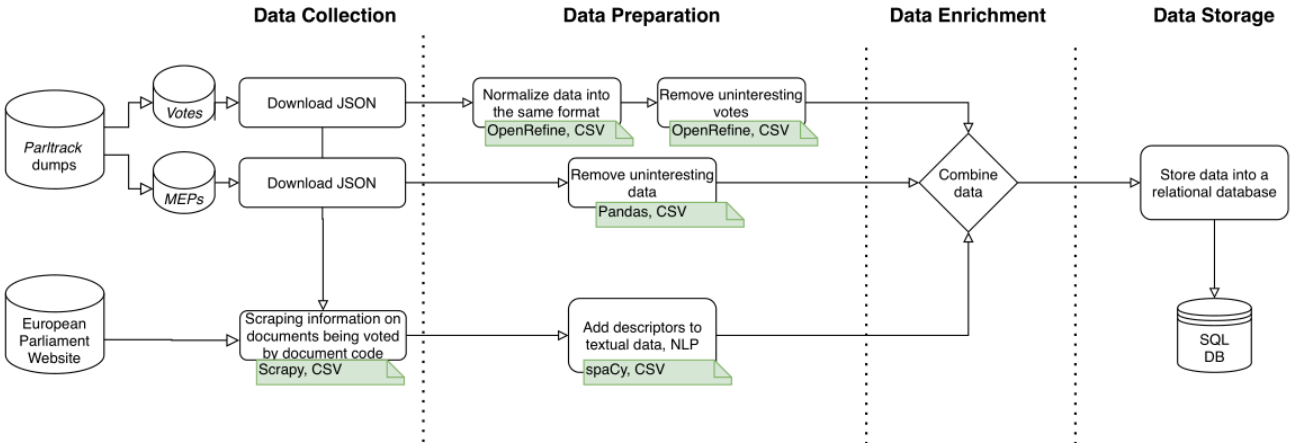

Figure 1: Data Pipeline Diagram.

2.2 Data preparation

In this stage we used **OpenRefine** [4], a specialized tool for data cleaning. We started by normalizing the data provided by the *Parltrack* dumps. Political group names' were normalized, since some of them were abreviated and others not, and the names refering to the same group were often in different languages. Insertion errors in the committees names were also fixed. These errors were mostly related with the arbitrary usage of single and double quotes interchangeably. Finally, we removed data that was not relevant for our project. The dumps including much more information that is not relevant for the goals of this project, for example the *Curriculum Vitae* of the MEPs. We also greatly reduced the number of plenary votes because we decided to include only final votes on final proposals, removing all the votes on amendments and paragraph changes.

The **Parltrack** JSON dumps contains, for each plenary vote, only some basic information regarding the corresponding resolution. On the other hand, the **EP** website contains the full text of each document, complete with the author and respective committee. In order to cross these two data sources, the following **Python** libraries where used: for each vote in the **Parltrack** dump, the respective document code was used to access the EP website URL and download the page with the content. Then, by using the **PyQuery** [5] library, the HTML code was parsed. Because the documents' pages don't follow a common structure, some attention was needed to deal with all the inconsistencies between different pages, requiring some extra steps to extract all needed details of each Resolution.

In the end, all the needed information was combined into a single **Pandas** [6] dataframe, ready to be stored.

To prepare and add descriptions to textual data of those reports we used **spaCy** [7]. By using spaCy we were able to extract the most important and common keywords present with the reports. Particularly, we applied the language processing pipeline on the reports by joining all the reports' texts, segmenting them into tokens, detecting and labeling named entities.

2.3 Conceptual Model

The main entities of our domain are Committee, Country, **MEPs**, Political Group, **Resolutions** and **Vote** as can be seen in Figure 2.

Each MEP can vote on several **Resolutions** and each can have several **Authors**, or rapporteurs. MEPs can belong to several **Political Groups** since they can change their political affiliation over time, and they are elected to represent a specific **Country**.

Resolutions can be proposed by a **Committee**, while MEPs can belong to several committees as they may change over time.

2.4 Data Storage

In order to make the collected data easily available for the next processing steps, it was decided to store everything in a relational database. To make it possible, the data needed to be normalized. This means ensuring, for example, that each cell contained only atomic information and that all repeated data was centralized into a single table, thus eliminating all data redundancy. This required additional processing of some generated **Pandas** dataframes, in order to normalize information. Some data frames were split, with primary/foreign keys being generated to establish a relationship. In resolutions, some identifier values were not numeric as they presented the following format "dd:mm:yyyy hh:mm:ss-k.", where the id consists of a datetime and a k value to guarantee uniqueness, separated by an hyphen.

The processed CSV files were imported into a relational database with the structured presented in the conceptual model by using the **MySQL** [8] database management system paired with an Kettle ETL (Extraction, Transform and Load) tool - **Pentaho Data Integration** [9] - to perform the storage and loading of the data respectively.

2.5 System Documents

We will have three documents in our system: **MEPs**, **reports** and **committees**.

