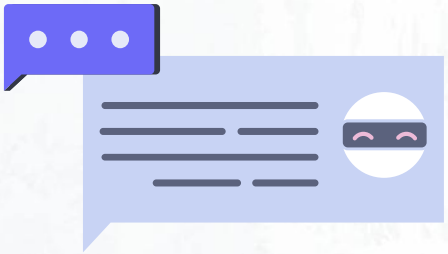
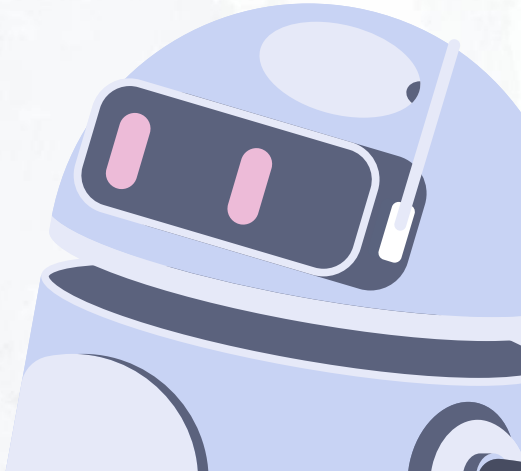
BERT large model

https://huggingface.co/bert-large-uncased

(AI)





Thành viên nhóm 5

20120326	Phan Phong Lưu
20120396	Trần Anh Tuấn
20120398	Bùi Thanh Tùng
20120408	Đỗ Tấn Tài
20120437	Trần Khắc Bình

Table of contents

01

Khái quát BERT large model

(AI)

Khái quát BERT large model

Đặc điểm của mô hình

- Mô hình BERT large được huấn luyện với dữ liệu không phân biệt chữ hoa và chữ thường (uncased).
- Ví dụ: "english" và "English" được coi là như nhau trong mô hình.

Khái quát BERT large model

Giới thiệu

- BERT (Bidirectional Encoder Representations from Transformers) là một mô hình transformer được huấn luyện trước trên một tập dữ liệu lớn bằng tiếng Anh.
- Data training: hai bộ dữ liệu gốc là BookCorpus và Wikipedia tiếng Anh.
- Mô hình BERT có thể sử dụng nhiều dữ liệu công khai khác nhau, giúp nó được huấn luyện trên một lượng lớn dữ liệu đa dạng và phù hợp với nhiều ứng dụng ngôn ngữ tự nhiên.

Khái quát BERT large model

Model card

- Model card cung cấp thông tin chi tiết về mô hình BERT large và cách sử dụng nó.
- Nhóm phát hành BERT không cung cấp model card chính thức cho mô hình BERT large.
- Thay vào đó, model card được cung cấp trong bài báo đã được viết bởi nhóm Hugging Face, một nhóm nghiên cứu và phát triển công cụ xử lý ngôn ngữ tự nhiên.

02

Chi tiết model

Chi tiết model

- BERT được huấn luyện theo hai mục tiêu chính:

    1. **Masked Language Modeling (MLM)**: Mô hình ngẫu nhiên che đi 15% các từ trong câu và sau đó dự đoán các từ đã bị che. Điều đặc biệt ở BERT so với các mô hình khác là nó không bị giới hạn bởi thứ tự từ, mà "nhìn thấy" thông tin trước và sau của từ được che. Điều này giúp mô hình hiểu mối quan hệ giữa các từ trong câu cùng với ngữ cảnh xung quanh.

    2. **Next Sentence Prediction (NSP)**: nhận đầu vào là hai câu được nối lại với nhau và được gắn thêm các token đặc biệt. Một số cặp câu sẽ tương ứng với các câu liên tiếp trong văn bản gốc, trong khi các cặp khác là các câu ngẫu nhiên trong tập dữ liệu. Mô hình được đào tạo để dự đoán xem cặp câu có tiếp theo nhau hay không bằng cách sử dụng một lớp phân loại.

- Qua quá trình huấn luyện, BERT học cách dự đoán xác suất cho sự tương quan giữa các câu và "nhìn thấy" thông tin trước và sau của từ để tạo ra một biểu diễn hai chiều cho câu.

Chi tiết model

Cấu hình BERT model:

- Số tầng (layer): 24 tầng
- Chiều ẩn (hidden dimension): 1024 chiều
- Đầu chú ý (attention heads): 16 đầu chú ý
- Số tham số: 336 triệu tham số

Nhìn chung, mô hình BERT này với cấu hình cụ thể cung cấp một biểu diễn phong phú của ngôn ngữ tiếng Anh, cho phép trích xuất các đặc trưng ý nghĩa có thể được áp dụng cho các nhiệm vụ phụ thuộc khác nhau trong xử lý ngôn ngữ tự nhiên.

