Natural Language Processing

•••

Poetry

Romain Jochum / Tom Lerat /
Yanis Barbieux / Ugo Peduzzi

Summary :

Problem presentation

Data Collection

Cleaning the data

Fine tuning

Discussion / Conclusion

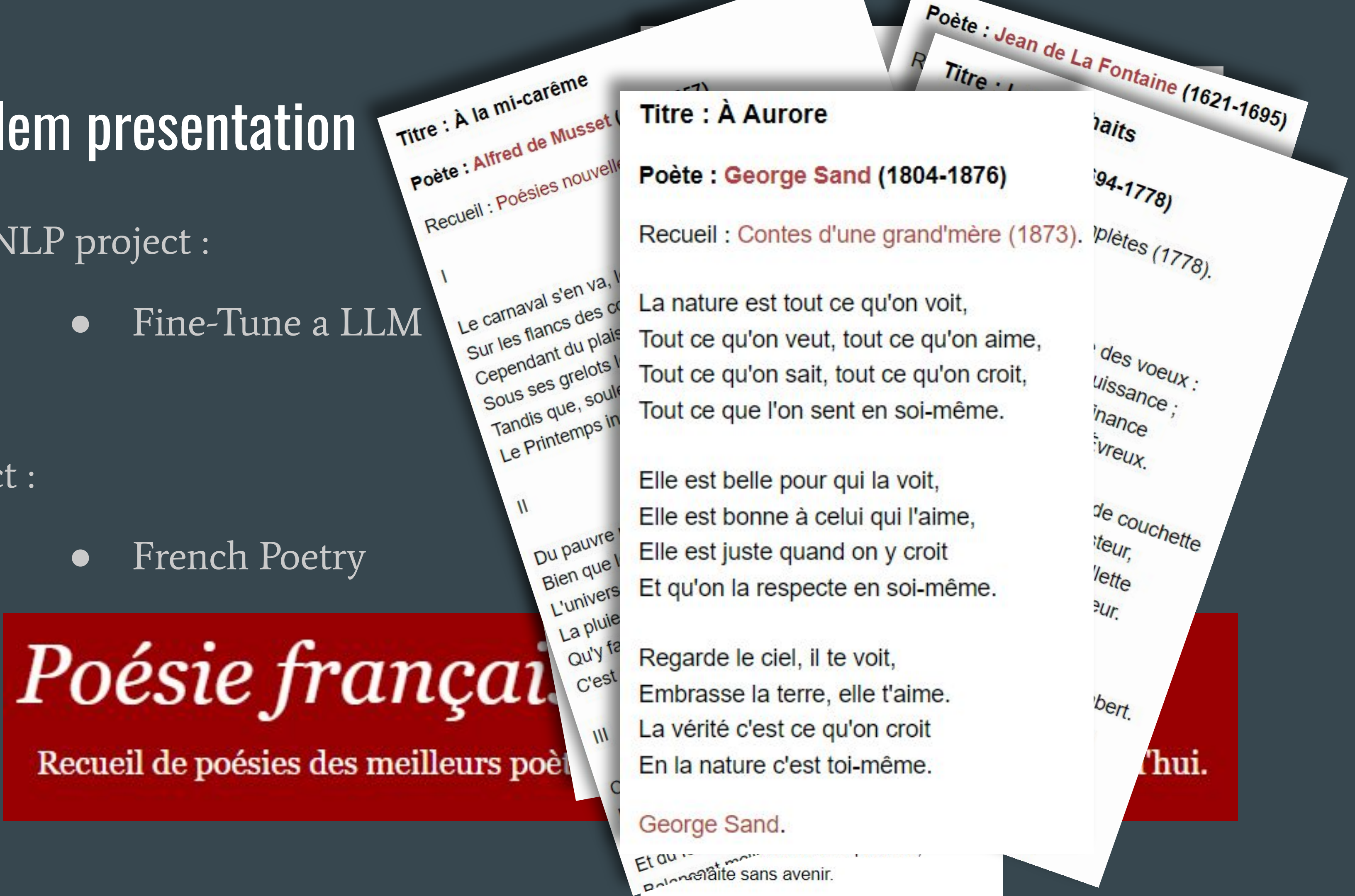
Problem presentation

Final NLP project :