The MEP document will have a brief bio section of a parliament member's information such as their name, gender, birth date, national party they belong to, their usernames on social media

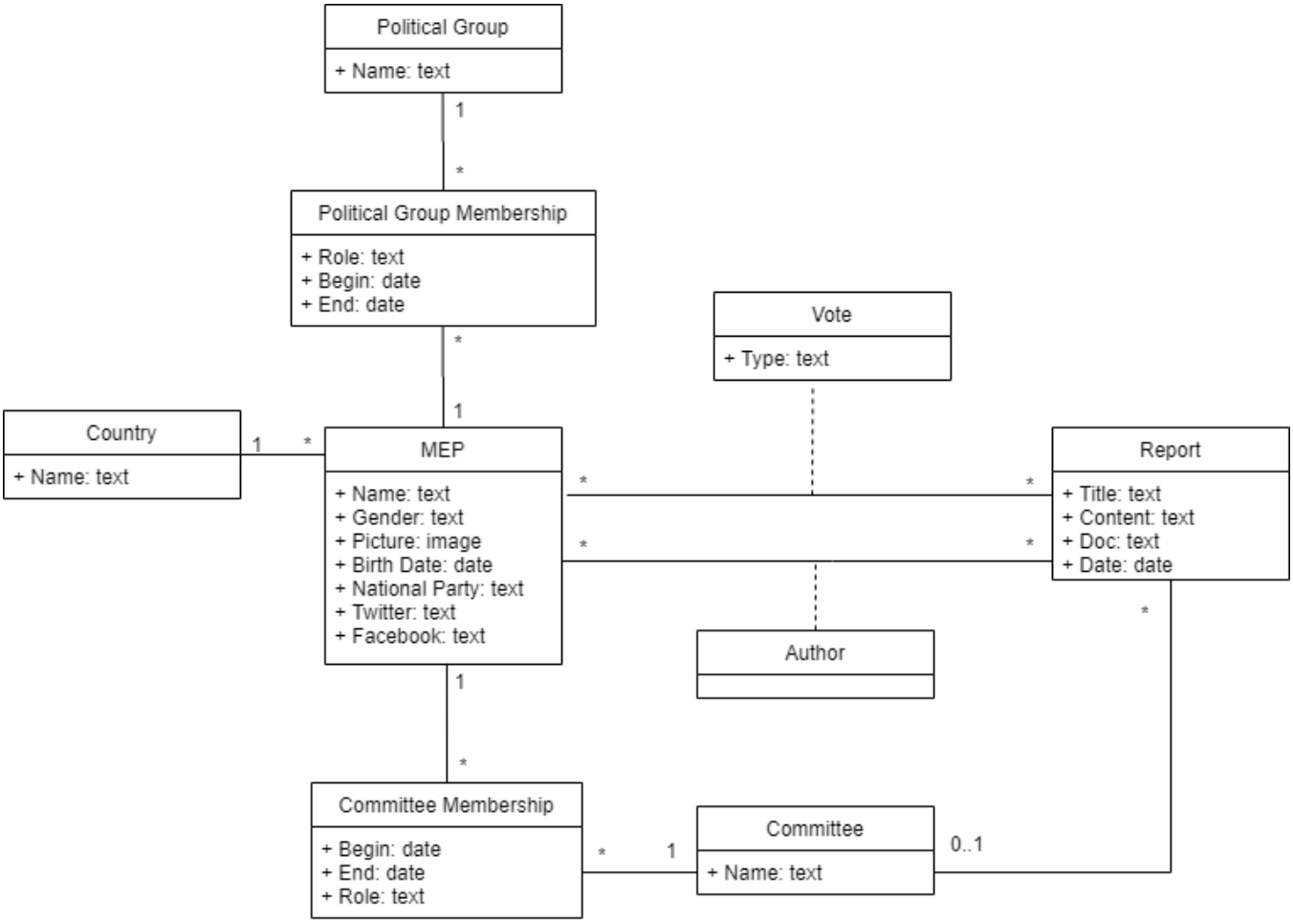

Figure 2: European Parliament's Conceptual Model.

platforms etc. It will also be possible to view the votes cast by them. A user can search for an MEP by their name.

The document Report will have a date, a title, content and voting results (total number of votes in favor, against and abstained, and information of who voted). A user can search for resolutions by its title or by keywords present in its content.

The Committee document will have the name of the committee and a list of reports motioned by it.

2.6 Collection characterization

2.6.1 ***Reports.*** The information contained in 6,396 reports, produced from 2004 to 2020, is being used. We have got this number after heavily filtering all the reports to include only final versions of them. The number of reports per years is very variable but we can see in Figure 3 that there was an unusual low number of reports during the eight term between 2009 and 2014.

The top 9 most common keywords present in the reports are represented in Figure 4. The most used terms are *Council*, *EU* and *Commission*.