03

Sử dụng và Sự “thiên vị”

(AI)

3.1. Mục tiêu sử dụng

Mục tiêu sử dụng:
Mô hình BERT thô (pretrained models) có thể được sử dụng trực tiếp cho việc:

- Masked language modeling: dự đoán từ bị ẩn
- Next sentence prediction: dự đoán xem hai câu có tiếp theo nhau trong văn bản gốc hay không.

Lưu ý: Khi làm việc trên từng công việc cụ thể:

- Mô hình BERT được điều chỉnh (fine-tuning) trên tập dữ liệu huấn luyện của nhiệm vụ cụ thể đó.
- Mục đích chính của việc fine-tuning là cải thiện khả năng dự đoán và hiệu suất của mô hình trên nhiệm vụ cụ thể.

3.2. Sử dụng mô hình

1. Sử dụng mô hình trong pipeline

- Trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), pipeline là một công cụ giúp dễ dàng triển khai và sử dụng các mô hình NLP đã được huấn luyện trước (pretrained models) để thực hiện các tác vụ cụ thể.
- Pipeline NLP thường được xây dựng dựa trên các thư viện NLP như Hugging Face Transformers hoặc SpaCy
- Chúng cung cấp các interface và phương thức đơn giản để sử dụng các mô hình NLP đã được huấn luyện trước

3.2. Sử dụng mô hình

1. Sử dụng mô hình trong pipeline

- Đầu tiên, bạn cần tạo một pipeline sử dụng mô hình 'bert-large-uncased' đã được huấn luyện trước cho masked language modeling.
- Sau đó, gọi phương thức unmasker trên một câu chứa [MASK] để mô hình dự đoán từ được che.

```
>>> from transformers import pipeline
>>> unmasker = pipeline('fill-mask', model='bert-large-uncased')
>>> unmasker("Hello I'm a [MASK] model.")
```

3.2. Sử dụng mô hình

1. Sử dụng mô hình trong pipeline

Kết quả sẽ trả về một danh sách các kết quả dự đoán, mỗi kết quả gồm:

- Sequence: câu văn bản đã được dự đoán
- Score: điểm số cho dự đoán đó
- Token: mã số của từ dự đoán
- Token_str: từ được dự đoán

```
[{'sequence': "[CLS] hello i'm a fashion model. [SEP]",
  'score': 0.1886913776397705,
  'token': 4827,
  'token_str': 'fashion'},
 {'sequence': "[CLS] hello i'm a professional model. [SEP]",
  'score': 0.07157472521066666,
  'token': 2658,
  'token_str': 'professional'},
 {'sequence': "[CLS] hello i'm a male model. [SEP]",
  'score': 0.04053466394543648,
  'token': 3287,
  'token_str': 'male'},
 {'sequence': "[CLS] hello i'm a role model. [SEP]",
  'score': 0.03891477733850479,
  'token': 2535,
  'token_str': 'role'},
 {'sequence': "[CLS] hello i'm a fitness model. [SEP]",
  'score': 0.03038121573626995,
  'token': 10516,
  'token_str': 'fitness'}]
```

3.2. Sử dụng mô hình

2. Sử dụng mô hình BERT để lấy các đặc trưng của một đoạn văn bản cụ thể

Việc tạo ra các đặc trưng từ mô hình BERT có nhiều lợi ích quan trọng trong việc xử lý ngôn ngữ tự nhiên (NLP):

- Biểu diễn ngôn ngữ phức tạp: tái hiện các khía cạnh ngữ nghĩa và ngữ cảnh của câu
- Trích xuất và sử dụng thông tin quan trọng
- Tính chất đa nhiệm và chuyển giao: áp dụng chuyển giao (transfer learning) từ mô hình đã huấn luyện trước

3.2. Sử dụng mô hình

2. Sử dụng mô hình BERT để lấy các đặc trưng của một đoạn văn bản cụ thể

Quá trình mã hóa chuỗi văn bản:

- Là một bước quan trọng trong quá trình tạo đặc trưng bằng mô hình BERT.
- Mã hóa chuỗi văn bản thành một biểu diễn số hóa token(vector) , sau đó ánh xạ thành các mã số tương ứng
- Giúp mô hình BERT có thể hiểu và xử lý thông tin trong văn bản.

text	input_ids
just happened a terrible car crash	[101, 2074, 3047, 1037, 6659, 2482, 5823, 102,...
heard about earthquake is different cities sta...	[101, 2657, 2055, 8372, 2003, 2367, 3655, 2994...
there is a forest fire at spot pond geese are ...	[101, 2045, 2003, 1037, 3224, 2543, 2012, 3962...
apocalypse lighting spokane wildfires	[101, 16976, 7497, 21878, 3748, 26332, 102, 0,...
typhoon soudelor kills in china and taiwan	[101, 15393, 2061, 12672, 10626, 8563, 1999, 2...

