Data acquisition extraction and storage project

Mariem Aalabou, Mohamed Amine Belkasmi , Mohamed Ali Srir

13th December 2024

Aim of our project

The aim of our project is to create a comprehensive dataset of government members and parliamentarians, and their interactions over time

Scraping of ministers

We developed a scraper using scrapy that goes through the Moroccan governments over time, stopping at the 2011's Benkirane government because it corresponds to a major change in the Moroccan constitution.
Thus, the Wikipedia pages scraped are :

- Gouvernement Akhannouch II (2024-present)
- Gouvernement Akhannouch I (2021-2024)
- Gouvernement El Otmani II (2019-2021)
- Gouvernement El Otmani I (2017-2019)
- Gouvernement Benkiran II (2013-2017)
- Gouvernement Benkiran I (2012-2013)

Example:

Let's check the scraping of a government's wikipedia link:
Gouvernement Akhannouch I

Scraping of the Moroccan Parliament website

We developed a generic scraper to extract detailed information about legislative activities, parliamentary sessions, members of Parliament (MPs) and committees from the Moroccan Parliament website.

Legislative Data (Laws)

Legislative data includes:

- Bills (Projets de loi), Legislative Proposals and Passed Bills.
- Votes and amendments over time.
- Temporal metadata to track changes (number of law readings, and involved committees and MPs).

Parliament Sessions" Questions Data

- The question details, including who asked it and who it was directed to.
- The date and time when the question was asked.
- The status of the question—whether it has been answered or remains pending.

Members of Parliaments (MPs) Data

- Name
- Party
- Commission

Parliamentarian cards

Scraping Process

The scraping process involved the following steps:

- Using `selenium` and `undetected-chromedriver` for handling dynamic content.
- Randomized user-agents and browser profiles to avoid Cloudflare bot detection.
- Extracting and structuring data into a clean JSON format.
- Handling errors and exceptions using automated cleanup methods.

Example: Scraping the legislative page: Legislation Page

Main Challenges and Solutions (1)

- ☞ *Challenge:* Dynamic website content requiring interaction with JavaScript.
  - **Pagination:** Data split across pages required automated navigation ("Next" button).
  - **Filters:** Selecting year, topic, or committee dynamically updated results (Filter search).
- ✓ **Solution:** Used `selenium.webdriver` to:
  - Scrap JavaScript-rendered information
  - Handle asynchronous page loading

Why Selenium Over Scrapy?

Selenium is a powerful tool for web scraping, particularly useful for handling dynamic content rendered by JavaScript.

- **WebDriver:** Automates browser actions.
- **Handling JavaScript:** Can interact with elements that require user actions.
- **Choice Over Scrapy:** Selected for its ability to handle dynamic content and complex interactions.

Main Challenges and Solutions (2)

- ☞ *Challenge:* Bypassing Cloudflare's bot checks.
- ✓ **Solution:** Used `undetected-chromedriver` to bypass Cloudflare"s bot checks and simulate user interactions through:
  - Randomized browser signatures
  - Dynamic user agent rotation
  - Simulated human browsing patterns

Using ChromeDriver

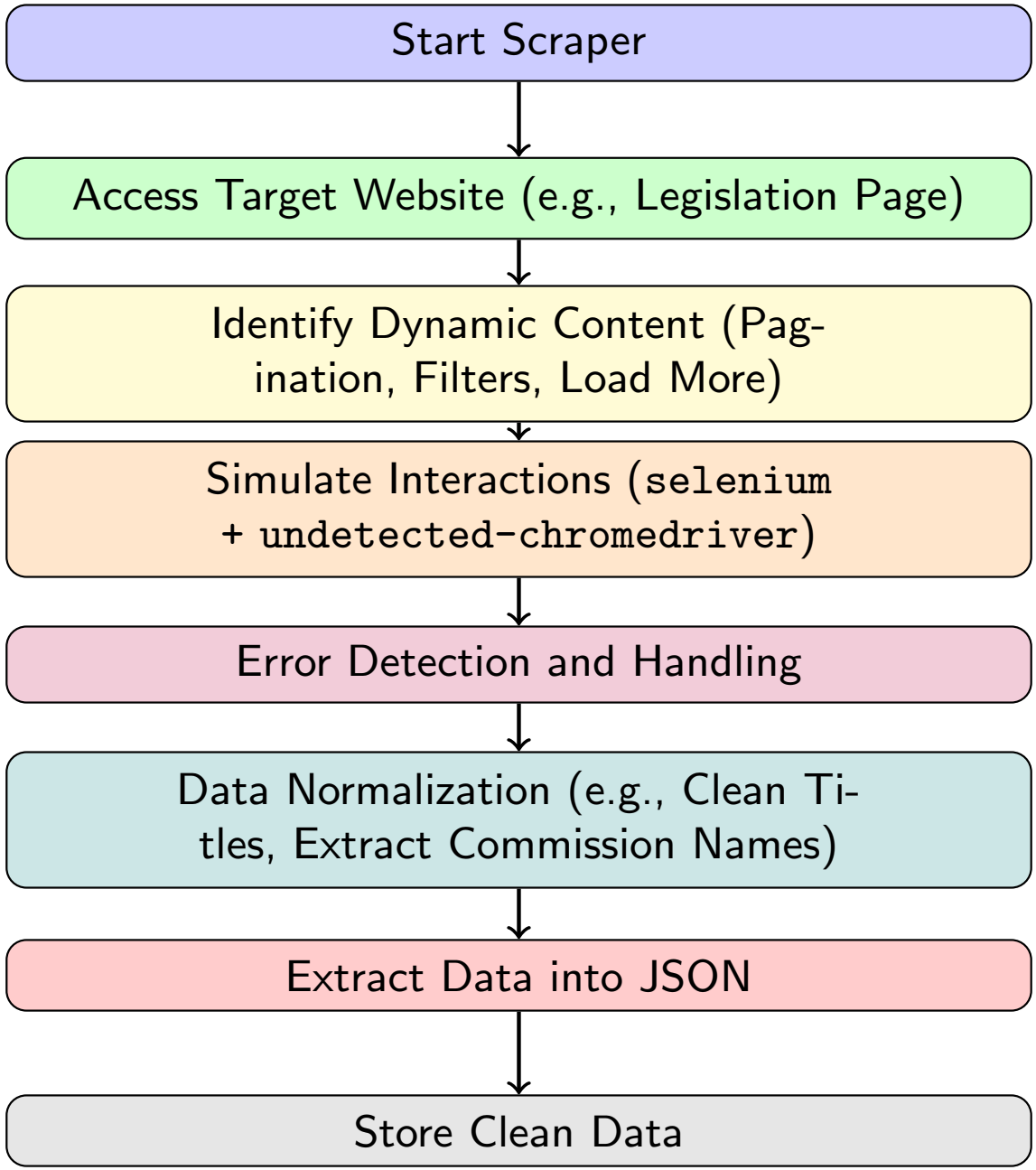
ChromeDriver is a standalone server that implements the WebDriver protocol for Chromium. It is used to control the Chrome browser programmatically.

