ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ

BỘ MÔN: PHÂN TÍCH DỮ LIỆU THÔNG MINH

Giảng viên hướng dẫn:

Phạm Trọng Nghĩa

Nhóm 4 - Vượt Khó

20120210 – TRẦN THỊ KIM TIẾN
20120224 – TRẦN THỊ MỸ TRINH
20120231 – PHAN HUY TRƯỜNG
20120307 – PHẠM GIA KHIÊM
20120328 – HOÀNG ĐỨC NHẬT MINH
20120578 – PHẠM QUỐC THÁI

Thành phố Hồ Chí Minh - 2023

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

BỘ MÔN: PHÂN TÍCH DỮ LIỆU THÔNG MINH

Giảng viên hướng dẫn:

Phạm Trọng Nghĩa

Thành phố Hồ Chí Minh - 2023

LỜI CẢM ƠN

Trong quá suốt quá trình học tập môn học Phân tích dữ liệu thông minh và thực hiện đồ án này, nhóm chúng em đã nhận được nhiều sự hướng dẫn góp ý, giúp đỡ tận tình từ thầy và bạn bè.

Nhóm em rất cảm ơn thầy Phạm Trọng Nghĩa, khoa Công nghệ thông tin trường Đại học Khoa học Tự nhiên, Đại học Quốc gia TPHCM đã hướng dẫn và truyền đạt rất nhiều kiến thức bổ ích với chúng em trong suốt thời gian qua.

Cảm ơn chân thành những lời góp ý của bạn bè và sự góp sức của bản thân các thành viên trong nhóm hoàn thành đồ án.

Trân trọng cảm ơn!

MỤC LỤC

TỔNG QUAN VỀ NHÓM

I. Thông tin nhóm

STT	Họ và tên	Mã số sinh viên	Email
1	Trần Thị Kim Tiến	20120210	20120210@student.hcmus.edu.vn
2	Trần Thị Mỹ Trinh	20120224	20120224@student.hcmus.edu.vn
3	Phan Huy Trường	20120231	20120231@student.hcmus.edu.vn
4	Phạm Gia Khiêm	20120307	20120307@student.hcmus.edu.vn
5	Hoàng Đức Nhật Minh	20120328	20120328@student.hcmus.edu.vn
6	Phạm Quốc Thái	20120578	20120578@student.hcmus.edu.vn

Bảng 1. Thông tin thành viên trong nhóm

II. Phân chia công việc

STT	Phần công việc	Phụ trách
1	Khám phá dữ liệu (EDA)	Mỹ Trinh
2	Tiền xử lý dữ liệu	Quốc Thái Huy Trường
3	Xây dựng mô hình	Gia Khiêm Nhật Minh Kim Tiến
4	Cải thiện mô hình	Gia Khiêm Nhật Minh
5	Tìm hiểu lý thuyết mô hình	Quốc Thái Huy Trường
6	Làm báo cáo	Kim Tiến Mỹ Trinh
7	Quay video	Cả nhóm

Bảng 2. Bảng phân chia công việc

III. Video trình bày đồ án

Đường dẫn:

https://drive.google.com/file/d/1P-iorRwk_3gTU3eBUQAGBCdA1SJfQzDQ/view?usp=sharing

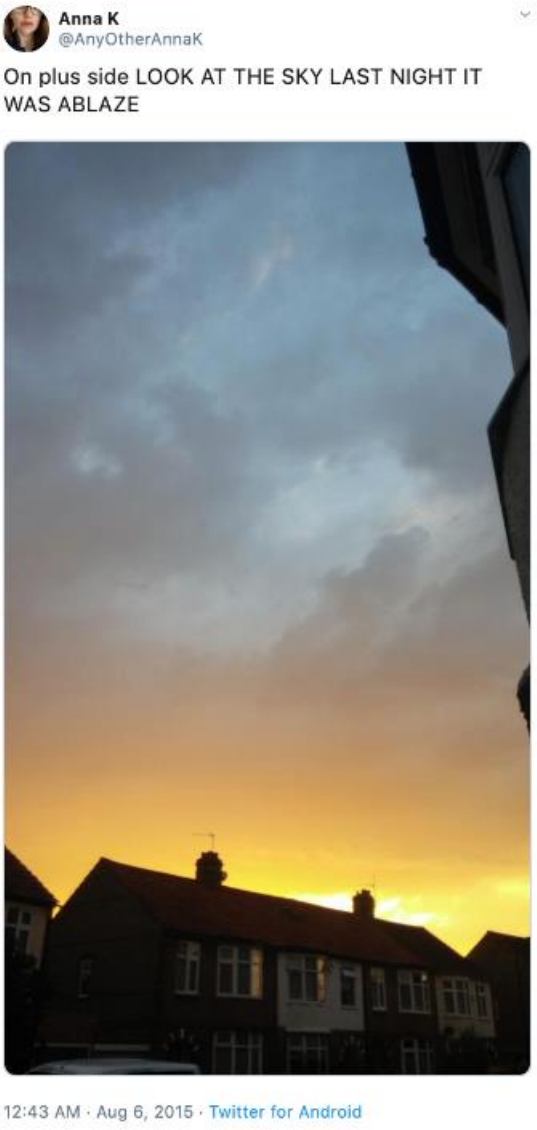
A – GIỚI THIỆU CHUNG VỀ ĐỒ ÁN

Twitter đã trở thành một kênh liên lạc quan trọng trong trường hợp khẩn cấp.

Sự phổ biến của điện thoại thông minh cho phép mọi người thông báo các trường hợp khẩn cấp mà họ đang quan sát thấy trong thời gian thực. Do đó, nhiều tổ chức quan tâm đến việc xây dựng một chương trình tự động theo dõi Twitter nhằm phát hiện các tin khẩn cấp được người dùng đăng lên Twitter (chẳng hạn như các tổ chức cứu trợ thảm họa và hãng thông tấn báo chí).

Tuy nhiên, không phải lúc nào bài đăng của người dùng cũng có thể xác định rõ đó có thực sự thông báo về một thảm họa hay không.

Ví dụ, ta có bài tweet sau:



Trong bài đăng này tác giả có dùng từ “ABLAZE”, nhưng không phải nói về thảm họa. Điều này rõ ràng đối với con người ngay lập tức, đặc biệt là với trợ giúp hình ảnh. Nhưng đối với máy tính thì sẽ không rõ ràng.

