A DIGITAL, SEARCHABLE TRANSCRIPT OF THE PORTUGUESE PARLIAMENT

A PREPRINT

Francisco L. Carvalho*
Department of Computer Science
Tsinghua University
francisco.de.carvalho@tecnico.ulisboa.pt

June 19, 2019

ABSTRACT

Transcripts from the Portuguese Parliamentary proceedings are available from 1975 on only in .pdf format. It's impossible to search them by any other 's factor than date. The role of the media in democracy is to scrutinize the government in the best interested of the people. To facilitate the process of Scrutinizing Parliamentary Discourse, I made a searchable, interlinked, digital transcript of the Portuguese Parliament.

***K*eywords** scrutinizing democracy · searchable transcript · Portuguese Parliament

1 Introduction

Portuguese Parliament debate transcripts are available on the PT Parliament website for debates since 1821, the establishment of the general assembly. Of these, transcripts from 1975 onward are available in .pdf format.

There is also recorded meta-data about the latest parliamentary proceedings that can eventually be matched with the speeches themselves.

Parliamentary proceedings and debates have structure, while the speeches themselves are free-form, creating a favorable environment for analysis with NLP, so creating a database that can used to enable researchers with complex filtering and querying options is very valuable.

As a web app, this can be useful to journalists and other civics-oriented professions to quicken the research and consulting process. This is particularly important as journalism has been particularly negatively affected in the new digital economy, and news organizations boast smaller teams than before.

For other citizens, it's easier to participate in democracy if data is easily accessible and discoverable, and discourse might be more easily grounded in fact by enabling the sharing of text snippets online.

1.1 Related Work

In a non-exhaustive list, I would like to mention the **Ugandan Parliament Watch**[1], which does great work for its country's young democracy, not only by delivering transcripts from the parliament, but also by functioning as hub for vote stats, law proposals and much more.

The **Parliamentwatch Network**[2] consists of independent legal entities – non-partisan and not for profit organizations that promote parliamentary openness, transparency, accountability and dialogue between citizens and elected representatives. This newtwork is a collaboration between member organizations in countries as diverse as Greece, Tunnisia,

*Student of Web Information Retrieval at Tsinghua University

Germany, Yemen, and Morocco. These organizations provide services that are frequently used by politicians to directly interact with their constituents and answer questions.

EU Votewatch[3], a huge figure in democratic oversight, provides all kinds of curated documentation about everything regarding the European Parliament including transcripts, votes, the texts of proposals and Laws passed, and much, much more.

In Portugal, the **Hemiciclo** [4] project, is a website that tracks the voting distributions and results, as well as statistics regarding voting in the parliament, but don't concern themselves with the text. Seeing this feature is missing in the Portuguese civic ecosystem, I decided to move forward with this project.

2 Data

In this prototype, only the transcripts for the XIII legislature were used, from 2015 to 2019. In total, there were 385 plenary days, and total of 167687 spoken interventions. The transcripts usually start with a summary of the session containing the date, the duration, the participants and highlights, before moving on to the body of the text - which has a " **speaker1:** *speech* ; **speaker2:** *speech* " structure.

3 Development

The realization of this project involved 4 distinct phases: *Web Crawling* for the .pdf files; structured *Information Retrieval* from the pdfs into .json files; devloping the *Search* feature; and finally producing a visually pleasing *Web Application*

3.1 Web Crawling

Selenium, BeautifulSoup and python were used to crawl the parliament website and download the pdf files. The website uses a javascript based interface so I had to use browser automation to navigate the menus.

Figure 1: Parliament website javascript interface.

3.2 Information Retrieval

To extract the texts and information from the transcripts, the following pipeline was used:

- pdf -> xml
- xml -> txt
- txt -> json

1. I used a tool called pdf2xml.py [needs source] to convert the files to extremely unstructured xml, where each character had its own element.

2. I wrote a script to convert the xml to txt, removing text formatting, layout elements, and meta-information like publishing notes and links.

3. To obtain the structured data format we are looking for, the scraping of the text has to account for typing mistakes, naming mistakes, and formatting insconsistencies. For example, when speakers are interrupted (sometimes by several people in a row) the "speaker" mark in their continuation consists of "Cont." so these kinds of dynamics have to be dealt with.

