BỘ GIÁO DỤC VÀ ĐÀO TẠO
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO THỰC TẬP CƠ SỞ

ĐỀ TÀI

PHÁT HIỆN TIN GIẢ SỬ DỤNG MÔ HÌNH BERT

Giảng viên: Đinh Xuân Trường

Thành viên nhóm

1. Cao Xuân Đạt - B19DCCN167
2. Nguyễn Quang Huy- B20DCCN318
3. Nguyễn Thị Ước- B20DCCN718

Hà Nội, năm 2023

Lời cảm ơn

Lời đầu tiên, nhóm em xin gửi lời cảm ơn chân thành đến Học viện Công nghệ Bưu chính Viễn thông đã đưa môn Thực tập cơ sở vào chương trình giảng dạy. Qua quá trình học tập, nhóm em được trang bị vô số kiến thức quý báu cũng như những kỹ năng cần thiết cũng như có định hướng rèn luyện, phát triển thêm sau khi hoàn thành môn học.

Đặc biệt, nhóm em xin gửi lời cảm ơn sâu sắc đến giảng viên bộ môn Thực tập cơ sở Thầy Đinh Xuân Trường - với sự tận tình và tâm huyết đã dạy dỗ, truyền đạt những kiến thức quý báu cho nhóm em trong suốt thời gian học tập vừa qua, giúp nhóm em trang bị cho mình hành trang để có thể vững bước sau này. Nhóm em xin chúc thầy mạnh khỏe, công tác tốt và đạt được những thành công trong sự nghiệp.

Nhóm em xin chân thành cảm ơn.

Mục lục

Danh sách hình ảnh

Bảng phân công công việc

Mã sinh viên	Họ và tên	Nội dung công việc
B19DCCN167	Cao Xuân Đạt	- Thuyết trình Demo Web - Thiết kế giao diện Web
B20DCCN318	Nguyễn Quang Huy	- Code + Train mô hình - Làm Slide - Thuyết trình
B20DCCN718	Nguyễn Thị Ước	- Tìm hiểu, tổng hợp nội dung - Làm báo cáo - Code Web

I. Tổng quan

Với sự phát triển của công nghệ thông tin, mạng Internet đã lan rộng và phủ sóng toàn cầu. Bên cạnh những lợi ích mà mạng xã hội mang lại, chúng ta đang đối mặt với nhiều nguy cơ, thách thức không nhỏ, thậm chí đe dọa đến an ninh quốc qia và trật tự an toàn xã hội. Trong đó phải để đến những ảnh hưởng tiêu cục từ các thông tin xấu, độc hại được lan truyền trên các mạng xã hội cũng như vấn nạn tin giả - Fake News. Tin tức giả mạo trên các nền tảng khác nhau đang lan truyền rộng rãi và là một vấn đề đáng lo ngại, vì nó gây ra các cuộc chiến xã hội và phá vỡ vĩnh viễn các mối quan hệ được thiết lập giữa mọi người. Hệ lụy của việc lan truyền "tin giả" không chỉ dừng lại ở những cá nhân đơn lẻ, những nhóm người ở từng địa phương nhất định mà còn có tác động lớn hơn, đe dọa trực tiếp đến an ninh quốc gia.

Có thể phân loại tin giả thành hai loại:

- Loại thứ nhất là những thông tin hoàn toàn không chính xác (bao gồm cả những thông tin thông thường và những thông tin được trình bày giống như một tin báo chí) được cố tình đăng tải, lan truyền vì một mục đích nào đó
- Loại thứ hai là những thông tin có thể có một phần sự thật nhưng không hoàn toàn chính xác do người viết không kiểm chứng toàn bộ sự thật tước khi đăng tải chia sẻ hoặc có thể những tin tức đã được phóng đại một phần.

Đối với lại tin tức thứ nhất, có thể thấy trường hợp điển hình là trong cuộc bầu cử Tổng thống Mỹ năm 2016. Đây được xem như là một môi trường gần như hoàn hảo cho sự nảy nở của tin tức giả. Sự kiện này được thảo luận trên toàn cầu với nhiều luồng ý kiến tranh luận khác nhau. Trong bầu không khí mà người ta chưa biết điều gì có thể xảy ra hoặc có thể tin vào điều gì thì họ sẽ trở nên dễ tiếp nhật những điều được cường điệu hóa hay xuyên tạc. "Giáo hoàng ủng hộ Trump", "Hillary bán vũ khí cho IS", "Mật vụ FBI tình nghi trong vụ rò rỉ thư điện tử của bà Hillary Clinton được tìm thấy đã chết" – những tin tức này đã được lan truyền trước thềm bầu cử, thu hút sự chú ý lớn của mọi người, vượt qua cả những tin tức chính xác được chia sẻ trên mạng xã hội Facebook.

Để có thể phân biệt được tin giả, nhóm em sẽ thực hiện dự án Phân biệt tin giả sử dụng mô hình Bert dựa trên tập dataset có sẵn.

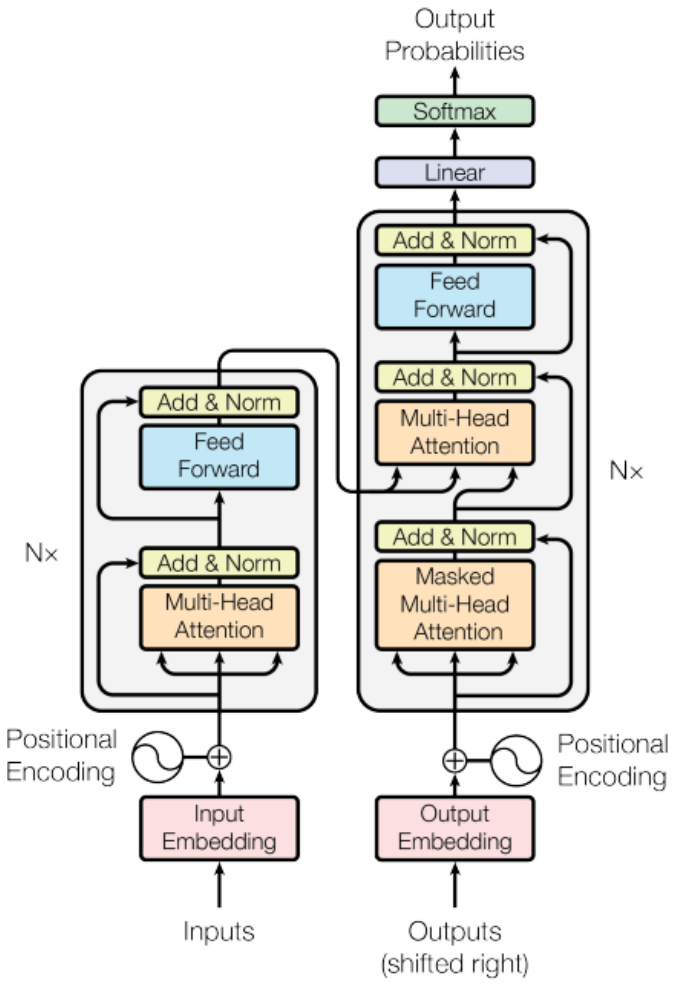
II. Kiến trúc Transformer

1. Giới thiệu về Transformer

Với sự ra đời của cơ chế attention thì vào năm 2017 paper Attention is all you need [1] đã giới thiệu một kiến trúc mới dành cho các bài toán NLP mà không có sự xuất hiện của các mạng nơ-ron hồi tiếp (RNN, LSTM,...) hay là mạng nơ-rơn tích chập (CNN) - đó là **Transformer**.

Mô hình Transformer là nền tảng của rất nhiều mô hình khác mà nổi tiếng nhất là BERT (Bidirectional Encoder Representations from Transformers) một mô hình dùng để học biểu diễn của các từ tốt nhất hiện tại và đã tạo ra một bước ngoặt lớn cho động đồng NLP trong năm 2019.

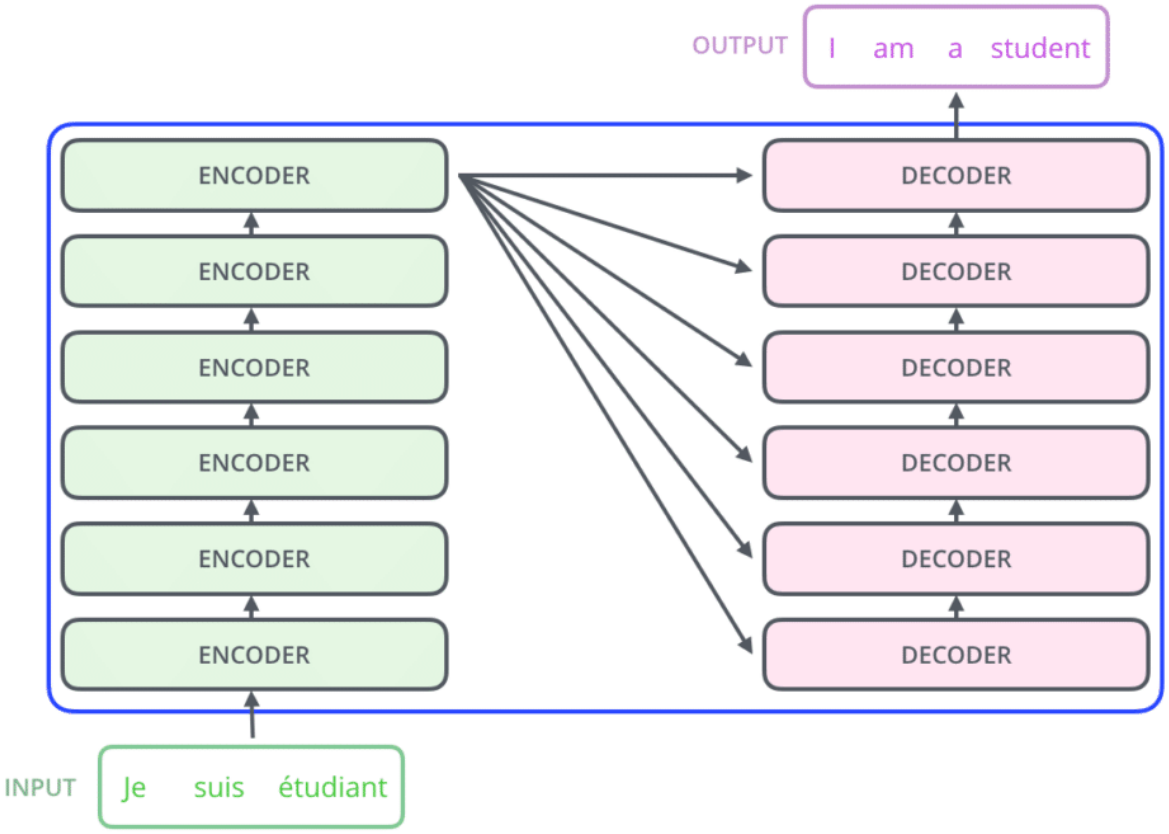
2. Kiến trúc tổng quan



Hình 1: Kiến trúc của Transformer

Giống như những mô hình dịch máy khác, kiến trúc tổng quan của mô hình transformer bao gồm 2 phần lớn là encoder và decoder. Encoder dùng để học vector biểu của câu với mong muốn rằng vector này mang thông tin hoàn hảo của câu đó. Decoder thực hiện chức năng chuyển vector biểu diễn kia thành ngôn ngữ đích.

Encoder thường có Nx = 6 layers chồng lên nhau. Mỗi layer sẽ có hai sub-layers bao gồm multi-head self-encode và feed forward neural network. Decoder, tương tự cũng có Nx = 6 layers chồng lên nhau. Kiến trúc thì khá giống encoder nhưng chỉ có thêm khối masked multi-head attention ở vị trí đầu tiên.

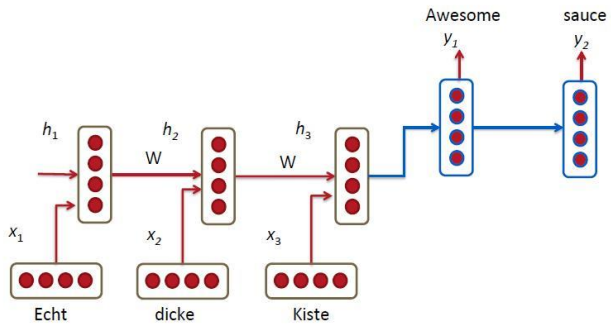


Hình 2: Ví dụ về bài toán sử dụng kiến trúc Transformer

Trong ví dụ ở dưới, encoder của mô hình transformer nhận một câu tiếng pháp, và encode thành một vector biểu diễn ngữ nghĩa của câu **je suis étudiant**, sau đó mô hình decoder nhận vector biểu diễn này, và dịch nó thành câu tiếng anh **I am a student**.

3. Tìm hiểu Positional Encoding

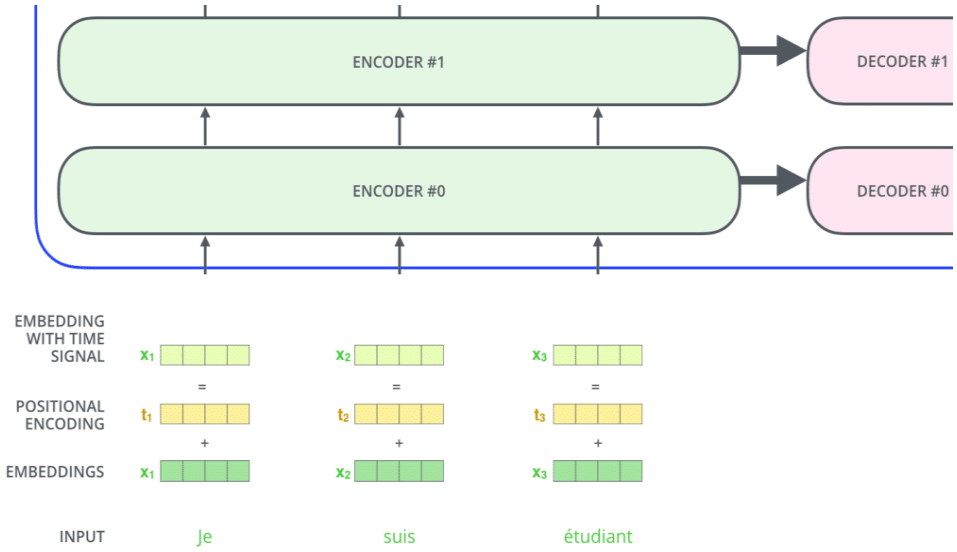
Khác với mạng RNNs các từ sẽ được đưa lần lượt vào xử lý dẫn đến điểm yếu là xử lý chậm, có gây ra mất thời gian, mất thông tin trong quá trình xử lý. Nhưng ưu điểm của RNNs là xác định được vị trí của các từ khi được vào encode.