Đồ án được xây dựng trên một cuộc thi của Kaggle, cuộc thi cụ thể này rất phù hợp cho các nhà khoa học dữ liệu muốn bắt đầu với Xử lý Ngôn ngữ Tự nhiên (NLP). Tập dữ liệu của cuộc thi không quá lớn và ngay cả khi không có nhiều sức mạnh tính toán cá nhân, chúng ta vẫn có thể làm tất cả công việc trong môi trường Jupyter Notebooks và không cần thiết lập gọi là Kaggle Notebooks.

Trong cuộc thi này, chúng ta cần xây dựng một mô hình học máy dự đoán Tweet nào nói về thảm họa thực sự và Tweet nào không. Chúng ta sẽ có quyền truy cập vào tập dữ liệu gồm 10.000 tweet đã được phân loại thủ công.

B – KHÁM PHÁ DỮ LIỆU VÀ TIỀN XỬ LÝ

I. Khám phá dữ liệu:

1. Sơ lược về dữ liệu:

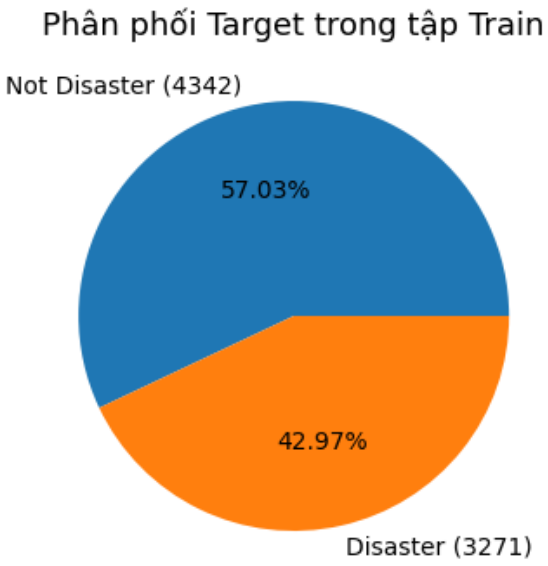
Dữ liệu gồm 2 file là train.csv và test.csv:

- Đối với file train.csv gồm: 7613 dòng và 5 cột.
- Đối với file test.csv gồm: 3263 dòng và 4 cột (không có cột target).

Ý nghĩa các cột:

- **id**: mã định danh cho mỗi tweet.
- **location**: địa điểm gửi tweet (có thể trống).
- **keyword**: từ khóa cụ thể của mỗi tweet (có thể trống).
- **text**: nội dung của tweet.
- **target**: chỉ có trong file train, xét xem tweet có nói về thiên tai, thảm họa thật hay không: (0) là không, (1) là có.

Phân phối target của dữ liệu:



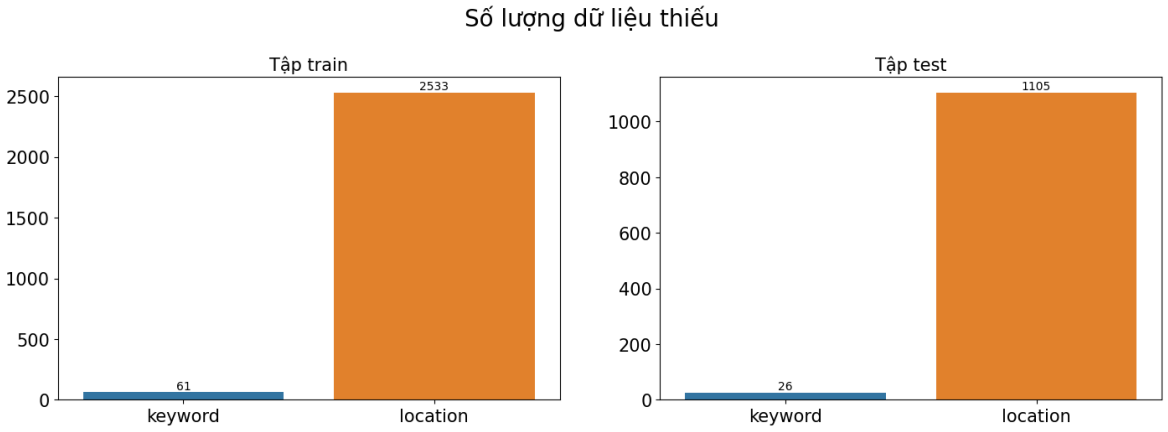
Hình 1. Phân phối target toàn bộ tập dữ liệu

Nhận xét: Dữ liệu không bị mất cân bằng

2. Xem phân bố dữ liệu:

2.1. Dữ liệu thiếu

Dữ liệu thiếu chỉ nằm trong 2 cột: keyword và location



Hình 2. Phân bố dữ liệu thiếu

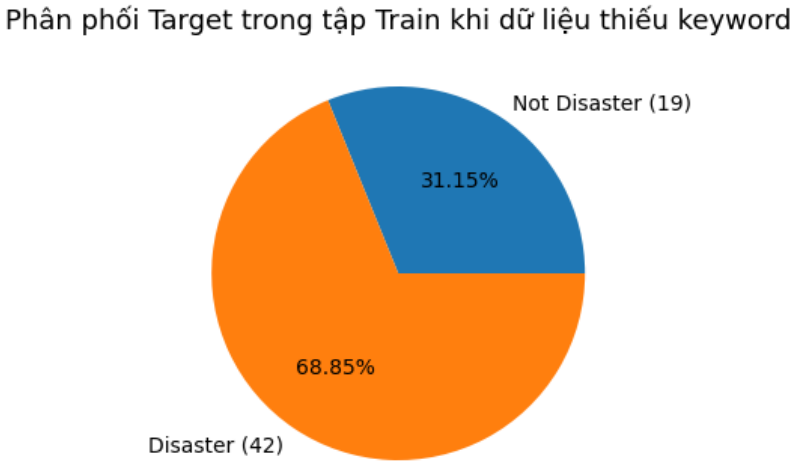
```
Tỉ lệ dữ liệu thiếu trong tập train:
- keyword: 0.801%
- location: 33.272%
Tỉ lệ dữ liệu thiếu trong tập test:
- keyword: 0.797%
- location: 33.865%
```

Hình 3. Tỉ lệ dữ liệu thiếu trong toàn bộ dữ liệu

Cả tập train và tập test đều có cùng tỷ lệ dữ liệu thiếu trong "keyword" và "location".