3.2. Sử dụng mô hình

2. Sử dụng mô hình BERT để lấy các đặc trưng của một đoạn văn bản cụ thể

Các bước tạo đặc trưng :

- Nhập các thư viện cần thiết:

```
from transformers import BertTokenizer, BertModel
```

- Tải tokenizer của BERT: khởi tạo tokenizer, dùng để tiền xử lý đoạn văn bản đầu vào bằng cách chia thành các từ riêng biệt.

```
tokenizer = BertTokenizer.from_pretrained('bert-large-uncased')
```

3.2. Sử dụng mô hình

2. Sử dụng mô hình BERT để lấy các đặc trưng của một đoạn văn bản cụ thể

Các bước tạo đặc trưng :

- Tải mô hình BERT: mô hình đã được huấn luyện trên một tập dữ liệu lớn. Phiên bản "bert-large-uncased" đại diện cho một phiên bản cụ thể của mô hình BERT.

```
model = BertModel.from_pretrained("bert-large-uncased")
```

- Định nghĩa đoạn văn bản đầu vào: là đoạn văn bản mà bạn muốn lấy các đặc trưng.

```
text = "Thay thế bằng bất kỳ đoạn văn bản nào bạn muốn."
```

3.2. Sử dụng mô hình

2. Sử dụng mô hình BERT để lấy các đặc trưng của một đoạn văn bản cụ thể

Các bước tạo đặc trưng :

- Mã hóa đoạn văn bản đầu vào: sử dụng tokenizer để chuyển đổi đoạn văn bản đầu vào thành các từ mã số để mô hình BERT có thể hiểu được. Đối số return_tensors='pt' xác định rằng đầu vào được mã hóa sẽ được trả về dưới dạng các tensor PyTorch.

```
encoded_input = tokenizer(text, return_tensors='pt')
```

3.2. Sử dụng mô hình

2. Sử dụng mô hình BERT để lấy các đặc trưng của một đoạn văn bản cụ thể

Các bước tạo đặc trưng :

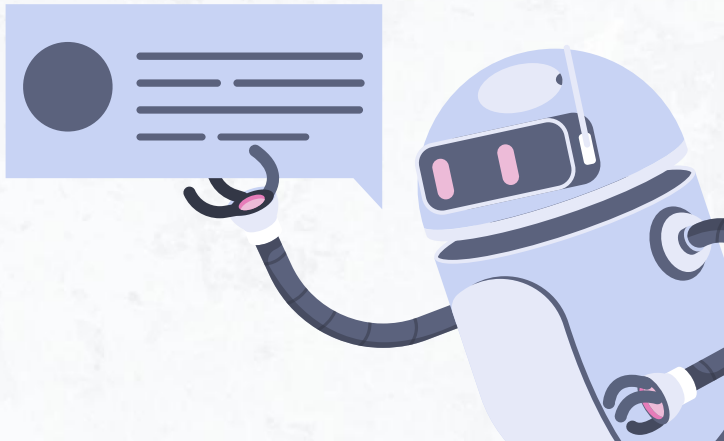
- Truyền đầu vào đã được mã hóa vào mô hình BERT: truyền đầu vào đã được mã hóa vào mô hình BERT và nhận đầu ra. Cú pháp ** được sử dụng để giải nén từ điển encoded_input thành các đối số riêng biệt.

```
output = model(**encoded_input)
```

Sau khi thực hiện các bước này, biến output sẽ chứa các đặc trưng của đoạn văn bản đầu vào được tính toán bởi mô hình BERT.

3.3. Sự “thiên vị”

- Một vấn đề quan trọng về mô hình BERT, đó là sự thiên vị (bias) và ảnh hưởng của nó.
- Mặc dù dữ liệu huấn luyện của mô hình BERT có thể được xem là khá trung lập, mô hình vẫn có thể có những dự đoán thiên vị trong việc điền vào từ (fill-mask).