- Fine-Tune a LLM

Subject :

- French Poetry



Titre : À la mi-carême

Poète : Alfred de Musset (

Recueil : Poésies nouvelle

I

Le carnaval s'en va, l
Sur les flancs des co
Cependant du plais
Sous ses grelots l
Tandis que, soule
Le Printemps in

II

Du pauvre
Bien que l
L'univers
La pluie
Qu'y fa
C'est

III

Et du
aite sans avenir.

Titre : À Aurore

Poète : George Sand (1804-1876)

Recueil : Contes d'une grand'mère (1873).

La nature est tout ce qu'on voit,
Tout ce qu'on veut, tout ce qu'on aime,
Tout ce qu'on sait, tout ce qu'on croit,
Tout ce que l'on sent en soi-même.

Elle est belle pour qui la voit,
Elle est bonne à celui qui l'aime,
Elle est juste quand on y croit
Et qu'on la respecte en soi-même.

Regarde le ciel, il te voit,
Embrasse la terre, elle t'aime.
La vérité c'est ce qu'on croit
En la nature c'est toi-même.

George Sand.

Poète : Jean de La Fontaine (1621-1695)

Titre :
haits

94-1778)

plètes (1778).

des voeux :
uissance ;
inance
Evreux.

de couchette
teur,
lette
eur.

bert.

Poésie françai

Recueil de poésies des meilleurs poèt
'hui.

Data Collection

There was no appropriate dataset for our project so we decided to create our own by scraping poems through a popular poetry website :

The data is scraped from “poesie-francaise.fr” on the 21 of April 2024

We scrapped around 1600 poems

Strictly from dead poets

From Middle Age - Now on

Initial scraping from copy-paste html code

We add another scraping to include the theme of each poems

Data Scraping :

In summary :

Why use poesie-francaise.fr ?

- Simple navigation
- No anti-scraping algorithms implemented
- No javascript loaded (no use of selenium)
- Great bank of classic french poetry

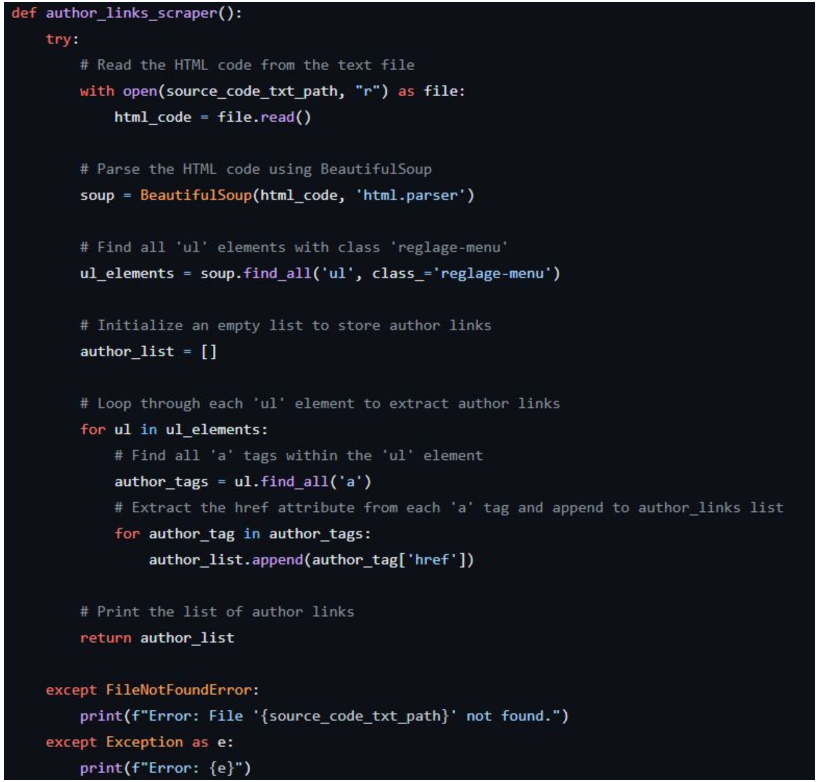
Library used :