Hình 3: Quá trình xử lý của RNNs

Transformer các từ cũng được vào cùng một lúc dẫn đến không thể xác định được thứ tự giữa các từ. Từ nào đến trước từ nào đến sau. Trong một số trường hợp, việc nhầm thứ tự sẽ gây ra hiểu nhầm về thông tin đầu ra và dẫn đến ý nghĩa của thông tin đầu ra sẽ được hiểu theo một hướng khác. Ví dụ như "Tôi thích bạn" khi không biết được vị trí chính xác sẽ được hiểu nhầm thành "Bạn thích tôi", lúc này ý nghĩa của câu đầu vào sẽ được hiểu theo một nghĩa khác với câu gốc. Vậy nên cần một cơ chế để ghi lại vị trí giữa các từ trong câu.

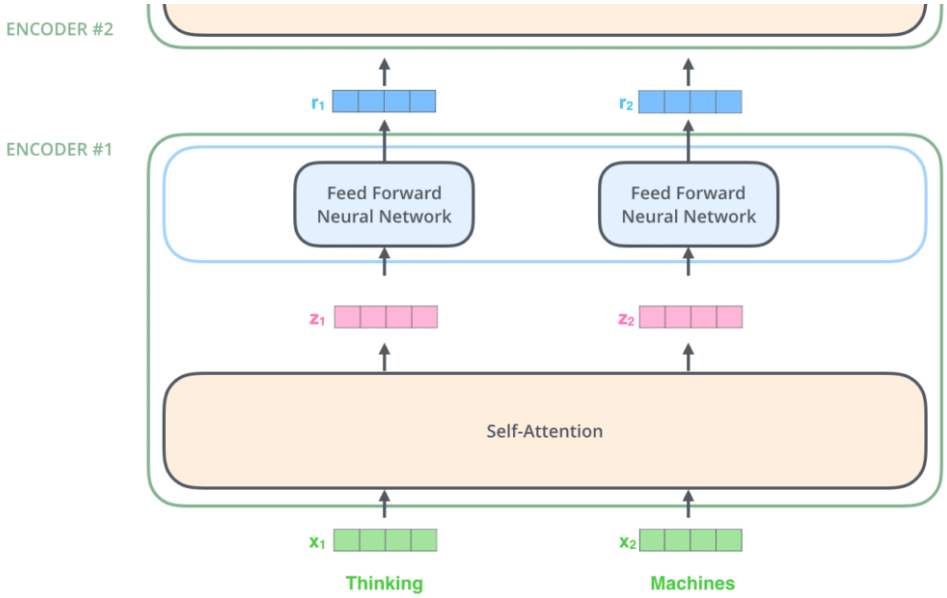
Để giải quyết vấn đề này, transformer thêm một véc tơ vào mỗi embedding đầu vào. Các véc tơ này tuân theo một mẫu cố định mà mô hình học được, giúp nó xác định vị trí của từng từ hoặc khoảng cách của các từ khác nhau trong chuỗi. Ý tưởng ở đây là việc thêm các giá trị đó sẽ cung cấp thông tin về khoảng cách giữa các véc tơ embedding khi chúng được phản ánh thông qua các véc tơ Q/K/V và thông qua phép lấy tích vô hướng.



Hình 4: Vector mã hóa vị trí (positional embedding)

4. Tìm hiểu Encoder

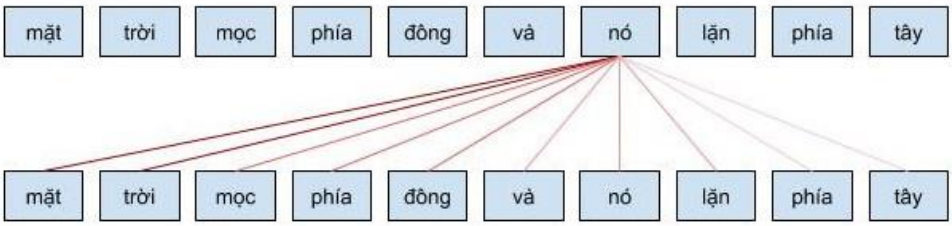
Một encoder nhận vào một danh sách các véc tơ và xử lý chúng bằng cách truyền các véc tơ này qua lớp self-attention và lớp feed-forward. Đầu ra của nó được gửi tới encoder tiếp theo.



Hình 5: Cấu trúc Encoder

Self-Attention Layer

Self-Attention tạo ra quan hệ giữa các từ trong câu, cho phép mô hình khi mã hóa một từ có thể sử dụng thông tin của những từ liên quan tới nó. Ví dụ khi từ **nó** được mã hóa, nó sẽ chú ý vào các từ liên quan như là **mặt trời**.



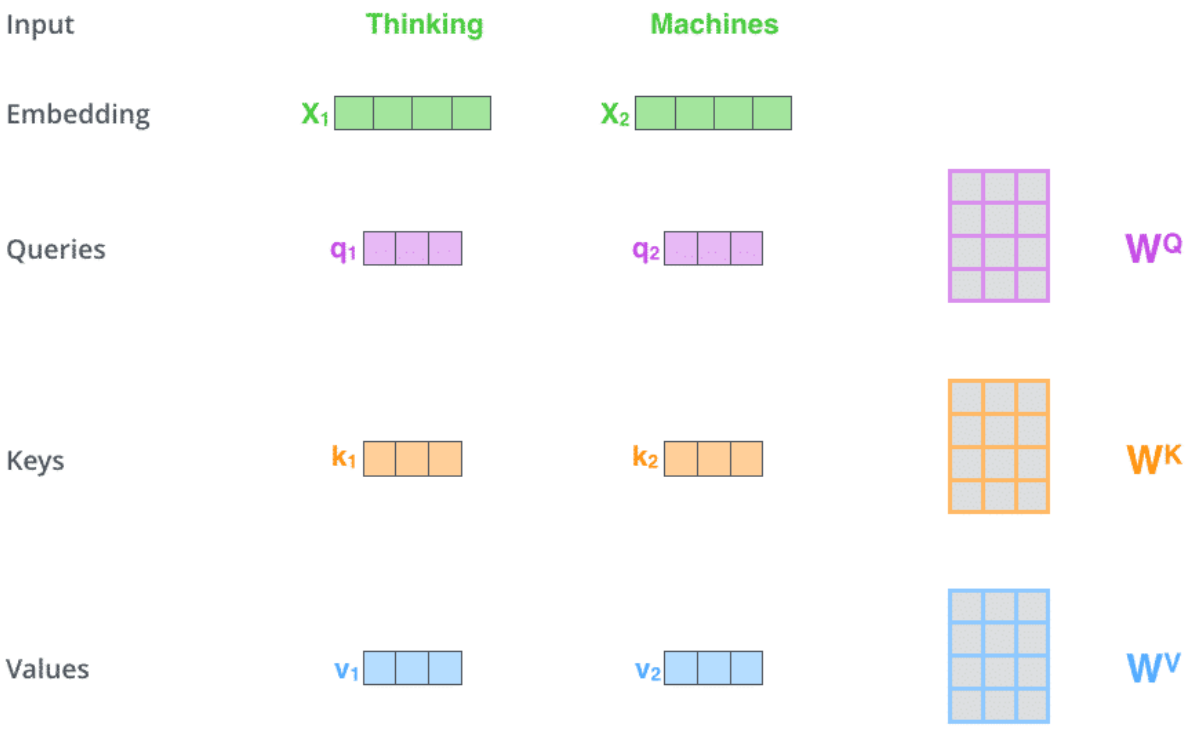
Hình 6: Mô hình mã hóa có thể sử dụng thông tin của các từ liên quan

Cơ chế self attention giống như cơ chế tìm kiếm. Với một từ cho trước, cơ chế này sẽ cho phép mô hình tìm kiếm trong cách từ còn lại, từ nào "giống" để sau đó thông tin sẽ được mã hóa dựa trên tất cả các từ trên.

Bước đầu tiên để tính self-attention là tạo ra ba vector từ mỗi vector đầu vào của encoder (trong trường hợp này là embedding của mỗi từ).

Với mỗi từ, sẽ tạo một vector truy vấn (Query), một vector khóa (Key), và một vector giá trị (Value). Các vector này được tạo ra bằng cách nhân embedding với ba ma trận được cập nhật trong quá trình huấn luyện.

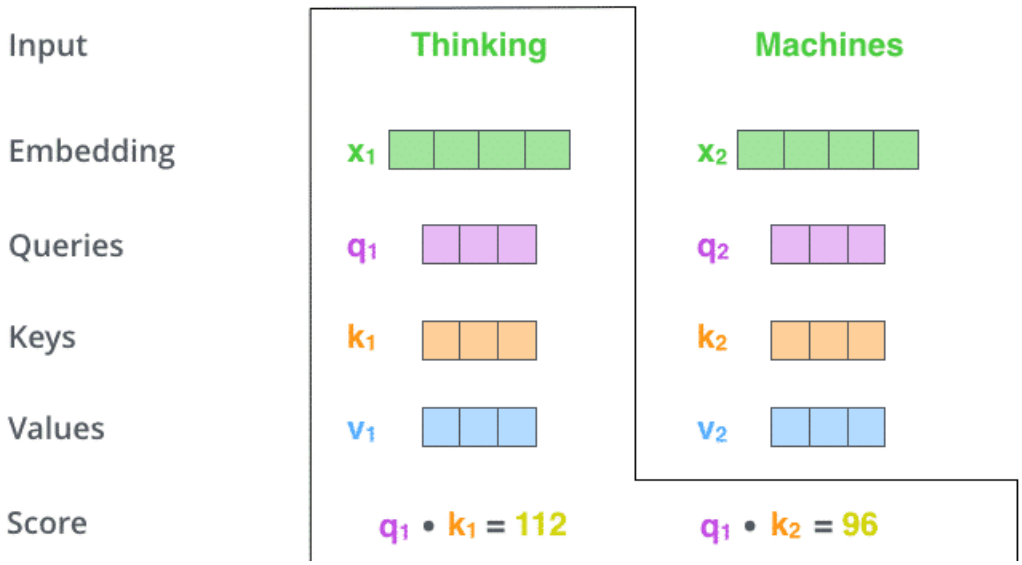
- Vector Query: vector dùng để chứa thông tin của từ được tìm kiếm, so sánh.
- Vector Key: vector dùng để biểu diễn thông tin các từ được so sánh với từ cần tìm kiếm ở trên.
- Vector Value: vector biểu diễn nội dung, ý nghĩa của các từ.



Hình 7: Cách thức tạo ra ba vector từ mỗi vector đầu vào của encoder

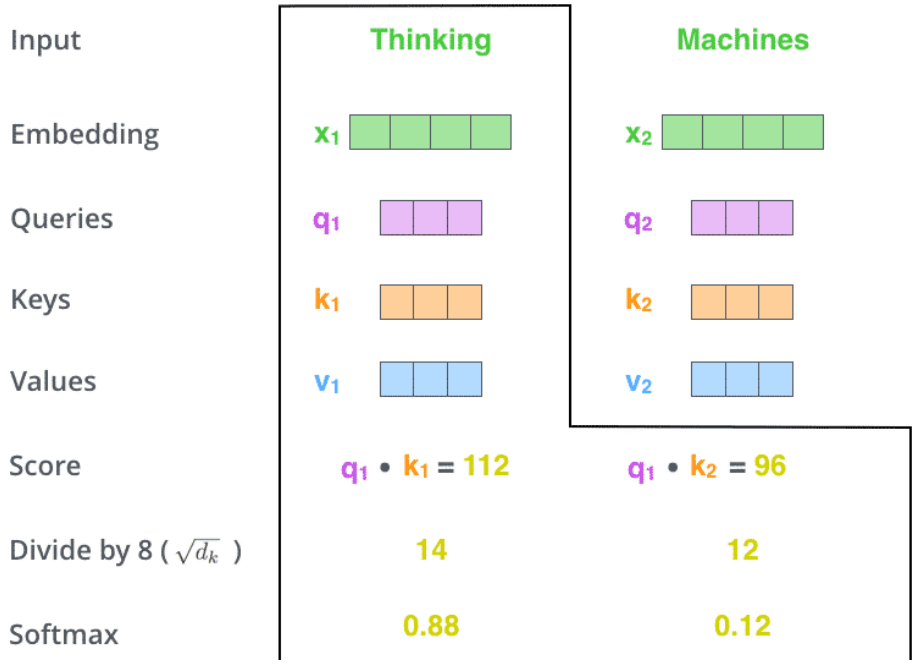
Bước thứ hai để tính self-attention là tính điểm. Giả sử tính self-attention cho từ đầu tiên trong ví dụ, "Thinking". Cần tính điểm cho mỗi từ trong câu đầu vào so với từ này. Điểm sẽ quyết định cần chú ý bao nhiêu vào các phần khác của câu đầu vào khi ta đang mã hóa một từ cụ thể.