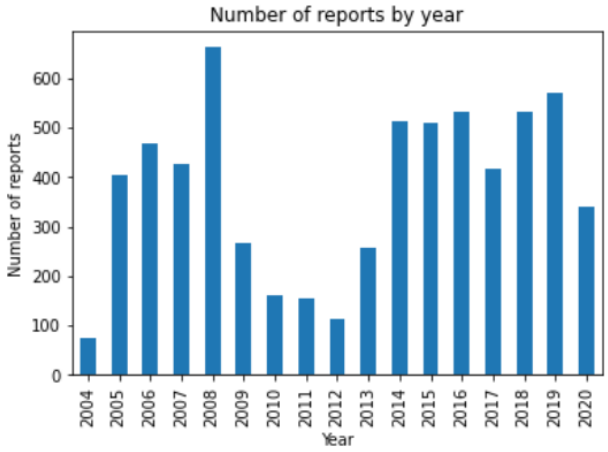

Figure 3: Number of reports per year.

Text normalization such as removing stopwords before applying the language processing pipeline was performed. Initially, we

considered extracting noun chunks with no regard to the number of words considered as a term as a means of exploring the content. After analysing the results, we decided to limit the keywords extraction to trigrams as it resulted in a cohesive ranking of entities.

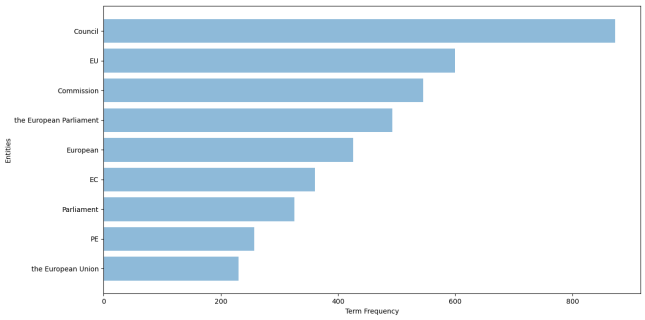

Figure 4: Term frequency of entities in reports.

Most of the reports are produced in the context of a committee, as can be seen in Figure 5. There are many however that are produced in other contexts, e.g. written by an individual MEP.

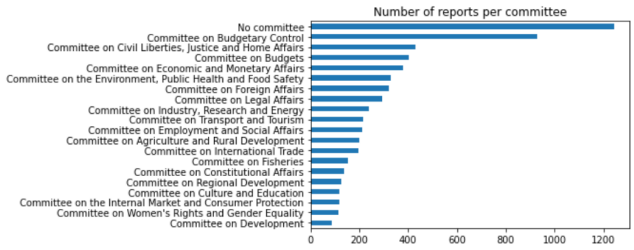

Figure 5: Number of reports per committee.

2.6.2 ***Meps.*** The information on 4,150 MEPs that were at some point part of the parliament from 2004 to 2020 is being used.

The activity of MEPs in terms of number of votes cast is very variable, as seen in Figure 6. One of the reasons for this is that there are many MEPs that did not spend an entire term - five years - in the parliament. Some of them stayed for a time period inferior to a month, as to replace a temporarily absent MEP for a few weeks. Figure 7 shows the distribution of gender for the entire set of MEPs. Historically, men have outnumbered women in the European Parliament by a very large margin.

2.6.3 ***Committees.*** The information on 32 committees is being used. Committees are characterized by the policy topic(s) they refer to.

2.7 Data Retrieval tasks

Our results will be focused on the previous documents listed: **MEPs**, **Committees** and **Reports**. Starting with the MEPs, the system will present their personal information, the votes they have cast, political groups they have been part of organized by periods of time and all the committees they have been part of, also organized by period of time. For the committees, the data presented will be about

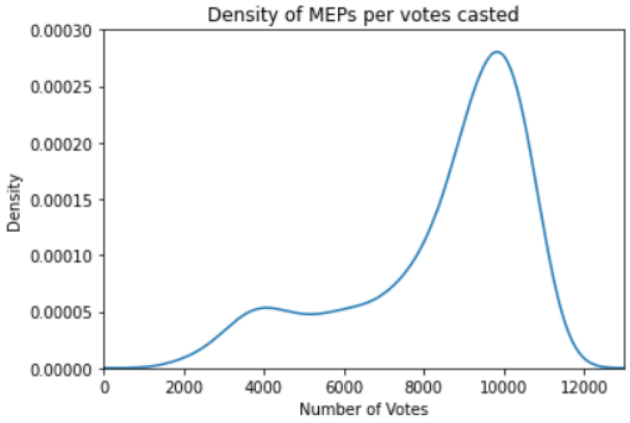

Figure 6: Density of MEPs votes.