- Requests in order (To retrieve the html code)
- BeautifulSoup (The html parser)

Data Scraping in details :

Code to scrap the authors :

We scrape the author first to ensure to only take dead poets

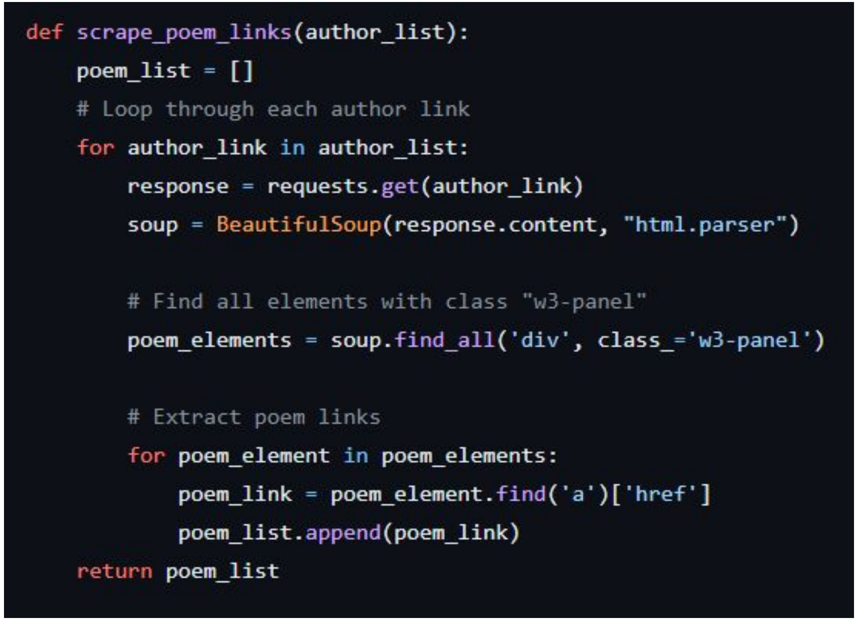


```python
def author_links_scraper():
    try:
        # Read the HTML code from the text file
        with open(source_code_txt_path, "r") as file:
            html_code = file.read()

        # Parse the HTML code using BeautifulSoup
        soup = BeautifulSoup(html_code, 'html.parser')

        # Find all 'ul' elements with class 'reglage-menu'
        ul_elements = soup.find_all('ul', class_='reglage-menu')

        # Initialize an empty list to store author links
        author_list = []

        # Loop through each 'ul' element to extract author links
        for ul in ul_elements:
            # Find all 'a' tags within the 'ul' element
            author_tags = ul.find_all('a')
            # Extract the href attribute from each 'a' tag and append to author_links list
            for author_tag in author_tags:
                author_list.append(author_tag['href'])

        # Print the list of author links
        return author_list

    except FileNotFoundError:
        print(f"Error: File '{source_code_txt_path}' not found.")
    except Exception as e:
        print(f"Error: {e}")
```

Data Scraping in details :

Code to scrap the poems :