Điểm được tính bằng phép nhân vô hướng giữa vector truy vấn với vector khóa của từ mà ta đang tính điểm. Nếu ta tiến hành self-attention cho từ ở vị trí thứ nhất, điểm đầu tiên sẽ là tích vô hướng của q1 và k1. Điểm thứ hai là tích vô hướng của q1 và k2.



Hình 8: Cách thức tính điểm cho mỗi từ trong câu đầu vào so với từ đang xét

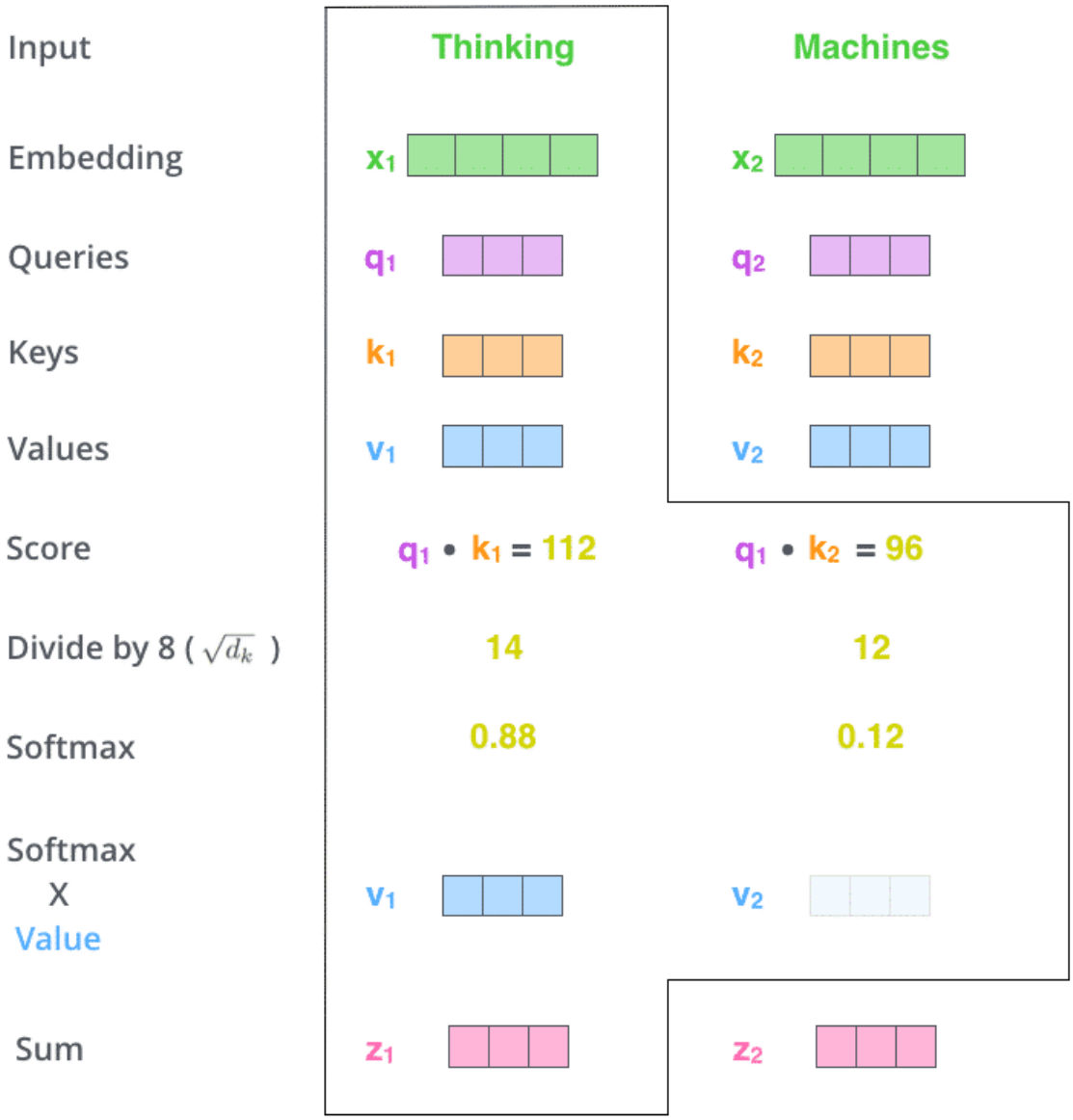
Bước thứ ba và bước thứ tư là chia điểm cho 8 (căn bậc hai của số chiều của véc tơ khóa trong bài báo gốc – 64. Điều này giúp cho độ dốc ổn định hơn. Có thể có các giá trị khả dĩ khác, nhưng đây là giá trị mặc định), và truyền kết quả qua một phép softmax. Softmax chuẩn hóa các điểm để chúng là các số dương có tổng bằng 1. Điểm softmax sẽ quyết định mỗi từ sẽ được thể hiện nhiều hay ít tại vị trí hiện tại.



Hình 9: Cách thức tính các giá trị softmax

Bước thứ năm là nhân mỗi vector giá trị với điểm softmax (trước khi cộng chúng lại). Một cách trực giác, việc này bảo toàn giá trị của các từ mà muốn chú ý và bỏ qua các từ không liên quan (nhân chúng với một số rất nhỏ, ví dụ 0.001).

Bước thứ sáu là cộng các vector giá trị đã được nhân trọng số. Kết quả chính là đầu ra của lớp self-attention tại vị trí hiện tại



Hình 10: Cách thức tính đầu ra của lớp self-attention tại vị trí hiện tại

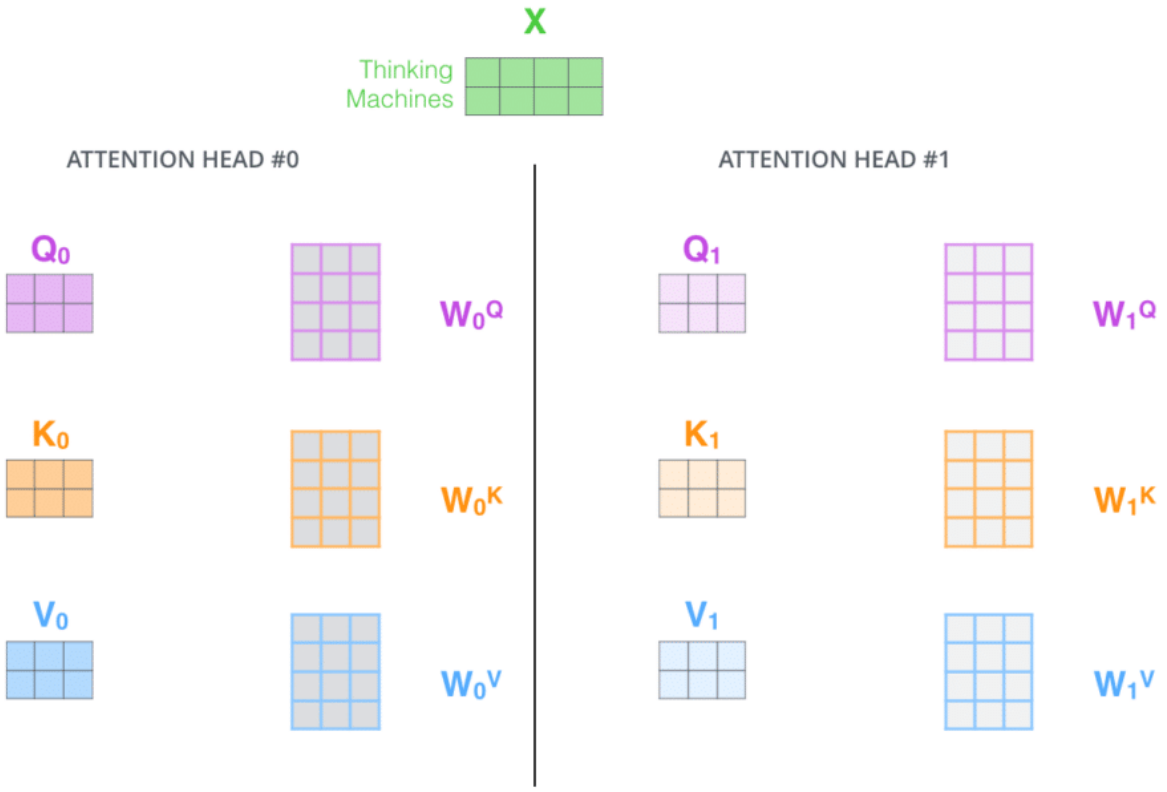
Đến đây là kết thúc việc tính toán self-attention. Vector kết quả có thể được gửi tới mạng truyền thẳng.

Multi – Head

Bài toán làm mịn lớp self-attention bằng cơ chế mang tên "multi-headed" attention. Cơ chế này cải thiện hiệu năng của lớp attention theo hai khía cạnh:

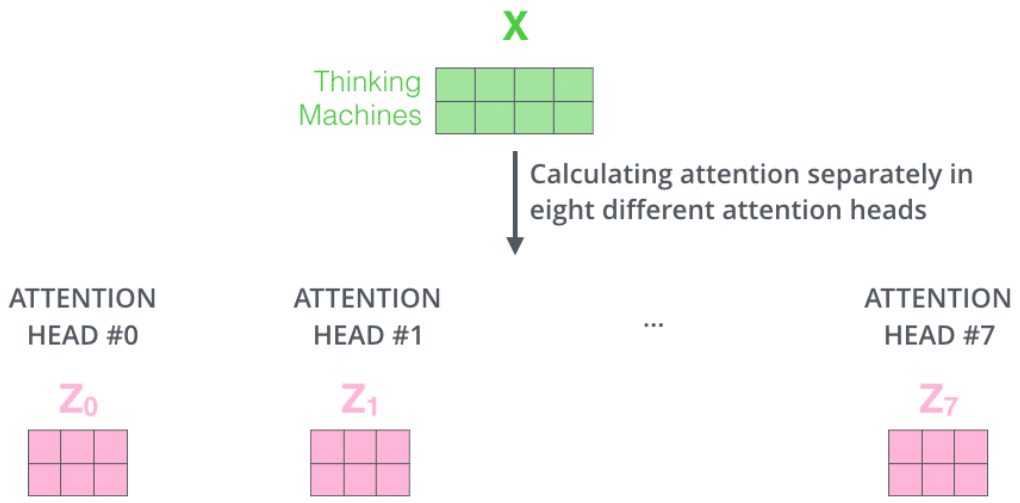
- Nó mở rộng khả năng của mô hình trong việc tập trung vào các vị trí khác nhau. Trong ví dụ phía trên, z1 chứa một số thông tin mã hóa từ các vị trí khác, nhưng bị lấn át bởi từ ở chính vị trí đó. Khi ta dịch câu "The animal didn't cross the street because it was too tired", ta muốn biết chính xác "it" dùng để chỉ cái gì.

- Nó mang lại cho lớp attention nhiều không gian con để biểu diễn. Như chúng ta sắp theo dõi, với multi-headed attention không chỉ có một mà nhiều bộ ma trận trọng số Query/Key/Value (Transformer sử dụng tám đầu attention, do đó sẽ có 8 bộ cho mỗi encoder/decoder). Mỗi bộ được khởi tạo ngẫu nhiên. Sau đó, kết thúc huấn luyện, mỗi bộ được dùng để phản ánh embedding đầu vào (hoặc vector từ các encoder/ decoder phía dưới) trong một không gian con riêng biệt.



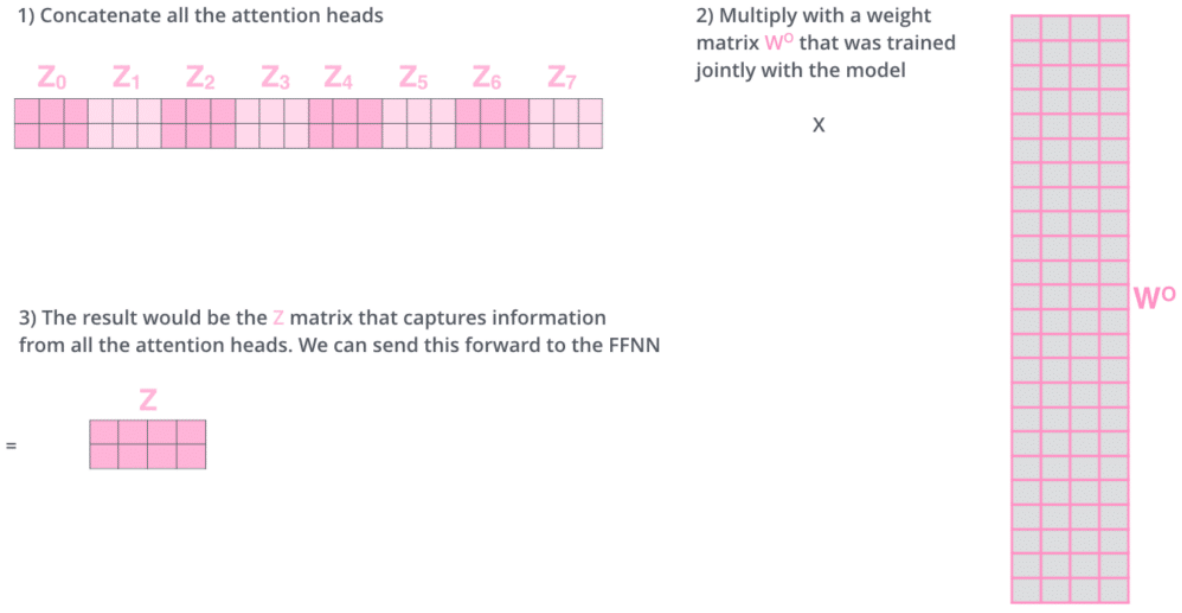
Hình 11: Transformer sử dụng tám đầu attention, do đó sẽ có 8 bộ cho mỗi encoder/decoder

Nếu thực hiện self-attention như đã vạch ra bên trên, với 8 lần tính với các ma trận khác nhau, ta có 8 ma trận Z khác nhau.



