AI VIET NAM – AI COURSE 2023

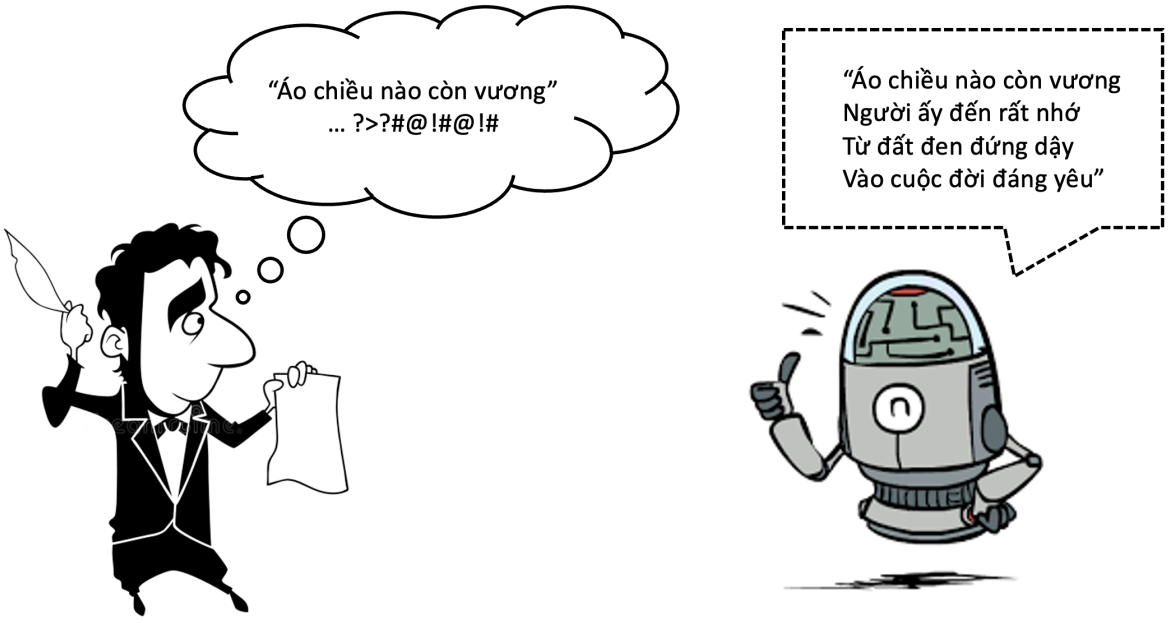
Xây dựng mô hình AI tạo sinh thơ tiếng Việt

Dinh-Thang Duong và Quang-Vinh Dinh

Ngày 25 tháng 2 năm 2024

Phần I. Giới thiệu

Text Generation là một trong các dạng bài toán thuộc lĩnh vực xử lý ngôn ngữ tự nhiên hiện đang rất được cộng đồng quan tâm, với khả năng tạo ra các câu từ mới dựa trên dữ liệu đầu vào. Chính bởi các kết quả trả về mang tính "sáng tạo", đáp ứng được các nhu cầu bài toán khác nhau từ mô hình này mà nó đã và đang được ứng dụng vào các sản phẩm nổi tiếng hiện nay, điển hình nhất chính là ChatGPT, một ứng dụng Chatbot đã tạo nên cơn sốt trên toàn thế giới thời gian qua.



Hình 1: Bài toán sinh thơ tiếng Việt (thơ trong hình được tạo từ mô hình sinh thơ)

Trong project này, chúng ta sẽ cùng triển khai một chương trình sử dụng mô hình Text Generation với chủ đề sinh thơ Tiếng Việt dựa vào một từ tiếng Việt đầu vào từ người dùng. **Như vậy, Input/Output của chương trình là:**

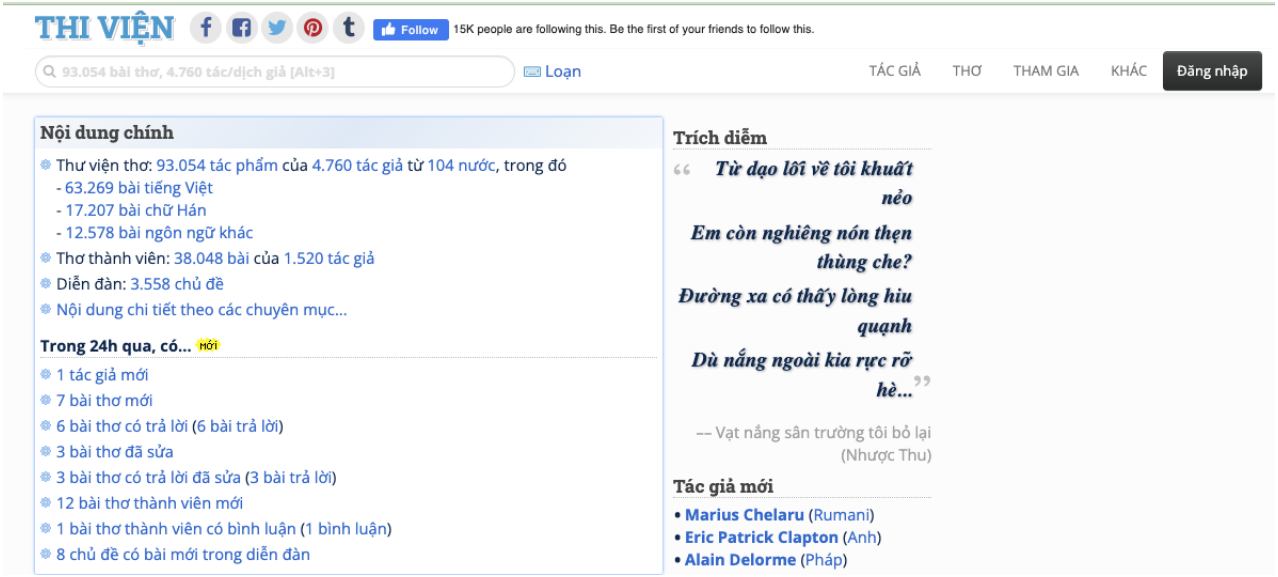
- **Input:** Một chuỗi gồm các kí tự mở đầu cho bài thơ.
- **Output:** Bài thơ hoàn chỉnh.