Để xem xét, dữ liệu thiếu có ý nghĩa khi cho ra kết quả cuối cùng hay không, nhóm sẽ xét phân bố của target với các dữ liệu thiếu.

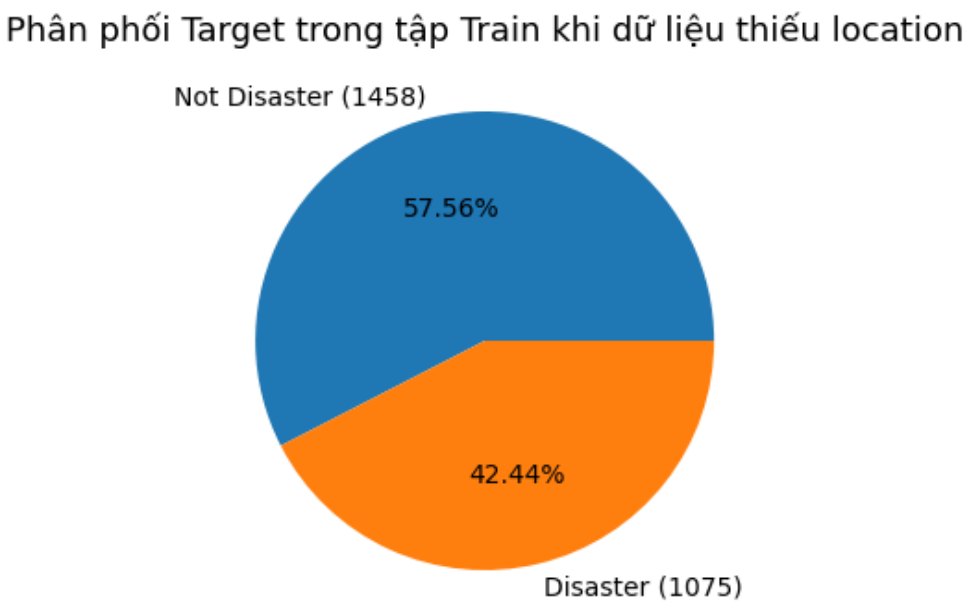
Xét target đối với dữ liệu thiếu keyword:



Hình 4. Phân phối target khi thiếu keyword

Khi thiếu keyword dữ liệu có xu hướng là "Disaster", tuy nhiên tỉ lệ chưa đủ lớn để đưa ra khẳng định thiếu keyword thì là "Disaster".

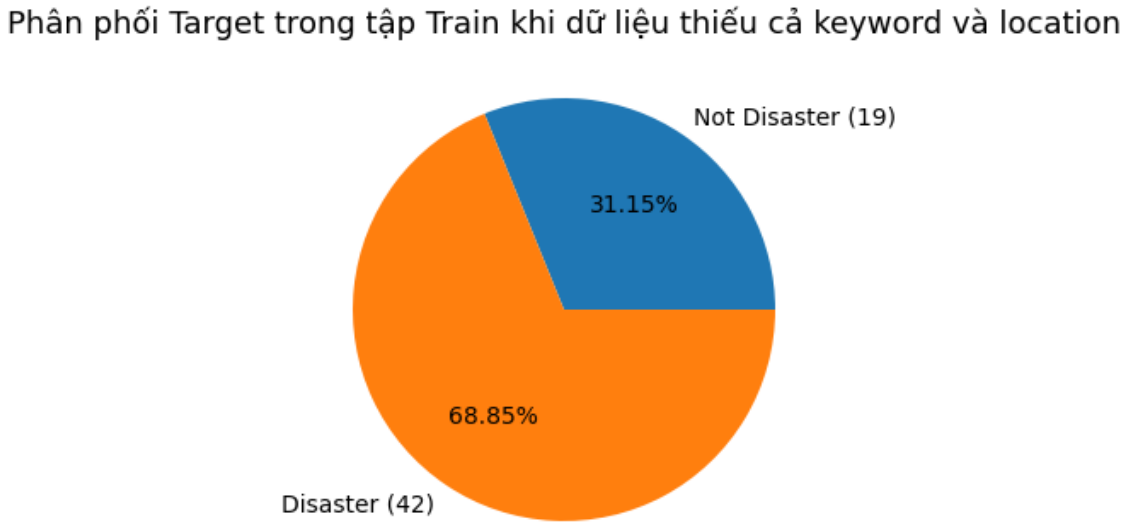
Xét target đối với dữ liệu thiếu location:



Hình 5. Phân phối target khi thiếu location

Khi thiếu location dữ liệu có xu hướng là "Not Disaster", tuy nhiên tỉ lệ chưa đủ lớn để đưa ra khẳng định thiếu location thì là "Not Disaster".

Khi thiếu cả 2 cột:



Hình 6. Phân phối target khi thiếu cả location và keyword

Khi thiếu cả keyword và location dữ liệu có xu hướng là "Disaster", tuy nhiên tỉ lệ chưa đủ lớn để đưa ra khẳng định thiếu keyword và location thì là "Disaster".

Nhận xét:

Các phân bố target của dữ liệu thiếu location và keyword không đủ để đưa ra kết luận "Not Disaster" và "Disaster".

Ngoài ra, số lượng thiếu của keyword ít so với tổng tập dữ liệu. Vì vậy, ta nên xóa dữ liệu thiếu của keyword.

Tiếp theo, để biết nên xử lý dữ liệu thiếu của location như thế nào? Ta sẽ, xem xét phân bố của "location".