Hình 12: Hình ảnh tám ma trận Z được tạo ra

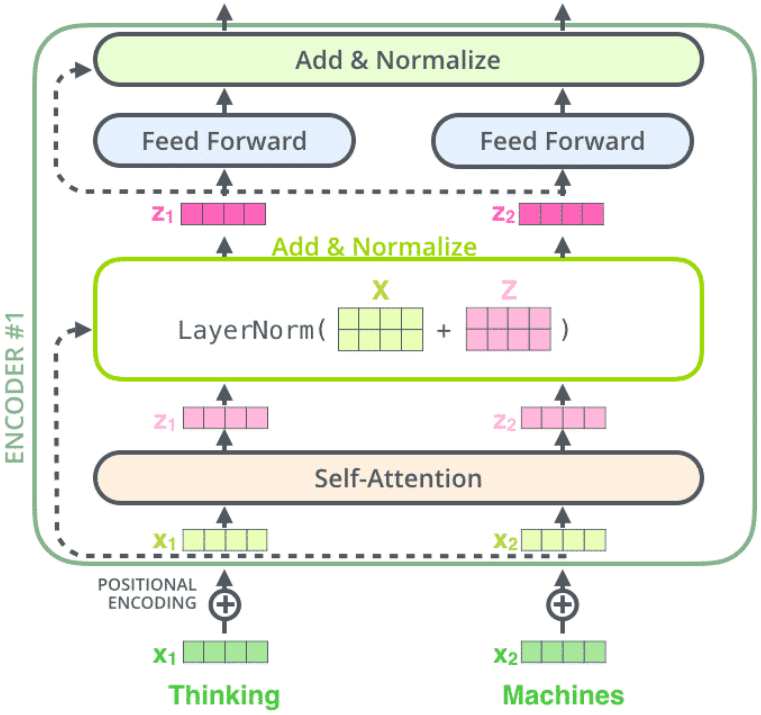
Mạng truyền thẳng không phù hợp để nhận vào 8 ma trận, thay vào đó nó cần 1 ma trận duy nhất (mỗi từ một véc tơ). Do đó, ta cần biến đổi 8 ma trận về 1 ma trận duy nhất. Vậy nên thực hiện nối các na trận lại và nhân chúng với một ma trận trọng số được bổ sung Wo.



Hình 13: Hình ảnh biến đổi tám ma trận Z về một ma trận duy nhất

Residuals Connection và Normalization Layer

Trong kiến trúc của mô hình transformer, residuals connection và normalization layer được sử dụng mọi nơi. Hai kỹ thuật giúp cho mô hình huấn luyện nhanh hội tụ hơn và trách mất mát thông tin trong quá trình huấn luyện mô hình, ví dụ như là thông tin của vị trí các từ được mã hóa.



Hình 14: Cách thức hoạt động của residuals connection và normalization layer

5. Tìm hiểu Decoder

Decoder thực hiện chức năng giải mã vector của câu nguồn thành câu đích, do đó decoder sẽ nhận thông tin từ encoder là 2 vector key và value. Kiến trúc của decoder rất giống với encoder, ngoại trừ có thêm một multi head attention nằm ở giữa dùng để học mối liên quan giữ từ đang được dịch với các từ được ở câu nguồn.

Encoder bắt đầu bằng việc xử lý câu đầu vào. Kế quả của encoder trên cùng được chuyển thành một bộ các véc tơ attention K và V. Chúng được sử dụng bởi mỗi decoder trong lớp "encoder-decoder attention" để giúp decoder tập trung vào phần quan trọng trong chuỗi đầu vào.

Các lớp self attention của decoder hoạt động hơi khác so với tại encoder:

- Trong decoder, lớp self-attention chỉ cho phép chú ý lên các vị trí phía trước của chuỗi đầu ra. Điều này được thực hiện bằng cách che đi các vị trí phía sau thông qua masking (đặt chúng về -inf) trước bước softmax khi tính self-attention.

- Lớp "Encoder-Decoder Attention" hoạt động như self-attention nhiều đầu, ngoại trừ việc nó tạo các ma trận Q từ lớp phía dưới và lấy các ma trận K và V từ đầu ra của ngăn xếp encoder.

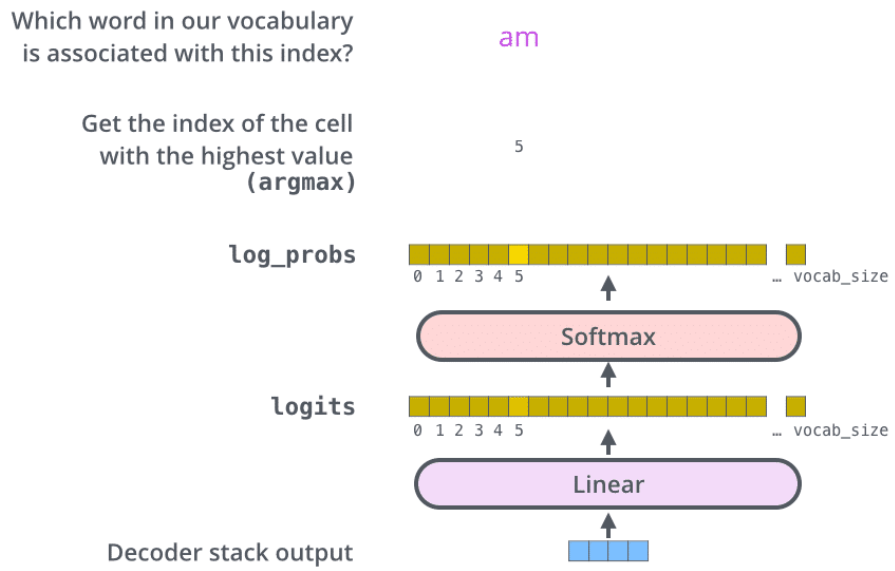
6. Tìm hiểu Linear và Softmax

Ngăn xếp decoder đưa ra vector của các số thực. Lớp tuyến tính cuối cùng và lớp Softmax sẽ chuyển nó thành các từ.

Lớp Linear là một mạng kết nối đầy đủ đơn giản, ánh xạ vector được tạo bởi ngăn xếp decoder thành một vector lớn hơn rất nhiều, được gọi là vector logit.

Giả sử mô hình biết 10K từ tiếng Anh (tập từ vựng đầu ra của mô hình) được học từ bộ huấn luyện. Nó sẽ tạo ra vector logit có độ rộng 10K ô – mỗi ô tương ứng với một điểm của một từ. Đó là cách chúng ta phiên dịch đầu ra của mô hình sau khi đi qua lớp tuyến tính.

Lớp softmax sẽ chuyển các điểm này thành xác suất (các số dương có tổng bằng 1). Ô với xác suất cao nhất được chọn, và từ ứng với nó là đầu ra của bước hiện tại.



Hình 15: Cách thức hoạt động của Linear và Softmax

III. Mô hình Bert

1. Bert là gì?

BERT [2] được biết tới đầy đủ với tên gọi "Bidirectional Encoder Representations from Transformers" (tạm dịch: Đại diện bộ mã hóa hai chiều từ Transformers) được công bố trên Blog của Google AI vào đầu tháng 11 năm 2018, được xem như một nghiên cứu mới mang tính đột phá của Google trong lĩnh vực xử lý ngôn ngữ tự nhiên.

Bert được hiểu là một mô hình học sẵn hay còn gọi là pre-train model, học ra các vector đại diện theo ngữ cảnh 2 chiều của từ, được sử dụng để transfer sang các bài toán khác trong lĩnh vực xử lý ngôn ngữ tự nhiên. BERT đã thành công trong việc cải thiện những công việc gần đây trong việc tìm ra đại diện của từ trong không gian số (không gian mà máy tính có thể hiểu được) thông qua ngữ cảnh của nó.

Ví dụ: nếu tìm kiếm cụm từ cow fishing, thạt chât đây là tên một loài cá (cá nược). Thế nhưng khi bạn gõ cụm từ về cow fishing, Google sẽ cung cấp kết quả liên quan đến chăn nuôi bò.

Mặc dù bạn đã cố tình sử dụng từ "fishing" để cung cấp ngữ cảnh, Google đã bỏ qua bối cảnh đó và cung cấp kết quả liên quan đến bò. Đó là vào ngày 1 tháng 10 năm 2019. Sau đó, ngày 25 tháng 10 năm 2019, cùng một kết quả truy vấn trong kết quả tìm kiếm có đầy đủ các kết quả liên quan đến cá này và việc câu cá. Thuật toán BERT dường như đã hiểu ngữ cảnh của từ "fishing" là quan trọng và thay đổi kết quả tìm kiếm để tập trung vào các trang web liên quan đến câu cá.

Điều này là do hệ thống máy móc không thể phân biệt chính xác ngữ nghĩa câu giống như con người. BERT ra đời để giải quyết điều này, nó giúp phân biệt những sắc thái nghĩa trừu tượng nhất, đưa ra những kết quả phù hợp và có liên quan nhất.

2. Sự ra đời của Bert

Sự thiếu hụt dữ liệu đào tạo là một trong những thách thức lớn nhất trong lĩnh vực xử lý ngôn ngữ tự nhiên. Đây là một lĩnh vực rộng lớn và đa dạng với nhiều nhiệm vụ riêng biệt, hầu hết các tập dữ liệu đều chỉ đặc thù cho từng nhiệm vụ. Để thực hiện được tốt những nhiệm vụ này cần những bộ dữ liệu lớn chứa hàng triệu thậm chí hàng tỷ ví dụ mẫu. Tuy nhiên, trong thực tế hầu hết các tập dữ liệu chỉ chứa vài nghìn hoặc vài trăm nghìn mẫu được đánh nhãn bằng tay bởi con người (các chuyên gia ngôn ngữ học). Sự thiếu hụt dữ liệu có nhãn chất lượng cao để đào tạo mô hình gây cản trở lớn cho sự phát triển của NLP nói chung.

Để giải quyết thách thức này, các mô hình xử lý ngôn ngữ tự nhiên sử dụng một cơ chế tiền xử lý dữ liệu huấn luyện bằng việc transfer từ một mô hình chung được đào tạo từ một lượng lớn các dữ liệu không được gán nhãn. Ví dụ một số mô hình đã được nghiên cứu trước đây để thực hiện nhiệm vụ này như Word2vec, Glove hay FastText.

Việc nghiên cứu các mô hình này sẽ giúp thu hẹp khoảng cách giữa các tập dữ liệu chuyên biệt cho đào tạo bằng việc xây dựng mô hình tìm ra đại diện chung của ngôn ngữ sử dụng một số lượng lớn các văn bản chưa được gán nhãn lấy từ các trang web.

Các pre-train model khi được tinh chỉnh lại trên các nhiệm vụ khác nhau với các bộ dữ liệu nhỏ như Question Answering, Sentiment Analysis,...sẽ dẫn đến sự cải thiện đáng kể về độ chính xác cho so với các mô hình được huấn luyện trước với các bộ dữ liệu này.

Tuy nhiên, các mô hình kể trên có những yếu điểm riêng của nó, đặc biệt là không thể hiện được sự đại diện theo ngữ cảnh cụ thể của từ trong từng lĩnh vực hay văn cảnh cụ thể.

Tiếp nối sự thành công nhất định của các mô hình trước đó, Google đã công bố thêm 1 kỹ thuật mới được gọi là Bidirectional Encoder Representations from Transformers(BERT). Với lần công bố này, Google khẳng định bất kỳ ai trên thế giới đều có thể đào tạo được các hệ thống hỏi đáp (Question Answering) cải tiến hơn cho riêng mình hoặc rất nhiều các mô hình NLP khác.

3. Bert có thể biểu diễn ngữ cảnh 2 chiều

Về mặt lý thuyết, các kỹ thuật khác như Word2vec, FastText hay Glove cũng tìm ra đại diện của từ thông qua ngữ cảnh chung của chúng. Tuy nhiên, những ngữ cảnh này là đa dạng trong dữ liệu tự nhiên. Ví dụ các từ như "con chuột" có ngữ nghĩa khác nhau ở các ngữ cảnh khác nhau như "Con chuột máy tính này thật đẹp!!" và "con chuột này to thật." Trong khi các mô hình như Word2vec, fastText tìm ra 1 vector đại diện cho mỗi từ dựa trên 1 tập ngữ liệu lớn nên không thể hiện được sự đa dạng của ngữ cảnh. Việc tạo ra một biểu diễn của mỗi từ dựa trên các từ khác trong câu sẽ mang lại kết quả ý nghĩa hơn nhiều. Như trong trường hợp trên ý nghĩa của từ con chuột sẽ được biểu diễn cụ thể dựa vào phần trước hoặc sau nó trong câu. Nếu đại diện của từ "con chuột" được xây dựng dựa trên những ngữ cảnh cụ thể này thì ta sẽ có được biểu diễn tốt hơn.

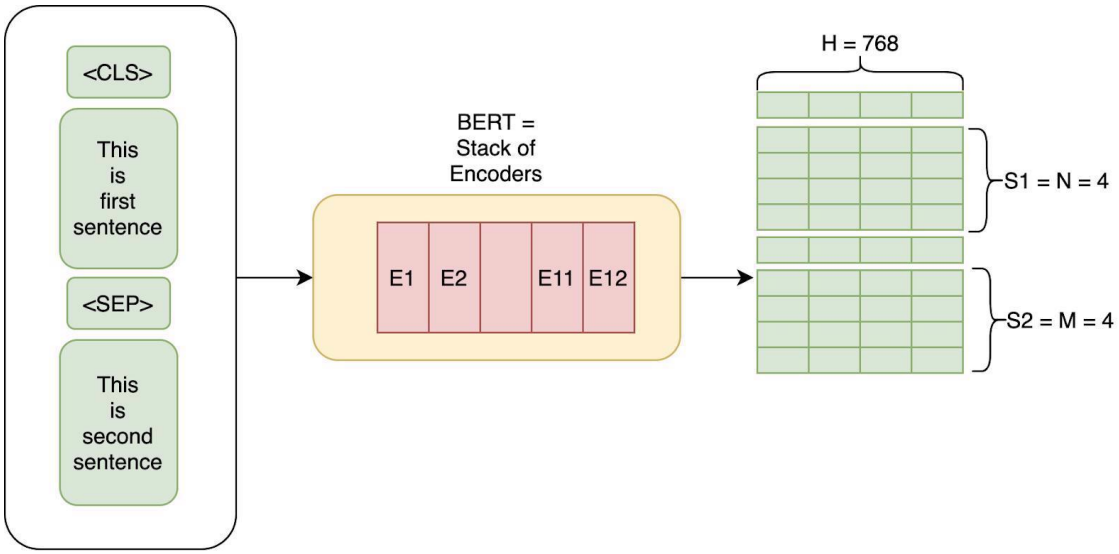
BERT mở rộng khả năng của các phương pháp trước đây bằng cách tạo các biểu diễn theo ngữ cảnh dựa trên các từ trước và sau đó để dẫn đến một mô hình ngôn ngữ với ngữ nghĩa phong phú hơn.