- **Automation:** Allows automation of browser actions.
- **Handling Dynamic Content:** Essential for interacting with JavaScript-rendered elements.
- **Bypassing Bot Checks:** Used to simulate human browsing patterns.

Main Challenges and Solutions (3)

- ☞ *Challenge:* Switching between French and Arabic pages.
- ✓ **Solution:** Adapted the scraper to handle new structures and encoding differences in the Arabic version.
- ☞ *Challenge:* Incomplete data across pages.
- ✓ **Solution:** Implemented automated checks and cross-referenced with additional sources.

Scraping Process Visualization



Output of Data Scraping

The final output for data includes:

- Over **1197 laws** with metadata.
- Detailed records of **12200 parliamentary questions and session data**.
- Structured information about **1126 MPs**, **9 committees**, and **24 ministries**.

What is Dgraph?

- Dgraph is a distributed, scalable graph database.
- Designed for managing highly connected data efficiently.
- Features include:
  - Graph data model with nodes and edges.
  - Native GraphQL support.
  - ACID transactions for reliability.
  - Built-in full-text search and geospatial queries.
- Open-Source with Python API and a simple query language (GraphQL)

Why Use Dgraph in Data Science?

- Ideal for analyzing complex relationships, such as:
  - Social networks.
  - Recommendation systems.
  - Knowledge graphs.
- Optimized for graph traversal, reducing computational overhead.
- Scalable for large datasets with distributed architecture.
- Supports real-time querying and analysis.

Our Usecase

- The Moroccan Parliament dataset represents the structure, entities, and activities of parliament members and ministries.
- Using Dgraph, we can effectively model and query relationships between deputies, ministers, laws, questions, commissions, and ministries.
- This presentation showcases the schema and justifies the use of Dgraph for representing this data.

Why Use Dgraph?

- **Graph Representation:**
  - Captures complex relationships (e.g., deputies asking questions to ministers).
- **Scalability:**
  - Supports a growing dataset as parliamentary activities evolve.
- **Efficient Queries:**
  - Retrieve connected data with minimal latency.
- **Flexible Schema:**
  - Easily add new entities or relationships (e.g., bills, votes).
- **Real-Time Analysis:**
  - Enables exploration of live parliamentary activity.

Entity Attributes

Deputy:

- `name: string`
- `party: string`
- `work_at: [uid]` (linked to Ministries or Commissions)
- `ask: [uid]` (linked to Questions)

Question:

- `title: string`
- `to: uid` (target, e.g., Minister or Ministry)
- `created_at: datetime`
- `state: string` (e.g., pending, answered)

Law:

- `title: string`
- `type: string`
- `link: string` (URL for law details)
- `created_at: datetime` (timestamp)
- `developed_by: uid` (linked to Commissions or Ministry)

Commission:

- `name: string`

Ministry:

- `name: string`

Entity Relationships

Deputy Relationships:

- Belongs to a `party`.
- Associated with `commissions` or `ministries` (`work_at`).
- Asks `questions` (`ask` predicate).