Once on the author page, we scrape avery poems submitted

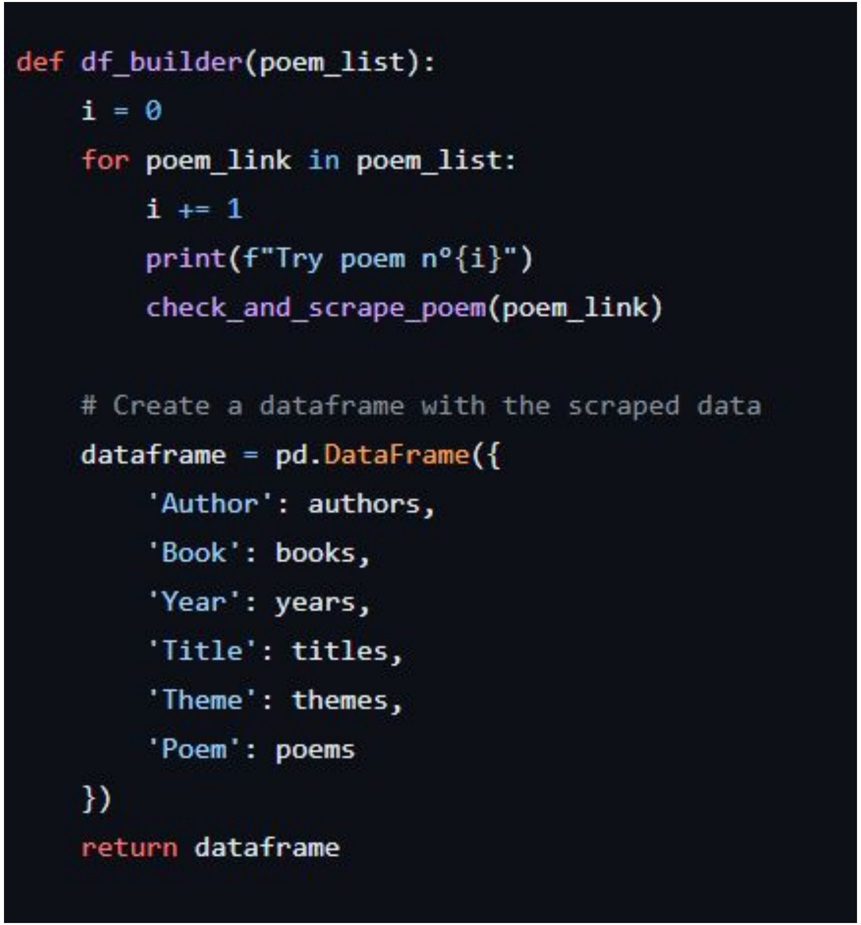


```python
def scrape_poem_links(author_list):
    poem_list = []
    # Loop through each author link
    for author_link in author_list:
        response = requests.get(author_link)
        soup = BeautifulSoup(response.content, "html.parser")

        # Find all elements with class "w3-panel"
        poem_elements = soup.find_all('div', class_='w3-panel')

        # Extract poem links
        for poem_element in poem_elements:
            poem_link = poem_element.find('a')['href']
            poem_list.append(poem_link)
    return poem_list
```

Data Scraping in details :

Builder to create the pandas dataframe :

The builder returns the author, book, year, poem itself and title
A bit tweaker for the themes, the first script without theme saved around 4200 poems while the one with themes saves around 1600 different poemes.



```
def df_builder(poem_list):
    i = 0
    for poem_link in poem_list:
        i += 1
        print(f"Try poem n°{i}")
        check_and_scrape_poem(poem_link)

    # Create a dataframe with the scraped data
    dataframe = pd.DataFrame({
        'Author': authors,
        'Book': books,
        'Year': years,
        'Title': titles,
        'Theme': themes,
        'Poem': poems
    })
    return dataframe
```

Data Cleaning :
What needed to be clean ?

“1830-1875 post-hum”

↓

Integer

“À Alex de Bertha.\n\n\nL'absent qu'on ...”