4. Kiến trúc của Bert

[CLS]: Mã thông báo này được gọi là mã thông báo ' *phân loại* '. Nó được sử dụng ở đầu một chuỗi.

[SEP]: Mã thông báo này cho biết sự tách biệt của 2 chuỗi tức là nó hoạt động như một dấu phân cách.

[MASK]: Được sử dụng để chỉ mã thông báo được che giấu trong nhiệm vụ MLM.



Hình 16: Mô tả một câu khi đưa vào Bert

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.

- ***BERT*$_{BASE}$: L=12, H=768, A=12, Total Parameters=110M**
- ***BERT*$_{LARGE}$: L=24, H=1024, A=16, Total Parameters=340M**

***BERT*$_{BASE}$** đã được chọn để có một kích thước mô hình giống hệt như mô hình OpenAI GPT để nhằm mục đích so sánh giữa 2 mô hình này. Tuy nhiên, một cách đơn giản để so sánh, BERT Transformer sử dụng các attention 2 chiều trong khi GPT Transformer sử dụng

các attention 1 chiều nơi mà tất cả các từ chỉ chú ý tới ngữ cảnh trái của nó. 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.

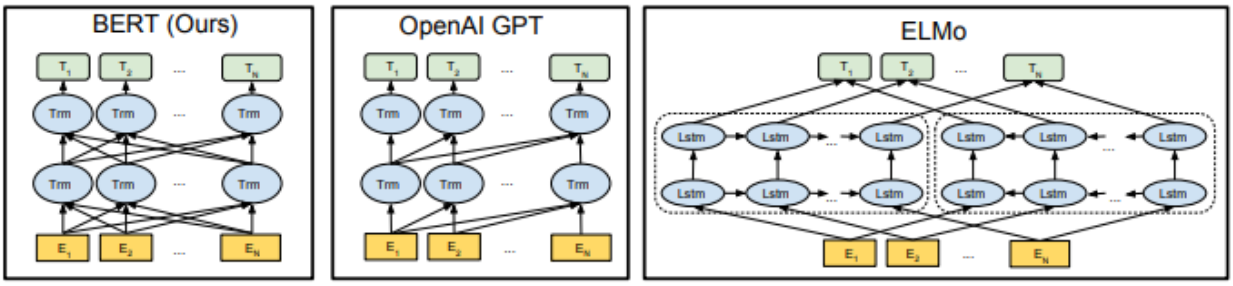


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

Hình 17: So sánh giữa mô hình Bert, ChatGPT và ELMo

5. Chi tiết bổ sung cho Bert

5.1. Biểu diễn đầu vào

Đầu vào 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.

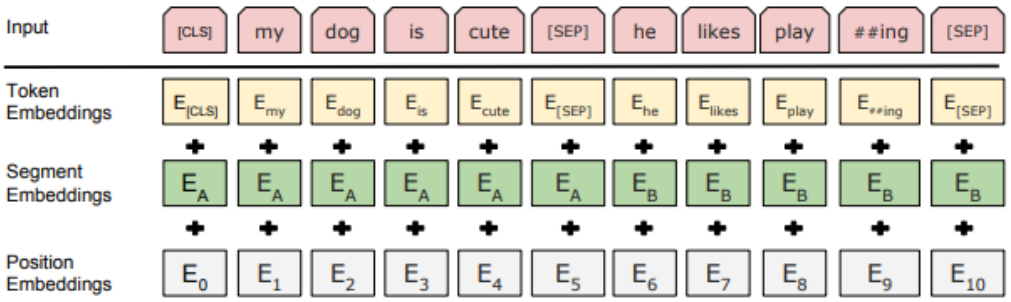


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Hình 18: Biểu diễn đầu vào Bert

Một số điểm cần lưu ý:

- Sử dụng WordPiece embeddings (Wu et al., 2016) với một từ điển 30.000 từ và sử dụng ## làm dấu phân tách. Ví dụng từ ***playing*** được tách thành ***play##ing***.
- 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, phân biệt các câu theo 2 cách. Đầu tiên, chúng ta tách chúng bởi 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.

5.2. Bert trước đào tạo

Đào tạo BERT bằng cách sử dụng 2 nhiệm vụ dự đoán không giám sát được gọi là **Masked LM** và **Next Sentence Prediction**.

Masked LM

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).

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, 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 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 mối 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.

Next Sentence Prediction (Dự đoán câu tiếp theo)

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, 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.

- **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.

5.3. Quá trình tiền đào tạo

Sử dụng 2 bộ dữ liệu cho quá trình đào tạo là BooksCorpus (800M words) (Zhu et al., 2015) và English Wikipedia (2,500M words). Đối với Wikipedia, chỉ trích xuất các đoạn văn bản và bỏ qua các danh sách, bảng và tiêu đề. Điều quan trọng là sử dụng một kho văn bản ở mức độ đoạn(bài văn) chứ không phải là một tập hợp các câu bị xáo trộn.

Để tạo ra một chuỗi đầu vào cho quá trình đào tạo, lấy mẫu gồm 2 spans liên tiếp nhau trong corpus mà tạm gọi đấy là các câu mặc dù chúng thường dài hơn nhiều so với các câu đơn thông thường(hoặc cũng có thể ngắn hơn). Lấy mẫu sao cho sau khi kết hợp, chiều dài của mẫu kết hợp tối đa chứ 512 tokens. Các mask cho MLM vẫn được sử dụng sau khi áp dụng WordPiece tokenization với một tỷ lệ thống nhất là 15%. Việc đào tạo $\boldsymbol{BERT_{BASE}}$ được thực hiện trên 4 Cloud TPUs với tổng số 16 chíp TPUs. Việc đào tạo $\boldsymbol{BERT_{LARGE}}$ được thực hiện trên 16 Cloud TPUs với tổng số 64 chíp. Thời gian thực hiện mỗi lần training là khoảng 4 ngày.

5.4. Tinh chỉnh Bert

Đối với các nhiệm vụ phân loại câu, BERT được fine-tuning rất đơn giản. Để có được biểu diễn của một chuỗi đầu vào với số chiều cố định, chúng ta chỉ cần lấy hidden state ở lớp cuối cùng, tức là đầu ra của lớp Transformer cho token đầu tiên(token đặc biệt [CLS] được xây dựng cho đầu chuỗi). Chúng ta gọi vector này là **C** ($\boldsymbol{C \in R^H}$). Chỉ có 1 tham số được thêm vào trong quá trình fine-tuning $\boldsymbol{W \in R^{K*H}}$ với K là số nhãn lớp phân loại. Xác suất của nhãn **P** là một phân phối với $\boldsymbol{P \in R^K}$ được tính toán bởi 1 hàm softmax $\boldsymbol{P = softmax(C * W^T)}$. Tất cả các tham số của BERT và W được fine-tuning để tối ưu hóa hàm lỗi.

IV. Giới thiệu Hugging Face Transformer

Hugging Face Transformers là một thư viện mã nguồn mở được phát triển bởi Hugging Face. Transformers cung cấp các API và công cụ để dễ dàng tải xuống và huấn luyện các mô hình tiền huấn luyện tiên tiến nhất như BERT, GPT, Transformer-XL, và nhiều mô hình NLP khác.. Việc sử dụng các mô hình được đào tạo trước có thể giảm chi phí tính toán, đồng thời giúp tiết kiệm thời gian và tài nguyên cần thiết để đào tạo một mô hình từ đầu. Các mô hình này hỗ trợ các tác vụ phổ biến theo các phương thức khác nhau, chẳng hạn như:

- Xử lý ngôn ngữ tự nhiên : phân loại văn bản, nhận dạng thực thể được đặt tên, trả lời câu hỏi, mô hình hóa ngôn ngữ, tóm tắt, dịch thuật, trắc nghiệm và tạo văn bản.
- Thị giác máy tính : phân loại hình ảnh, phát hiện đối tượng và phân đoạn.
- Âm thanh : tự động nhận dạng giọng nói và phân loại âm thanh.
- Đa phương thức : trả lời câu hỏi trên bảng, nhận dạng ký tự quang học, trích xuất thông tin từ tài liệu được quét, phân loại video và trả lời câu hỏi bằng hình ảnh.

Hugging Face Transformers là một thư viện quan trọng và mạnh mẽ trong lĩnh vực xử lý ngôn ngữ tự nhiên, giúp người dùng tải về, tinh chỉnh và sử dụng các mô hình ngôn ngữ tiên tiến một cách dễ dàng.

V. Dataset

Dataset[3] là những dữ liệu được tổng hợp về những tin tức liên quan đến cuộc bầu của tổng thống Mỹ năm 2016 và những thông tin xoay xung quanh cuộc bầu cữ. Dữ liệu trong tập train.csv bao gồm những thông tin đúng sự thật, và những thông tin chưa đúng sự thật, những thông tin sai, những thông tin bị thiếu.

train.csv: Tập huấn luyện đầy đủ với các thuộc tính sau:

- **id**: id của bài báo
- **title**: tiêu đề của bài báo
- **author**: tác giả của bài báo
- **text**: nội dung của bài báo, có thể không đầy đủ
- **label**: nhãn dán của bài báo, đánh dấu là có đáng tin hay không đáng tin
    - 1: không đáng tin
    - 0: đáng tin

test.csv: Tập dữ liệu huấn luyện thử nghiệm có tất cả các thuộc tính giống nhau tại train.csv không có nhãn.

VI. Tinh chỉnh mô hình Bert cho Fake News Detection

Cài đặt thư viện:

pip install transformers nltk pandas numpy matplotlib seaborn wordcloud

Trong đó:

- Transformers: Thư viện Transformers cung cấp các công cụ và giao diện để làm việc với các mô hình xử lý ngôn ngữ tự nhiên (NLP), bao gồm cả BERT. Thư viện này cho phép tải và sử dụng các mô hình đã được huấn luyện trước (pre-trained models) và tiến hành tinh chỉnh (fine-tuning) trên tập dữ liệu.
- NLTK (Natural Language Toolkit): NLTK là một thư viện Python mạnh mẽ để xử lý ngôn ngữ tự nhiên. Nó cung cấp các công cụ cho việc tách từ (tokenization), phân loại từ loại (part-of-speech tagging), xử lý ngữ pháp, và các công cụ khác để tiền xử lý và phân tích văn bản.
- Pandas: Pandas là một thư viện phổ biến để làm việc với dữ liệu dạng bảng. Nó cung cấp cấu trúc dữ liệu gọi là DataFrame,tiện lợi trong việc xử lý, lọc và biến đổi dữ liệu.
- NumPy: NumPy là một thư viện Python để làm việc với ma trận và mảng nhiều chiều. Nó cung cấp các công cụ và hàm tính toán mạnh mẽ để thực hiện các phép toán số học và đại số tuyến tính trên dữ liệu số.
- Matplotlib: Matplotlib là một thư viện trực quan hóa dữ liệu trong Python. Nó cho phép tạo ra các biểu đồ, đồ thị, sơ đồ và hình ảnh khác để hiển thị dữ liệu theo cách dễ hiểu và trực quan.
- Seaborn: Seaborn là một thư viện trực quan hóa dữ liệu dựa trên Matplotlib. Nó cung cấp các chủ đề và mô-đun trực quan hóa cao cấp để tạo ra các biểu đồ và đồ thị chuyên nghiệp, hiển thị dữ liệu một cách thu hút và dễ hiểu hơn.
- Wordcloud: Thư viện Wordcloud giúp tạo ra hình ảnh từ dữ liệu văn bản, trong đó kích thước và tần suất xuất hiện của các từ sẽ được biểu diễn dưới dạng hình dạng đám mây. Điều này có thể giúp trực quan hóa và hiển thị các từ quan trọng hoặc phổ biến trong văn bản.

Các bước thực hiện:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk

nltk.download('stopwords')
nltk.download('wordnet')
```

Hình 19: Khai báo các thư viện cần thiết

Đoạn code khai báo các thư viện cần thiết và tải xuống các corpus ‘stopwords’ và ‘wordnet’

Stopwords là một danh sách các từ phổ biến trong ngôn ngữ tự nhiên như "a", "the", "is" và nhiều từ khác không mang ý nghĩa quan trọng trong việc phân tích ngôn ngữ tự nhiên. Các từ dừng này thường không đóng góp nhiều thông tin ngữ nghĩa và có xu hướng xuất hiện rất phổ biến trong các văn bản.

Khi làm việc với xử lý ngôn ngữ tự nhiên, một bước phổ biến là loại bỏ các từ dừng khỏi văn bản. Việc loại bỏ các từ dừng giúp giảm kích thước của dữ liệu và tập trung vào các từ quan trọng hơn. Điều này có thể cải thiện hiệu suất và độ chính xác của các tác vụ như phân loại văn bản, phân cụm và trích xuất thông tin.

Thư viện NLTK cung cấp một danh sách các từ dừng cho các ngôn ngữ khác nhau.

WordNet là một từ điển từ vựng tổ chức theo hướng đồ thị trong thư viện NLTK. Nó cung cấp một bộ từ vựng rộng lớn được chia thành các từ loại và các quan hệ từ vựng giữa chúng. WordNet có thể được sử dụng để tìm kiếm đồng nghĩa, đối nghịch, mệnh đề, và các mối quan hệ từ vựng khác.

WordNet cung cấp một giao diện dễ sử dụng để truy vấn và khám phá từ vựng theo các mối quan hệ.

Tiến hành tải dữ liệu để train model. Dữ liệu lúc này chưa được xử lý.

