

Ngày 10 – Chủ đề: Natural Language Processing

Một trong các mục tiêu của ngành Trí Thông Minh Nhân Tạo (AI – Artificial Intelligent) là hiểu được ngôn ngữ của con người để thực hiện mệnh lệnh. Các hệ thống sửa lỗi chính tả (Spellcheck), phân tích cảm xúc (sentiment analysis), hỏi đáp (question answering), chat bots, và trợ lý ảo (virtual assistants) đều có lõi là module xử lý ngôn ngữ tự nhiên (NLP – Natural Language Process).

Các bài tiếp theo sẽ thảo luận về xử lý textual data (dữ liệu văn bản). Phần code minh họa chủ yếu là dùng Python.

Ngày thứ mười này sẽ gồm 4 bài:

- **Bài 41:** Giúp bạn làm quen với vài kỹ thuật cơ bản để xử lý nội dung văn bản (text).
- **Bài 42:** Giúp rút trích đặc trưng của văn bản.
- **Bài 43:** Giúp bạn trải nghiệm một chút qua ứng dụng phân tích cảm xúc qua văn bản (sentiment).
- **Bài 44:** Giúp bạn trải nghiệm một chút qua ứng dụng phân tích từ vựng.

Bài 42: Các kỹ thuật cơ bản

Trước khi đi vào từng bài cụ thể thì chúng ta bàn một chút về các kỹ thuật cơ bản trong xử lý text. Trước là để làm quen thêm với Python. Sau là biết thêm một kỹ thuật trích thông tin từ văn bản, nghe thì có vẻ đơn giản nhưng sử dụng thành thạo thì sẽ rất hữu ích. Đây là một trong các việc đầu tiên khi xử lý dữ liệu về văn bản để trích xuất các thông tin quan trọng.

Văn bản thô (raw text)

Xem ví dụ sau đây:

```
print("https://thachln.github.io \ntouches on data science.")
```

Kết quả là nội dung text trong tham số của hàm `print` được hiển thị ra 2 dòng:

```
https://thachln.github.io
touches on data science.
```

Lý do là có kí tự đặc biệt `\n` trong văn bản. Nó là một kí tự đặc biệt có tên là newline character. Khi Python gặp kí tự này thì Python có xử lý đặc biệt là xuống hàng rồi mới hiển thị tiếp thông tin.

Để báo cho hàm `print` trong Python này biết `\n` là một nội dung bình thường thì thêm chữ `r` phía trước dấu nháy của chuỗi như sau:

```
print(r"https://thachln.github.io \ntouches on data
science.")
print(r'https://thachln.github.io \ntouches on data
science.')
```

```
https://thachln.github.io \ntouches on data science.
https://thachln.github.io \ntouches on data science.
```

Cách sử dụng kí tự `r` (raw) như thế này gọi là raw text.

Regular Expression

Regular Expression (RegEx) là cách để mô tả hoặc nhận diện hoặc so khớp (matching) một chuỗi các kí tự theo một qui tắc hoặc khuôn mẫu (pattern) nào đó. Ví dụ để kiểm tra một câu tiếng Anh bắt đầu bằng chữ Hello rồi **đến chữ gì gì đó tiếp theo** (có thể là tên một người) thì chúng ta viết Hello+. Nếu để kiểm tra câu bắt đầu bằng chữ Hello, sau đó **có hoặc không có** chữ gì đấy thì chúng ta viết Hello*.

Dưới đây là bảng tóm tắt cách viết RegEx:

Kí hiệu	Ý nghĩa	Ví dụ
^ và \$	^: Bắt đầu với... \$: Kết thúc với...	“&The” có nghĩa bất cứ câu chữ nào bắt đầu với chữ The

		“you\$” có nghĩa là bất cứ câu chữ nào kết thúc với chữ you
x*	Có (nhiều) hoặc không có bất cứ kí tự x nào	“Hello*” sẽ khớp với bất cứ chuỗi Hello bắt đầu bằng chữ Hell rồi sau đó có các chữ o theo sau như: Hello, Helloo
x+	Có ít nhất một kí tự x	“Hello+” sẽ khớp với bất cứ chuỗi nào bắt đầu bằng chữ Hello rồi sau đó có ít nhất một kí tự o như: Helloo
x?	Có một kí tự x hoặc không có kí tự nào	“Hello?” sẽ khớp với các từ như: Hell, Hello. Nhưng không khớp với HelloY, Helloi Hello Team
x{n,m}	kí tự x bất kì xuất hiện liên tục từ n đến m lần	Hello(2,5) sẽ khớp với với các chuỗi Helloo, Hellooo, Helloooo, Hellooooo
.	Bất kỳ kí tự nào	“a.” sẽ khớp với các chuỗi at, ac...
	Hoặc.	“a the” sẽ khớp với chuỗi kí tự “a”, “the”.
[]	Gặp bất kì kí tự nào bên trong cặp dấu móc vuông	“[ai]t” sẽ khớp với chuỗi “at”, “it”. Ví dụ khác kết hợp và []: “[a the] book” sẽ khớp với các từ “a book”, “the book”.
\w	Một kí tự trong bảng chữ cái alphabet	“\w\s\d” sẽ khớp với các chuỗi như: “a 1”, “x 9”. Tức là bắt đầu bằng một chữ, sau đó là khoảng cách, tiếp theo là 1 số.
\d	Một kí số từ 0 đến 9	
\s	Khoảng trắng (space)	
\S	Một kí tự không phải là khoảng trắng	Chú ý chữ S HOA

