Database Project

Parliament Data Extraction

Camille Dubois, Alejandro Jorba, Killian Varescon

Introduction

Goal :

Extract the voting data from different parliament to compare stances on different matters.

Data sources :

- EU Parliament Data : Votes & Member of European Parliament (MEPs)
- UK Parliament Data : Votes & Member of Parliament (MPs)

Table of contents

I. The Data : Access, Structure and Limits

Access :

- UK & EU Parliaments both uses **APIs** to make their data available.
- Wide range of data available :
  - Votes,
  - M(E)Ps information, pictures, party, etc.
  - Debates.

Structure :

- **JSON** and XML response, we **chose JSON** for **simplicity.**

Limits :

- API **Call rate limited** for some APIs.
- **API badly structured** resulting in **high number of** API **calls required.**
- Missing Data.

II. LegiScraper : The Data Extraction Pipeline

It started from an observation…

- UK & EU **Parliaments**, as well as many others, **share a lot of things** :
  - **Member** of **Parliament** : they are elected, often come from a party and can vote.
  - **Voting** sessions : M(E)Ps vote for the adoption of a text.
  - **Documents** : Parliaments produces all sorts of documents, laws and texts on political matters
  - …
  - **A system for accessing the data : an API !**

II. LegiScraper : The Data Extraction Pipeline

… which yielded an idea : a standard scraper for parliaments, LegiScraper !

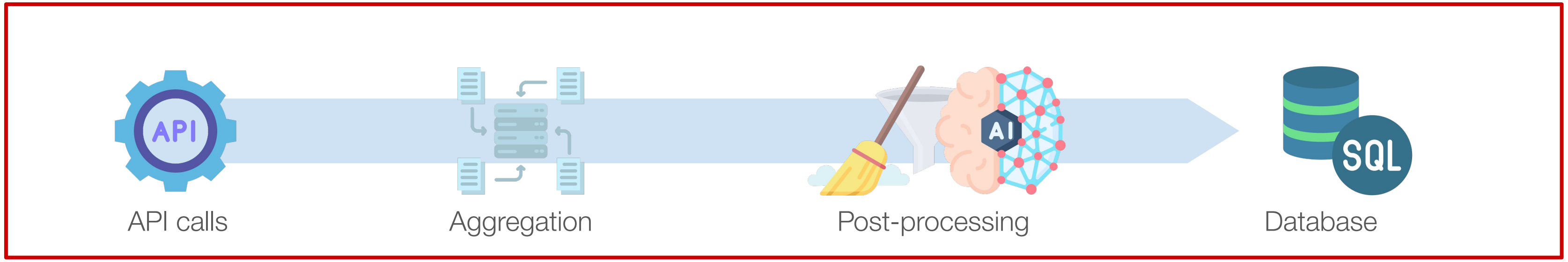
- A **unique scraping object** : the “Scraper” class
    - That automatically configures itself based on configuration files.
    - Manages all APIs uniformly, with standard methods in the package.
    - Adapts to different APIs requests.
- A standard **sub-module framework for data preprocessing**
    - Each Parliament has a sub-module : “eu”, “uk”
    - Each sub-module contains standardized mps.py and votes.py files, with standardized classes.
    - Sub-modules operate with the same config files, Scraper class, and common methods

II. LegiScraper : The Data Extraction Pipeline

While the overall extraction process is orchestrated by a single class :