Phần II. Cài đặt chương trình

Trong phần này, chúng ta sẽ thực hiện hai giai đoạn chính của project để hoàn thiện được mô hình yêu cầu, bao gồm: Thu thập dữ liệu và Xây dựng mô hình. **Nội dung cụ thể như sau:**

1. **Thu thập dữ liệu:** Để huấn luyện được mô hình với Input/Output theo đúng như yêu cầu đã đề ra ở phần trước, chúng ta cần thu thập vào xây dựng một bộ dữ liệu theo đúng mô tả. Đối với dữ liệu thơ, có rất nhiều trang web tổng hợp các văn thơ của Việt Nam cũng như thế giới. Tuy nhiên ở project này, ta sẽ thu thập các văn thơ ngũ ngôn trên trang web thivien.net, một trang web lớn chuyên tổng hợp các văn thơ gồm đủ các thể loại của Việt Nam.

   ***Note:** Các bạn có thể tải bộ dữ liệu đã được thu thập sẵn tại đây và bỏ qua phần này.*



Hình 2: Trang chủ thivien.net

Có rất nhiều thư viện Python giúp ta có thể tương tác và trích xuất thông tin từ trang web một cách dễ dàng. Song ở project này, ta sẽ dùng thư viện Selenium để thực hiện việc thu thập dữ liệu trên Google Colab. **Các bước thực hiện như sau:**

(a) **Tải thư viện Selenium:** Với môi trường máy tính cá nhân, ta đơn giản cài đặt bằng dòng lệnh **pip install selenium webdriver_manager**. Tuy nhiên với môi trường Google Colab, ta sẽ có cách cài đặt phức tạp hơn (chi tiết các bạn coi tại đây), các bạn hãy copy và chạy đoạn code bên dưới trong Colab:

```
%%shell
# Ubuntu no longer distributes chromium-browser outside of snap
#
# Proposed solution: https://askubuntu.com/questions/1204571/how-to-install-chromium-without-snap

# Add debian buster
cat > /etc/apt/sources.list.d/debian.list <<'EOF'
deb [arch=amd64 signed-by=/usr/share/keyrings/debian-buster.gpg] http://deb.debian.org/debian buster main
```

```
deb [arch=amd64 signed-by=/usr/share/keyrings/debian-buster-updates.gpg] http://deb.debian.org/debian buster-updates main
deb [arch=amd64 signed-by=/usr/share/keyrings/debian-security-buster.gpg] http://deb.debian.org/debian-security buster/updates main
EOF

# Add keys
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys DCC9EFBF77E11517
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 648ACFD622F3D138
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 112695A0E562B32A

apt-key export 77E11517 | gpg --dearmour -o /usr/share/keyrings/debian-buster.gpg
apt-key export 22F3D138 | gpg --dearmour -o /usr/share/keyrings/debian-buster-updates.gpg
apt-key export E562B32A | gpg --dearmour -o /usr/share/keyrings/debian-security-buster.gpg

# Prefer debian repo for chromium* packages only
# Note the double-blank lines between entries
cat > /etc/apt/preferences.d/chromium.pref << 'EOF'
Package: *
Pin: release a=eoan
Pin-Priority: 500


Package: *
Pin: origin "deb.debian.org"
Pin-Priority: 300


Package: chromium*
Pin: origin "deb.debian.org"
Pin-Priority: 700
EOF

# Install chromium and chromium-driver
apt-get update
apt-get install chromium chromium-driver

# Install selenium
pip install selenium
```