Để làm quen RegEx trong Python thì sử dụng thư viện `re`. Bạn cũng nên đọc tài liệu giải thích ở đây <https://docs.python.org/2/library/re.html>.

```
import re
```

Lệnh `sub` sau đây có dạng `rs.sub(pattern, repl, string)` là để thay thế các mẫu (tham số `pattern`) trong tham số `string` bằng giá trị của tham số `repl`.

```
re.sub("https?:\/\/\S+\s", '', "https://thachln.github.io touches on data science.")
```

```
'touches on data science.'
```

Phân tích một chút pattern trong tham số đầu tiên:

Pattern “s?” có nghĩa là có kí tự s có hoặc không có. Vì thế https? có nghĩa là http hoặc https. Pattern “https?:\\.” có nghĩa là đường link web có thể là http:// hoặc https://. Hãy thử lệnh sau sẽ cho ra kết quả tương tự. Link lúc này là http://thachln...

```
re.sub("https?:\\S+", '', "http://thachln.github.io  
touches on data science.")
```

\\S+ có nghĩa là các kí tự không phải khoảng trắng.

\\s là space (khoảng trắng)

Tức là pattern "https?:\\S+" có nghĩa là sau http:// hoặc https:// là các kí tự liên tục nhau không có space. Vì thế nội dung string trong tham số thứ ba chính là phần này: "https://thachln.github.io ". Phần này sẽ được thay thế bởi tham số thứ hai " (chuỗi rỗng). Thay bằng chuỗi rỗng tức là xóa đi.

Sau khi xóa theo pattern thì nội dung string còn lại là “touches data science”.

Chuẩn bị dữ liệu

Tiếp theo để minh họa cho các kỹ thuật xử lý dữ liệu văn bản, chúng ta sẽ lấy dữ liệu từ một bài báo. Kỹ thuật lấy dữ liệu gọi là scraping. Sau đó chúng ta sẽ xử lý nội dung bằng cách xóa bớt các từ “không có giá trị” để giữ lại các từ khóa của văn bản. “Không có giá trị” sẽ được hiểu khác nhau tùy theo tiêu chuẩn và mục tiêu của vấn đề đang xử lý.

Cần cài đặt vài thư viện

```
pip install requests  
pip install cssselect
```

Bước 1: Lấy nội dung của trang web.

Đoạn code Python sau sử dụng thư viện requests để lấy nội dung trang “https://www.nguyenvantuan.info/research-interests”. Trong trường hợp link này không còn thì bạn có thể thay thế bởi link khác và tập thực hành tương tự. Nội dung cả trang web được lưu vào biến content.

```
from bs4 import BeautifulSoup  
import requests  
  
url = 'https://www.nguyenvantuan.info/research-interests'  
  
response = requests.get(url)
```

```
html_soup = BeautifulSoup(response.text, 'html.parser')

article_containers = html_soup.find('div', {'id':
'yi9cainlineContent-gridWrapper'})

content = article_containers.text
print(content)
```

Research interestsMy research work is primarily based at the Garvan Institute of Medical Research's Bone Biology Division. I pursue both epidemiological and genetic research, often combining the two, to address issues that are transformational, shaping policy and practice leading to better treatment and control of osteoporosis.

...

Ghi chú: Phần ... là còn nữa.

Xử lý kỹ thuật

Tiếp theo là sẽ làm gọn thông tin này bằng cách xóa các nội dung không liên quan. Đồng thời kích thước của nội dung sẽ được làm giảm lại để phù hợp cho các phân tích tiếp theo.

Bước 2: Xem kích thước của nội dung (tính bằng số kí tự)

```
len(content)
```

4029

Bước 3: Trong nội dung phân tích thì chúng ta không quan tâm đến các liên kết (hyperlink) nên sẽ loại bỏ nó (nếu có) bằng cách sử dụng thư viện `re`, hàm `sub`.

```
import re
content = re.sub(r"https?:\/\/\S+", '', content)
```

Bước 4: Xóa các kí tự đặc biệt

```
content = re.sub(r'[_"\-;%()|+&=*%.,!?:#$@\[\\]/][\xa0]', '',
content)
len(content)
```

4015

Bước 6: Xóa các thẻ web

```
content = re.sub(r'<.*?>', '', content)
```

Bước 7: Thay thế các thẻ xuống hàng, kí tự xuống hàng \n bởi khoảng trắng

```
content = re.sub(r'<br/>', ' ', content)
content = re.sub(r'\n', ' ', content)
```

Bước 8: Xóa bớt các từ như cannot can't, it's, it is:

```
content = re.sub(r"[can\t][cannot][it\s][it is]", '',
content)

len(content)
```

3477

Bước 9: Tìm các từ có 1 hoặc 2 kí tự

```
re.findall(r"\b[A-z]{1,2}\b", content)
```

Kết quả sẽ có rất nhiều từ như:

```
['is',
 'at',
 'of',
 's',
 ...
```

Bước 10: Tìm các từ bắt đầu bằng chữ hoa

```
re.findall(r"[A-Z][a-z]*", content)
```

```
['Research',
 'My',
 'Garvan',
 'Institute',
 'Medical',
 ...
```

Trên đây là các thao tác cơ bản mà bạn nên làm quen để xử lý văn bản.

