Introduction to Python Midterm Report: Measuring the French Government Position on Promoting Electric Vehicles

Xiaoyan HU
x.hu@mpp.hertie-school.org

Abstract

In 2013, Bradley W. Lane and others identified two motivations for why governments seek to promote the electric car: risk management in response to ecological or energy concern versus industrial policy seeking an economic upgrade. In this framework, jurisdictions suite of policies falls in one of three categories: primary risk management, primary industrial policy, or a substantial blend of the two. By reading government documents, Lane concluded that France was an intermediate case with a substantial combination of industrial policy and risk management.

Lane's framework of analysis provided useful guidelines, but it has been seven years and two government changes since their research. It is time to apply the theory on more recent policy texts to update our understanding of the intention of this French government on the French EV industry. Instead of repeating the conventional way of generating policy positions by clicking, reading and downloading all the search result on the open web, this team project wishes to draw inspiration from computer sciences and explore a more efficient way of collecting and classifying text.

At the current stage of the the experiment, we used the Scrapy library to set up a spider. We successfully scraped all the relevant article title and links from the French government website with the search keyword "véhicules électrique". However, this is not the endpoint of our project. As for future work, we will design an additional level of code to fetch the corresponding text content of the respective web links and apply non-deep learning natural language processing methods to generate topics from the scraped text.

1. Proposed Method[1]

Successful completion of this team project consists of two parts. The first part is the definitive collection of text from the open web specified in the resources section. The second part involved topic classification and results representation.

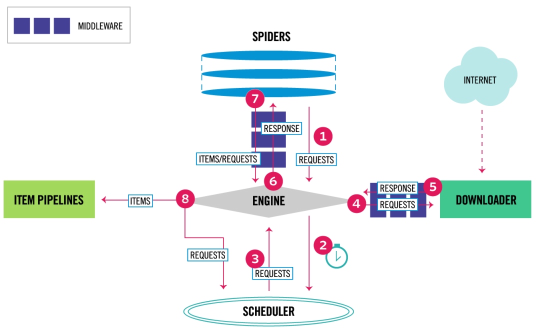


Figure 1. Scrapy Architecture

There are multiple packages available online that allow us to scrape data from the open web such as Scrapy, Selenium, and BeautifulSoup. While the last one is prevalent, this project decided to proceed with the more robust framework, Scrapy. The following diagram shows an overview of the Scrapy architecture.

As we can observe from this photo, Scrapy is a complete framework which is designed around the concept of Item. Thus, we cannot just command our spider isolated from other files that support its operation, such as pipelines and items. The detailed description of the code is saved from this midterm report but will be presented in a separated document on the GitHub repository. Here we will list the intuitive steps of our scraping project:

1. Find the URL that you want to scrape;
2. Inspecting the Page;
3. Find the data you want to extract;
4. Write the code;
5. Run the code and extract the data;
6. Store the data in the required format;

[1]GitHub page link:https://github.com/Xiaoyan-Adele/Political-Motivation-behind-French-EV-Promotion

2. Experiments

2.1. Defining the Target Domains for Web Scraping

This project decided to scrape the French government website (https://www.gouvernement.fr). First, it is representative. This site is the official platform for publishing opinion piece, press release and legislation record of the French prime minister office. Thus it represents the official standing of the French government. Secondly, comparing to the website of the French president, prime minister website yields more content during our exploratory research and it is much more up-to-date as it focuses on the position of the current administration.