(b) **Import các thư viện cần thiết:**

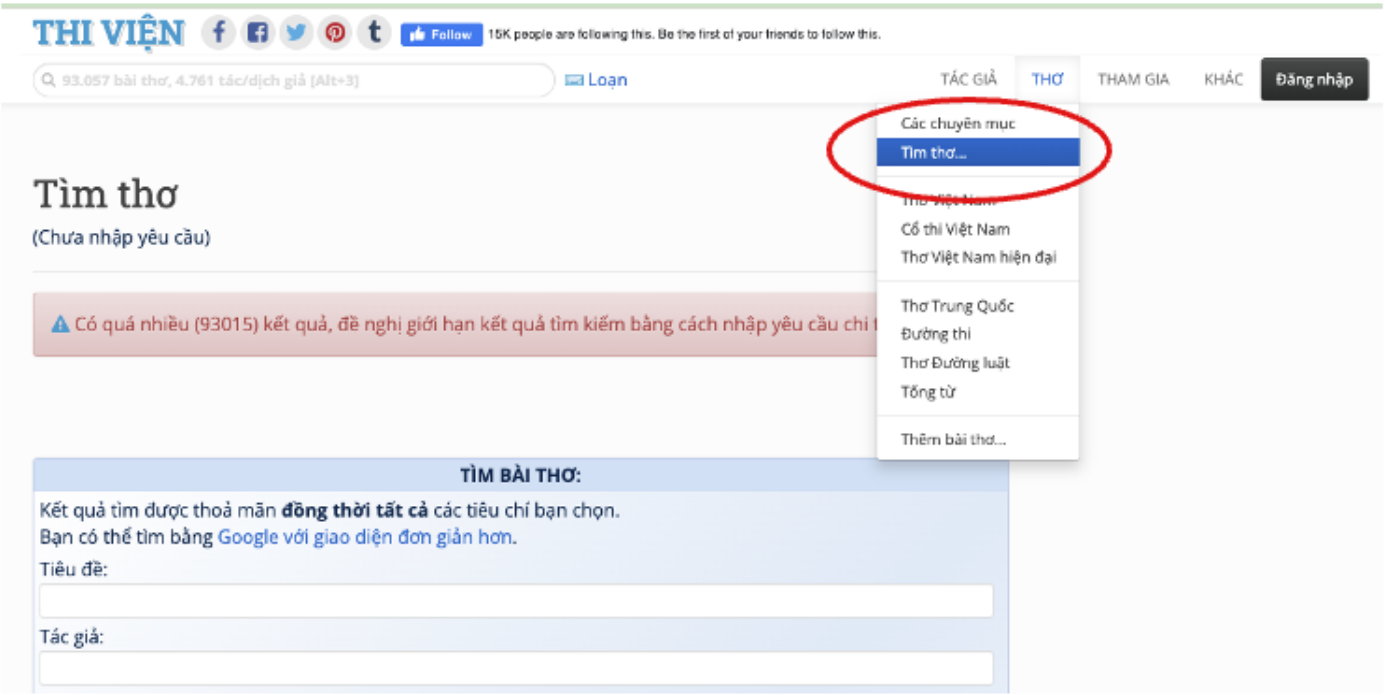
```
import pandas as pd
import os
import requests
import time
import random

from tqdm import tqdm
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

(c) **Khởi tạo Selenium driver:** Với selenium, ta có thể hiểu đơn giản rằng việc truy cập vào một trang web sẽ được thực hiện như chính chúng ta sử dụng trình duyệt web hằng ngày. Đầu tiên, ta khởi tạo một driver sử dụng đoạn code sau:

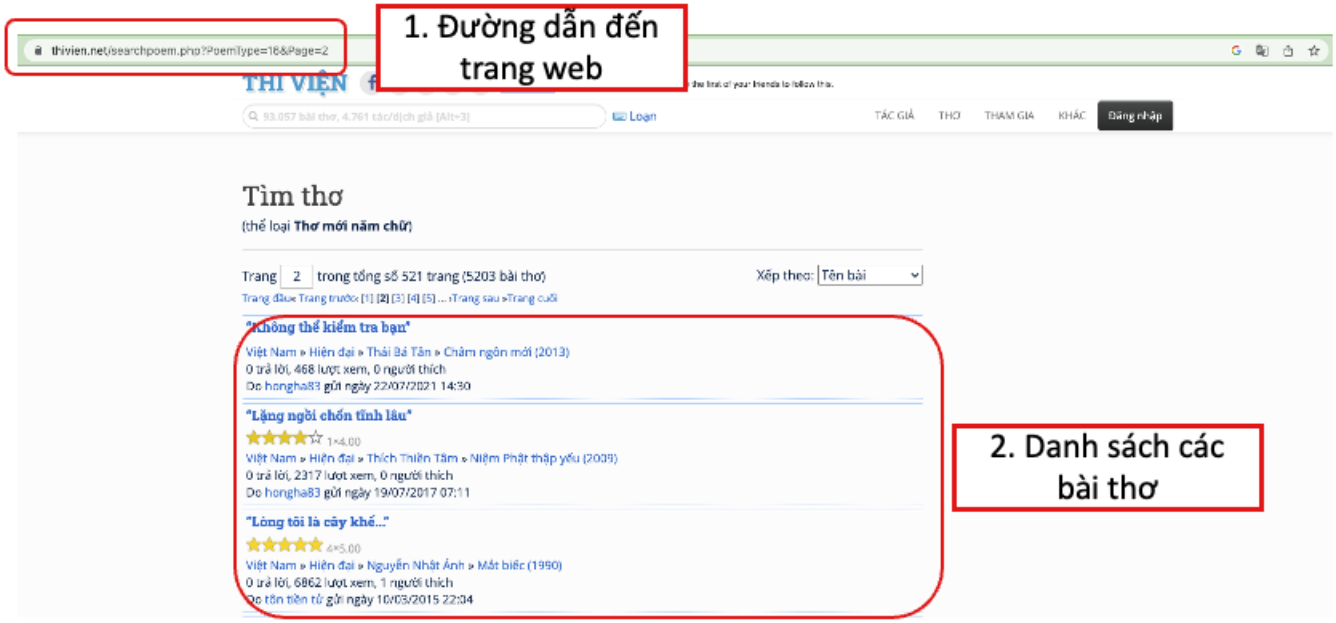
```
WEBDRIVER_DELAY_TIME_INT = 10

service = Service(executable_path=r'/usr/bin/chromedriver')
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--no-sandbox')
chrome_options.headless = True
driver = webdriver.Chrome(service=service, options=chrome_options)
driver.implicitly_wait(5)
wait = WebDriverWait(driver, WEBDRIVER_DELAY_TIME_INT)
```

Driver trong Selenium đóng vai trò như trình duyệt web, giúp ta thực hiện các thao tác như truy cập vào trang web dựa vào đường dẫn, thao tác chuyển trang...

(d) **Phân tích nội dung cần trích xuất:** Để việc triển khai code được thuận lợi, ta cần xác định rõ kiến trúc file html của trang web cũng như các thành phần, nội dung mà ta mong muốn trích xuất. Một cách tìm nhanh chóng nhất đó là ta nên tìm đến trang tìm kiếm, từ đó sử dụng selenium để duyệt qua toàn bộ các bài viết được liệt kê trong trang tìm kiếm đó. Trong thivien.net, ta chọn mục **Tìm thơ...** để đến trang này.



Sau đó, các bạn hãy điền một số thông tin trong bảng **TÌM BÀI THƠ:**, ở đây ta chỉ cần quan tâm đến trường thông tin **Thể thơ:** được chọn vào **"Thơ mới năm chữ"**. Sau khi bấm tìm kiếm, một trang web với các bài thơ với thể thơ ngũ ngôn xuất hiện.

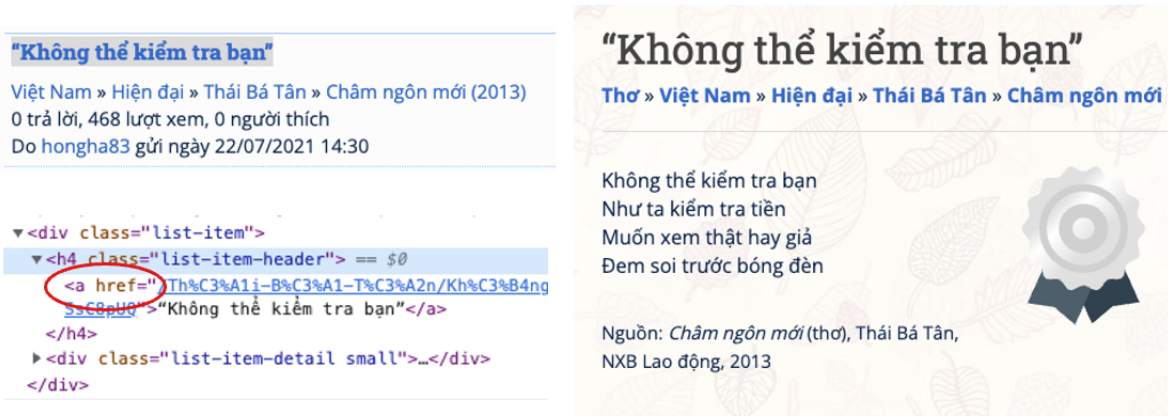


Tại đây, ta đã có được thông tin đường dẫn của trang web (mainpage_url). Chúng ta sẽ sử dụng driver đã định nghĩa ở trên để truy cập vào trang này bằng lệnh **driver.get()**:

```
datasets = []
deletion_script = 'arguments[0].parentNode.removeChild(arguments[0]);'
for page_idx in tqdm(range(1, 11)):
    main_url = f'https://www.thivien.net/searchpoem.php?PoemType=16&ViewType=1&Country=2&Age[]=3&Page={page_idx}'
    driver.get(main_url)
```

Nhận thấy đường dẫn trang web có chứa các trường thông tin để ta có thể di chuyển qua lại tại các trang tiếp theo của bảng tìm kiếm (**'Page=2'** tức đang ở trang 2 của bảng). Vì vậy ta có thể tận dụng điều này để tạo một vòng lặp lặp qua từng trang một cách tự động (với đoạn code trên ta sẽ duyệt từ trang thứ 1 đến trang thứ 10).

Ta tiếp tục phân tích các thành phần ta cần trích xuất đối với một trang thơ thông qua việc đọc cấu trúc html của trang web (cấu trúc html của trang web có thể được tìm thấy thông qua tính năng Inspect trên trình duyệt). Nhận thấy, thông tin duy nhất ta quan tâm đó chính là nội dung bài thơ. Song, để tiện cho các tác vụ sau này nếu có, ta sẽ trích xuất thêm các thông tin khác của bài thơ như Tựa đề, Nguồn...

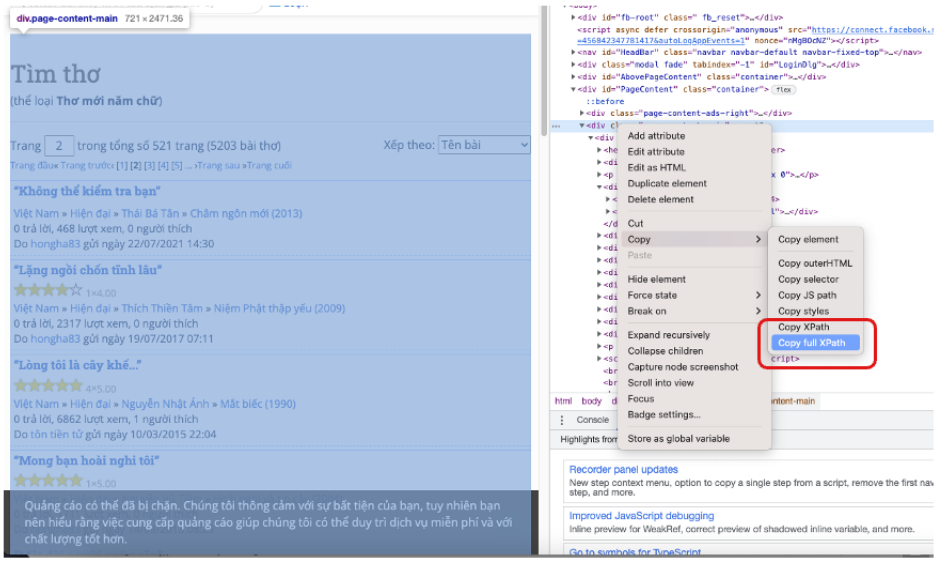


Hình 3: Ví dụ về một bài thơ được liệt kê trong bảng tìm kiếm và thẻ html của nó

Như vậy, mỗi bài thơ được liệt kê tại trang tìm kiếm là một thẻ , bên trong có chứa đường dẫn đến trang chứa bài thơ chính khi ta click chuột vào. Từ đây, ta dễ dàng truy cập vào mỗi bài thơ bằng cách đọc đường dẫn của thuộc tính 'href' chứa tại thẻ <a> (như hình). Để thực hiện được điều này, đầu tiên ta cần đọc được thẻ html chứa bảng dữ liệu, thực hiện như sau:

```
content_tags_xpath = '//*[@class="page-content container"]/div[2]/div/div[@class="list-item"]'
content_tags = driver.find_elements(By.XPATH, content_tags_xpath)
```

Trong Selenium, có nhiều cách để xác định và đọc thẻ html từ trang web thông qua hai phương thức **driver.find_element()** (tìm một thẻ khớp) và **driver.find_elements()** (tìm nhiều thẻ khớp) (chi tiết tại đây) song tìm kiếm bằng XPATH là một cách nhanh chóng nhất. Ở đây, ta quan tâm đến các thẻ div chứa thông tin bài thơ nên ta sẽ dùng find_elements() để tìm toàn bộ các thẻ này. Và với danh sách các thẻ của từng bài thơ (**content_tags**), ta đã có thể truy cập vào nội dung chi tiết của từng bài thơ.



Hình 4: Tìm XPATH của một thẻ html trên trình duyệt

(e) **Thực hiện trích xuất nội dung thơ:** Cuối cùng, với từng thẻ html của trang thơ, ta sẽ thực hiện trích xuất các thông tin mà ta đã xác định (gồm Nội dung bài thơ, Tựa đề, Nguồn) của bài thơ tương ứng. Các thông tin này sẽ được lưu thành một dictionary và đẩy vào một list lưu trữ chung (**datasets**):