Bài 43: Trích đặc trưng (Feature extraction)

Để hiểu văn bản thì một trong các kỹ thuật là phải biết được các thông tin của văn bản qua các đặc trưng của nó. Ví dụ trong dữ liệu mà chúng ta sẽ minh họa trong bài này, quan sát số từ trung bình trong một câu văn, hoặc số kí tự đặc biệt trong một câu văn cũng nói lên nhiều thông tin.

Dữ liệu chúng ta sử dụng là dataset về đánh giá phim từ người xem (movie review dataset).

Đọc dữ liệu

Đọc dữ liệu file .zip từ Internet vào dataframe:

```
import pandas as pd
fp = 'https://thachln.github.io/datasets/movie_reviews.zip'
df = pd.read_csv(fp, compression='zip', encoding='latin-1')
df.head()
```

	SentimentText	Sentiment
0	first think another Disney movie, might good, ...	1
1	Put aside Dr. House repeat missed, Desperate H...	0
2	big fan Stephen King's work, film made even gr...	1
3	watched horrid thing TV. Needless say one movi...	0
4	truly enjoyed film. acting terrific plot. Jeff...	1

Dữ liệu này gồm có 2 cột: SentimentText là nội dung người xem góp ý, hoặc bày tỏ cảm xúc. Cột Sentiment có 2 giá trị:

- 1 là tích cực (positive sentiment).
- 0 là tiêu cực (negative sentiment).

Các khác để xem data:

```
df.iloc[0]
```

```
SentimentText    first think another Disney movie, might good, ...
Sentiment                                              1
Name: 0, dtype: object
```

```
df.SentimentText[0]
```

```
"first think another Disney movie, might good, it's kids movie. watc
h it, can't help enjoy it. ages love movie. first saw movie 10 8 yea
rs later still love it! Danny Glover superb could play part better.
Christopher Lloyd hilarious perfect part. Tony Danza believable Mel
Clark. can't help, enjoy movie! give 10/10!"
```

Tiền xử lý dữ liệu văn bản (Text pre-processing)

Để “hiểu” thêm về dữ liệu thì chúng ta sẽ phân tích một vài chỉ số thống kê. Có thể coi đây là một bài tập thống kê mô tả về xử lý văn bản kết hợp với xử lý dữ liệu ban đầu (gọi là pre-processing).

Số từ (number words)

Đếm số từ của 'SentimentText' và thêm cột word_count vào dataframe:

```
df['word_count'] = df['SentimentText'].apply(lambda x:
len(str(x).split(" ")))
df.head()
```

	SentimentText	Sentiment	word_count
0	first think another Disney movie, might good, ...		152
1	Put aside Dr. House repeat missed, Desperate H...		86
2	big fan Stephen King's work, film made even gr...		193
3	watched horrid thing TV. Needless say one movi...		63
4	truly enjoyed film. acting terrific plot. Jeff...		65

Ý tưởng của hàm apply đối với cột SentimentText là quét từng dòng dữ liệu rồi áp dụng kỹ thuật lamda: từng dữ liệu được lưu vào biến x và áp dụng kết quả xử lý bằng biểu thức sau dấu :

Biểu thức lamda ở đây là: `len(str(x).split(" "))` có cú pháp là `len(a)`. Trong đó a là `str(x).split(" ")` có nghĩa là:

- Chuyển x thành chuỗi bởi hàm `str(x)`
- Sau đó tách các từ của chuỗi x bởi dấu cách (space): " "

`len(a)` là đếm số từ sau khi tách.

Kết quả của hàm apply ở trên là danh sách số lượng từ của mỗi dòng dữ liệu trong cột word_count: 52, 86,...

Số từ trung bình

Tính số từ trung bình bằng cách sử dụng hàm `mean()` đối với cột dữ liệu trong pandas dataframe.

Lệnh sau tính số từ trung bình của góp ý tiêu cực:

```
df.loc[df.Sentiment == 0, 'word_count'].mean()
```


129.08048

Lệnh sau tính số từ trung bình của góp ý tích cực:

```
df.loc[df.Sentiment == 1, 'word_count'].mean()
```

132.48864

Nói chung là số từ trung bình giữa hai loại góp ý không có khác biệt lớn.