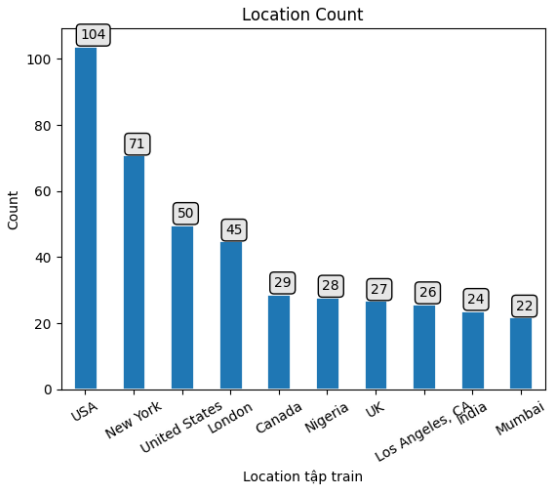
2.2. Xét phân bố location

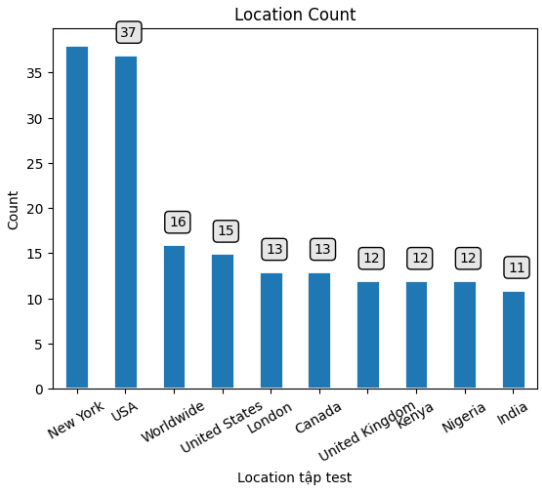
Tập train:

- Số dòng có giá trị trong **location** thuộc tập **train**: 5080.
- Số lượng giá trị khác nhau trong **location** thuộc tập **train**: 3341.

Tập test:

- Số dòng có giá trị trong **location** thuộc tập **test**: 2158.
- Số lượng giá trị khác nhau trong **location** thuộc tập **test**: 1602.





Nhận xét:

- Có thể thấy "**location**" trong **train** nhiều nhất cũng chỉ có 104 dòng trên tổng dữ liệu là 5080, và giá trị khác nhau có số lượng 3341 nên gần như mỗi dòng có 1 giá trị location khác nhau. Nên lúc tiền xử lý sẽ không lấy dữ liệu cột này.
- Tương tự, giá trị nhiều nhất của "**location**" trong **test** cũng chỉ có 38 dòng trên tổng dữ liệu là 2158 dòng, và giá trị khác nhau là 1602 chiếm hơn 1 nửa của dữ liệu có giá trị. Có thể thấy cột location cũng có thể coi như cột định danh nên tiền xử lý test cũng sẽ bỏ cột này.

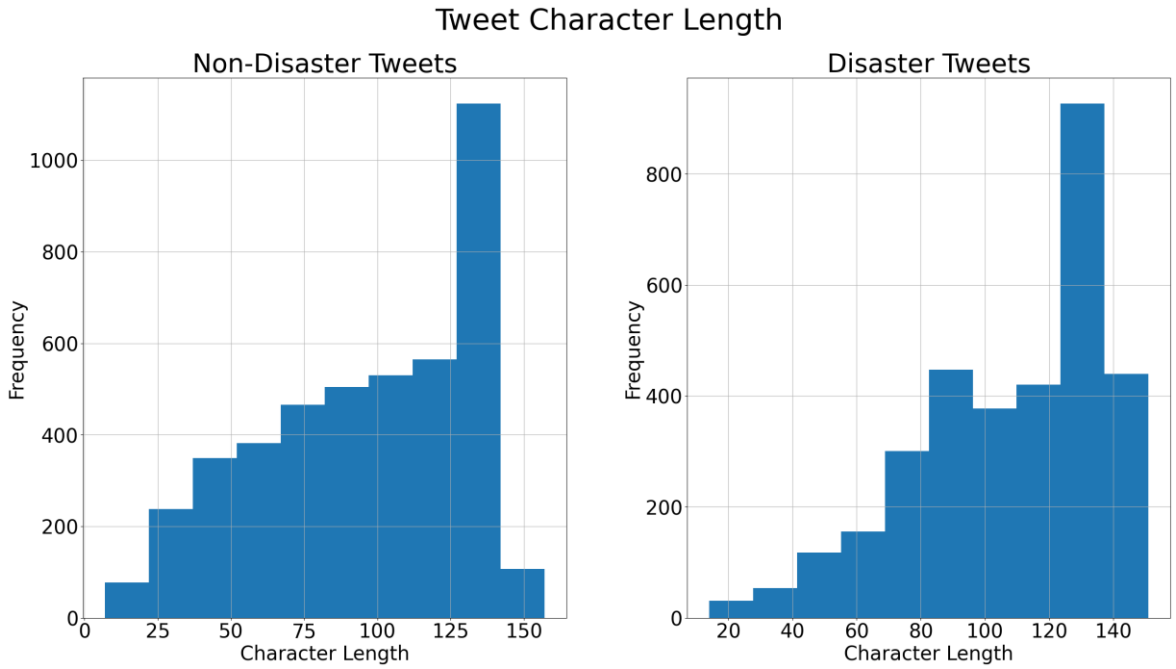
2.3 Xét phân bố keyword

Số dòng mà keyword cũng nằm trong text là: 3934 dòng.

Có thể thấy, đa phần nội dung của keyword đều nằm trong text, nên đồ án này nhóm em chỉ tập trung phần text.

2.4 Xét phân bố text

Xem xét độ dài của câu có ảnh hưởng đến target hay không?



Hình 7. Phân bố độ dài của câu xét theo target

Nhận xét:

- Có thể thấy độ dài của những câu có target là 0 và 1 cùng phân phối.
- Điều này có nghĩa là 1 câu ngắn hay dài không ảnh hưởng tới kết quả của target.

3. Xét nội dung của câu:

Với các dòng có độ giống nhau giữa các "text" > 70% thì ta sẽ xem như hai dòng này tương đồng nhau (duplicated). Sau khi xem xét nội dung của các câu:

Ta có 2 vấn đề sau với dữ liệu:

- **Vấn đề 1 – Dữ liệu trùng:** các dòng dữ liệu text có nội dung giống nhau, nhãn target cũng giống nhau
- **Vấn đề 2 – Dữ liệu mâu thuẫn:** các dòng dữ liệu text có nội dung giống nhau, nhãn target khác nhau.

II. Tiền xử lý:

1. Tổng kết các bước tiền xử lý:

- Xóa cột id.
- Xóa cột location.
- Xóa cột keyword.
- Xử lý dữ liệu trùng và mâu thuẫn.