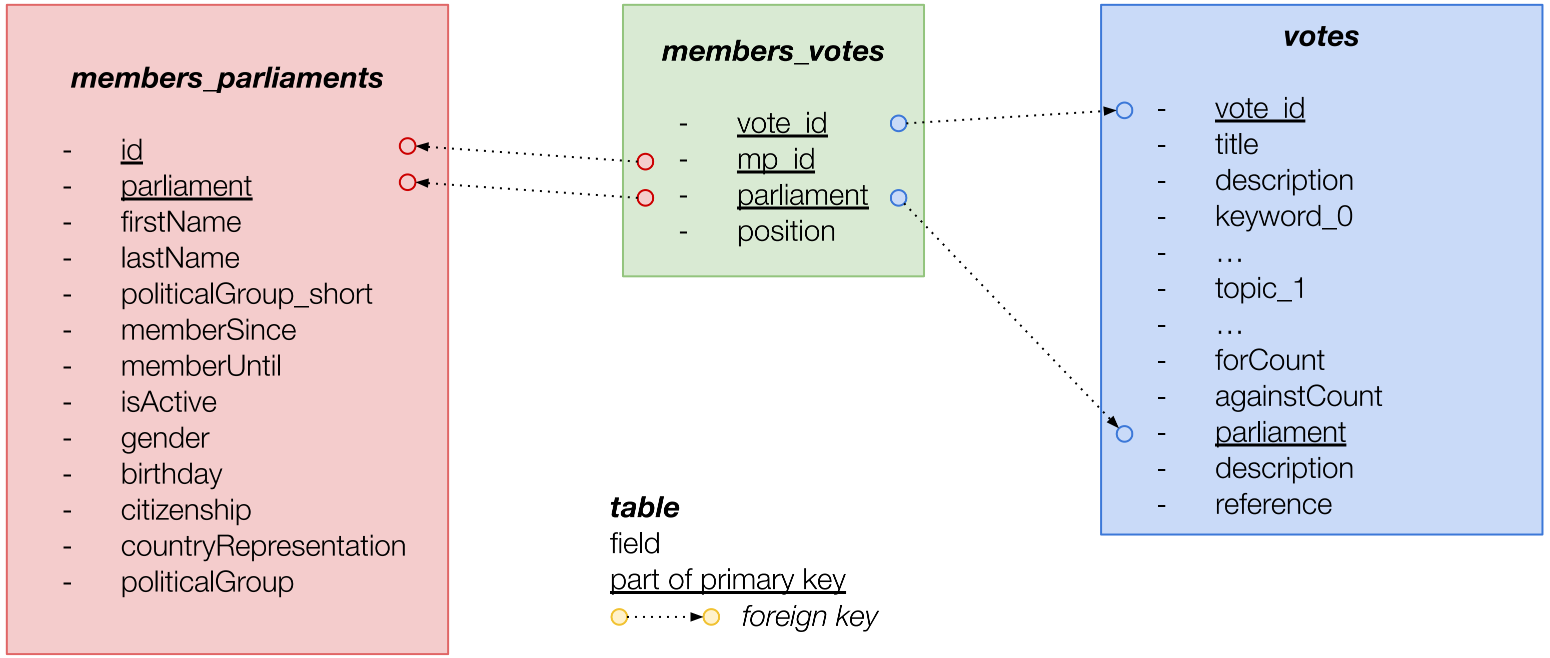
- Database from the database.py file
    - Automatically reads config files to detect available parliaments and data (votes, MPs, …)
    - Dynamically imports the corresponding sub-modules and classes for each parliament
    - Automatically generates the datasets, post-processes and merges them

And all of that in a nice python package ;)



III. Database - Architecture

SQL : the data is **intrinsically relational**



III. Database - Request example

When a vote in the EU parliament was concerning Ukraine, how did each group of french representatives vote ?

members_parliaments

- id
- parliament
- politicalGroup_short
- countryRepresentation

members_votes

- vote_id
- mp_id
- parliament
- position

votes

- vote_id
- keyword_0
- keyword_1
- keyword_2

Group	FOR	AGAINST	NA
PPE	70	0	30
PfE	0	72	28
Renew	98	0	2
S&D	92	2	6
The Left	50	20	30

SQL request return (truncated)

III. Database - Missing or incoherent data

Parliament	Id	last name	PG short	Active (T/F)	member until	birthday	total
EU	0	0	0	718 (100%)	718 (100%)	0	718
UK	0	0	0	927 (38%)	650 (26%)	2454	2458

Extract of members_parliaments table null values quality check

Quality criteria	Description	Qualitative criteria	Explanation/Solution
Missing values	Missing MEPs who voted in this term	10/725	Collect non active MEPs
Incoherence	A member who left is still “active”	277/2454	Error from original database, manual correction or get in touch with UK parliament

IV. Challenges & Limits

APIs :

- Badly configured ⇒ forced to do many API calls
- Limited call rate ⇒ very often got “Access Denied”

Data :

- Missing data from some parliaments (data on MPs in UK)
- Heterogeneous Structure between parliaments ⇒ heterogeneous transformations

Processing :

- Keyword Extraction & Topic Classification computationally intensive ⇒ optimization of the process with batching, parallelization