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

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 added another scraping to include the theme of each poems

Data Scraping : What is it ?

The process of extracting information from websites . We can decompose it in 4 steps :

- Identify the target website
- Send a request
- Parse the HTML
- Extract Data / Store Data

Here a simple example in python :

```python
import requests
from bs4 import BeautifulSoup

# Send a request to the website
url = 'http://example.com'
response = requests.get(url)

# Parse the HTML content
soup = BeautifulSoup(response.text, 'html.parser')

# Extract data (for example, all the titles on the page)
titles = soup.find_all('h1')
for title in titles:
    print(title.get_text())
```

Data Scraping :

In our case :

Library used :

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

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

Data Scraping in details :

Code to scrap the authors :

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

```
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 every 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 :

A bit tweaker for the themes, 4200 poems without, 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

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 :

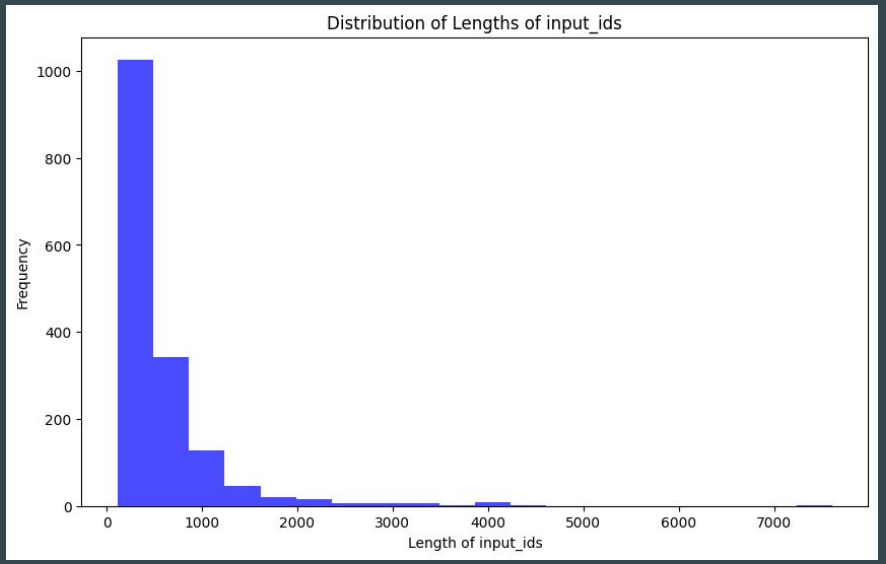
Large Language Model (LLM) is a pre trained generative text model with 7 billion parameters.

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

Project :

Poetry generation via Fine tuning a LLM

Accomplishments :

Data scraping poesie-francaise.fr

Cleaning the Data obtained from the scraping

Fine tuning Mistral 7B using the the clean Data

Conclusion

What should be done ?

- Merge both Data scraping scripts
- Improve the Metering function
- Finetune the metadata of the model
- Adjust the model to use longer tokens
- Update to Mistral 7B v0.2

Conclusion

Thank you for listening