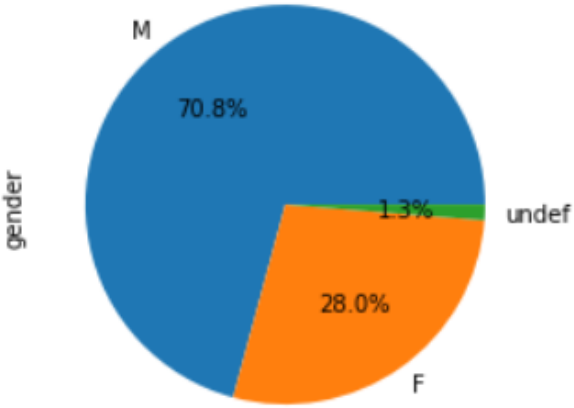

Figure 7: Gender distribution.

all the reports produced by it and all of its members. Finally, for the reports, the system will present its title and text along with the votes cast for it by MEP and organized by political group.

Possible Data Retrieval tasks:

- Search for a specific report
- Search for a specific MEP
- Search for MEPs of a certain country
- Search for MEPs of a certain political party
- Search for MEPs of a certain political group
- Get all the votes cast by a MEP
- Get the votes cast on a report

3 INFORMATION RETRIEVAL

Information Retrieval is the process of retrieving documents of an unstructured nature from large collections that satisfies an information need [10].

An experiment was conducted to develop systems that retrieve relevant documents within a collection according to a given information need, i.e., an ad hoc retrieval task. Three types of systems were set up: a control system with indexes to allow searches (System 1), one where filters were applied (System 2) and another where weights were applied (System 3).

To assess the results we calculated and analysed three statistics: Precision at 10 (P@10) and Average Precision (AP). Precision at k measures precision at fixed low levels of retrieved results. However, since its an evaluation measure that does not average well given that the number of relevant documents greatly influences precision at k, Average Precision complements it. AP is the average of the precision values calculated for the top k documents after each relevant document is retrieved.

3.1 Tool Selection

From a variety of tools, we considered the platforms Apache Solr and Elasticsearch. From the literary research [11] conducted, there are advantages and disadvantages to both, even though they are both similar search platforms based in Lucene.

Solr is an established and popular platform, while Elasticsearch is a newer platform, which gives Solr the advantage of having better support. However, since Elasticsearch was created as a response to Solr's deficiencies, it has recently become better liked and more used in comparison to Solr according to an analysis the authors [11] made in regards to data from the DB-Engines ranking in July 2016. Elasticsearch had a rank of popularity of 88.62, while Solr had a rank of 64.69. Currently, as of December 2020, Elasticsearch has a rank of 152.49, while Solr has a rank of 51.24.

Considering our context and data size and disregarding popularity, Solr was chosen as the better suited tool since it has better performance in long data regarding indexing duration and it uses less disk space when comparing the size of data after indexing.

3.2 Collections and documents

Members of the European Parliament (MEPs) and Resolutions make up the documents of our collection. Initially, we had three documents, including Commissions. However, the latter document was removed as it made more sense to merge it as a field, named *committee*, in the Resolution document as a means of searching for resolutions proposed by an arbitrary commission.

3.3 Indexing process

In order to import our data from the relational database to Apache Solr's document representation, the request handler Data Import Handler [12] was configured. In the configuration file for data, the documents and entities were defined, demonstrated in Figure 8, with the MEPs and Resolutions being set by querying the database as to enable the data aggregation and creation of Solr documents.

```
<dataConfig>
    <dataSource type="JdbcDataSource"
    driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://192.168.1.127:3306/dapi"
    user="root"
    password="password"/>
    <document>
        <entity name="mep"
     query="SELECT m.id, m.name, m.gender, m.`national party`,
     m.`birth date`, c.name as country FROM mep m, country c
        where m.country= c.id">
            <field column="id" name="id"/>
            <field column="name" name="name"/>
            <field column="gender" name="gender"/>
            <field column="country" name="country"/>
        <field column="national party" name="national_party"/>
         <field column="birth date" name="birth_date"/>
        </entity>
    </document>
</dataConfig>
```

Figure 8: Code portion for data configuration file that enables import in Solr regarding MEPs.

The indexing process involved the selection of fields we determined would be most useful for searching operations given the information needs we had in mind.

Regarding the MEP document, the text fields name, country and national_party are of the type text_general which applies an analyser upon the document's indexing and querying with a standard tokenizer that splits text fields into tokens by considering whitespace and punctuation as delimiters. It also applies a stop, synonym graph, lower case, Porter Stem, English possessive and hyphenated words filters. The birth date field is indexed and stored as a pdate type. The indexed fields can be consulted in Table 1 and the filters applied to the field types used in Table 3.

Field	Type	Indexed
name	text_general	Y
country	text_general	Y
national_party	text_general	Y
birth_date	pdate	Y
gender	gender_field	Y

Table 1: Indexed fields for MEPs document.

Finally, the field gender was stored as a gender_field type, a custom type, listed in Table 3, that applies a standard tokenizer and both the synonym graph and lower case filters where the values in the database "M" or "F" are searchable by synonyms given, i.e., "male" and "female", detailed in the portion of the synonyms' text file in Figure 9.

```
F,Female,Woman,Women,Feminine,Girl
M,Male,Man,Men,Masculine,Boy,Guy
```

Figure 9: Synonyms in text file for gender_field type.

