**MỤC LỤC**

[DANH MỤC HÌNH VẼ 3](#_Toc76055443)

[CHỮ VIẾT TẮT VÀ THUẬT NGỮ CHUYÊN NGÀNH 4](#_Toc76055444)

[LỜI NÓI ĐẦU 6](#_Toc76055445)

[CHƯƠNG 1: CÔNG VIỆC CÓ LIÊN QUAN 7](#_Toc76055446)

[**1.** **Phương pháp tiếp cận dựa trên tính năng không được giám sát ( Unsupervised Feature-based Approaches)** 7](#_Toc76055447)

[**2.** **Chuyển giao việc học dữ liệu được giám sát** 7](#_Toc76055448)

[**3.** **Phương pháp tiếp cận dựa trên tinh chỉnh không giám sát** 7](#_Toc76055449)

[CHƯƠNG 2: TRANSFORMER 9](#_Toc76055450)

[**1.** **Giới thiệu chung về Transformer architect** 9](#_Toc76055451)

[**2.** **Tổng quan về mô hình** 9](#_Toc76055452)

[**3.** **Kiến trúc mô hình Transformer** 10](#_Toc76055453)

[**3.1** **Embeding Layer với Position Encoding** 12](#_Toc76055454)

[**3.2** **Encoder** 14](#_Toc76055455)

[**3.3** **Decoder** 17](#_Toc76055456)

[CHƯƠNG 3: BERT 21](#_Toc76055457)

[**1.** **Giới thiệu về Bert model** 21](#_Toc76055458)

[**2.** **Kiến trúc của BERT** 22](#_Toc76055459)

[**2.1 Fine-tuning BERT ( tinh chỉnh BERT)** 23](#_Toc76055460)

[**2.2 Masked Language Model (MLM)** 24](#_Toc76055461)

[**2.3** **Next Sentence Prediction (NSP)** 25](#_Toc76055462)

[CHƯƠNG 4: BÀI TOÁN 30](#_Toc76055463)

[**1.** **Tổng qua về hệ thống Question Answering** 30](#_Toc76055464)

[**2.** **Bài toán “ Xây dựng hệ thống trả lời câu hỏi luật”** 30](#_Toc76055465)

[**2.1** **Các nghiên cứu trước** 30](#_Toc76055466)

[**2.2** **Xây dựng hệ thống trả lời câu hỏi luật ( BERT)** 30](#_Toc76055467)

[**3.** **Xây dựng chương trình** 34](#_Toc76055468)

[**3.1 Chuẩn bị dữ liệu** 34](#_Toc76055469)

[**3.2 Xây dựng chương trình** 35](#_Toc76055470)

[CHƯƠNG 5: PHÂN CÔNG CÔNG VIỆC NHÓM 10 39](#_Toc76055471)

[Tài liệu tham khảo 39](#_Toc76055472)

# DANH MỤC HÌNH VẼ

[Hình ảnh 1. Mô hình kiến trúc Transformer 10](#_Toc67596856)

[Hình ảnh 2. Hai mô hình kiến trúc BERT và LSTM 10](#_Toc67596857)

[Hình ảnh 3 Kiến trúc mô hình Transformer 11](#_Toc67596858)

[Hình ảnh 4 Kiến trúc chi tiết các lớp của mô hình transformer 12](#_Toc67596859)

[Hình ảnh 5. Word embedding cho đầu vào lớp encoder 13](#_Toc67596860)

[Hình ảnh 6. Minh hoạ cách tính position encoding 13](#_Toc67596861)

[Hình ảnh 7. Cấu trúc của Encoder 14](#_Toc67596862)

[Hình ảnh 8. Tính toán self-attention 15](#_Toc67596863)

[Hình ảnh 9. Tinh hoạ công thức tính 16](#_Toc67596864)

[Hình ảnh 10. Tính toán multi-head attention 17](#_Toc67596865)

[Hình ảnh 11. Minh hoạ cấu trúc decoder 18](#_Toc67596866)

[Hình ảnh 12. Cách tạo input mask trước khi đưa vào decoder 19](#_Toc67596867)

[Hình ảnh 13. Minh hoạ Encoder-decoder attention 19](#_Toc67596868)

[Hình ảnh 14. Neural của BERT 22](#_Toc67596869)

[Hình ảnh 15. Neural của mô hình BERT, OpenAI GPT, ELMo 22](#_Toc67596870)