```
# Tải lên bộ dữ liệu
news_d = pd.read_csv("/content/train.csv")
```

```
print("Shape of News data:", news_d.shape) # Hiển thị số hàng , số cột
print("News data columns", news_d.columns) # Hiển thị các thuộc tính của tập dữ liệu
```

```
Shape of News data: (20800, 5)
News data columns Index(['id', 'title', 'author', 'text', 'label'], dtype='object')
```

```
# Hiển thị một số hàng đầu tiên trong dataframe df.
# Mặc định là hiển thị 5 hàng đầu tiên.
news_d.head()
```

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

Hình 20: Tải bộ dữ liệu và xem trước

Tiến hành clean dữ liệu:

Việc làm sạch dữ liệu trước khi huấn luyện là một bước quan trọng trong quá trình xử lý dữ liệu.

- Loại bỏ dữ liệu nhiễu: Dữ liệu thô thường chứa nhiễu như các ký tự đặc biệt, ký tự không phải chữ cái, dấu câu, đường dẫn URL, v.v. Việc loại bỏ các nhiễu này giúp tinh chỉnh dữ liệu và làm tăng độ chính xác của mô hình.
- Chuẩn hóa dữ liệu: Việc chuẩn hóa dữ liệu giúp đồng nhất hóa dữ liệu trong tập huấn luyện. Ví dụ, chuyển đổi chữ hoa thành chữ thường, loại bỏ khoảng trắng không cần thiết và dấu câu không cần thiết.
- Loại bỏ từ dừng (stop words): Các từ dừng là những từ phổ biến trong ngôn ngữ như "a", "an", "the", v.v. Những từ này thường không mang nhiều ý nghĩa và không đóng góp nhiều cho quá trình phân loại hoặc phân tích văn bản. Loại bỏ các từ dừng giúp làm giảm kích thước của dữ liệu và tập trung vào các từ quan trọng hơn.
- Chuyển đổi từ về dạng gốc (stemming hoặc lemmatization): Việc chuyển đổi các từ về dạng gốc giúp làm giảm đa dạng từ vựng trong dữ liệu. Stemming và lemmatization là các kỹ thuật để chuyển đổi từ về dạng gốc, ví dụ như chuyển đổi từ "running" thành "run" hoặc "went" thành "go". Điều này giúp tạo ra các nhóm từ tương đồng và giảm không gian đặc trưng.

- Tăng tính nhất quán: Bằng cách làm sạch dữ liệu, chúng ta đảm bảo rằng cùng một thông tin sẽ được biểu diễn theo cách nhất quán trong toàn bộ tập dữ liệu. Điều này giúp mô hình học được các mẫu và quy tắc chung hơn.

Đặt các hàng số để clean dữ liệu

```
# Các hằng số được sử dụng để clean datasets
column_n = ['id', 'title', 'author', 'text', 'label']
remove_c = ['id','author']
categorical_features = []
target_col = ['label']
text_f = ['title', 'text']
```

Hình 21: Thiết lập các hằng số để tiến hành clean dữ liệu

Tạo ra các hàm để clean dữ liệu như xóa các cột không sử dụng, đặt các giá trị null thành none, loại bỏ các ký tự không cần thiết

```
# Chuyển đổi các từ về dạng gốc (stem) của chúng
ps = PorterStemmer()
wnl = nltk.stem.WordNetLemmatizer()

# Tải danh sách các từ dừng (stop words) trong
# tiếng Anh từ thư viện NLTK
stop_words = stopwords.words('english')
# Tạo một từ điển đếm (counter) từ danh sách các từ dừng
stopwords_dict = Counter(stop_words)

# Xóa các cột không sử dụng
def remove_unused_c(df,column_n=remove_c):
    df = df.drop(column_n,axis=1)
    return df

# Đặt các giá trị null bằng None
def null_process(feature_df):
    for col in text_f:
        feature_df.loc[feature_df[col].isnull(), col] = "None"
    return feature_df
```

Hình 22: Thiết lập các hàm để tiến hành clean dữ liệu

```python
def clean_dataset(df):
    df = remove_unused_c(df)
    df = null_process(df)
    return df

# Loại bỏ các ký tự không cần thiết.
def clean_text(text):
    # loại bỏ các đường dẫn (URL)
    text = str(text).replace(r'http[\w:/\.]+', ' ')
    # loại bỏ mọi ký tự không phải là chữ cái, chữ số, dấu chấm
    # và khoảng trắng
    text = str(text).replace(r'[^\.\w\s]', ' ')
    # loại bỏ mọi ký tự không phải là chữ cái.
    text = str(text).replace('[^a-zA-Z]', ' ')
    # thay thế nhiều khoảng trắng liên tiếp bằng một khoảng
    # trắng duy nhất.
    text = str(text).replace(r'\s\s+', ' ')
    # Dòng này chuyển đổi văn bản thành chữ thường và loại bỏ
    # các khoảng trắng ở đầu và cuối chuỗi.
    text = text.lower().strip()

    return text
```

Hình 23: Thiết lập các hàm để tiến hành clean dữ liệu (p2)

Hàm clean_text sẽ thực hiện quá trình clean dữ liệu, hàm sẽ loại bỏ các đường dẫn URL, loại bỏ mọi ký tự không phải là chữ cái, chữ số, dấu chấm và khoảng trắng, chuyển đổi văn bản thành các chữ thường và loại bỏ khoảng trắng ở đầu và cuối câu.

```python
def nltk_preprocess(text):
    text = clean_text(text)
    wordlist = re.sub(r'[^\w\s]', '', text).split()
    text = ' '.join([wnl.lemmatize(word) for word in wordlist if word not in stopwords_dict])
    return  text
    # Kết quả cuối cùng là một chuỗi văn bản đã qua xử lý.
```

Hình 24: Thiết lập hàm xử lý để clean dữ liệu

Clean dữ liệu

```
df = clean_dataset(news_d)
# tiền xử lý văn bản
df["text"] = df.text.apply(nltk_preprocess)
# tiền xử lý tiêu đề
df["title"] = df.title.apply(nltk_preprocess)
# bộ dữ liệu sau khi tiền xử lý
df.head()
```

Hình 25: Clean dữ liệu

Kết quả của quá trình clean dữ liệu

	title	text	label
0	house dem aide didnt even see comeys letter ja...	house dem aide didnt even see comeys letter ja...	1
1	flynn hillary clinton big woman campus breitbart	ever get feeling life circle roundabout rather...	0
2	truth might get fired	truth might get fired october 29 2016 tension ...	1
3	15 civilian killed single u airstrike identified	video 15 civilian killed single u airstrike id...	1
4	iranian woman jailed fictional unpublished sto...	print iranian woman sentenced six year prison ...	1

Hình 26: Hình ảnh xem trước một số kết quả sau khi tiến hành clean dữ liệu

So sánh trước và sau khi clean dữ liệu

Trước khi clean dữ liệu	Sau khi clean dữ liệu
Sunday on CNN's "State of the Union," in reacting to reports about President Donald Trump's national security adviser Michael Flynn's phone conversations with a Russian ambassador, Sen. Al Franken ( ) said, "We need to have an independent investigation on it" because what he said was Trump having a "Putin crush. " Partial transcript as follows: FRANKEN: There is	sunday cnns state union reacting report president donald trump national security adviser michael flynns phone conversation russian ambassador sen al franken said need independent investigation said trump putin crush partial transcript follows franken lot need look need independent investigation tapper say independent mean independent counsel select committee

a lot here that we need to look at, and we need to have an independent investigation on it. TAPPER: When you say independent, what do you mean, independent counsel, select committee? FRANKEN: I think an independent counsel would be terrific but I know that Lindsey Graham and Sheldon Whitehouse in Judiciary are doing — did doing hearinngs and investigation. I trust those guys. There's something going on in intelligence and that's opaque. We need something transparent and we need an investigation because we don't know what he owes Russia.	franken think independent counsel would terrific know lindsey graham sheldon whitehouse judiciary hearinngs investigation trust guy there something going intelligence thats opaque need something transparent need investigation dont know owes russia dont know many russian oligarch invested business saddled putin many way he syria great tapper yeah franken didnt know annex crimea going nato there something he got bit putin crush there want know much tied maybe financial string follow pam key twitter pamkeynen

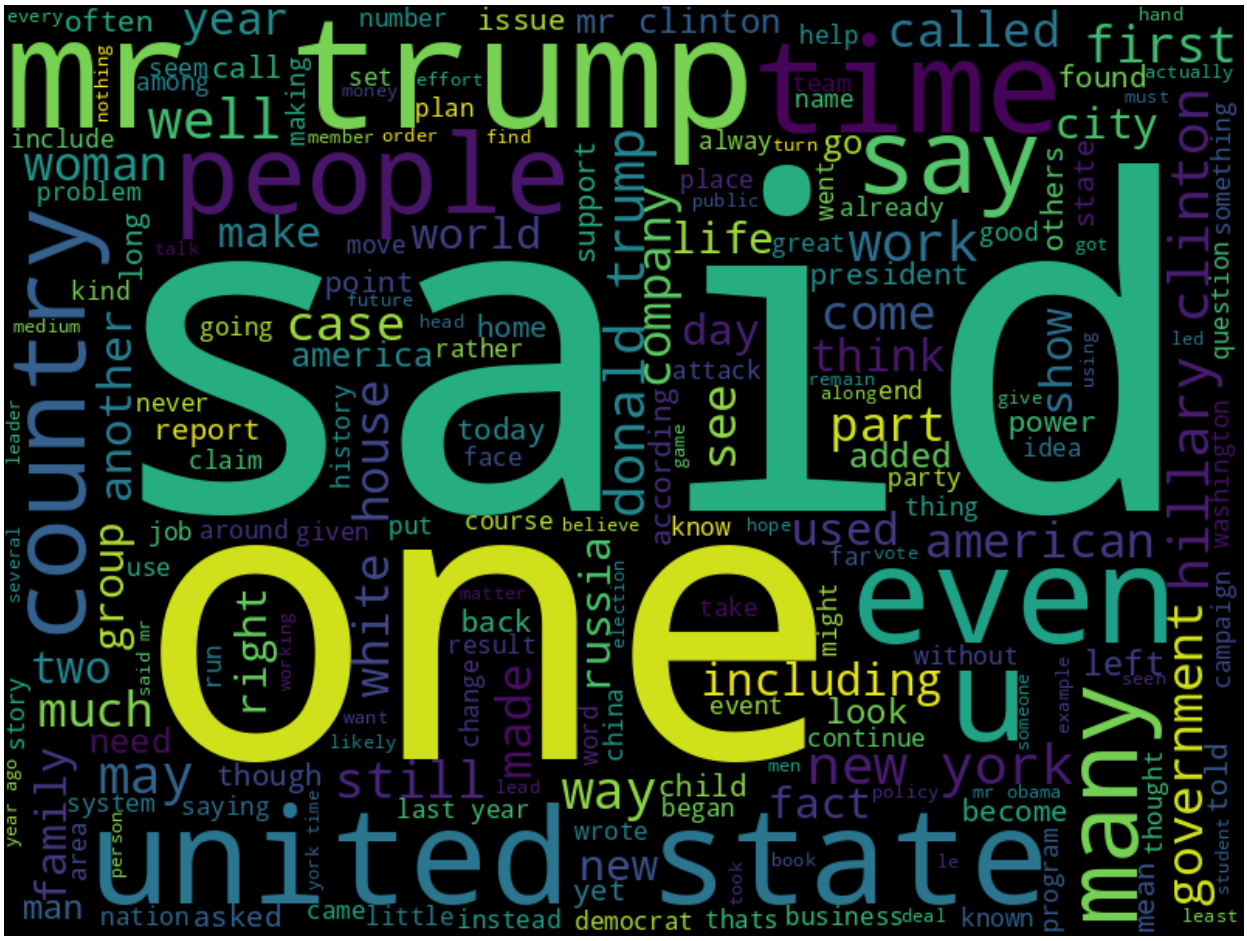
```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt

# Khởi tạo đối tượng WordCloud với các tham số
wordcloud = WordCloud( background_color='black', width=800, height=600)
# tạo ra WordCloud từ các đoạn văn bản trong cột 'text'
text_cloud = wordcloud.generate(' '.join(df['text']))
# hiển thị WordCloud đã được tạo ra
plt.figure(figsize=(20,30))
plt.imshow(text_cloud)
plt.axis('off')
plt.show()
```

Hình 27: Tạo ra đối tượng WordCloud từ các đoạn văn bản

Đoạn code trên giúp tạo ra một biểu đồ WordCloud từ các đoạn văn bản trong cột 'text', trong đó các từ xuất hiện nhiều hơn được hiển thị lớn hơn và có màu sắc khác nhau. WordCloud giúp trực quan hóa các từ quan trọng và phổ biến trong dữ liệu văn bản.

Kết quả:



Hình 28: Hiển thị các từ xuất hiện trong text

```
# sơ đồ bigram phổ biến nhất trên các tin tức
def plot_top_ngrams(corpus, title, ylabel, xlabel="Number of Occurences", n=2):
  true_b = (pd.Series(nltk.ngrams(corpus.split(), n)).value_counts())[:20]
  true_b.sort_values().plot.barh(color='blue', width=.9, figsize=(12, 8))
  plt.title(title)
  plt.ylabel(ylabel)
  plt.xlabel(xlabel)
  plt.show()