Considering the Resolution document, the text fields text, content and committee are of the text_general type and doc is stored as a string since there is no need to perform tokenization for exact matches. For the lists of votes, for and against, from the MEPS, another custom field type, listed in Table 3, named comma_list was created. Its analyzer applies a Pattern tokenizer with a pattern that defines commas as delimiters. The indexed fields can be consulted in Table 2

Field	Type	Indexed
committee	text_general	Y
content	text_general	Y
doc	string	Y
mep_against	comma_list	Y
mep_favor	comma_list	Y
title	text_general	Y

Table 2: Indexed fields for Resolutions document.

Type	Tokenizer	Filters
gender_field	Standard	SynonymGraph, LowerCase
comma_list	Pattern	-
pdate	-	-
string	-	-
text_general	Standard	Stop, SynonymGraph, LowerCase, PorterStem, EnglishPossessive, HyphanatedWords

Table 3: Tokenizer and filters in field types used in MEP and Resolution documents.

3.4 Retrieval process

In order to explore and demonstrate Solr's available features and possible weights, we defined two information needs and formulated retrieval queries accordingly. To assess the tasks, the top ten results will be presented with each results' relevance judgement and the analysis metrics identified earlier.

The first information need is "MEPs who voted in favor of resolutions regarding the animal welfare of rabbits that were proposed by the Committee on Agriculture and Rural Development". The query chosen is [committee agriculture rural development rabbits].

It retrieved 5,145 documents. The high amount of documents returned can be explained by the way the query was written. We experimented with adding quotation marks to part of the query referent to the committee and it returned 5 documents.

We used the qf paramater from the DisMax query parser to assign boost factors that increase or decrease fields' importance. In Table 4, there are the boosts chosen after experimentation for the fields title, committee, content and mep_favor related to this query where title and committee are valued as the most important.

Field	Weight
title	7
committee	5
content	2
mep_favor	0.5

Table 4: Field weights for Resolutions document.

Between System 2 and System 3 the ranking results, in Table 5, are similar regarding the relevant documents that are retrieved. However, the non relevant documents in System 3 all refer to the Committee of Agriculture and Rural Development and have a content unrelated to the animal welfare of rabbits, while the non relevant documents in System 2 include documents that do not refer to the Committee of Agriculture and Rural Development. This may be explained given that we assigned a boost of 5 to the committees and 2 to the content in System 3.

Rank	System 1	System 2	System 3
1	1	1	1
2	1	1	1
3	1	1	1
4	0	1	1
5	0	1	1
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0

Table 5: Top 10 results for query [committee agriculture rural development rabbits].

Regarding metrics, System 1 has a Precision @ 10 of 30%, shown in Figure 10, and an AP of 0.63, while both system 2 and 3 have a Precision @ 10 of 50% and an AP of 0.82.

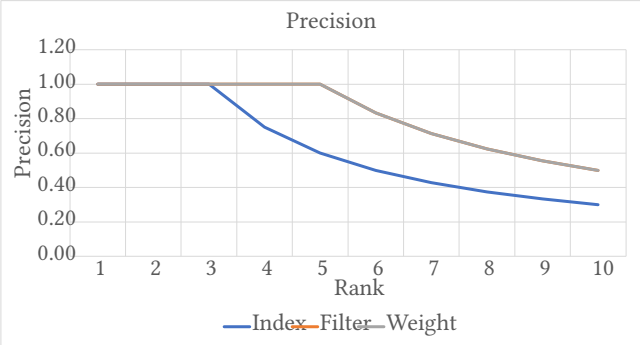

Figure 10: Precision @ 10 for query [committee agriculture rural development rabbits].

Considering our exploration over the different ways to elaborate this query, we can conclude that, with 5 relevant documents returned, both systems 2 and 3 are enough to satisfy the information need.

The second information need is "Portuguese female MEPs in the EP". The query chosen is [portugal female].

For System 1 there are 0 documents returned, since the filter that allows the recognition of "female" as a synonym of the gender field value "F" is not applied. The top 10 results can be consulted in Table 6.

Rank	System 1	System 2	System 3
1	-	0	0
2	-	1	1
3	-	0	0
4	-	0	0
5	-	1	1
6	-	0	0
7	-	0	0
8	-	0	0
9	-	0	0
10	-	0	0

Table 6: Top 10 results for query [portugal female].

It retrieved 1,278 documents for the other two systems. Considering the qf paramater, in Table 7 we can see the boosts chosen after experimentation for the fields title, committee, content and mep_favor where name is the most important followed by country and gender which has a default boost.

Field	Weight
name	5
country	3
gender	-

Table 7: Field weights for MEPs document.

Both systems 2 and 3 have similar ranking results, which is shown in Table 6. That may be explained by our decision to boost the country field by 3, which resulted in the non relevant documents that are male MEPs. Regarding metrics, in Figure 11, both System 2 and 3 had a Precision @ 10 of 0.20 and an AP of 0.28. Considering how this task was chosen specifically to test how the systems interpreted the gender field, the results obtained were unsatisfactory.