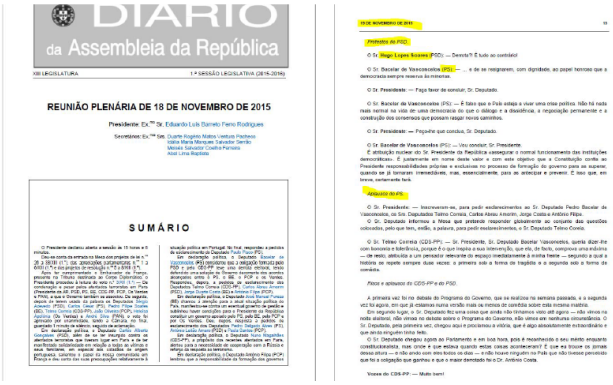


Figure 2: Example of the layout of a .pdf transcript of the parliament.

In the end, we obtain an ordered list in a .json file where each entry is a dictionary with the following parameters.

- Name:
- Date:
- Party:
- Type: [MP, Secretary, President, Secretary of State, Prime-Minister]
- Speech:

The data is then uploaded to a Mongo DB which is to be made accessible as an API for anyone who wishes to process parliamentary discourse.

3.3 Search

For search, I made use of a BM25 ranking function to retrieve the most relevant parliamentary interventions matching queries.

3.3.1 Preprocessing

Before producing the index, I preprocessed the interventions, discarding speeches shorter than 50 characters (usually just interruptions and protests) and speeches by the president and secretary which might occasionaly be useful, but in this case were discarded to speed up the search.

The texts were tokenized, stopwords removed, and words were lemmatized - but not stemmed, as I considered that might remove too much relevant information.

An index was produced as a dictionary with every word as a key and a list of ids of the documents(speech) it appears in.

3.3.2 BM25

In information retrieval, Okapi BM25 is a ranking function used by search engines to estimate the relevance of documents to a given search query. It is based on the probabilistic retrieval framework developed in the 1970s and 1980s by Stephen E. Robertson, Karen Spärck Jones, and others. The name of the actual ranking function is BM25; the fuller name "Okapi BM25" includes the name of the first system to use it, the Okapi information retrieval system.

BM25 is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of their proximity within the document. It is a family of scoring functions with slightly different components and parameters.

Given a query Q, containing keywords $q_1, \ldots, q_n$, the $\mathrm{BM25}$ score of a document D is:

$$score(D,Q) = \sum_{i=1}^{n} IDF\left(q_i\right) \cdot \frac{f\left(q_i, D\right) \cdot \left(k_1 + 1\right)}{f\left(q_i, D\right) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{argd}\right)} \tag{1}$$

where $f(q_i, D)$ is $q_i's$ term frequency in the document D. |D| is the length of D in words, and avgdl is the average document length. k_1 and b_1 are free parameters. IDF is the inverse document frequency weight of the query term q_i, usually computed as:

$$IDF\left(q_i\right) = \log \frac{N - n\left(q_i\right) + 0.5}{n\left(q_i\right) + 0.5} \tag{2}$$

where N is the total number of documents in the collection, and $n(q_i)$ is the number of documents containing q_i.

3.3.3 Results

To execute the searches, the query is preprocessed like the documents and a tfidf vector produced. The matchiing documents are obtained from the index and ranked according to the similarity between the tfidf vectors.

4 Web Application

4.1 Architecture

In this section I will explain the decisions made for the architecture of the web page.

4.1.1 Database: MongoDB

I stored my data in a MongoDB database because it was easy to integrate into the project. Furthermore, evolutions in the project might require unstructured data, which MongoDB can handle unlike traditional SQL databases.

4.1.2 Backend: Flask

Flask is the most popular python framework for serving back end functionality. It's lightweight and has a vibrant community which makes it resilient if support is ever needed. Flask is a good web framework choice because "Flask is considered more Pythonic than the Django web framework because in common situations the equivalent Flask web application is more explicit. Flask is also easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running."

4.1.3 Frontend: Vue.js

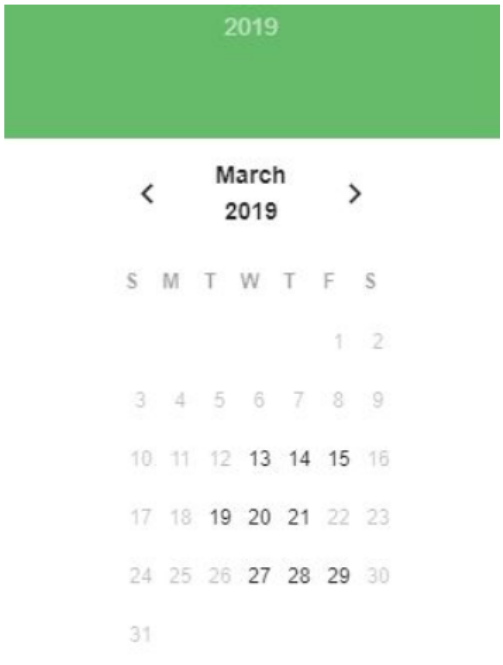
Vue.js is a progressive, component based javascript framework for front-end development. I chose it because it is approachable, versatile, and performant