```
    for idx in range(len(content_tags)):
        content_tag_xpath = f'/html/body/div[4]/div[2]/div/div[{2+idx}]'
        content_title_xpath = f'/html/body/div[4]/div[2]/div/div[{2+idx}]/h4/a'
        content_tag = wait.until(
            EC.presence_of_element_located(
                (By.XPATH, content_tag_xpath)
            )
        )
        poem_title = wait.until(
            EC.presence_of_element_located(
                (By.XPATH, content_title_xpath)
            )
        ).text
        poem_url = wait.until(
            EC.presence_of_element_located(
                (By.XPATH, content_title_xpath)
            )
        ).get_attribute('href')

        try:
            driver.get(poem_url)

            poem_src_xpath = '//div[@class="small"]'
            poem_content_tag = wait.until(
                EC.presence_of_element_located(
                    (By.CLASS_NAME, 'poem-content')
                )
            )

            try:
                poem_content_i_tag = poem_content_tag.find_element(
                    By.TAG_NAME,
                    'i'
                )
                driver.execute_script(deletion_script, poem_content_i_tag)
            except:
                pass

            try:
                poem_content_b_tag = poem_content_tag.find_element(
                    By.TAG_NAME,
                    'b'
                )
                driver.execute_script(deletion_script, poem_content_b_tag)
            except:
                pass

            poem_content = poem_content_tag.text

            try:
                poem_src_tag = wait.until(
                    EC.presence_of_element_located(
                        (By.XPATH, poem_src_xpath)
                    )
```

```
                )
                poem_src = poem_src_tag.text
            except:
                poem_src = ''

            poem_info = {
                'title': poem_title,
                'content': poem_content,
                'source': poem_src,
                'url': poem_url
            }

            datasets.append(poem_info)

            driver.back()
        except Exception as e:
            print(e)
            print(poem_url)
```

Các kỹ thuật tại bước này đều xoay quanh việc tìm kiếm bằng XPATH kèm theo một số logic python phát sinh trong quá trình thử nghiệm code, các bạn nên đọc qua từng dòng cũng như kiểm nghiệm lại giá trị các biến dữ liệu để có thể hiệu tường tận các bước trích xuất trên. Thông qua đó, các bạn cũng có thể áp dụng để thu thập dữ liệu từ các trang web khác theo nhu cầu của các bạn (Ví dụ: thu thập các bài thơ lục bát tại thivien.net...)

(f) **Lưu bộ dữ liệu thành file .csv:** Sau khi hoàn tất giai đoạn thu thập trên, ta sẽ có một danh sách các dictionary (record) chứa thông tin của một bài thơ. Lúc này, để thuận tiện trong việc lưu trữ, ta sẽ lưu danh sách này thành một file .csv.

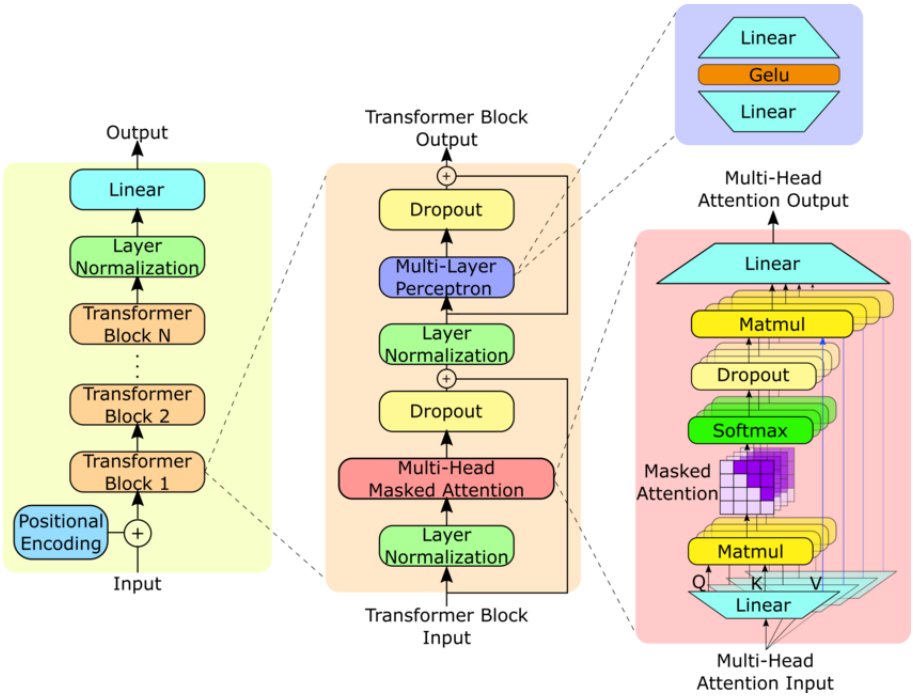
	title	content	source	url
0	“Bạn xấu như chiếc bóng”	Bạn xấu như chiếc bóng\nCứ bám riết theo anh\n...	[Thông tin 1 nguồn tham khảo đã được ẩn]	https://www.thivien.net/Th%C3%A1i-B%C3%A1-T%C3...
1	“Cái làm ta hạnh phúc”	Cái làm ta hạnh phúc\nThực ra cũng chẳng nhiều...	[Thông tin 1 nguồn tham khảo đã được ẩn]	https://www.thivien.net/Th%C3%A1i-B%C3%A1-T%C3...
2	“Chiều vừa xốp trên tay”	Chiều vừa xốp trên tay\nChợt nghe thoáng ong b...	[Thông tin 1 nguồn tham khảo đã được ẩn]	https://www.thivien.net/L%C3%A2m-Huy-Nhu%E1%BA...
3	“Chơi thân không có nghĩa”	Chơi thân không có nghĩa\nKhông cãi nhau bao g...	[Thông tin 1 nguồn tham khảo đã được ẩn]	https://www.thivien.net/Th%C3%A1i-B%C3%A1-T%C3...
4	“Có thể buồn chút ít”	Có thể buồn chút ít\nMột mình, không người yêu...	[Thông tin 1 nguồn tham khảo đã được ẩn]	https://www.thivien.net/Th%C3%A1i-B%C3%A1-T%C3...
...	...	...	...	...

Hình 5: Dữ liệu thu thập được trong bảng dữ liệu