Figure 11: Precision @ 10 for query [portugal female].

3.5 Tool evaluation

Despite being a powerful search platform, Solr's lack of relevant documentation made the tasks we performed much more difficult. We spent almost half of this milestone trying to understand how Solr works, how to connect Solr to our database and figuring out why some errors were appearing. If Solr documentation were more complete it could have saved us a lot of time.

The hardest task was understanding how to implement the whole system, specifically how to make a successful connection with our database. The easiest one was assigning weights and performing query text searches.

4 SEMANTIC WEB

Nowadays databases are available on the Web in some form where users and application programs are dynamic. As a result, the data's semantics must be available along side the data itself. For application programs, it means that semantics must have a formal and machine-processable form [13]. That is the goal of the **Semantic Web** - to represent Web content in a more easily machine-processable form.

Semantic Web aims to have applications in more advanced knowledge management systems where knowledge is organized in conceptual spaces according to its meaning, automated tools will be maintained by checking for inconsistencies and extraction of new knowledge and query answering over several documents will take over keyword-based search to present requested knowledge in a human friendly way [13].

In this section we identified entities from our domain and represented them in an ontology using **Protégé**, an ontology tool, which allows them to be machine-readable. Existing ontologies were identified and useful entities were used.

4.1 Data domain and concepts

As the project's theme is European Parliament Data, we searched for ontologies related with the European Union. The various organisms of the EU produce millions of pages of content each year that need to be organized and accessible to all citizens. On those grounds, the **Publications Office of the European Union** [14] is the entity in charge of creating and keeping up vocabularies like thesauri and taxonomies and various data models like schemas and ontologies, among other things, in all EU's 24 official languages.

The Publications Office of the European Union provides three different ontologies - the Common Data Model, the European Research Information Ontology, and the European Legislation Identifier.

The **Common Data Model Ontology** [15] is based on the FRBR model [16], which is a conceptual model for Bibliographic Records created within IFLA [17] to describe entities, relationships, and attributes. This ontology is described by using RDF(S)/OWL technologies, able to represent the relationships between the resource types managed by the Publications Office and their views according to the FRBR model in terms of Work, Expression, Manifestation and Item.

The **European Research Information Ontology** [18] (EURIO) conceptualises, formally encodes and makes available in an open, structured and machine-readable format data about research projects funded by the EU's framework programmes for research and innovation.

At last, the **European Legislation Identifier** [19] is an ontology regarding legislation. This ontology has three major components: Identification of legislation - URI templates at the European, national and regional levels based on a defined set of components, Properties describing each legislative act - Definition of a set of metadata and its expression in a formal ontology and Serialisation of ELI metadata elements - Integration of metadata into the legislative websites using RDFa.

4.2 Existing ontologies

Within the ontologies found, we decided to focus on the **Common Data Model** since it is the most generic and the most related to our project. The selected ontology provides some entities that are useful for our project: Committee, Country, Parliamentary Group and Person. However, we still had to create most of our own ontology, since the Common Data Model was not completely compatible with the information we intended to work with in the context of this project.

4.3 Ontology creation

The ontology developed for this project was created using **Protégé** [20], an open-source ontology editor that provides a graphic interface that is very intuitive and made the ontology creation process simpler.

This process started by the identification of relevant classes in a way that made them as compatible as possible with the existing ontologies described in Section 4.2 in the spirit of open linked data.

There are three main classes: **Report**, **Agent** and **Membership**. The Report class relates to the textual documents produced by the European Parliament, while the Agent class refers to various kinds of political entities which are then discriminated as sub-classes. Finally, there is the Membership class that allows the expression of membership relationships that have initial and end dates associated with them between individuals, such as Committees and MEPs or Political Groups and MEPs. Some of the created classes include the prefix *cdm:*, in the case of being a class that was imported from the previously presented Common Data Model. The implemented classes are visible in Figure 12.

Figure 12: Ontology classes.

The object properties are all new with the exception of *cdm:authors-report*. As our ontology is much more specific than the one found in the Common Data Model, most of the object properties had to be reinvented. Many object properties are the inverse of each other, like *vote_in_favor* and *was_voted_against_by*, that relate a MEP to a Report. All the object properties are asymmetric and irreflexive, while *committee_membership_mep, committee_membership_committee, political_group_membership_mep* and *political_group_membership_political_group* are also functional. The object properties can be found in Figure 13.

Figure 13: Ontology object properties.

As stated above for the object properties, only two data properties were reused from the Common Data Model. Those properties are *cdm:title* and *cdm:date-adopted*. The implemented object properties can be found in Figure 14.