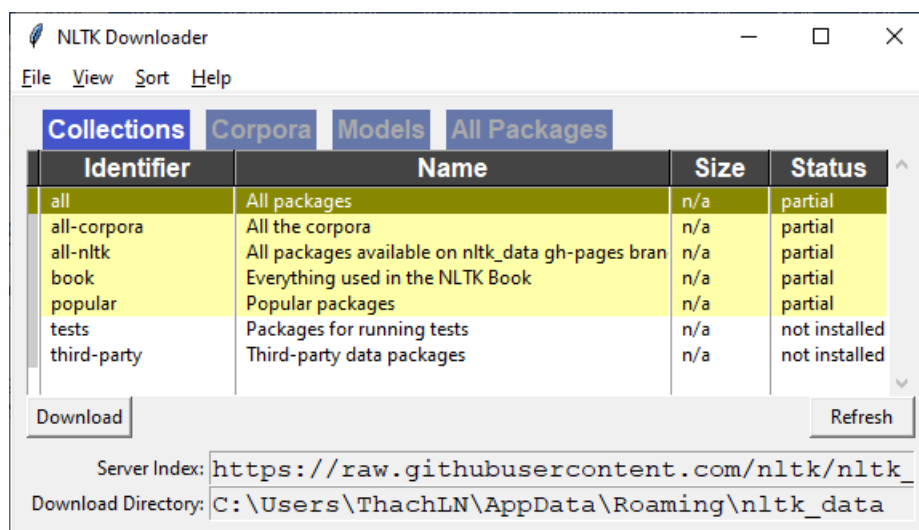
Stop words

Stop words được hiểu là những từ chung chung trong (common words) ngôn ngữ. Ví dụ trong tiếng Anh, stop words là "I", "me", "my", "yours", and "the.". Trong xử lý ngôn ngữ tự nhiên bằng máy tính cho đến thời điểm hiện tại thì các từ này nói chung là chưa có ý nghĩa gì thực sự. Vì vậy trong Python có thư viện `nltk` cung cấp sẵn các stop words (có hỗ trợ rất nhiều ngôn ngữ).

Để sử dụng thư viện `nltk` cần download dữ liệu bởi lệnh sau:

```
import nltk
nltk.download()
```

Nếu bạn chạy lệnh trên trong Spyder thì cửa sổ sau sẽ bật lên. Bấm nút **Download** để tiếp tục.



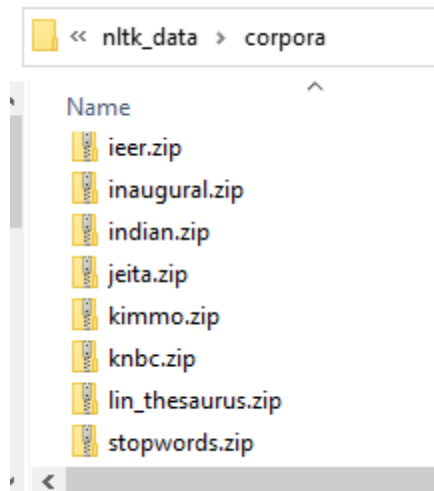
Khi nào cột Status có giá trị Installed cho biết đã tải xong. Khi tất cả các mục đã được tải xong thì bạn có thể đóng cửa sổ NLTK Downloader.

Dữ liệu của `nltk` được tải và lưu vào thư mục:

`%USERPROFILE%\AppData\Roaming\nltk_data`

`USERPROFILE` là biến môi trường trên Windows chứa đường dẫn thư mục của người dùng. Ví dụ trên máy tôi là `C:\Users\ThachLN`. Bạn chỉ cần copy đường

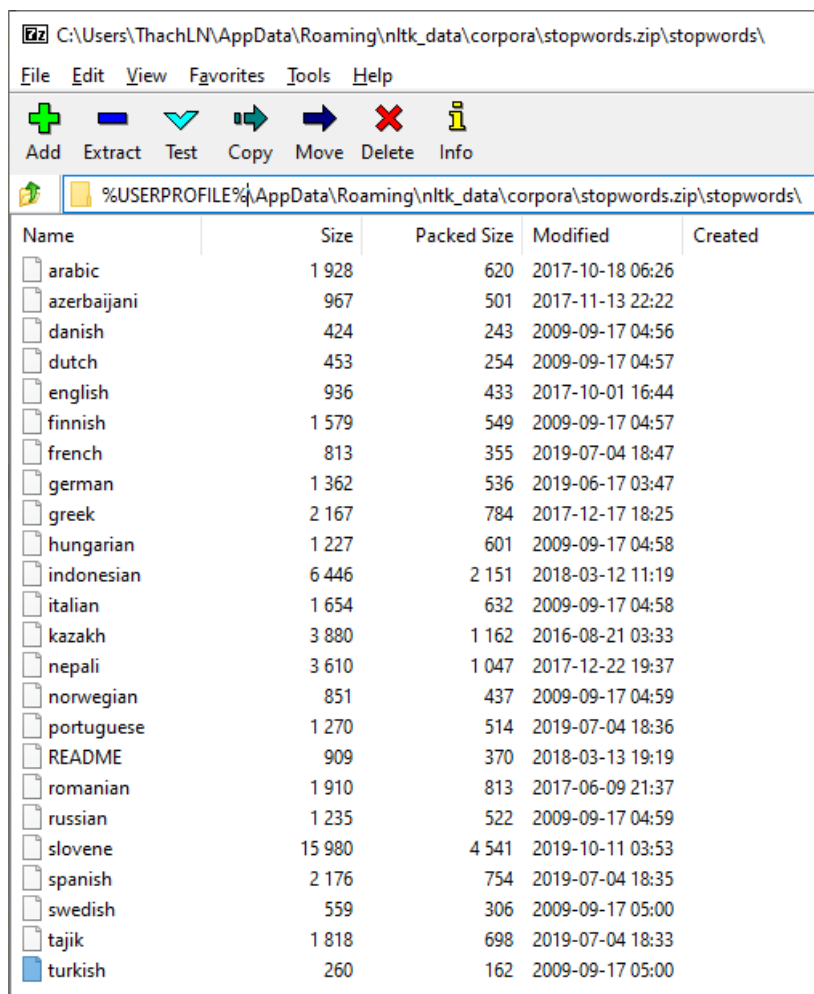
dẫn trên và paste vào thanh Address của phần mềm quản lý file Windows Explorer rồi nhấn Enter để xem nội dung thư mục.