Ngoài ra, thực hiện tiền xử lý văn bản bằng các cách phổ biến như:

- Xóa các đường link URL
  Ví dụ:
  - Trước: "@bbcmtd ablaze http://t.co/lHYXEOHY6C ab".
  - Sau : "@bbcmtd ablaze ab".
- Hàm chuyển chữ in sang chữ thường
  Ví dụ:
  - Trước: "HELLO".
  - Sau : "hello".
- Chuyển từ viết tắt sang từ đầy đủ.
  Ví dụ:
  - Trước: "You ain't fun".
  - Sau : "You are not fun".
- Xóa dấu câu
  Ví dụ:
  - Trước: "Hello, I am A".
  - Sau : "Hello I am A".
- Xóa các stopwords
  Ví dụ:
  - Trước: "You are so cute".
  - Sau : "You cute".
- Xóa các ký hiệu đặc biệt (icon, emoji, …)
  Ví dụ:
  - Trước: "Hello 😊".
  - Sau : "Hello".
- Sửa lỗi các từ viết sai
  Ví dụ:
  - Trước: "helleo".
  - Sau : "hello".
- Lemmatization (Kỹ thuật thay đổi 1 từ về từ gốc)
  Ví dụ:
  - Trước: "went".
  - Sau : "go".
- Xóa bỏ các ký tự không nằm trong ASCII
  Ví dụ:
  - Trước: "hello µ"
  - Sau : "hello".
- Xóa bỏ các khoảng trắng dư thừa trong câu
  Ví dụ:

- Trước: “Hello , I am A”
- Sau : “Hello , I am A”

2. Quá trình tiền xử lý:

Chỉ giữ lại cột text để đi qua các bước tiền xử lý văn bản trên. Tuy nhiên, trong quá trình này nhóm em nhận thấy bỏ 1 số bước xử lý văn bản đi thì hiệu quả sẽ tốt hơn.

– **Cách 1:** khi thực hiện tất cả các bước tiền xử lý cho văn bản như trên, nhưng ra kết quả khá thấp.
– **Cách 2:** khi chỉ áp dụng 1 số bước:
  - Hàm chuyển chữ in sang chữ thường
  - Xóa bỏ các ký tự không nằm trong ASCII
  - Chuyển từ viết tắt sang từ đầy đủ
  - Xóa dấu câu
  - Xóa khoảng trắng dư thừa

  Thì kết quả vượt trội hơn hẳn việc tiền xử lý như ở cách 1.

Tiếp theo, nhóm sẽ giải quyết vấn đề nêu ra khi **Khám phá dữ liệu** là:

– **Vấn đề 1 – Dữ liệu trùng:** các dòng dữ liệu text có nội dung giống nhau, nhãn target cũng giống nhau
– **Vấn đề 2 – Dữ liệu mâu thuẫn:** các dòng dữ liệu text có nội dung giống nhau, nhãn target khác nhau.

Cách giải quyết:

– **Vấn đề 1 – Dữ liệu trùng**: Xóa tất cả, giữ lại một mẫu dữ liệu. Minh họa:

id	keyword	location	text	target
59	ablaze	Live On Webcam	Check these out: http://t.co/rOI2NSmEJJ http://t.co/3Tj8ZjiN21 http://t.co/YDUiXEfIpE http://t.co/LxTjc87KLS #nsfw	0
68	ablaze	Live On Webcam	Check these out: http://t.co/rOI2NSmEJJ http://t.co/3Tj8ZjiN21 http://t.co/YDUiXEfIpE http://t.co/LxTjc87KLS #nsfw	0

Hình 8. Hai mẫu dữ liệu trùng nhau

Sau khi xử lý, sẽ xóa dòng 68, giữ lại dòng 59.

id	keyword	location	text	target
59	ablaze	Live On Webcam	Check these out: http://t.co/rOI2NSmEJJ http://t.co/3Tj8ZjiN21 http://t.co/YDUiXEfIpE http://t.co/LxTjc87KLS #nsfw	0

Hình 9. Kết quả sau khi xử lí

– **Vấn đề 2 – Dữ liệu mâu thuẫn**: Điều tra, và quyết định nhãn cho các dòng dữ liệu bị mâu thuẫn này, sau đó xóa toàn bộ và giữ lại một mẫu dữ liệu.
  - <u>Trường hợp dữ liệu mâu thuẫn số lẻ:</u>
    Chọn target của text theo target của có số lượng nhiều nhất.

446	armageddon	California, United States	#PBBan (Temporary:300) avYsss @'aRmageddon | DO NOT KILL | FLAGS ONLY | Fast XP' for Reason	0
447	armageddon	California, United States	#PBBan (Temporary:300) Russaky89 @'aRmageddon | DO NOT KILL | FLAGS ONLY | Fast XP' for Reason	0
462	armageddon	California, United States	#PBBan (Temporary:300) hyider_ghost2 @'aRmageddon | DO NOT KILL | FLAGS ONLY | Fast XP' for Reason	1

Hình 10. Ba dữ liệu bị mâu thuẫn với nhau

446	armageddon	California, United States	#PBBan (Temporary:300) avYsss @'aRmageddon | DO NOT KILL | FLAGS ONLY | Fast XP' for Reason	0

Hình 11. Dữ liệu sau khi xử lí

- Trường hợp dữ liệu mâu thuẫn số chẵn:

  Vì lượng dữ liệu mâu thuẫn ít nên sẽ xét từng trường hợp. Nếu câu có nghĩa thì sẽ xem xét nghĩa để quyết định target. Nếu không tìm ra nghĩa thì nhóm em sẽ bỏ luôn cả 2, vì tỉ lệ rất ít.