# trên các tin tức đáng tin với 2 từ
plot_top_ngrams(true_n, 'Top 20 Frequently Occuring True news Bigrams', "Bigram", n=2)
```

Hình 29: Sơ đồ bigram phổ biến nhất trên các tin tức

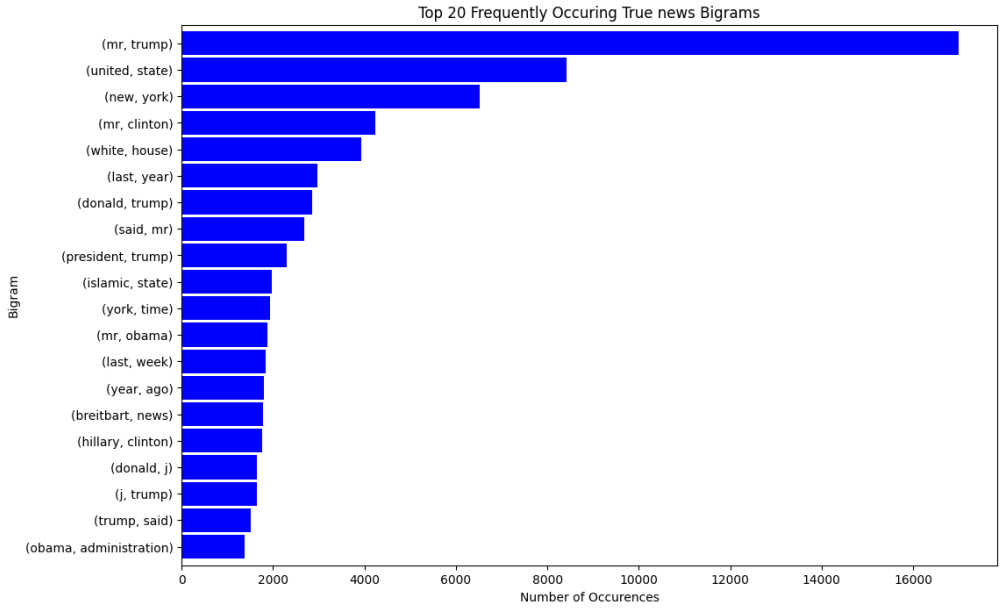
Hàm plot_top_ngrams được sử dụng để vẽ biểu đồ của các bigram phổ biến nhất trên một đoạn văn bản (corpus). Các tham số của hàm như sau:

- corpus: Đoạn văn bản đầu vào để trích xuất các bigram phổ biến.
- title: Tiêu đề của biểu đồ.
- ylabel: Nhãn trục y của biểu đồ.
- xlabel (tùy chọn): Nhãn trục x của biểu đồ, mặc định là "Number of Occurences".
- n = 2: Số từ trong một bigram, mặc định là 2 (bigram).

Hàm này thực hiện các bước sau:

- Sử dụng nltk.ngrams để trích xuất các bigram từ đoạn văn bản (corpus).
- Đếm số lần xuất hiện của các bigram bằng cách sử dụng pd.Series(nltk.ngrams(corpus.split(), n)).value_counts(). Kết quả là một Series chứa các bigram và số lần xuất hiện tương ứng.
- Chọn ra các bigram phổ biến nhất (top 20) bằng cách sử dụng phép cắt [:20].
- Sắp xếp các bigram theo thứ tự tăng dần của số lần xuất hiện bằng cách sử dụng true_b.sort_values().
- Vẽ biểu đồ cột ngang (barh) với các bigram và số lần xuất hiện tương ứng bằng cách sử dụng true_b.plot.barh(color='blue', width=.9, figsize=(12, 8)).
- Đặt tiêu đề, nhãn trục y và nhãn trục x cho biểu đồ sử dụng plt.title, plt.ylabel và plt.xlabel.
- Hiển thị biểu đồ bằng cách sử dụng plt.show().

Kết quả:



Hình 30: Top20 từ xuất hiện trong tin thật

Các bước chuẩn bị để train mô hình

```python
def set_seed(seed: int):
    """
     Hàm trợ giúp để đảm bảo tính nhất quán trong việc thiết lập seed trong các thư viện
     "random", "numpy", "torch" và/hoặc "tf" (nếu đã được cài đặt).
    Args:
        seed (:obj:`int`): The seed to set.
    """
    random.seed(seed)
    np.random.seed(seed)
    #Kiểm tra xem thư viện torch có khả dụng không
    if is_torch_available():
        torch.manual_seed(seed)
        torch.cuda.manual_seed_all(seed) # có thể gọi hàm này ngay cả khi CUDA không khả dụng
    if is_tf_available():
        import tensorflow as tf

        tf.random.set_seed(seed)

set_seed(1)
```

Hình 31: Thiết lập hàm trợ giúp

Hàm set_seed (seed: int) được sử dụng để thiết lập các giá trị hạt giống (seed) cho các thư viện "random", "numpy", "torch" và/hoặc "tf" (TensorFlow) (nếu được cài đặt) trong mục đích giữ cho quá trình ngẫu nhiên nhất quán và tái tạo được. Mục đích của việc thiết lập seed là để đảm bảo tính nhất quán trong việc sinh các số ngẫu nhiên, đặc biệt là trong quá trình huấn luyện mô hình máy học.

- Thiết lập hạt giống cho thư viện "random" bằng cách sử dụng random.seed(seed) và np.random.seed(seed).
- Kiểm tra xem thư viện "torch" có khả dụng không bằng cách sử dụng is_torch_available().
- Nếu thư viện "torch" khả dụng, thiết lập hạt giống cho "torch" bằng cách sử dụng torch.manual_seed(seed) và torch.cuda.manual_seed_all(seed). Hàm torch.manual_seed(seed) thiết lập hạt giống cho các phép tính trên CPU, trong khi torch.cuda.manual_seed_all(seed) thiết lập hạt giống cho các phép tính trên GPU (nếu có).
- Kiểm tra xem thư viện "tf" (TensorFlow) có khả dụng không bằng cách sử dụng is_tf_available().
- Nếu thư viện "tf" khả dụng, thiết lập hạt giống cho "tf" bằng cách sử dụng tf.random.set_seed(seed).

- Việc thiết lập seed giúp đảm bảo rằng các quá trình ngẫu nhiên, chẳng hạn như việc chọn ngẫu nhiên các mẫu huấn luyện, trọng số khởi tạo ngẫu nhiên của mô hình, sẽ được tái tạo và nhất quán mỗi khi bạn chạy mã. Điều này giúp bạn kiểm soát được tính ngẫu nhiên và tái tạo được kết quả.

```
# tên mô hình
model_name = "bert-base-uncased"
# độ dài tối đa mỗi câu/văn bản
max_length = 512
```

```
# chúng ta sử dụng tokenizer của BERT để chia văn bản thành các thành phần nhỏ hơn để mô hình có thể xử lý.
tokenizer = BertTokenizerFast.from_pretrained(model_name, do_lower_case=True)
```

```
Downloading (…)okenizer_config.json: 100% 28.0/28.0 [00:00<00:00, 1.49kB/s]
Downloading (…)solve/main/vocab.txt: 100% 232k/232k [00:00<00:00, 4.16MB/s]
Downloading (…)/main/tokenizer.json: 100% 466k/466k [00:00<00:00, 16.9MB/s]
Downloading (…)lve/main/config.json: 100% 570/570 [00:00<00:00, 23.9kB/s]
```

```
# chọn các hàng trong đó có cột k rỗng
news_df = news_d[news_d['text'].notna()]
news_df = news_df[news_df["author"].notna()]
news_df = news_df[news_df["title"].notna()]
```

```
def prepare_data(df, test_size=0.2, include_title=True, include_author=True):
  texts = []
  labels = []
  for i in range(len(df)):
    text = df["text"].iloc[i]
    label = df["label"].iloc[i]
    if include_title:
      text = df["title"].iloc[i] + " - " + text
    if include_author:
      text = df["author"].iloc[i] + " : " + text
    if text and label in [0, 1]:
      texts.append(text)
      labels.append(label)
  return train_test_split(texts, labels, test_size=test_size)

 # lưu trữ kết quả trả về vào các biến tương ứng
train_texts, valid_texts, train_labels, valid_labels = prepare_data(news_df)
```

Trong đoạn mã trên, chúng ta định nghĩa hàm prepare_data để chuẩn bị dữ liệu huấn luyện và dữ liệu kiểm tra từ DataFrame df

Hàm prepare_data sẽ lặp qua từng hàng của DataFrame và lấy văn bản và nhãn tương ứng. Nếu include_title là True, tiêu đề sẽ được thêm vào trước văn bản. Tương tự, nếu include_author là True, tác giả sẽ được thêm vào trước văn bản. Chỉ các văn bản có nội dung và nhãn thuộc [0, 1] sẽ được thêm vào danh sách texts và labels.

Cuối cùng, hàm train_test_split từ thư viện sklearn.model_selection được sử dụng để chia dữ liệu thành hai phần: train_texts và train_labels cho huấn luyện, và valid_texts và valid_labels cho kiểm tra. Tỷ lệ chia được xác định bởi test_size.

Kết quả của hàm prepare_data là bốn biến: train_texts, valid_texts, train_labels, và valid_labels, lưu trữ dữ liệu huấn luyện và kiểm tra tương ứng.

```
# ta sử dụng tokenizer để mã hóa các văn bản
# văn bản sẽ bị cắt bớt hoặc đệm bằng số 0 (zero-padding) nếu lớn hơn/nhỏ hơn max_length
train_encodings = tokenizer(train_texts, truncation=True, padding=True, max_length=max_length)
valid_encodings = tokenizer(valid_texts, truncation=True, padding=True, max_length=max_length)
```

Sử dụng phương thức tokenizer để mã hóa train_texts bằng cách truyền các đối số sau:

- truncation=True: Đảm bảo rằng các văn bản sẽ được cắt bớt nếu dài hơn max_length.
- padding=True: Đảm bảo rằng các văn bản ngắn hơn max_length sẽ được đệm bằng số 0 (zero-padding) để có cùng độ dài.
- max_length: Độ dài tối đa của các văn bản sau khi mã hóa.

Tương tự, chúng ta mã hóa valid_texts bằng cách truyền cùng các đối số vào phương thức tokenizer.

Kết quả là hai đối tượng train_encodings và valid_encodings, lưu trữ các mã hóa của các văn bản huấn luyện và kiểm tra tương ứng.

```
# chuyển đổi dữ liệu thành tập dữ liệu PyTorch
class NewsGroupsDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {k: torch.tensor(v[idx]) for k, v in self.encodings.items()}
        item["labels"] = torch.tensor([self.labels[idx]])
        return item

    def __len__(self):
        return len(self.labels)

# chúng ta sử dụng lớp NewsGroupsDataset để tạo ra các tập dữ liệu huấn luyện và tập dữ liệu xác thực
train_dataset = NewsGroupsDataset(train_encodings, train_labels)
valid_dataset = NewsGroupsDataset(valid_encodings, valid_labels)
```

```
# tạo một mô hình mới từ mô hình được chỉ định
model = BertForSequenceClassification.from_pretrained(model_name, num_labels=2)
```

Định nghĩa lớp NewsGroupsDataset là một lớp con của torch.utils.data.Dataset. Lớp này sẽ chuyển đổi các dữ liệu đã được mã hóa (encodings) và nhãn (labels) thành một tập dữ liệu PyTorch có thể sử dụng cho việc huấn luyện và kiểm tra.

Lớp NewsGroupsDataset có các phương thức sau:

__init__(self, encodings, labels): Khởi tạo đối tượng NewsGroupsDataset với encodings là mã hóa các văn bản và labels là nhãn tương ứng.

__getitem__(self, idx): Trả về một mục dữ liệu tại vị trí idx. Đối tượng được trả về là một từ điển có khóa k là tên của thuộc tính và giá trị v là tensor PyTorch tương ứng với mã hóa của thuộc tính đó. Nhãn cũng được chuyển thành một tensor PyTorch và có khóa labels.

__len__(self): Trả về số lượng mục dữ liệu trong tập dữ liệu.

Sau khi định nghĩa lớp NewsGroupsDataset, tạo ra hai đối tượng train_dataset và valid_dataset tương ứng với tập dữ liệu huấn luyện và tập dữ liệu xác thực bằng cách truyền train_encodings và valid_encodings cùng với train_labels và valid_labels vào lớp NewsGroupsDataset.

model = BertForSequenceClassification.from_pretrained(model_name, num_labels=2) Sử dụng phương thức from_pretrained() của lớp BertForSequenceClassification để tạo một mô hình mới từ mô hình đã được chỉ định (được lưu trữ trong model_name).

Tham số num_labels=2 được sử dụng để chỉ định số lượng nhãn trong bài toán phân loại. Trong trường hợp này, chúng ta đang làm việc với bài toán phân loại giữa hai nhãn: tin tức đáng tin cậy và tin tức giả mạo.

```python
from sklearn.metrics import accuracy_score
# để tính toán các độ đo (metrics) cho mô hình dự đoán.
def compute_metrics(pred):
  labels = pred.label_ids # Trích xuất các nhãn
  preds = pred.predictions.argmax(-1) #  Trích xuất các dự đoán của mô hình
  # tính toán độ chính xác (accuracy) giữa nhãn thực tế (labels) và dự đoán của mô hình (preds).
  acc = accuracy_score(labels, preds)
  #Trả về một từ điển chứa kết quả tính toán độ chính xác
  return {
      'accuracy': acc,
  }
```

Định nghĩa hàm compute_metrics để tính toán các độ đo (metrics) cho mô hình dự đoán.

- Trích xuất các nhãn thực tế từ đối tượng dự đoán (pred.label_ids).
- Trích xuất các dự đoán của mô hình bằng cách chọn lớp có giá trị dự đoán cao nhất (pred.predictions.argmax(-1)).
- Sử dụng hàm accuracy_score từ thư viện sklearn.metrics để tính toán độ chính xác giữa nhãn thực tế và dự đoán của mô hình.
- Trả về một từ điển chứa kết quả tính toán độ chính xác.