Nếu bạn xem nội dung file stopwords.zip trong thư mục:

“%USERPROFILE%\AppData\Roaming\nltk_data\corpora\stopwords.zip\stop words” bằng phần mềm giải nén như 7-zip²¹ thì sẽ không thấy file cho tiếng Việt. Hy vọng sớm có ai đó đóng góp stopwords tiếng Việt vào thư viện nltk này.

²¹ <https://www.7-zip.org/>



Name	Size	Packed Size	Modified	Created
arabic	1 928	620	2017-10-18 06:26	
azerbaijani	967	501	2017-11-13 22:22	
danish	424	243	2009-09-17 04:56	
dutch	453	254	2009-09-17 04:57	
english	936	433	2017-10-01 16:44	
finnish	1 579	549	2009-09-17 04:57	
french	813	355	2019-07-04 18:47	
german	1 362	536	2019-06-17 03:47	
greek	2 167	784	2017-12-17 18:25	
hungarian	1 227	601	2009-09-17 04:58	
indonesian	6 446	2 151	2018-03-12 11:19	
italian	1 654	632	2009-09-17 04:58	
kazakh	3 880	1 162	2016-08-21 03:33	
nepali	3 610	1 047	2017-12-22 19:37	
norwegian	851	437	2009-09-17 04:59	
portuguese	1 270	514	2019-07-04 18:36	
README	909	370	2018-03-13 19:19	
romanian	1 910	813	2017-06-09 21:37	
russian	1 235	522	2009-09-17 04:59	
slovene	15 980	4 541	2019-10-11 03:53	
spanish	2 176	754	2019-07-04 18:35	
swedish	559	306	2009-09-17 05:00	
tajik	1 818	698	2019-07-04 18:33	
turkish	260	162	2009-09-17 05:00	

Thử khai báo thư viện và nạp kho từ vựng stopwords cho tiếng Anh

```
from nltk.corpus import stopwords  
stop = stopwords.words('english')
```

Để sử dụng stopwords tiếng Việt thì hãy tải file tại raw.githubusercontent.com/stopwords/vietnamese-stopwords/refs/heads/master/vietnamese-stopwords.txt và lưu vào %USERPROFILE%\AppData\Roaming\nltk_data\corpora\stopwords. Chú ý lưu thành file “vietnamese” không có phần đuôi .txt.

Xem danh sách từ trong stopwords tiếng Anh:

```
print(stop)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',  
"you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',  
'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her',  
'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 't  
heir', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this  
, 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'w  
ere', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do',  
'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',  
'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', ']
```

```
about', 'against', 'between', 'into', 'through', 'during', 'before',  
'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',  
'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'h  
ere', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',  
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor',  
'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't',  
'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',  
'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'coul  
dn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn  
't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mi  
ghn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "  
shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't  
't", 'won', "won't", 'wouldn', "wouldn't"]
```

Đếm thử số stopwords trong dataset:

```
df['stop_count'] = df['SentimentText'].apply(lambda x:  
                                              len([x for x in x.split() if x in stop]))  
  
print(df['stop_count'])
```

```
0      1  
1      2  
2      3  
3      1  
4      2  
..  
24995   0  
24996   0  
24997   2  
24998   1  
24999   1  
Name: stop_count, Length: 25000, dtype: int64
```

Thử đếm số từ stopwords trung bình trong hai nhóm góp ý:

```
agv_sw_active = df.loc[df.Sentiment == 1,  
                        'stop_count'].mean()  
  
agv_sw_passive = df.loc[df.Sentiment == 0,  
                        'stop_count'].mean()  
  
print('Số từ stopwords trung bình trong góp ý tích cực:',  
      agv_sw_active)  
  
print('Số từ stopwords trung bình trong góp ý tiêu cực:',  
      agv_sw_passive)
```

```
Số từ stopwords trung bình trong góp ý tích cực: 1.49064  
Số từ stopwords trung bình trong góp ý tiêu cực: 1.94104
```