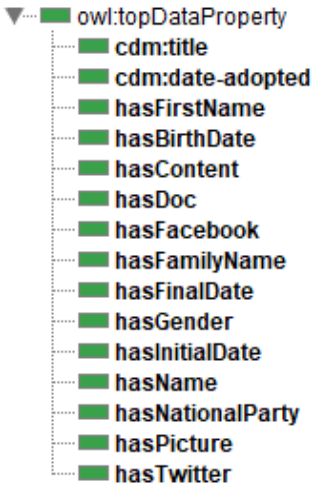

Figure 14: Ontology data properties.

4.4 Ontology population

In order to populate the ontology, the .csv files generated previously on the API scraping process were reused. Each .csv represents an entity or a relationship between multiple ones. All .csv files were merged into a single Excel book with each file being on a separate sheet. The data was treated and optimized for the ontology: a letter was added before each **ID** so that individuals with the same **ID**, but from different entities could be distinguished and most of the votes were removed so that *Protégé* was able to read the file.

An option to use a very reduced set of individuals was considered to not affect *Protégé* performance. In order for this to work, only relationships that involved entities that were removed could also be removed, otherwise most of the relationships present would be useless (between non existent entities). This proved to be impossible to do on Excel due to the huge amount of lines present, so we decided that it would be best to compromise the *Protégé* performance and populate with all the lines, which, while sacrificing the performance, obtains more meaningful results to the queries at the end.

Afterwards, the *Cellfie* plugin was used to import the final Excel file. Several transformations were defined in the *transform.xml* file, converting the several columns of each sheet to individual ones and their respective properties onto the ontology. After processing everything, all individuals were added to the ontology.

4.5 Ontology queries

In this section we will present some examples of SPARQL queries. Apart from the standard ontology queries prefixes, the prefix *cdm:* for the European Union Common Data Model ontology and the prefix *:* for our own ontology were included.

The query in Figure 15 answers the question *What are the names of the committees that the European Parliament has or had?*

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX cdm: <http://publications.europa.eu/resource/cdm/>
PREFIX : <http://www.semanticweb.org/joobernardo
    ↪ /ontologies/2020/11/DAPI#>

SELECT ?committee_name

WHERE {
  ?committee a cdm:Committee .
  ?committee :hasName ?committee_name .
}
```

Figure 15: SPARQL query related to the names of the committees.

The query in Figure 16 answers the question *How many reports did each Member of the European Parliament author?*

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX cdm: <http://publications.europa.eu/resource/cdm/>
PREFIX : <http://www.semanticweb.org/joobernardo
    ↪ /ontologies/2020/11/DAPI#>

SELECT ?mep_name (count(?report) as ?n_reports)

WHERE {
  ?mep a :Member_of_the_European_Parliament .
  ?mep :hasName ?mep_name .
  ?mep cdm:authors-report ?report .
}

GROUP BY ?mep_name
HAVING (?n_reports > 1)
ORDER BY DESC (?n_reports)
```

Figure 16: SPARQL query related to the number of reports for each MEP.

The query in Figure 17 answers the question *What reports did each Member of the European Parliament author?*

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX cdm: <http://publications.europa.eu/resource/cdm/>
PREFIX : <http://www.semanticweb.org/joobernardo
    ↪ /ontologies/2020/11/DAPI#>

SELECT ?mep_name ?report_title

WHERE {
   ?mep a :Member_of_the_European_Parliament .
   ?mep :hasName ?mep_name .
   ?mep cdm:authors-report ?report .
   ?report :hasTitle ?report_title .
}
```

Figure 17: SPARQL query related to the content of the reports authored by each MEP.

The query in Figure 18 answers the question *How many Members of the European Parliament are there for each gender?*

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX cdm: <http://publications.europa.eu/resource/cdm/>
PREFIX : <http://www.semanticweb.org/joobernardo
    ↪ /ontologies/2020/11/DAPI#>

SELECT ?gender (count(distinct ?gender) as ?count)

WHERE {
   ?mep a :Member_of_the_European_Parliament .
   ?mep :hasGender ?gender .
}

GROUP BY ?gender
```

Figure 18: SPARQL query related to the number of MEPs by gender.

4.6 Evaluation

Protégé is the main tool used in Section 4. It is a very intuitive tool since almost every task that needed to be done has the ability to be completed using the graphical user interface. The interface is divided in a clear way with different tabs for different elements of the ontology or tools (like query tools). The usage of different colors for classes, object properties and data properties is particularly clever and helpful.

However, this tool is not adequate for the population of ontologies with thousands of individuals since it takes a very long time. The heavy memory usage caused by this also makes it very hard to work with and query on an ontology with the described characteristics.

The results for all queries were as expected.

4.7 SW and IR tools evaluation

As discussed previously in Section 3.5 and Section 4.6 where Information Retrieval and Semantic Web tools were respectively analysed in terms of advantages and limitations, the main pros and cons will be used for comparison of both tools in this subsection.

The population process using Semantic Web tools was arduous given the large volume of data and proved to be a big step back for us. Previously, information was structured directly into a relational database where the population process was simpler. However, it was much more complicated to define the concepts, structure and constraints. When using ontologies, information can be defined much more easily and accurately before building the database. The built ontology could serve as a base for an improved database on the search system. Therefore both areas of Information Retrieval and Semantic Web are seen as complementary and useful in different aspects of our project.