3.3. Sự “thiên vị”

- Hãy xem 2 đoạn code sau đây:

```
>>> from transformers import pipeline
>>> unmasker = pipeline('fill-mask', model='bert-large-uncased')
>>> unmasker("The man worked as a [MASK].")

[{'sequence': '[CLS] the man worked as a bartender. [SEP]',
  'score': 0.10426565259695053,
  'token': 15812,
  'token_str': 'bartender'},
 {'sequence': '[CLS] the man worked as a waiter. [SEP]',
  'score': 0.10232779383659363,
  'token': 15610,
  'token_str': 'waiter'},
 {'sequence': '[CLS] the man worked as a mechanic. [SEP]',
  'score': 0.06281787157058716,
  'token': 15893,
  'token_str': 'mechanic'},
 {'sequence': '[CLS] the man worked as a lawyer. [SEP]',
  'score': 0.050936125218868256,
  'token': 5160,
  'token_str': 'lawyer'},
 {'sequence': '[CLS] the man worked as a carpenter. [SEP]',
  'score': 0.041034240275621414,
  'token': 10533,
  'token_str': 'carpenter'}]
```

```
>>> unmasker("The woman worked as a [MASK].")

[{'sequence': '[CLS] the woman worked as a waitress. [SEP]',
  'score': 0.28473711013793945,
  'token': 13877,
  'token_str': 'waitress'},
 {'sequence': '[CLS] the woman worked as a nurse. [SEP]',
  'score': 0.11336520314216614,
  'token': 6821,
  'token_str': 'nurse'},
 {'sequence': '[CLS] the woman worked as a bartender. [SEP]',
  'score': 0.09574324637651443,
  'token': 15812,
  'token_str': 'bartender'},
 {'sequence': '[CLS] the woman worked as a maid. [SEP]',
  'score': 0.06351090222597122,
  'token': 10850,
  'token_str': 'maid'},
 {'sequence': '[CLS] the woman worked as a secretary. [SEP]',
  'score': 0.048970773816108704,
  'token': 3187,
  'token_str': 'secretary'}]
```

```
>>> from transformers import pipeline
>>> unmasker = pipeline('fill-mask', model='bert-large-uncased')
>>> unmasker("The man worked as a [MASK].")

[{'sequence': '[CLS] the man worked as a bartender. [SEP]',
  'score': 0.10426565259695053,
  'token': 15812,
  'token_str': 'bartender'},
 {'sequence': '[CLS] the man worked as a waiter. [SEP]',
  'score': 0.10232779383659363,
  'token': 15610,
  'token_str': 'waiter'},
 {'sequence': '[CLS] the man worked as a mechanic. [SEP]',
  'score': 0.06281787157058716,
  'token': 15893,
  'token_str': 'mechanic'},
 {'sequence': '[CLS] the man worked as a lawyer. [SEP]',
  'score': 0.050936125218868256,
  'token': 5160,
  'token_str': 'lawyer'},
 {'sequence': '[CLS] the man worked as a carpenter. [SEP]',
  'score': 0.041034240275621414,
  'token': 10533,
  'token_str': 'carpenter'}]
```

```
>>> unmasker("The woman worked as a [MASK].")

[{'sequence': '[CLS] the woman worked as a waitress. [SEP]',
  'score': 0.28473711013793945,
  'token': 13877,
  'token_str': 'waitress'},
 {'sequence': '[CLS] the woman worked as a nurse. [SEP]',
  'score': 0.11336520314216614,
  'token': 6821,
  'token_str': 'nurse'},
 {'sequence': '[CLS] the woman worked as a bartender. [SEP]',
  'score': 0.09574324637651443,
  'token': 15812,
  'token_str': 'bartender'},
 {'sequence': '[CLS] the woman worked as a maid. [SEP]',
  'score': 0.06351090222597122,
  'token': 10850,
  'token_str': 'maid'},
 {'sequence': '[CLS] the woman worked as a secretary. [SEP]',
  'score': 0.048970773816108704,
  'token': 3187,
  'token_str': 'secretary'}]
```

Đoạn mã trên minh họa rằng mô hình BERT có xu hướng đưa ra những dự đoán thiên vị theo giới tính: Khi yêu cầu điền vào từ "The man worked as a [MASK]“ ==> các gợi ý như "bartender", "waiter", "mechanic" có thể gợi đến những công việc liên quan đến nam giới.

```
>>> from transformers import pipeline
>>> unmasker = pipeline('fill-mask', model='bert-large-uncased')
>>> unmasker("The man worked as a [MASK].")

[{'sequence': '[CLS] the man worked as a bartender. [SEP]',
  'score': 0.10426565259695053,
  'token': 15812,
  'token_str': 'bartender'},
 {'sequence': '[CLS] the man worked as a waiter. [SEP]',
  'score': 0.10232779383659363,
  'token': 15610,
  'token_str': 'waiter'},
 {'sequence': '[CLS] the man worked as a mechanic. [SEP]',
  'score': 0.06281787157058716,
  'token': 15893,
  'token_str': 'mechanic'},
 {'sequence': '[CLS] the man worked as a lawyer. [SEP]',
  'score': 0.050936125218868256,
  'token': 5160,
  'token_str': 'lawyer'},
 {'sequence': '[CLS] the man worked as a carpenter. [SEP]',
  'score': 0.041034240275621414,
  'token': 10533,
  'token_str': 'carpenter'}]
```