Kí tự đặc biệt

Tùy theo vấn đề chúng ta đang phân tích thì các kí tự đặc biệt như ^, &, *, \$, @, # có ý nghĩa gì hay không? Trong trường hợp này chúng ta chưa quan tâm đến các từ này nên có thể loại bỏ chúng đi. Trước tiên đếm chúng xem bao nhiêu:

```
df['special_count'] = df['SentimentText'].apply(lambda x:
                                                len(re.sub('[^\^&*$@#]+'
, '', x)))
print(df['special_count'])
```

```
0      0
1      1
2      4
3      0
4      0
..
24995   0
24996   1
24997   0
24998   0
24999   0
Name: special_count, Length: 25000, dtype: int64
```

Xử lý nội dung văn bản (Text processing)

Để áp dụng Machine learning vào giải quyết vấn đề thì chúng ta cần thêm vài bước xử lý nữa.

Dùng chữ thường (Lowercase)

Về ý nghĩa nội dung đối với chúng ta thì chữ Hoa hay chữ thường thì như nhau. Nhưng đối với máy tính thì là khác nhau. Vì vậy cần chuyển đổi hết nội dung về một loại. Ví dụ đoạn code sau sẽ chuyển nội dung góp ý về chữ thường (lowercase)

```
df['SentimentText'] = df['SentimentText'].apply(lambda x:
                                                " ".join(x.lower() for x in
x.split()))
```

Loại bỏ stopwords

```
df['SentimentText'] = df['SentimentText'].apply(lambda x:
                                                " ".join(x for x in x.split() if x not
in stop))
```

Quan sát các từ lặp lại

Thử liệt kê các từ lặp lại nhiều nhất:

```
word_freq = pd.Series(''.join(df['SentimentText']).split()).value_counts()
word_freq.head()
```

```
 /><br      50931
movie      30883
film       27774
one        22476
like       18776
dtype: int64
```

Kết quả cho thấy các từ xuất hiện nhiều nhưng không liên quan đến “cảm xúc” tích cực/tiêu cực của lời góp ý mà chúng ta đang phân tích như: /><br, movie, file.

Vì thế sẽ loại bỏ chúng:

```
df['SentimentText'] = df['SentimentText'].str.replace(r'<br />', '')
df['SentimentText'] = df['SentimentText'].apply(lambda x:
        " ".join(x for x in x.split() if x not in ['movie',
        'film']))
```

Kiểm tra các từ không phải chữ cái, hoặc số (Punctuation)

```
test_punc = df['SentimentText'].apply(lambda x:
        re.findall(r"^[A-Za-z0-9\s]+", x))

test_punc
```

```
0      [, , , , , , ' , , , ! , , , , , ' , , , ! , / , !]
1      [ , , , ( , ) , , , , ' , , , , , , , ' , , , ( , ) , , ...
2      [' , , , , , , , , , ' , ( , ) , , , , ( , ) , , , ...
3      [ , , , , , , - , , , " , " , , , , " , " , , , , , ...
4      [ , , , , , , , , , ( , ) , - , , , ' , ' , , , , , .]

24995      [' , ' , , , ? , , , , , , , , , , , ! , , , , .]
24996      [ , , ' , ' , , , * , , , , , , , ' , ' , - , ' , , , , ' , ...
24997      [" , ' , " , ' , , , , ' , , , , , , , ; , ' , , , , - , ...
24998      [ , , , - , , , - , , , , / , , , , , .]
24999      [ , , ' , ' , , , , , , , , , , , , , / , .]
Name: SentimentText, Length: 25000, dtype: object
```

Loại bỏ các Punctuation:

```
punc_special = r"^[A-Za-z0-9\s]+"  
df['SentimentText'] =  
df['SentimentText'].str.replace(punc_special, '')
```

Sửa chính tả

Đôi khi người dùng gõ sai chính tả, nếu không sửa lại thì chất lượng phân tích sẽ ảnh hưởng xấu. Vì thế Python có thư viện để sửa lại lỗi sai này.

Ví dụ:

Cài thêm thư viện:

```
pip install autocorrect
```

Đoạn code sau minh họa sử dụng module Speller để sửa lỗi chính tả tiếng Anh:

```
from autocorrect import Speller  
spell = Speller(lang='en')  
spell('Teh')
```

Kết quả là từ Teh sẽ được sửa lại là:

```
'The'
```

Áp dụng cho dữ liệu Sentiment của chúng ta.

```
from autocorrect import Speller  
spell = Speller(lang='en')  
df['SentimentText'] = [' '.join([spell(i) for i in  
x.split()]) for x in df['SentimentText']]
```

Đoạn code này chạy hơi lâu tùy theo cấu hình máy tính của bạn.