4.2 Layout

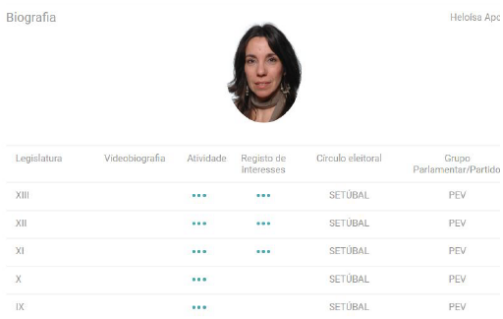
Presenting the information in an easily parseable, accessible way is important for the dual purpose of empowering journalists and engaging the public at the same time. This is why I chose to chronologically display the speeches with a layout that resembles instant messaging apps, with speech cards marked with the speaker names, avatars, and a timestamp, interspersed by narrations of what is happening physically at the time.

4.3 Features

- The front page presents a calendar that highlights the dates where parliament meetings took place, allowing an intuitive **selection of transcripts by date**.



(a) Date selection calendar



(b) MP page in the Parliament website

- **MP personal pages** provide access all of an MPs interventions organized by most recent day, but chronologically within the same day.
- **Term-based search** to find relevant sessions from within the whole dataset.
- **Clickable dates and names** in each speech card to speed up navigation.

5 Future Work

- **API wrapper for the database**: Parliamentary analytics and NLP is a fertile field of research with much work to be done, from NLP to statistic inquiries. An API wrapper for the database for NLP researches to easily access and make complex queries to this previously unsearchable data could kickstart some work in this area.
- **Advanced search**: selecting by time intervals, speakers parties, length, topics. This is an obvious feature, especially for supporting professionals who may use this tool for policy research and government oversight.
- **Medium-like highlighting and sharing**: As a feature, being able to simply select and directly share a snippet of text both enables people to easily share a reference to particular passages in a debate, and drives civic engagement by the fact that it might be entertaining to share particularly funny or controversial passages. Heatmaos for selections could be made persistent among all users, in order to create a sort of "landscape" of highlights across the whole corpus.
- **Wikipedia-like URLs for MP names and laws**: To further facilitate navigation and research, automatically transforming MP names and references to certain laws or proposals into their respective pages and texts into urls can be a very useful measure. With entity recognition technology this can potentially be applied to other types of words like mentioned court cases, or other notable people and members of government.
- **Complete MP profiles** with pictures and statistics about the particular member of parliament.
- Easier "**big-picture" visualization**: Many speeches are very long and frankly quite boring, as they usually involve many formalities, greetings, and procedural language, it would be very useful to be able ton understand an intervention at a glance. Measures could include: Summarization of interventions; Word clouds; Persistant highlight heatmaps.
- Integrated **NLP and statistics**: This is likely out of the scope of my abilities so it might not happen without some kind of funding or collaboration. Interesting high-level projects include: Comparing participation length and frequency with demographic information; Detecting topics discussed and analysing sentiment.

6 Conclusion

The resulting prototype is satisfactory in that it includes all the basic features one could hope from a digital version of the transcripts. They're searchable, they're interlinked, and they're in plain text. There is, however, much work to be done with relative ease, and high potential impact. It is my hope that this can be a useful tool for journalists and researchers alike, and to keep developing it further into a form in which it is approachable to the regular user.

References

[1] Parliament Watch Uganda.
In *parliamentwatch.ug*,

[2] Parliament Watch Network.
In *https://parliament.watch*

[3] VoteWatch Europe: European Parliament, Council of the EU
In *https://www.votewatch.eu*

[4] Hemiciclo, Democracia em tempo real
In *http://www.hemiciclo.pt/*

[5] Stephen Robertson and Hugo Zaragoza (2009).
"The Probabilistic Relevance Framework: BM25 and Beyond".
In *Foundations and Trends in Information Retrieval.* 3 (4): 333–389. CiteSeerX 10.1.1.156.5282. doi:10.1561/1500000019.