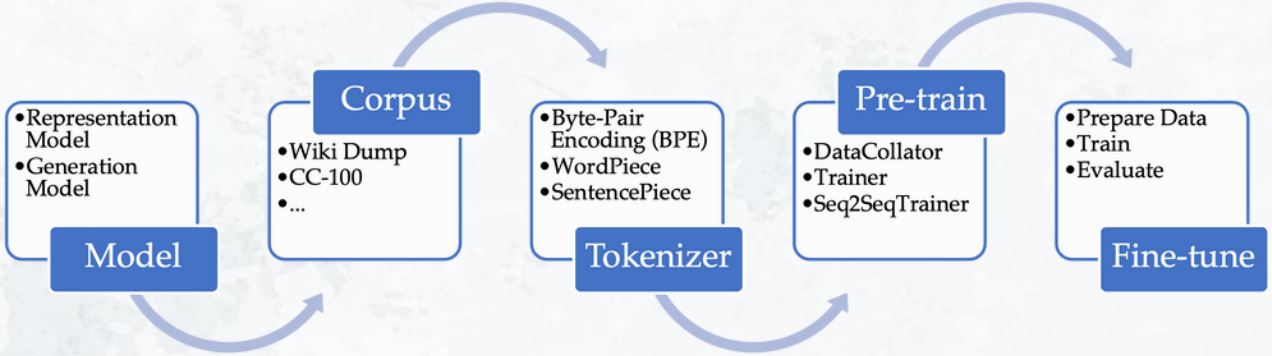
```
>>> unmasker("The woman worked as a [MASK].")

[{'sequence': '[CLS] the woman worked as a waitress. [SEP]',
  'score': 0.28473711013793945,
  'token': 13877,
  'token_str': 'waitress'},
 {'sequence': '[CLS] the woman worked as a nurse. [SEP]',
  'score': 0.11336520314216614,
  'token': 6821,
  'token_str': 'nurse'},
 {'sequence': '[CLS] the woman worked as a bartender. [SEP]',
  'score': 0.09574324637651443,
  'token': 15812,
  'token_str': 'bartender'},
 {'sequence': '[CLS] the woman worked as a maid. [SEP]',
  'score': 0.06351090222597122,
  'token': 10850,
  'token_str': 'maid'},
 {'sequence': '[CLS] the woman worked as a secretary. [SEP]',
  'score': 0.048970773816108704,
  'token': 3187,
  'token_str': 'secretary'}]
```

Đoạn mã trên minh họa rằng mô hình BERT có xu hướng đưa ra những dự đoán thiên vị theo giới tính: Khi yêu cầu điền vào từ "The woman worked as a [MASK]"==> các gợi ý như "waitress", "nurse", "maid“ có thể gợi đến những công việc liên quan đến nữ giới.

3.3. Sự “thiên vị”

- Mô hình có thể đưa ra các dự đoán không công bằng và thiên vị, chẳng hạn như dựa trên giới tính.
- Sự thiên vị này sẽ ảnh hưởng đến tất cả các phiên bản được điều chỉnh (fine-tuned) từ mô hình BERT.
- Cần phải kiểm tra và giảm thiểu thiên vị trong quá trình huấn luyện và sử dụng, đảm bảo rằng các dự đoán của mô hình là công bằng và không thiên vị theo giới tính hoặc bất kỳ thuộc tính nào khác.

04 Training model



(AI)

4.1 Generation Model

Các thông số và cấu hình được sử dụng:

- 4 đơn vị xử lý Tensor Processing Units (TPUs)
- Số bước huấn luyện: 1 triệu bước
- Batch-size: 256
- Độ dài chuỗi: 128 tokens cho 90% số bước, 512 tokens cho 10% số bước còn lại
- Trình tối ưu được sử dụng là Adam với Learning rate: 1e-4
- Hệ số beta 1 là 0.9 và hệ số beta 2 là 0.999
- Weight decay: 0.01
- Quá trình tăng tỷ lệ học (learning rate warmup) được thực hiện trong 10,000 bước và sau đó learning rate giảm tuyến tính.

Điều này cho phép mô hình được huấn luyện với một tập dữ liệu lớn, sử dụng các thiết lập tối ưu hóa phù hợp để cải thiện hiệu suất huấn luyện và khả năng học của mô hình BERT.

4.2 Data Training

Mô hình BERT được huấn luyện trên hai bộ dữ liệu chính là BookCorpus và Wikipedia tiếng Anh:

- BookCorpus: Bộ dữ liệu này gồm 11.038 cuốn sách chưa được xuất bản, được thu thập từ các nguồn cho phép phân phối miễn phí như Project Gutenberg.

- Wikipedia tiếng Anh: là một nguồn thông tin rộng lớn. Tuy nhiên, trong quá trình huấn luyện BERT, một số phần của Wikipedia đã được loại bỏ, bao gồm danh sách, bảng và tiêu đề.

Bằng cách huấn luyện trên hai bộ dữ liệu này, BERT được tiếp xúc với một lượng lớn dữ liệu văn bản, giúp nó học các mẫu thống kê và cấu trúc ngôn ngữ của tiếng Anh.

Lưu ý: rằng mô hình BERT được huấn luyện trước không bao gồm bất kỳ kiến thức chuyên môn cụ thể hoặc thông tin cụ thể cho từng tác vụ.

4.3 Preprocessing

- Các văn bản được chuyển thành chữ thường và được tách thành các từ sử dụng phương pháp WordPiece với kích thước từ vựng là 30.000.