```python
training_args = TrainingArguments(
output_dir='./results',
num_train_epochs=1,              # tổng số epoch huấn luyện
per_device_train_batch_size=10, # kích thước batch cho mỗi thiết bị trong quá trình huấn luyện
per_device_eval_batch_size=20,
warmup_steps=100,                # số lượng bước khởi đầu để điều chỉnh tốc độ học
logging_dir='./logs',            # thư mục lưu trữ các file log
load_best_model_at_end=True,     # tải mô hình tốt nhất khi hoàn thành huấn luyện (độ đo mặc định là loss)
# tuy nhiên, bạn có thể chỉ định đối số metric_for_best_model để thay đổi thành độ chính xác hoặc độ đo khác
logging_steps=200,               # ghi log và lưu trọng số sau mỗi logging_steps bước
save_steps=200,
evaluation_strategy="steps",     # đánh giá sau mỗi logging_steps bước
```

- output_dir: Đường dẫn đến thư mục lưu trữ các kết quả huấn luyện, bao gồm mô hình và các file liên quan.
- num_train_epochs: Số lượng epoch (vòng lặp) trong quá trình huấn luyện.
- per_device_train_batch_size: Kích thước batch (số lượng mẫu đầu vào) cho mỗi thiết bị trong quá trình huấn luyện.
- per_device_eval_batch_size: Kích thước batch cho mỗi thiết bị trong quá trình đánh giá.
- warmup_steps: Số lượng bước khởi đầu để điều chỉnh tốc độ học.
- logging_dir: Đường dẫn đến thư mục lưu trữ các file log.

- load_best_model_at_end: Tải mô hình tốt nhất khi hoàn thành huấn luyện (sử dụng độ đo mặc định là loss).
- logging_steps: Ghi log và lưu trọng số sau mỗi số bước nhất định.
- save_steps: Lưu trọng số sau mỗi số bước nhất định.
- evaluation_strategy: Chiến lược đánh giá sau mỗi số bước nhất định.

Train mô hình

```
trainer = Trainer(
    model=model,                         # mô hình Transformers đã được khởi tạo để được huấn luyện
    args=training_args,                  # các tham số huấn luyện, đã được định nghĩa ở trên
    train_dataset=train_dataset,         # tập dữ liệu huấn luyện
    eval_dataset=valid_dataset,          # tập dữ liệu đánh giá
    compute_metrics=compute_metrics, # hàm tính các độ đo quan trọng
)
```

```
# train mô hình
trainer.train()
# đánh giá mô hình trên tập dữ liệu đánh giá
trainer.evaluate()
```

Lưu lại mô hình

```
 # lưu lại model & tokenizer
 model_path = "fake-news-bert-base-uncased"
 model.save_pretrained(model_path)
 tokenizer.save_pretrained(model_path)
```

Hàm get_prediction để thực hiện dự đoán trên mô hình đã huấn luyện.

```
def get_prediction(text, convert_to_label=False):
    # Chuẩn bị văn bản thành chuỗi token hóa
    inputs = tokenizer(text, padding=True, truncation=True, max_length=512, return_tensors="pt").to("cpu")
    # Thực hiện dự đoán trên mô hình
    outputs = model(**inputs)
    # Lấy xác suất đầu ra bằng cách sử dụng softmax
    probs = torch.softmax(outputs.logits, dim=1)
    # Thực hiện argmax để lấy nhãn dự đoán
    d = {
        0: "reliable",
        1: "fake"
    }
    if convert_to_label:
        return d[int(probs.argmax())]
    else:
        return int(probs.argmax())
```

Ví dụ:

```
news="""Trump is USA's antique hero. Clinton will be next president"""
get_prediction(news,convert_to_label=True)
```

```
'fake'
```

Hình 32: Dự đoán một thông tin sai

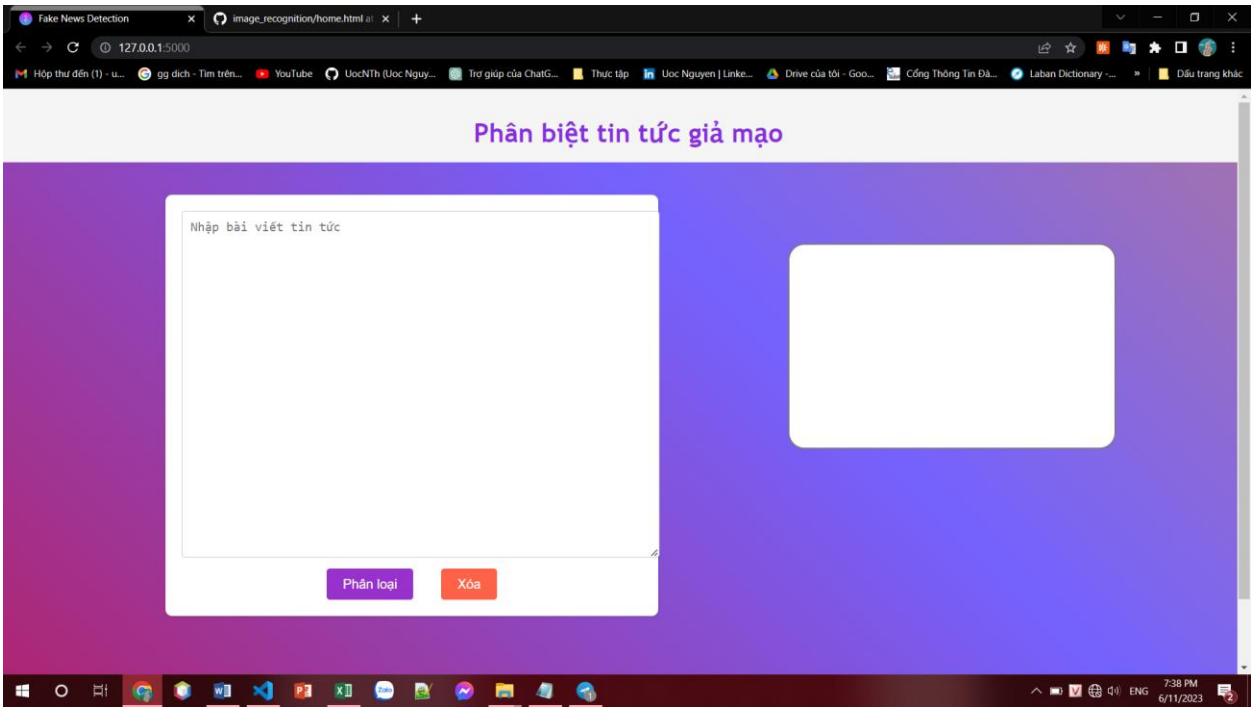
```
get_prediction("""Weekly Featured Profile – Randy Shannon",Trevor Loudon,"You are here: Home / *Articles of
the Bound* / Weekly Featured Profile – Randy Shannon Weekly Featured Profile – Randy Shannon
October 31, 2016, 7:21 am by Trevor Loudon Leave a Comment 0
KeyWiki.org Randy Shannon
Randy Shannon is a Beaver County , Pennsylvania Democratic Party activist. “A Democratic victory
in 2016 with a bigger progressive caucus can tax Wall Street, end austerity and discrimination,
and put the nation to work building the solar infrastructure we desperately need.”
“We need progressives like Sanders, who support working families, running for President,
for Senate, and for Congress wherever possible,” said Randy Shannon , convener of the Sanders for President PA Exploratory Committee.
Randy Shannon was a student leader in the 1960’s at Duke University.""",convert_to_label=True)
```

```
'fake'
```

Hình 33: Dự đoán một thông tin sai

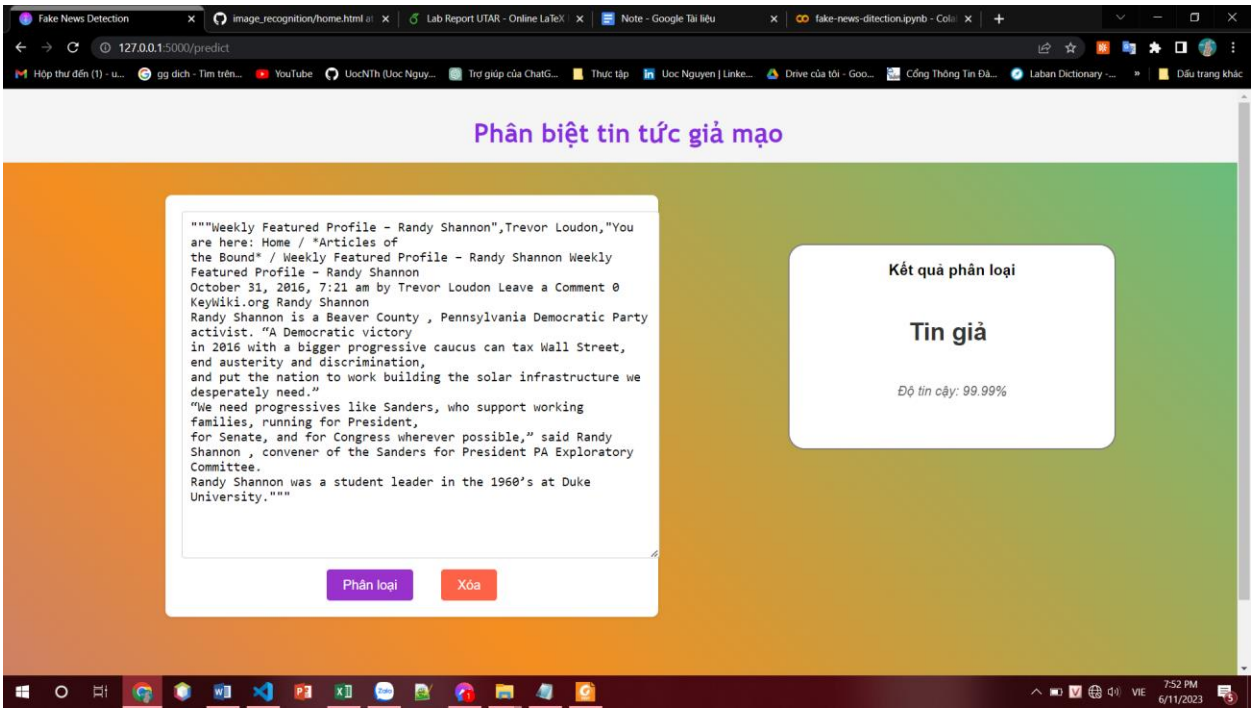
VII. Demo Web

Giao diện của Web



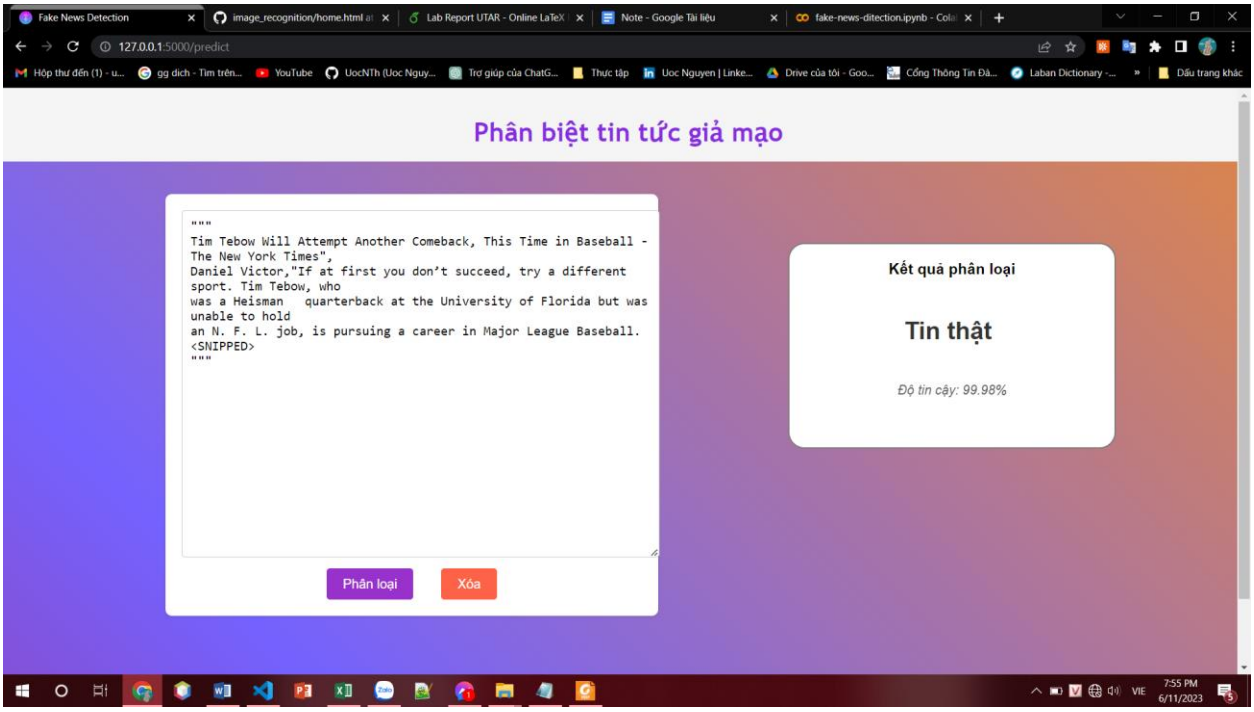
Hình 34: Giao diện của Web

Giả sử khi nhập một tin tức giả và ấn phân loại thì web sẽ trả về như hình ảnh sau:



Hình 35: Web trả về kết quả khi dự đoán một tin giả

Khi nhập vào một tin tức đáng tin cậy thì Web sẽ trả về thông báo như hình sau:



Hình 36: Web trả về kết quả khi dự đoán một tin thật

Tài liệu tham khảo

[1] Attention Is All You Need, 2017

[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019

[3] Dataset Fake News Detection