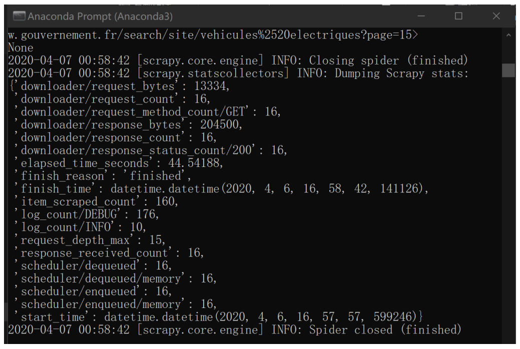
Secondly, it is vital to decide which keyword to use for navigating the search engine of our target websites (https://www.gouvernement.fr/search/). While being able to extract the maximum information possible, this keyword also needs to remain neutral so that our search result will not be too biased towards either "industrial policy" or "ecological concerns". During the initial inspection, this project tested several keywords such as "automobile", "PSA Renault", "véhicules électrique" and "batterie (battery)". Each yield a slightly different assortment of results. According to the first result page of each word, "automobile", "PSA Renault" seems to be closer with industry probably due to the contextual usage of those words. Between "batterie" and "véhicules électrique", although both seem to be somewhat unbiased, we decided to proceed with "véhicules électrique" since it is closer to the core content that we want to examine in our project.

2.2. Experimental Setup: Constructing the Scrapy Architecture

We rely on two major sources to acquire a hands-on understanding of the coding mechanism from scratch: DataCamp course *Web Scraping in Python* and an article series on Medium *A Minimalist End-to-End Scrapy Tutorial*. Despite the fact that both were comprehensive tutorials, this project still encountered much-expected bugs from missing spiders to empty output. With the help of friends and classmates and countless debugging attempts, we realized the importance to create an empty scrapy project with its own command code other than mimic manually the structures described in the tutorials. Using scrapy's own code prevent negligence of specific settings, especially for the file settings.py.

2.3. Initial Results Analysis

Our spider managed to scrape three types of information (title, link to the article and publishing date) from eleven webpages. It generated a csv file of 160 scraped items in less than one-minute time. With this result, we have a list of all the articles in the French government website with the keyword "véhicules électrique". It indicates a promising alternative to systematically collect data comparing to the manual copy-paste methods that we used to adopt.



```
w.gouvernement.fr/search/site/vehicules%2520electriques?page=15>
None
2020-04-07 00:58:42 [scrapy.core.engine] INFO: Closing spider (finished)
2020-04-07 00:58:42 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/request_bytes': 13334,
 'downloader/request_count': 16,
 'downloader/request_method_count/GET': 16,
 'downloader/response_bytes': 204500,
 'downloader/response_count': 16,
 'downloader/response_status_count/200': 16,
 'elapsed_time_seconds': 44.54188,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2020, 4, 6, 16, 58, 42, 141126),
 'item_scraped_count': 160,
 'log_count/DEBUG': 176,
 'log_count/INFO': 10,
 'request_depth_max': 15,
 'response_received_count': 16,
 'scheduler/dequeued': 16,
 'scheduler/dequeued/memory': 16,
 'scheduler/enqueued': 16,
 'scheduler/enqueued/memory': 16,
 'start_time': datetime.datetime(2020, 4, 6, 16, 57, 57, 599246)}
2020-04-07 00:58:42 [scrapy.core.engine] INFO: Spider closed (finished)
```

Figure 2. Spider Crawl Demostration Screenshot

3. Future work

In this midterm report, we demonstrate a successful set up of web scraping spider with the Scrapy library in Python. From a coding exercise perspective, this project already completed its goal. However, judging from its practical motivation, the current achievement is insufficient. Ultimately, this scraping algorithm is deployed to gather text for future analysis. Therefore, title and corresponding links cannot perform this task. We need to complete the spider with additional code to auto click into each link and scrape the text content.

Moreover, since we want to explore the potential replacement of human subjective reading of text and model sensible topics with the algorithm, the future work of this project will also focus on non-deep learning natural language processing.

References

[1] Lane, B. W.; Messer-Betts, N.; Hartmann, D.; Carley, S. (2013). *Government Promotion of the Electric Car: Risk Management or Industrial Policy?*. European Journal of Risk Regulation, 4(2), 227-246.

[2] Fabrizio Gilardi and Bruno Wüest (2018). *DataCamp course Web Scraping in Python*. https://learn.datacamp.com/courses/web-scraping-with-python

[3] Harry Wang.(2019). *A Minimalist End-to-End Scrapy Tutorial*. https://towardsdatascience.com/a-minimalist-end-to-end-scrapy-tutorial-part-i-11e350bcdec0