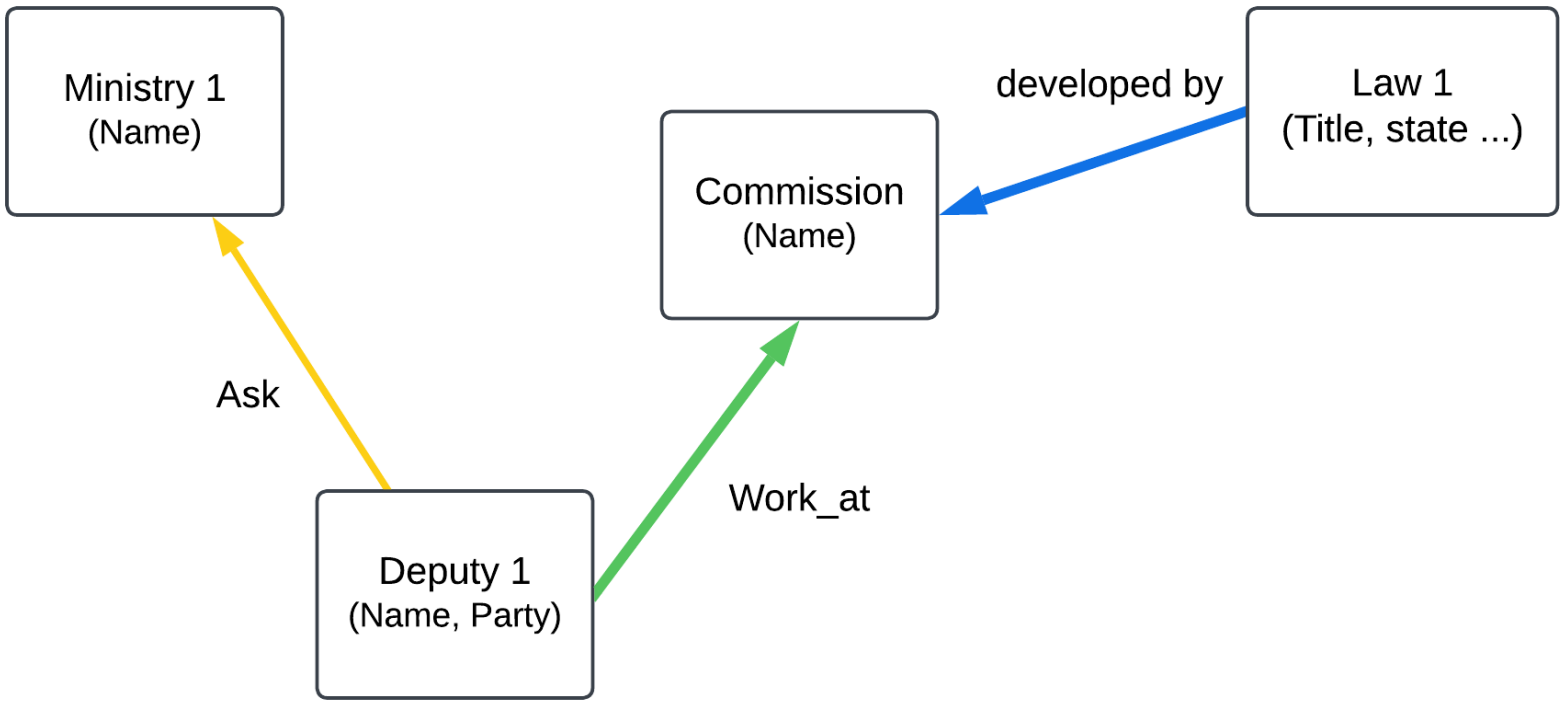
Law Relationships:

- Developed by `commissions`.
- Associated with specific parliamentary activities.

Question Relationships:

- Asked by `deputies`.
- Directed to `ministers` or `ministries`.
- Status tracked via the `state` predicate.

Query Language



Example Query in Dgraph

Retrieve Deputies and Their Commissions for a Specific Term

```
{
  deputy(func: has(name)) @cascade {
    name
    party
    work_at @filter(eq(term, "2016_2021")) {
      commission {
        name
      }
      term
    }
  }
}
```

Query Result (Translated)


```
{
  "data": {
    "deputy": [
      {
        "uid": "0x3b1f",
        "name": "Mustafa Brahimi",
        "party": "Parti de la Justice et du Développement",
        "work_at": [
          {
            "commission": {
              "uid": "0x3af6",
              "name": "Secteurs Sociaux"
            },
            "uid": "0x3dfb",
            "term": "2016_2021"
          }
        ]
      },
      {
        "uid": "0x3b23",
        "name": "Mohamed Barkane",
        "party": "Union Socialiste des Forces Populaires",
        "work_at": [
          {
            "commission": {
              "uid": "0x3af6",
```

Updating Party in Dgraph

Example: Change the Party of Mustafa Brahimi to a New Party
Mutation:

```
{
  "set": [
    {
      "uid": "0x3b1f",
      "party": "New Party Name"
    }
  ]
}
```

Explanation:

- `"uid": "0x3b1f"` specifies the existing node for Mustafa Brahimi.
- Updates the `"party"` attribute to `"New Party Name"`.
- No other attributes or relationships are affected.

Demo Time