353	annihilation	Subconscious LA	World Annihilation vs Self Transformation http://t.co/pyehwodWun Aliens Attack to Exterminate Humans http://t.co/pB2N77nSKz	0
390	annihilation	Subconscious LA	World Annihilation vs Self Transformation http://t.co/pyehwodWun Aliens Attack to Exterminate Humans http://t.co/8jxqL8Cv8Z	1

Hình 12. Hai dữ liệu bị mâu thuẫn

Nhóm sẽ bỏ dòng có giá trị nhãn 1 vì có "Aliens".

id	keyword	location	text	target
353	annihilation	Subconscious LA	World Annihilation vs Self Transformation http://t.co/pyehwodWun Aliens Attack to Exterminate Humans http://t.co/pB2N77nSKz	0

Hình 13. Dữ liệu sau khi xử lí

C – TẠO ĐẶC TRƯNG VÀ XÂY DỰNG MÔ HÌNH

I. Lý thuyết BERT

1. Giới thiệu về BERT

BERT được viết tắt của Bidirectional Encoder Representations from Transformers, một kiến trúc mới cho lớp bài toán Language Representation đươc Google công bố. Không giống như các mô hình trước đó, BERT được thiết kế để đào tạo ra các vector đại diện cho ngôn ngữ văn bản thông qua ngữ cảnh 2 chiều (trái và phải) của chúng. Kết quả là, vector đại diện được sinh ra từ mô hình BERT được tính chỉnh với các lớp đầu ra bổ sung đã tạo ra nhiều kiến trúc cải tiến đáng kể cho các nhiệm vụ xử lý ngôn ngữ tự nhiên như Question Answering, Language Inference,...mà không cần thay đổi quá nhiều từ các kiến trúc cũ.

Lớp bài toán Representation cho mô hình ngôn ngữ đã cho thấy hiệu quả trong việc cải thiện nhiều nhiệm vụ trong lĩnh vực xử lí ngôn ngữ tự nhiên. Những nhiệm vụ này có thể là những nhiệm vụ cấp câu như Natural language inference, Paraphrasing nhằm dự đoán mối quan hệ giữa các câu bằng cách phân tích tổng thể chúng và cũng có thể là những nhiệm vụ cấp từ như nhận dạng thực thể có tên(NER), Question Answering với yêu cầu trả ra kết quả chính xác cho câu hỏi ở dạng từ,...

Có 2 chiến lược để sử dụng các biểu diễn ngôn ngữ được huấn luyện trước này cho các nhiệm vụ về sau, gồm feature-based và fine-tuning

Tuy nhiên, các kỹ thuật hiện tại bị hạn chế rất nhiều trong việc thể hiện khả năng của các mô hình vector đại diện, đặc biệt là hướng tiếp cận fine-tuning. Hạn chế chính ở đây là do các mô hình ngôn ngữ được xây dựng dựa trên ngữ cảnh 1 chiều gây nên sự hạn chế trong việc lựa chọn mô hình kiến trúc được sử dụng trong quá trình sử dụng pre-training. Ví dụ như trong OpenAI GPT, các tác giả sử dụng kiến trúc left-to-right, nghĩa là các tokens chỉ phụ thuộc vào các token ở trước đó.

2. Chi tiết về BERT

2.1. Kiến trúc của BERT

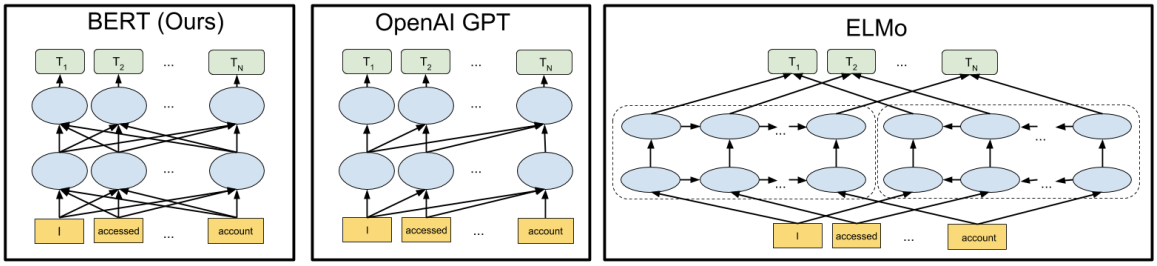
Kiến trúc của mô hình BERT là một kiến trúc đa tầng gồm nhiều lớp Bidirectional Transformer encoder dựa trên bản mô tả đầu tiên của Vaswani et al. (2017).

Trong bài báo này, chúng ta sẽ gọi L là số lớp Transformer (blocks) được sử dụng với kích thước của các lớp ẩn là H và số heads ở lớp attention là A. Trong mọi trường hợp, kích thước của bộ lọc(filter size) luôn được đặt bằng 4H. Điều này có nghĩa là khi H = 768 thì filter size = 3072 và hoặc khi H = 1024 thì filter size = 4096. Báo cáo chủ yếu lấy kết quả trên 2 kích thước mô hình:

- $\text{BERT}_{\text{BASE}}$: L = 12, H = 768, A = 12, Total parameters = 110M.
- $\text{BERT}_{\text{LARGE}}$: L = 24, H = 1024, A = 16, Total parameters = 340M.

Có một chú thích nhỏ rằng, một Transformer 2 chiều thường được gọi là **Transformer encoder** trong khi các phiên bản Transformer chỉ sử dụng ngữ cảnh bên trái thường được

gọi là **Transformer decoder** vì nó có thể được sử dụng để tạo ra văn bản. Sự so sánh giữa BERT, OpenAI GPT và ELMo được hiện thị 1 cách trực quan dưới đây:

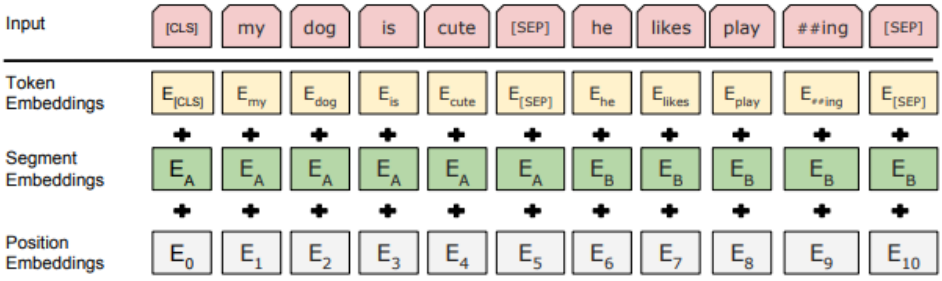


2.2. Input Representation

Ở đây, đầu vào của chúng ta có thể là biểu diễn của một câu văn bản đơn hoặc một cặp câu văn bản(ví dụ: [Câu hỏi, câu trả lời]) được đặt thành 1 chuỗi tạo bởi các từ.

Khi có một chuỗi đầu vào cụ thể, biểu diễn đầu vào của chúng ta được xây dựng bằng cách tính tổng các token đó với vector phân đoạn và vị trí tương ứng của các từ trong chuỗi.