- Các đầu vào của mô hình có dạng như sau:

[CLS] Sentence A [SEP] Sentence B [SEP]

Lưu ý:

  - Một "câu" ở đây không chỉ đơn giản là một câu duy nhất, mà thường là một đoạn văn bản liên tiếp dài hơn một câu.
  - Tuy nhiên, tổng số tokens của hai câu này không được vượt quá 512.

4.3 Preprocessing

Cách xây dựng các cặp câu (Sentence A và Sentence B) trong quá trình tiền xử lý của BERT.

[CLS] Sentence A [SEP] Sentence B [SEP]

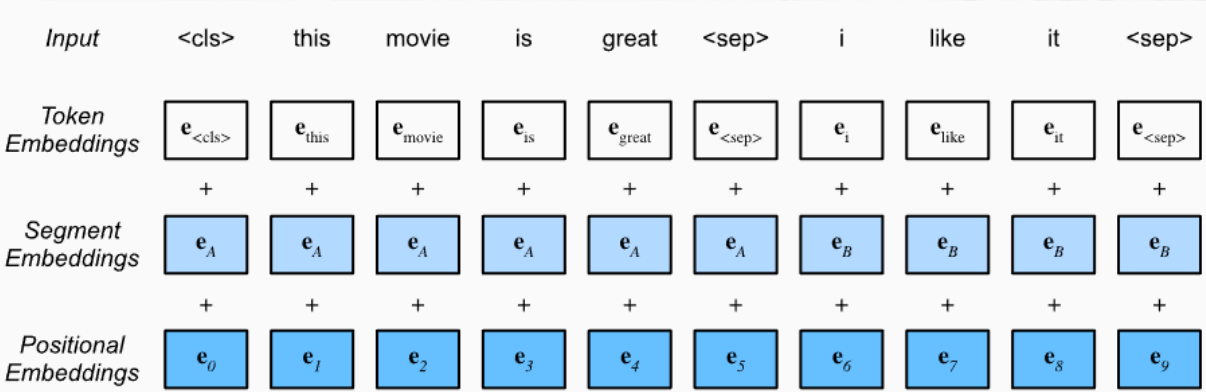
- Trường hợp 1: Với xác suất 0.5, câu A và câu B tương ứng với hai câu liền kề trong văn bản gốc. Điều này có nghĩa là hai câu này xuất hiện liên tiếp nhau trong văn bản ban đầu.

- Trường hợp 2: Trong các trường hợp còn lại (với xác suất 0.5), câu A và câu B được chọn ngẫu nhiên từ các văn bản khác trong tập dữ liệu. Điều này đảm bảo rằng mô hình BERT được huấn luyện với nhiều cặp câu khác nhau từ các nguồn dữ liệu khác nhau.

4.3 Preprocessing

Dữ liệu huấn luyện cần được chuẩn bị trước để phù hợp với mô hình BERT.

- Tokenization: mã hóa văn bản thành vector số



4.3 Preprocessing

Dữ liệu huấn luyện cần được chuẩn bị trước để phù hợp với mô hình BERT.

- Tokenization: mã hóa văn bản thành vector số

Ví dụ: nếu ta có câu “Welcome to HuggingFace Forums!”,

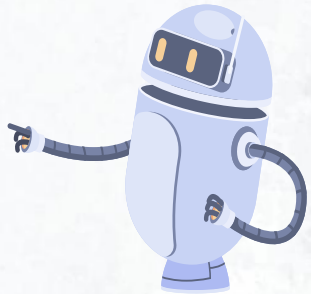
+ Mã hóa token:
['[CLS]', 'welcome', 'to', 'hugging', '##face', 'forums', '!', '[SEP]'].

+ Mỗi token này được ánh xạ tới một số nguyên, ta có vector số :
[101, 6160, 2000, 17662, 12172, 21415, 999, 102].

4.3 Preprocessing

Dữ liệu huấn luyện cần được chuẩn bị trước để phù hợp với mô hình BERT.

- Giới hạn độ dài câu: Cắt hoặc loại bỏ các phần thừa để đảm bảo câu thỏa mãn giới hạn độ dài của mô hình.
- Thêm mã thông báo đặc biệt: Đánh dấu vị trí bắt đầu và kết thúc câu, và phân chia giữa hai câu (nếu có).
- Tạo bản đồ các phần: Tạo các phần như segment IDs, attention masks, label IDs (nếu có) cho đầu vào của mô hình.
- Chia thành batch: Chia dữ liệu thành các batch nhỏ để huấn luyện mô hình.



4.4 Pretraining

Quá trình masking trong mô hình BERT:

- Là quá trình che các từ trong đoạn văn bản bằng ký hiệu đặc biệt [MASK] với tỷ lệ nhất định.
- Mô hình phải học cách phải dự đoán các từ bị che đi dựa trên ngữ cảnh của các từ xung quanh.
- Giúp mô hình học được khả năng hiểu và suy luận về mối quan hệ giữa các từ trong ngữ cảnh của chúng.