Semantic web and Information retrieval and not opposite to each other. Instead, they are concepts that complement each other.

In the beginning of this project, the data scraped from the REST API was directly inserted into a relational database built from scratch. This revealed to be a difficult task, as manually defining all entities, relations, constraints and other rules on a relational database is a fairly complex and error-prone process. If instead, a ontology was first defined in a tool like *Protégé*, which has a much easier interface to use and allows for a much more precise representation of the data structure, the process would then be much straightforward, being only needed a conversion of the ontology to a relational model.

Information retrieval tools are focused in making relevant information available as fast and efficiently as possible, while Semantic Web is more dedicated to formally define the structure of this same data. Trying to use *Protégé* to retrieve information will result in a slow performance as critical features like indexing are not available. In a similar fashion, *Solr* doesn't have many options to formally define the data structure that is indexed. This shows the complementary nature of both concepts. Both are needed in a complete Information System.

5 CONCLUSIONS

In this article we explored a system that returns information related to the European Parliament. From Data Preparation to Information Retrieval and Semantic Web, the data has been cleaned, stored, indexed and analysed in order to answer to information needs of users. The results obtained through querying the indexed documents and calculating statistics, were mostly as expected, although it made us question our method of ranking we assigned by the boosts applied. Perhaps a more thorough experimentation of different boosting options could better our results. The produced ontology with *Protégé* allowed for a much better understanding of how the European Parliament data is structured. The results of querying in order to answer retrieval tasks were as expected and allowed us to compare Information Retrieval and Semantic Web tools regarding our project. Ontologies give a much clearer picture over the different entities and how they relate to one another, in addition to the applied

rules and constraints. The tools used, disregarding the population process, were very intuitive and simple, unlike the Information Retrieval tool used.

REFERENCES

[1] *ParlTrack*. Accessed on 01-10-2020. URL: https://parltrack.org/.

[2] *The European Parliament website*. Accessed on 01-10-2020. URL: https://www.europarl.europa.eu/portal/en.

[3] Parltrack. *Dumps*. data retrieved from Parltrack, https://parltrack.org/dumps. 2011.

[4] *OpenRefine*. Accessed on 09-10-2020. URL: https://openrefine.org/.

[5] *PyQuery*. Accessed on 09-10-2020. URL: https://pythonhosted.org/pyquery/.

[6] *Pandas*. Accessed on 09-10-2020. URL: https://pandas.pydata.org/.

[7] *spaCy*. Accessed on 09-10-2020. URL: https://spacy.io/.

[8] *MySQL*. Accessed on 13-10-2020. URL: https://www.mysql.com/.

[9] *Pentaho Data Integration*. Accessed on 13-10-2020. URL: https://help.pentaho.com/Documentation/8.1/Products/Data_Integration.

[10] Di Worth. "Introduction to Modern Information Retrieval, 3rd Edition". In: *Australian Academic and Research Libraries* 41.4 (2010). ISSN: 00048623. DOI: 10.1080/00048623.2010.10721488.

[11] Tuncay Aydoğan, Muhammer İlkuçar, and Mustafa Ali AKCA. "An Analysis on the Comparison of the Performance and Configuration Features of Big Data Tools Solr and Elasticsearch". In: *International Journal of Intelligent Systems and Applications in Engineering* 4.Special Issue-1 (2016), pp. 8–12. ISSN: 2147-6799. DOI: 10.18201/ijisae.271328.

[12] *Data Import Handler*. Accessed on 06-11-2020. URL: https://cwiki.apache.org/confluence/display/SOLR/DataImportHandler.

[13] Grigoris Antoniou et al. *A Semantic Web Primer*. Vol. 24. 4. M.I.T. Press, 2009. ISBN: 9780262012423. DOI: 10.1017/s0269888909990117.

[14] *Publications Office of the European Union*. Accessed on 9-01-2021. URL: https://op.europa.eu/pt/home.

[15] *Common Data Model*. Accessed on 23-12-2020. URL: https://op.europa.eu/en/web/eu-vocabularies/model/-/resource/dataset/cdm.

[16] *Functional Requirements for Bibliographic Records*. Accessed on 23-12-2020. URL: https://www.ifla.org/publications/functional-requirements-for-bibliographic-records.

[17] *International Federation of Library Associations and Institutions*. Accessed on 9-01-2021. URL: https://www.ifla.org/.

[18] *EUropean Research Information Ontology*. Accessed on 23-12-2020. URL: https://op.europa.eu/en/web/eu-vocabularies/model/-/resource/dataset/eurio.

[19] *European Legislation Identifier (ELI)*. Accessed on 23-12-2020. URL: https://op.europa.eu/en/web/eu-vocabularies/model/-/resource/dataset/eli.

[20] *Protégé*. Accessed on 23-12-2020. URL: https://protege.stanford.edu/.