Cho dễ hình dung, biểu diễn đầu vào được trực quan hóa trong hình dưới đây:



Một số điểm cần chú ý:

- Chúng ta sử dụng WordPiece embeddings (Wu et al., 2016) với một từ điển 30.000 từ và sử dụng ## làm dấu phân tách. Ví dụ: từ playing được tách thành play##ing.
- Chúng ta sử dụng positional embeddings với độ dài câu tối đa là 512 tokens.
- Token đầu tiên cho mỗi chuỗi được mặc định là một token đặc biệt có giá trị là [CLS]. Đầu ra của Transformer (hidden state cuối cùng) tương ứng với token này sẽ được sử dụng để đại diện cho cả câu trong các nhiệm vụ phân loại. Nếu không trong các nhiệm vụ phân loại, vector này được bỏ qua.
- Trong trường hợp các cặp câu được gộp lại với nhau thành một chuỗi duy nhất, chúng ta phân biệt các câu theo 2 cách.
  - Đầu tiên, chúng ta tách chúng bỏ một token đặc biệt [SEP].
  - Thứ hai, chúng ta thêm một segment embedding cho câu A và một segment embedding khác cho câu B như hình vẽ.
- Khi chỉ có 1 câu đơn duy nhất, segment embedding của chúng ta chỉ có cho câu A.

2.3. Pretrain Tasks

Chúng ta đào tạo BERT bằng cách sử dụng 2 nhiệm vụ dự đoán không giám sát được gọi là **Masked LM** và **Next Sentence Prediction**. Cả 2 sẽ được trình bày ngay trong phần nội dung dưới đây.

2.3.1. Masked LM

Trực quan mà thấy, một mô hình học sâu được học dựa trên ngữ cảnh 2 chiều là tự nhiên và mạnh mẽ hơn nhiều so với một mô hình chỉ dùng ngữ cảnh từ trái qua phải (hoặc ngược lại).

Tuy nhiên, thật không may, các mô hình ngôn ngữ trước đây chỉ có thể đào tạo từ trái qua phải hoặc từ phải qua trái. Lý do được lý giải là vì khi sử dụng ngữ cảnh 2 chiều sẽ gây ra một nghịch lý là một từ có thể gián tiếp tự nhìn thấy nó trong một ngữ cảnh nhiều lớp.

Để đào tạo một mô hình tìm ra đại diện dựa vào ngữ cảnh 2 chiều, chúng ta sử dụng một cách tiếp cận đơn giản để che giấu đi một số token đầu vào một cách ngẫu nhiên và sau đó chúng ta chỉ dự đoán các token được giấu đi đó và gọi nhiệm vụ này như là một "masked LM" (MLM). Trong trường hợp này, các hidden vectors ở lớp cuối cùng tương ứng với các tokens được ẩn đi được đưa vào 1 lớp softmax trên toàn bộ tự vựng để dự đoán. Các nhà nghiên cứu của Google đã thử nghiệm mask 15% tất cả các token lấy từ từ điển của WordPiece trong câu một cách ngẫu nhiên là chỉ dự đoán các từ được mask.

Mặc dù điều này cho phép chúng ta có được một mô hình đào tạo 2 chiều, nhưng có 2 nhược điểm tồn tại. Đầu tiên là chúng ta đang tạo ra một sự không phù hợp giữa pre-train và fine-tuning vì các token được [MASK] không bao giờ được nhìn thấy trong quá trình tinh chỉnh mô hình. Để giảm thiểu điều này, chúng ta sẽ không phải lúc nào cũng thay thế các từ được giấu đi bằng token [MASK]. Thay vào đó, trình tạo dữ liệu đào tạo chọn 15% tokens một cách ngẫu nhiên và thực hiện các bước như sau:

Ví dụ với câu: "*con_chó của tôi đẹp quá*" Từ được chọn để mask là từ "*đẹp*".

- Thay thế 80% từ được chọn trong dữ liệu huấn luyện thành token [MASK] → "*con_chó của tôi [MASK] quá*"
- 10% các từ được chọn sẽ được thay thế bởi 1 từ ngẫu nhiên. → "*con_chó của tôi máy_tính quá*"
- 10% còn lại được giữ không thay đổi → "*con_chó của tôi đẹp quá*"

Transformer encoder không hề biết được từ nào sẽ được yêu cầu dự đoán hoặc từ nào đã được thay thế bằng một từ ngẫu nhiên, do đó, nó buộc phải giữ một biểu diễn theo ngữ cảnh của mỗi token đầu vào. Ngoài ra, do thay thế 1.5% tất cả các tokens bằng một từ ngẫu nhiên nên điều này dường như sẽ không làm ảnh hưởng tới khả năng hiểu ngôn ngữ của mô hình.

Nhược điểm thứ 2 của việc sử dụng MLM là chỉ có 15% tokens được dự đoán trong mỗi lô, điều này gợi ý cho ta 1 điều là có thể cần thêm các các bước sử dụng các pre-train model khác để mô hình hội tụ.

2.3.2. Next Sentence Prediction

Nhiều nhiệm vụ quan trọng trong xử lý ngôn ngữ tự nhiên như Question Answering yêu cầu sự hiểu biết dựa trên mối quan hệ giữa 2 câu văn bản, không trực tiếp sử dụng được các mô hình ngôn ngữ.

Để đào tạo được mô hình hiểu được mối quan hệ giữa các câu, chúng ta xây dựng một mô hình dự đoán câu tiếp theo dựa vào câu hiện tại, dữ liệu huấn luyện có thể là một corpus bất kỳ nào.

Cụ thể, khi chọn câu A và câu B cho mỗi training sample, 50% khả năng câu B là câu tiếp theo sau câu A và 50% còn lại là một câu ngẫu nhiên nào đó trong corpus.