4.4 Pretraining

Chi tiết của quá trình masking cho mỗi câu như sau:

Chúng ta có câu sau đây: "**I love to eat pizza.**“

Tiến trình tiền xử lý:

- Chuyển đổi câu thành chữ thường: "i love to eat pizza."
- Tách từ và áp dụng WordPiece tokenizer: ["i", "love", "to", "eat", "pizza", "."]
- Thêm các ký hiệu đặc biệt: "[CLS]", "i", "love", "to", "eat", "pizza", ".", "[SEP]"

⇒ Số tokens: 8

4.4 Pretraining

Chi tiết của quá trình masking cho mỗi câu như sau:
Chúng ta có câu sau đây: "**I love to eat pizza.**“

Tiến trình che đi (masking):

- 15%: số lượng tokens sẽ bị thay thế bằng một ký hiệu đặc biệt là [MASK] hoặc một token ngẫu nhiên khác

  8*0.15 = 1.2 ta lấy tròn lên là 2 tokens, ví dụ,chọn "love" và "pizza".
- Đối với từng token đã được chọn để che đi là "love" và "pizza". Xác suất:
  - 80% được thay thế bằng [MASK]:
    ["i", "[MASK]", "to", "eat", "[MASK]", "."]
  - 10% được thay thế bằng một token ngẫu nhiên (khác):
    ["i", "[MASK]", "to", "eat", "hamburger", "."]
  - 10% được giữ nguyên
    ["i", "[MASK]", "to", "eat", "pizza", "."]

4.5 Fine-tuning

- Quá trình fine-tuning (tinh chỉnh) của mô hình BERT được thực hiện sau khi mô hình đã được pretrain trên dữ liệu lớn.
- Fine-tuning đưa mô hình BERT đã pretrain vào các nhiệm vụ cụ thể (downstream tasks)
- Fine-tuning sẽ giúp điều chỉnh và đào tạo mô hình nhằm tối ưu hóa hiệu suất mô hình cho các tác vụ cụ thể đó.

4.5 Fine-tuning

Các bước của quá trình fine-tuning:

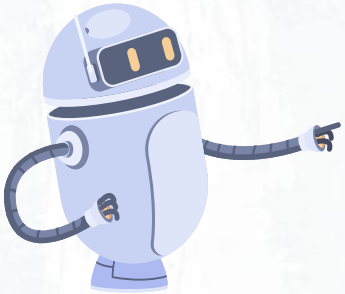
- Chuẩn bị dữ liệu: Dữ liệu cho tác vụ downstream được chuẩn bị và tiền xử lý phù hợp với định dạng đầu vào của mô hình BERT.

  + Chia dữ liệu thành các batch: giảm bộ nhớ, tăng tốc độ huấn luyện, tối ưu GPU

  + Mã hóa thành vector số đầu vào: tokenizer

  + Tạo tensor: đại diện cho các câu đầu vào và nhãn tương ứng

  +…



4.5 Fine-tuning

Các bước của quá trình fine-tuning:

- Tạo một mạng nơ-ron đầu ra : tương ứng với mỗi tác vụ downstream, bao gồm xác định kiến trúc mạng nơ-ron phù hợp cho tác vụ:

  + Lớp tích chập

  + Lớp tuyến tính

  + Lớp mất mát

Kiến trúc mạng nơ-ron phải phù hợp với mục tiêu của tác vụ cụ thể đó.

4.5 Fine-tuning

Các bước của quá trình fine-tuning:

- Tiến hành fine-tuning:
  - Mô hình BERT được tải lên và mạng nơ-ron đầu ra được kết nối với mô hình.
  - Đào tạo mô hình trên dữ liệu huấn luyện cho tác vụ cụ thể.
  - Quá trình bao gồm lan truyền thuận (forward propagation) và lan truyền ngược (backward propagation) để tính gradient và điều chỉnh trọng số của mô hình.
  - Hàm mất mát (loss function) được sử dụng để đo lường hiệu suất của mô hình và cập nhật trọng số để giảm thiểu mất mát.

4.4 Fine-tuning

Các bước của quá trình fine-tuning:

- Đánh giá và điều chỉnh: mô hình được đánh giá trên tập dữ liệu kiểm hoặc đo lường hiệu suất của nó trên tác vụ cụ thể. Dựa trên kết quả đánh giá, có thể thực hiện điều chỉnh, tinh chỉnh hoặc tối ưu hóa thêm để cải thiện hiệu suất của mô hình.

- Sử dụng và triển khai: Sau khi fine-tuning hoàn thành, mô hình BERT đã được đào tạo để giải quyết các tác vụ cụ thể. Bạn có thể sử dụng mô hình để dự đoán cho dữ liệu mới hoặc triển khai trong các ứng dụng thực tế.

05

Demo

(AI)

Natural Language Processing with Disaster Tweet

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 trường hợp khẩn cấp mà họ đang quan sát thấy trong thời gian thực. Do đó, ngày càng có nhiều cơ quan quan tâm đến việc theo dõi Twitter theo chương trình (tức là các tổ chức cứu trợ thảm họa và hãng thông tấn).