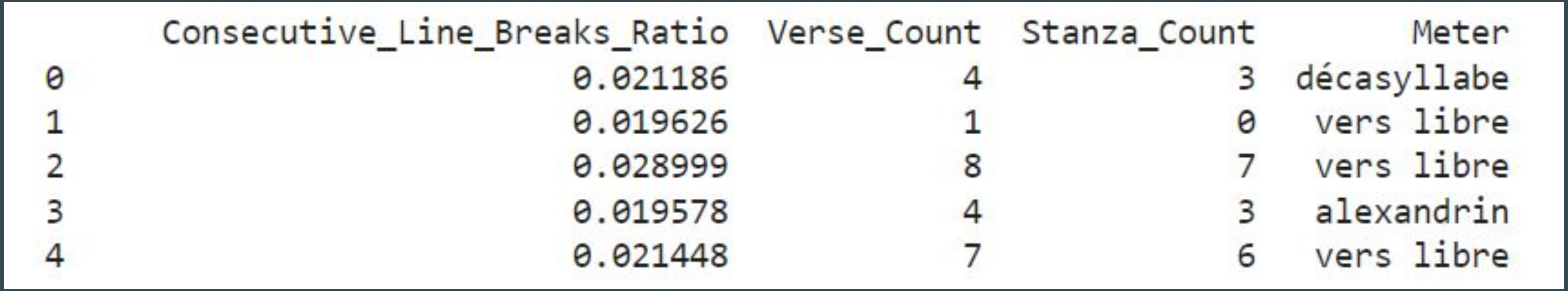
↓

“À Alex de Bertha.

L'absent qu'on ...”

Data Cleaning :

Metering of the poems



```
   Consecutive_Line_Breaks_Ratio  Verse_Count  Stanza_Count        Meter
0                       0.021186            4             3  décasyllabe
1                       0.019626            1             0   vers libre
2                       0.028999            8             7   vers libre
3                       0.019578            4             3   alexandrin
4                       0.021448            7             6   vers libre
```

Fine tuning :
Using Mistral-7B

We fine tuned using Mistral-7B

Helped by a project from brev.dev

Went for a paid version :

- Easier to use
- More documentation
- Only 4.75$
- Big save on time

Could have use the Centrale Supercomputer but the documentation was limited.

We run roughly 225 epochs.

Mistral-7B :

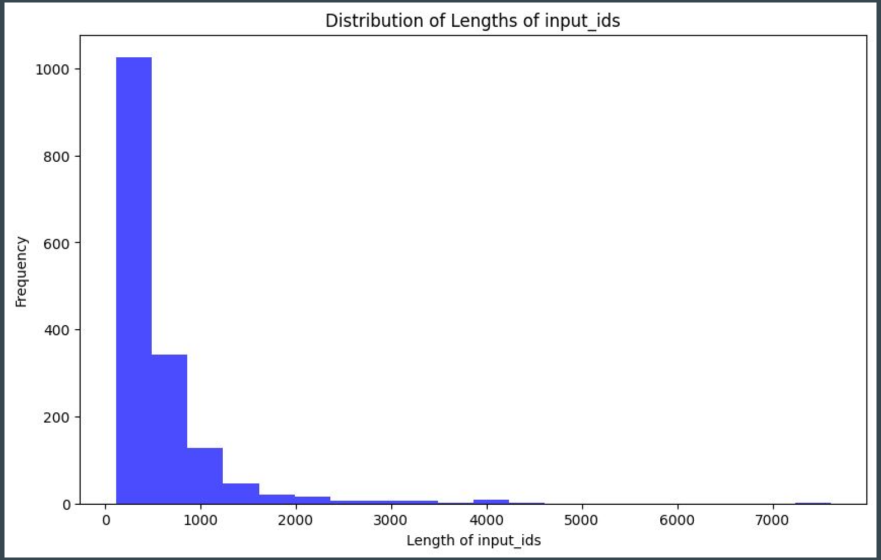
Large Language Model (LLM) is a pre trained generative text model with 7 billion parameters.

Why use it ?

- Lightness of the model
- Small computing power required
- Openness
- Huge documentation
- Only 4,95$

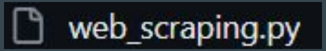
Fine tuning : the results

The fine tuning took around 30’ but due to VRam limitation we were required to regularize the size to 516 tokens



Conclusion

What could be done ?

- Combine web_scraping_with_theme.py and  web_scraping.py in a single script.
- Improve the Performance of the metering function for data cleaning as it is currently not performing flawlessly.
- Finetune the metadata of the model and the model itself using longer tokens
- Update to Mistral 7B v0.2