Ví dụ:

- **Input**: [CLS] người đàn_ông làm [MASK] tại cửa_hàng [SEP] anh_ta rất [MASK] và thân_thiện [SEP]
- **Label**: isNext
- **Input**: [CLS] người đàn_ông làm [MASK] tại cửa_hàng [SEP] cô_ta đang cầm súng [SEP]
- **Label**: notNext

Chúng ta chọn những câu **notNext** một cách ngẫu nhiên và mô hình cuối cùng đạt được độ chính xác 97%-98% trong nhiệm vụ này

II. Chuẩn bị dữ liệu

1. Tổng quan

- Chia dữ liệu trên train_df theo tỉ lệ 90:10 tương ứng để fine tune trên mô hình BERT với 90% dữ liệu và đánh giá mô hình trên 10% dữ liệu còn lại.
- Sao chép 100% dữ liệu trên train_df sang train_df_shuffle, sau đó xáo trộn 100% dữ liệu đó, và tiếp tục fine tune với mô hình BERT đã được fine tuning trước đó, trước khi nộp bài đánh giá trên kaggle.
- Dùng mô hình pretrained bert-base-uncased (tiếng anh) và module BertTokenizer từ thư viện transformer để tokenize văn bản(text) thành vector tương ứng. Thực hiện trên cả train_df, test_df và train_df_shuffle.
- Dựa vào số lượng token (từ) được tách ra (tokenize) từ các câu trên train_df, để lựa chọn max_length - số chiều của vector embedding khi fine tune bằng mô hình BERT.

2. Sao chép và xáo trộn dữ liệu:

Sao chép dữ liệu train_df sang train_df_shuffle và xáo trộn nó. Phương thức shuffle () trong Python có thể được sử dụng để xáo trộn một chuỗi, như một danh sách và tổ chức lại thứ tự của các mục. Lưu ý: Phương pháp này thay đổi danh sách ban đầu, nó không trả về một danh sách mới.

Tuy nhiên, việc xáo trộn dữ liệu có thể được sử dụng để làm cho dữ liệu ngẫu nhiên hơn và giúp cho việc huấn luyện mô hình tốt hơn.

3. Tokenize văn bản và lựa chọn siêu tham số max_length

3.1. Giới thiệu sơ về Tokenize và siêm tham số max_length

Việc tokenize văn bản là quá trình chuyển đổi văn bản thành các token (những đơn vị nhỏ nhất của văn bản) để có thể xử lý bởi mô hình.

Siêu tham số max_length được sử dụng để giới hạn số lượng token tối đa trong một câu. Nếu số lượng token trong câu vượt quá giới hạn này, chúng ta có thể sử dụng một trong hai phương pháp để giảm số lượng token xuống: cắt ngắn câu hoặc thêm các token đặc biệt vào câu để đạt được số lượng token mong muốn.

Ví dụ, nếu bạn muốn tokenize một câu và giới hạn số lượng token tối đa là 10, bạn có thể sử dụng phương pháp cắt ngắn câu để giảm số lượng token từ 15 xuống còn 10.

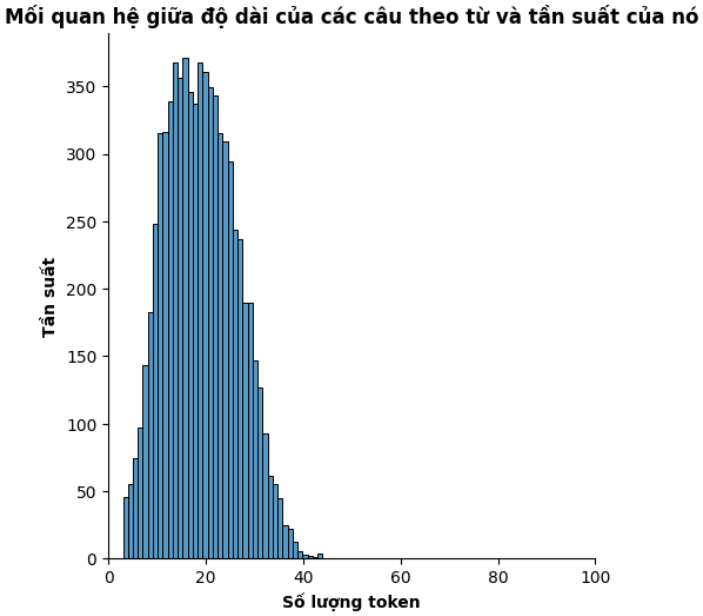
3.2. Mô hình pretrained bert-base-uncased

Trong bài này, nhóm chúng em sẽ dùng BertTokenizer để xử lý tập train và test. BertTokenizer là một trong những tokenizer được sử dụng phổ biến trong xử lý ngôn ngữ tự nhiên và học sâu. Nó được sử dụng để chuyển đổi văn bản thành các vector số để có thể xử lý bởi mô hình. BertTokenizer sử dụng phương pháp WordPiece để tách các từ thành các subword units và sau đó chuyển đổi chúng thành các vector số.

Sử dụng mô hình pretrained bert-base-uncased của BertTokenizer từ thư viện của Hugging Face Transformers, nghĩa là dùng mô hình $\text{BERT}_{\text{BASE}}$ với các từ không dấu để tokenize văn bản thành vector.

3.3. Siêu tham số max_length

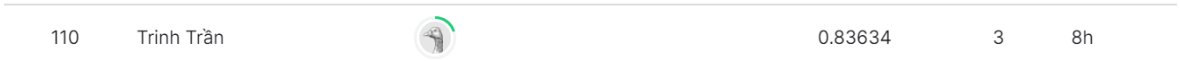
Ở đây max_length = 512 sẽ giới hạn là độ dài của mỗi câu. Nếu câu nào có độ dài lớn hơn 512, thì sẽ cắt bỏ (truncation = True). Tuy nhiên, toàn bộ các câu trong bộ corpus không tồn tại câu nào có độ dài lớn như vậy. Ta thử xem phân phối của chúng trên tập train:



Dựa vào đồ thị, ta sẽ chọn siêu tham số max_length = 45 hoặc 50 hoặc 64 để train mô hình.

D – KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN

I. Kết quả trên Kaggle:



Với số lượng đội tham dự là 1048 đội. Tuy nhiên, có 34 đội nộp kết quả ảo, nên khi trừ ra chỉ còn 1014 đội. Thứ hạng của nhóm khi trừ 34 đội ra là 76, nằm trong top 10% của cuộc thi.

II. Hướng phát triển:

Hướng phát triển đồ án này là có thể cải thiện ở bước tiền xử lý, xây dựng mô hình.

Bước tiền xử lý có thể thay thế những cách tiền xử lí để đưa ra bộ dữ liệu tốt hơn.

Tại bước xây dựng mô hình, có thể tìm siêu tham số tối ưu hơn hoặc có thể thay thế mô hình BERT bằng mô hình khác hiệu quả hơn như RoBERTa, ALBERT, …