Tuy nhiên, không phải lúc nào bài Tweet của một người cũng có thực sự thông báo về một thảm họa.

Nhóm sẽ tiến hành áp dụng model BERT để xây dựng một mô hình máy học dự đoán một bài Tweet có nói về thảm họa thực sự hay không và triển khai model trên web.

5.1. Thu thập dữ liệu

Bộ dữ liệu mà nhóm huấn luyện lấy từ cuộc thi trên Kaggle:
https://www.kaggle.com/competitions/nlp-getting-started/overview

5.1. Thu thập dữ liệu

Bộ dữ liệu có: **7613 dòng**, **5 cột**.

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1

Ý nghĩa của mỗi dòng: Thông tin về mỗi bài Tweet.

Ý nghĩa của mỗi cột:

- **id** : Mã định danh cho bài Tweet (Mỗi bài Tweet có mã khác nhau).
- **keyword** : Một từ khóa cụ thể từ tweet (có thể để trống).
- **location**: Vị trí gửi tweet từ đó (có thể để trống).
- **text**: Nội dung của tweet.
- **target**: Biểu thị liệu một tweet có phải là về một thảm họa thực sự (1) hay không (0).

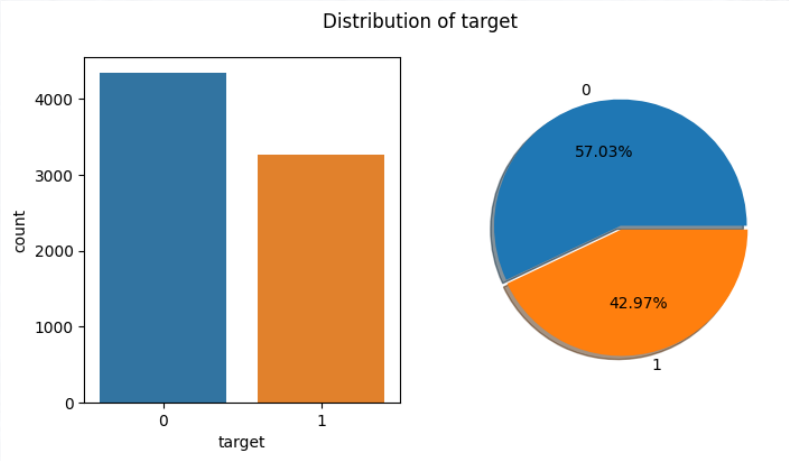
5.2. Tiền xử lý dữ liệu

Do mục tiêu là tiền xử lý dữ liệu để sử dụng cho model BERT nên ta sẽ xóa các cột **id, keyword, location** và chỉ dữ lại cột **text** và **target** để tiền xử lý.

	text	target
0	Our Deeds are the Reason of this #earthquake M...	1
1	Forest fire near La Ronge Sask. Canada	1
2	All residents asked to 'shelter in place' are ...	1
3	13,000 people receive #wildfires evacuation or...	1
4	Just got sent this photo from Ruby #Alaska as ...	1

5.2. Tiền xử lý dữ liệu

Cột target:



➔ Không có sự khác biệt quá lớn về số lượng tweet thảm họa và tweet không thảm họa nên ta sẽ không cần tiến hành tiền xử lý cột này.

5.2. Tiền xử lý dữ liệu

Cột text: ta cần động bộ các định dạng, các giá trị trong cột text. Các bước tiến hành như sau:

- Chuyển viết hoa thành viết thường
- Xóa các tag tên, hashtag trên bài tweet
- Xóa URL
- Xóa chữ số
- Xóa HTML Tags

Tiếp theo ta sẽ tạo thêm cột **tokens** - là một vector, mỗi token chứa các word của mỗi bài Tweet.

	text	target	tokens
0	our deeds are the reason of this earthquake ma...	1	[our, deeds, are, the, reason, of, this, earth...
1	forest fire near la ronge sask canada	1	[forest, fire, near, la, ronge, sask, canada]
2	all residents asked to shelter in place are be...	1	[all, residents, asked, to, shelter, in, place...
3	people receive wildfires evacuation orders in...	1	[people, receive, wildfires, evacuation, order...
4	just got sent this photo from ruby alaska as s...	1	[just, got, sent, this, photo, from, ruby, ala...

5.3. Xây dựng model

Trình bày trong file **Build_BERT_Model.ipynb**

5.4. Triển khai trên web

1. Vào thư mục BE, chạy server bằng command:

python model.py

2. Vào thư mục FE, chạy client:

Sử dụng Extensions "Live Server" trên vscode và **go live** file **index.html**

Tài liệu tham khảo

[1]. https://huggingface.co/bert-large-uncased
[2]. https://www.kaggle.com/code/canamika27/disasater-tweet-classification/notebook
[3]. https://github.com/chiragsamal/Flask-Tutorial

Thanks!

Any questions?