Chuyển về từ gốc (Stemming & Lemmatization)

Trong tiếng Anh, các hậu tố ly, ness, iest, ing có thể ghép với các từ để diễn đạt các tính chất, trạng thái, hoặc theo thì (tense) khác nhau. Tuy nhiên trong phân tích của chúng ta thì cần quy về từ gốc (root form) trước khi phân tích. Ví dụ các từ happy, happily, happiest thì đều có một nghĩa là happy. Trong Python, thư viện nltk, có module PorterStemmer giúp chúng ta làm việc này.

```
from nltk.stem import PorterStemmer  
stemmer = PorterStemmer()  
stemmer.stem("happy")  
stemmer.stem("happily")
```

```
stemmer.stem("happinest")
```

Kết quả

```
'happi'  
'happili'  
'happinest'
```

Thật sự thì module PorterStemmer không tốt mấy!

Thử áp dụng vào dữ liệu Sentiment:

```
from nltk.stem import PorterStemmer  
stemmer = PorterStemmer()  
df['SentimentText'] = df['SentimentText'].apply(lambda x:  
                                                    " ".join([stemmer.stem(word) for word in  
                                                    x.split()])))
```

Chú ý PorterStemmer có thể chạy rất lâu. Nếu bạn đợi không được thì hãy ngừng và bỏ qua bước này.

Một cách khác bổ sung thêm cho Stemming là Lemmatization:

Ví dụ:

```
import nltk  
lemmatizer = nltk.stem.WordNetLemmatizer()  
lemmatizer.lemmatize('walks')
```

Kết quả trả về từ gốc của walks là:

```
'walk'
```

Thử tiếp:

```
lemmatizer.lemmatize('walking')
```

Kết quả trả về:

```
'walking'
```

Bạn có thể thử thêm để đánh giá chất lượng của Stemming và Lemmatization.

Áp dụng vào để xử lý Sentiment của bài toán hiện tại:

```
import nltk  
lemmatizer = nltk.stem.WordNetLemmatizer()  
df['SentimentText'][:5].apply(lambda x: "  
".join([lemmatizer.lemmatize(word) for word in x.split()])))
```


Tách từ (Tokenization)

Tiếp theo là kỹ thuật để tách nội dung văn bản thành các từ (gọi là token). Trong Python có thể dùng thư viện `keras` như sau:

Cài thêm thư viện `keras`:

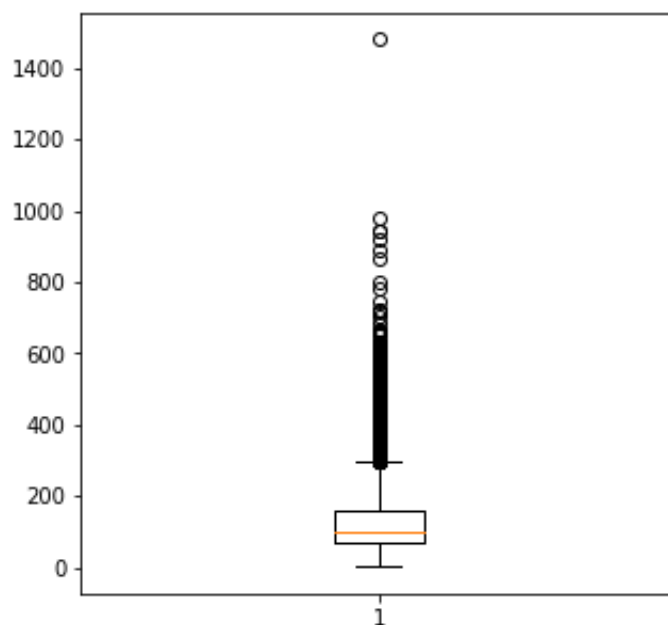
```
pip install keras
```

Dùng thư viện `keras` để tách từ:

```
from keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words=250)
tokenizer.fit_on_texts(list(df['SentimentText']))
sequences = tokenizer.texts_to_sequences(df['SentimentText'])
```