```
df = pd.DataFrame(datasets)
df.to_csv('poem_dataset.csv', index=True)
```

Trong phần code ví dụ trên, ta chỉ cài đặt để thu thập 10 trang đầu tiên trong trang tìm kiếm (vì trang web giới hạn số trang hiển thị).

2. **Xây dựng mô hình:** Trong project này, chúng ta sẽ sử dụng dữ liệu thơ đã thu thập được và thực hiện fine-tuning mô hình GPT2. GPT2 (Generative Pre-trained Transformer 2) là một mô hình ngôn ngữ lớn phiên bản thứ 2 trong chuỗi các mô hình họ GPT được phát triển bởi OpenAI. GPT2 được xây dựng dựa trên kiến trúc Transformer Decoder-only, các bạn có thể coi ảnh minh họa kiến trúc của GPT2 ở hình dưới đây:



Hình 6: Kiến trúc mô hình của GPT2. Nguồn: link

Ở phần này, chúng ta sẽ sử dụng mô hình GPT2 để thực hiện fine-tuning cho mục đích sinh thơ tiếng Việt. Theo đó, các bước thực hiện như sau:

(a) **Tải các thư viện cần thiết:**

```
!pip install -qq datasets==2.16.1 evaluate==0.4.1 transformers[sentencepiece]==4.35.2
!pip install -qq accelerate==0.26.1
!apt install git-lfs
```

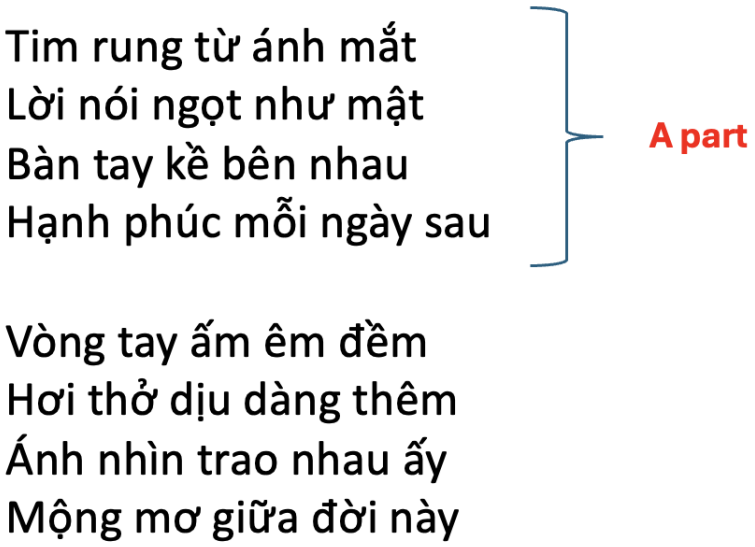
(b) **Import các thư viện cần thiết:** Chúng ta sẽ sử dụng thư viện HuggingFace với 2 module quan trọng là GPT2Tokenizer và GPT2LMHeadModel.

```
import os
import math
import torch
import pandas as pd

from transformers import GPT2Tokenizer, GPT2LMHeadModel
from transformers import DataCollatorForLanguageModeling
from transformers import TrainingArguments, Trainer
from huggingface_hub import notebook_login
from datasets import Dataset
```

(c) **Load bộ dữ liệu:** Với bộ dữ liệu đã thu thập được, chúng ta sẽ tiến hành đọc file .csv lên như sau:

```
DATASET_PATH = 'poem_final.csv'
df = pd.read_csv(DATASET_PATH)
df
```

(d) **Chuẩn bị bộ dữ liệu:** Hiện tại, đoạn thơ ta tách được có cấu trúc như sau:



Hình 7: Cấu trúc của một mẫu dữ liệu (một bài thơ) ta thu thập được.

Một bài thơ có thể có nhiều khổ thơ (part), một khổ thơ sẽ gồm 4 dòng thơ, mỗi dòng gồm 5 chữ. Để giảm độ phức tạp của bài toán, chúng ta sẽ coi mỗi khổ thơ là một data sample, và dùng chúng cho việc huấn luyện mô hình. Đầu tiên, ta xây dựng hàm tách nội dung thơ của một hành dữ liệu thành các danh sách chứa 4 dòng thơ:

```
def split_content(content):
    samples = []

    poem_parts = content.split('\n\n')
    for poem_part in poem_parts:
        poem_in_lines = poem_part.split('\n')
        if len(poem_in_lines) == 4:
            samples.append(poem_in_lines)

    return samples

df['content'] = df['content'].apply(lambda x: split_content(x))
df
```

Nhận thấy nội dung cột content của mỗi hàng dữ liệu là một list chứa các sublist. Ta sẽ thực hiện tách các sublist này thành một hàng trong bảng dữ liệu mới, coi như là một sample mới trong bộ dữ liệu. Cách thực hiện như sau:

```
df_exploded = df.explode('content')
df_exploded.reset_index(drop=True, inplace=True)
df_exploded = df_exploded.dropna(subset=['content'])
df_exploded
```

Hàm `df.explode()` sẽ giúp ta tách các phần tử trong một list thành các hàng mới. Khi thực thi xong đoạn code trên, ta có bảng dữ liệu mới như sau:

	Unnamed: 0	title	content	source	url
0	0	“Cái làm ta hạnh phúc”	[Cái làm ta hạnh phúc, Thực ra cũng chẳng nhiề...	Nguồn: Châm ngôn mới (thơ), Thái Bá Tân, NXB L...	https://www.thivien.net/Th%C3%A1i-B%C3%A1-T%C3...
1	0	“Cái làm ta hạnh phúc”	[Rồi thêm chút công việc, Cho ta làm hàng ngày...	Nguồn: Châm ngôn mới (thơ), Thái Bá Tân, NXB L...	https://www.thivien.net/Th%C3%A1i-B%C3%A1-T%C3...
2	1	“Chiều vừa xốp trên tay”	[Chiều vừa xốp trên tay, Chợt nghe thoáng ong ...	Nguồn: Lâm Huy Nhuận, Chiều có thật (thơ), NXB...	https://www.thivien.net/L%C3%A2m-Huy-Nhu%E1%BA...
3	1	“Chiều vừa xốp trên tay”	[Ớt đỏ sao cứ đỏ, Táo chín cho thật vàng, Em đ...	Nguồn: Lâm Huy Nhuận, Chiều có thật (thơ), NXB...	https://www.thivien.net/L%C3%A2m-Huy-Nhu%E1%BA...
4	2	“Dưới giàn hoa thiên lý...”	[Dưới giàn hoa thiên lý, Một mình anh đang ngồ...	Nguồn: Nguyễn Nhật Ánh, Mắt biếc, NXB Trẻ, 2004	https://www.thivien.net/Nguy%E1%BB%85n-Nh%E1%B...
...	...	...	...	...	...

Hình 8: Bảng dữ liệu sau khi sử dụng hàm explode của pandas để tách các khổ thơ thành các hàng dữ liệu mới. Có thể nhận thấy số hàng trong bảng dữ liệu đã tăng lên.

Ta cần nội dung thơ (giá trị của cột content) phải ở dạng string. Vì vậy, ta sẽ thực hiện convert nội dung content sang string như sau:

```
df_exploded['content'] = df_exploded['content'].apply(lambda x: '\n'.join(x))
df_exploded
```

	Unnamed: 0	title	content	source	url
0	0	“Cái làm ta hạnh phúc”	Cái làm ta hạnh phúc\nThực ra cũng chẳng nhiều...	Nguồn: Châm ngôn mới (thơ), Thái Bá Tân, NXB L...	https://www.thivien.net/Th%C3%A1i-B%C3%A1-T%C3...
1	0	“Cái làm ta hạnh phúc”	Rồi thêm chút công việc\nCho ta làm hàng ngày\...	Nguồn: Châm ngôn mới (thơ), Thái Bá Tân, NXB L...	https://www.thivien.net/Th%C3%A1i-B%C3%A1-T%C3...
2	1	“Chiều vừa xốp trên tay”	Chiều vừa xốp trên tay\nChợt nghe thoáng ong b...	Nguồn: Lâm Huy Nhuận, Chiều có thật (thơ), NXB...	https://www.thivien.net/L%C3%A2m-Huy-Nhu%E1%BA...
3	1	“Chiều vừa xốp trên tay”	Ớt đỏ sao cứ đỏ\nTáo chín cho thật vàng\nEm đẹ...	Nguồn: Lâm Huy Nhuận, Chiều có thật (thơ), NXB...	https://www.thivien.net/L%C3%A2m-Huy-Nhu%E1%BA...
4	2	“Dưới giàn hoa thiên lý...”	Dưới giàn hoa thiên lý\nMột mình anh đang ngồi...	Nguồn: Nguyễn Nhật Ánh, Mắt biếc, NXB Trẻ, 2004	https://www.thivien.net/Nguy%E1%BB%85n-Nh%E1%B...
...	...	...	...	...	...

Hình 9: Bảng dữ liệu sau khi chuyển đổi nội dung cột content sang dạng string.

Với DataFrame đã chuẩn bị xong, chúng ta sẽ đổi dạng dữ liệu pandas này sang HuggingFace dataset để thuận tiện trong việc sử dụng thư viện:

```
TEST_SIZE = 0.1
poem_dataset = Dataset.from_pandas(df_exploded)
poem_dataset = poem_dataset.train_test_split(test_size=TEST_SIZE)
```

```
poem_dataset = Dataset.from_pandas(df_exploded)
poem_dataset
```

```
Dataset({
    features: ['Unnamed: 0', 'title', 'content', 'source', 'url', '__index_level_0__'],
    num_rows: 441
})
```

```
TEST_SIZE = 0.1
poem_dataset = poem_dataset.train_test_split(test_size=TEST_SIZE)
poem_dataset
```

```
DatasetDict({
    train: Dataset({
        features: ['Unnamed: 0', 'title', 'content', 'source', 'url', '__index_level_0__'],
        num_rows: 396
    })
    test: Dataset({
        features: ['Unnamed: 0', 'title', 'content', 'source', 'url', '__index_level_0__'],
        num_rows: 45
    })
})
```

Hình 10: Bảng dữ liệu sau khi được đổi sang HuggingFace dataset.

(e) **Tiền xử lý dữ liệu:** Với bộ dữ liệu thơ đã sẵn sàng, chúng ta bắt đầu quy trình tiền xử lý bộ dữ liệu để chuẩn bị cho việc huấn luyện mô hình. Đầu tiên, ta khai báo tokenizer:

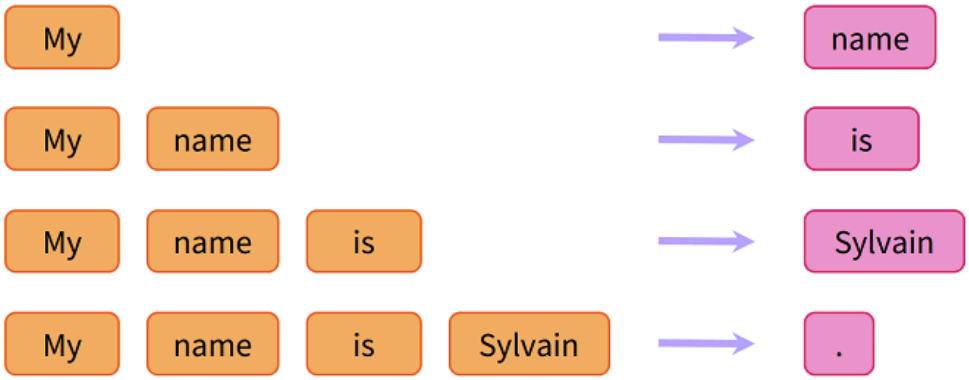
```
MODEL_NAME = 'danghuy1999/gpt2-viwiki'

tokenizer = GPT2Tokenizer.from_pretrained(MODEL_NAME)
```

Sau đó, xây dựng hàm chạy tokenization cho mỗi sample và thực thi chúng lên bộ dữ liệu:

```
tokenizer.pad_token = tokenizer.eos_token
MAX_SEQ_LEN = 100

def preprocess_function(row):
    return tokenizer(
        row['content'],
        max_length=MAX_SEQ_LEN,
        padding='max_length',
        truncation=True
    )

tokenized_poem_dataset = poem_dataset.map(
    preprocess_function,
    batched=True,
    num_proc=4,
    remove_columns=poem_dataset['train'].column_names,
)
```

Khi huấn luyện mô hình ngôn ngữ trong HuggingFace, chúng ta sẽ khai báo một instance từ class DataCollatorForLanguageModeling. Việc này nhằm giúp HuggingFace hỗ trợ chúng ta việc batching dữ liệu để việc huấn luyện mô hình ngôn ngữ trở nên hiệu quả hơn.

```
data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=
    False)
```



Hình 11: Ảnh minh họa việc đoán từ tiếp theo dựa trên các từ trước đó. Nguồn: link.

(f) **Huấn luyện mô hình:** Với bộ dữ liệu đã được tiền xử lý, ta sẽ bắt đầu việc huấn luyện mô hình. Đầu tiên, ta load pre-trained model GPT2:

```
model = GPT2LMHeadModel.from_pretrained(MODEL_NAME)
```

Sau đó, khai báo một vài các config trong việc huấn luyện mô hình:

```
training_args = TrainingArguments(
    output_dir='gpt2_viet_poem_generation',
    save_strategy='epoch',
    learning_rate=2e-5,
    num_train_epochs=10,
    weight_decay=0.01,
    fp16=True
)
```

Cuối cùng, ta thực thi trainer để tiến hành training:

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_poem_dataset['train'],
    eval_dataset=tokenized_poem_dataset['test'],
    data_collator=data_collator,
    tokenizer=tokenizer
)

trainer.train()
```

Như vậy, sau khi hoàn thành các bước trên, ta đã hoàn tất quá trình huấn luyện một mô hình sinh thơ tiếng Việt.

(g) **Inference:** Để sử dụng mô hình này, chúng ta có thể đưa lên HuggingFace hub và gọi mô hình xuống để sử dụng hoặc chúng ta có thể làm theo cách sau:

```
prompt = 'Học học nữa học mãi\n'
device = 'cuda' if torch.cuda.is_available() else 'cpu'
inputs = tokenizer(prompt, return_tensors="pt").input_ids.to(device)
outputs = model.generate(
    inputs,
    max_new_tokens=50,
    do_sample=True,
    top_k=50,
    top_p=0.95,
    temperature=0.8,
```

```
    repetition_penalty=1.2
)
results = tokenizer.batch_decode(outputs, skip_special_tokens=True)
results = results[0]
print()
for line in results.split('\n'):
    print(line)
```

```
Học học nữa học mãi
Hàng trăm ngàn đêm dài
Nhưng, không ai quên
Quanh ta thấy chuyện? đỏ?
```

Hình 12: Kết quả một khổ thơ được sinh ra từ mô hình.

Phần III. Câu hỏi trắc nghiệm

1. Mô hình Text Generation là?
   (*a*) Mô hình sinh chữ từ ảnh.
   (*b*) Mô hình sinh chữ từ video.
   (*c*) Mô hình sinh chữ từ một input nào đó.
   (*d*) Mô hình sinh chữ từ bản ghi âm thanh.

2. Ứng dụng nào sau đây thuộc về Text Generation?
   (*a*) Image Captioning.
   (*b*) Text Summarization.
   (*c*) Automatic Speech Recognition.
   (*d*) Tất cả các phương án trên.

3. Mục tiêu của bài toán Text Generation là?
   (*a*) Copy văn bản đầu vào.
   (*b*) Tạo văn bản mới dựa trên dữ liệu đầu vào.
   (*c*) Sửa văn bản đầu vào.
   (*d*) Không phương án nào đúng.

4. Các thách thức trong bài Text Generation?
   (*a*) Văn bản đầu ra có ngữ pháp đúng.
   (*b*) Văn bản đầu ra có nghĩa.
   (*c*) Văn bản đầu ra phải mạch lạc.
   (*d*) Tất cả các phương án trên.

5. Khi mô hình được thiết kế để dự đoán từ tiếp theo dựa trên một chuỗi các từ trước đó, ta gọi bài toán này là gì?
   (*a*) Causal Language Modeling.
   (*b*) Masked Language Modeling.
   (*c*) Sequence to Sequence
   (*d*) Denoising.

6. Mô hình GPT2 được xây dựng dựa theo kiến trúc nào?
   (*a*) Transformer Encoder-Decoder.
   (*b*) Transformer Encoder-only.
   (*c*) Transformer Decoder-only.
   (*d*) Long Short-Term Memory.

7. Selenium là?
   (*a*) Ngôn ngữ lập trình.
   (*b*) Trình duyệt web.
   (*c*) Một thư viện trong Python.

(*d*) Thiết kế mô hình học sâu.

8. Selenium thường được sử dụng trong việc?
   (*a*) Thu thập dữ liệu trên web.
   (*b*) Thiết kế giao diện web.
   (*c*) Tối ưu siêu tham số mô hình học sâu.
   (*d*) Thiết kế mô hình học sâu.
9. Dòng lệnh nào sau đây dùng để truy cập vào một trang web trong Selenium với đường dẫn cho trước:
   (*a*) `driver.get()`
   (*b*) `driver.switch_to_window()`
   (*c*) `driver.execute_script()`
   (*d*) `driver.close()`
10. Dòng lệnh nào sau đây dùng để tìm một thẻ html trong Selenium:
   (*a*) `driver.get()`
   (*b*) `driver.find_element()`
   (*c*) `driver.back()`
   (*d*) `driver.execute_async_script()`

- Hết -