Quan sát biểu đồ

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(5, 5))
plt.boxplot(df.word_count)
plt.show()
```



Tóm tắt:

Như vậy bạn đã hình dung và làm quen được vài kỹ thuật cơ bản để xử lý text trong việc xử lý ngôn ngữ tự nhiên.

Bài 43 Mở rộng

Sử dụng ChatGPT để học phương pháp “**Thematic Analysis**” và thử áp dụng quy trình sau để phân tích bài viết của TBT Tô Lâm ngày nhân ngày 30/4/2025.

Thematic Analysis – Quy trình 6 bước (theo Braun & Clarke)

1. Familiarization (Làm quen với dữ liệu)

- Đọc kỹ bài báo nhiều lần.
- Ghi chú ban đầu, xác định các ý chính hoặc điểm nổi bật.

2. Generating Initial Codes (Tạo mã sơ bộ)

- Mã hóa (code) các đoạn dữ liệu có liên quan theo ý nghĩa (dạng nhãn).
- Một đoạn văn có thể chứa nhiều mã.

3. Searching for Themes (Tìm kiếm chủ đề)

- Nhóm các mã có liên quan lại thành **chủ đề (themes)**.
- Dùng sơ đồ cây hoặc bảng để tổ chức các nhóm mã.

4. Reviewing Themes (Rà soát chủ đề)

- Đánh giá xem các chủ đề có logic, đầy đủ, và liên kết tốt với dữ liệu gốc không.
- Gộp, chia nhỏ hoặc bỏ các chủ đề nếu cần thiết.

5. Defining and Naming Themes (Định nghĩa và đặt tên chủ đề)

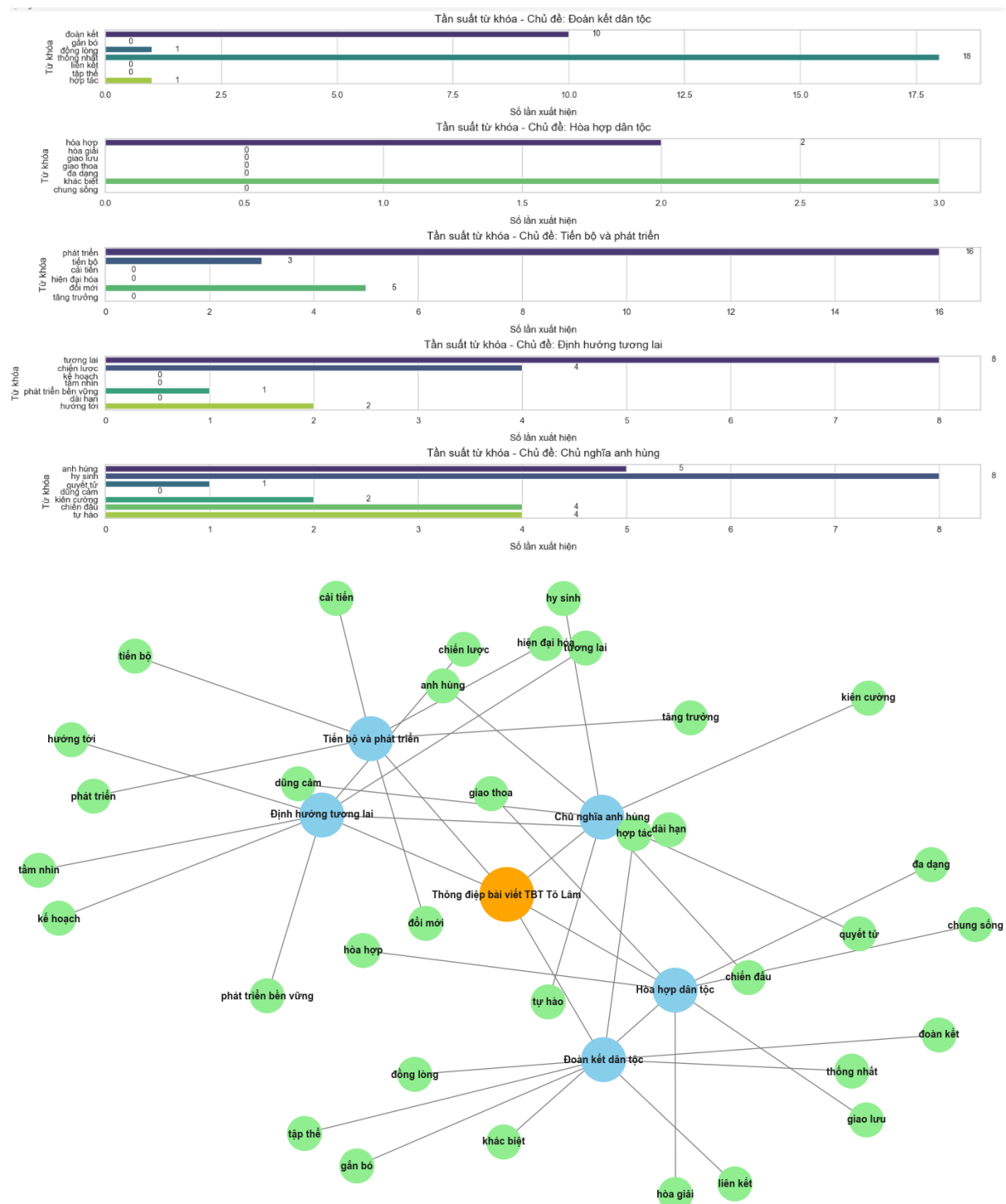
- Viết mô tả ngắn gọn và rõ ràng cho từng chủ đề.
- Đặt tên chủ đề sao cho phản ánh được nội dung cốt lõi.

6. Writing the Report (Viết báo cáo kết quả)

- Tóm tắt các chủ đề chính.
- Minh họa bằng trích dẫn từ bài báo (nếu cần).
- Liên hệ với câu hỏi nghiên cứu và các khung lý thuyết.

Biểu đồ minh họa:

Ứng dụng Phân tích dữ liệu và Trí tuệ nhân tạo với Python



Xem mã nguồn gợi ý từ ChatGPT tại <https://thachln.github.io/datasets/text/thematic-analysis.py>