[Hình ảnh 16. Toàn bộ tiến trình pre-traning và fine-tuning của BERT. Một kiến trúc tương tự được sử dụng cho cả pretrain-model và fine-tuning model. 23](#_Toc67596871)

[Hình ảnh 17. Sơ đồ kiến trúc BERT cho tác vụ Masked ML 24](#_Toc67596872)

[Hình ảnh 18. Sơ đồ kiến trúc BERT cho tác vụ NSP 26](#_Toc67596873)

[Hình ảnh 19. Minh hoạ cách tính vector đầu vào cho BERT 27](#_Toc67596874)

[Hình ảnh 20. Tiến trình của hệ thốn” hỏi đáp văn bản luật” 31](#_Toc67596875)

# CHỮ VIẾT TẮT VÀ THUẬT NGỮ CHUYÊN NGÀNH

Attention:  Thuật toán Attention trong lĩnh vực Deep learning cho dạng bài toán Sequence to sequence.

Self\_attention: Kĩ thuật tạo ra ma trận với hàng và cột là cùng một câu để hiểu những phần nào của câu sẽ liên quan đến nhau.

Softmax:  Một phương pháp classification.

RNN (Recurrent Neural Network): Mạng nơ-ron hồi quy là một thuật toán được chú ý rất nhiều trong thời gian gần đây bởi các kết quả tốt thu được trong lĩnh vực xử lý ngôn ngữ tự nhiên

LSTM (Long Short Term Memory): Một dạng đặc biệt của mạng neuron hồi quy

GRU (Gated Recurrent Unit): Mô hình tương tự như LSTM.

Encoder: Là phrase chuyển input thành những features learning có khả năng học tập các task.

Decoder: Đầu ra của encoder chính là đầu vào của các Decoder. Phrase này nhằm mục đích tìm ra phân phối xác xuất từ các features learning ở Encoder từ đó xác định đâu là nhãn của đầu ra.

Bidirectional LSTM: Mạng thần kinh tái phát hai chiều kết nối hai lớp ẩn của các hướng ngược nhau với cùng một đầu ra.

Layer: Lớp

Feed forward neural net: Dữ liệu được truyền thẳng từ Input vào trong mạng. Trong mạng này thì không có feedback connections cũng như loop trong mạng.

Embedding: Nhúng input vào model đã đào tạo

Word embedding: Từ được nhúng mạng nơ ron

Position embedding: Vị trí được nhúng vào mạng nơ ron

Position encoding: Phương pháp thêm vào vector đầu ra của Embedding để tạo ra tính tuần tự cho câu nhưng vẫn sử dụng Self-Attention được.

Multi head attention: Sub-layer thứ nhất của Transformer

Cross entropy: Đại lượng đánh giá sự khác nhau (hay khoảng cách) giữa hai phân bố xác suất (probability distributions).

Residuals connection: Là một mạng CNN được thiết kế để làm việc với hàng trăm hoặc hàng nghìn lớp chập.

Normalixation layer: Là một mạng CNN được thiết kế để làm việc với hàng trăm hoặc hàng nghìn lớp chập.

# LỜI NÓI ĐẦU

Tai nạn giao thông 2 tháng đầu năm 2021: 2.355 vụ, 1.230 người chết. Để lý giải cho con số khổng lồ này thì có rất nhiều lí do nhưng theo tôi lí do quan trọng nhất chủ yếu là do người dân chưa chấp hành đúng luật giao thông. Ở Việt Nam là một quốc gia có mật độ dân số cao, số lượng phương tiện lưu thông lớn dẫn đến việc quản lý của các cơ quan chức năng còn gặp nhiều khó khan. Do đó việc duy trì trật tự giao thông chủ yêu phụ thuộc vào ý thức và hiểu biết của người tham gia giao thông.

Mặc dù vậy nhưng phần lớn người tham gia giao thông không hoàn toàn nắm vững luật giao thông đặc biệt các quy định về xử phạt và các quy định mới. Đặc biệt sau khi vi phạm thì phần lớn mọi người sẽ đi tìm các nguồn thông tin về quy định xử phạt nhưng những thông tin trên mạng lại quá nhiều và có cả các thông tin về các quy định cũ dẫn đến một số người cho là mình rất hiểu luật nhưng lại vi phạm chồng vi phạm thậm trí chống đối cơ quan chức năng.

Nhận thấy được sự bất cập này nên nhóm chúng tôi đã quyết định xây dựng hệ thống nhằm mục đích cung cấp cho người dân một kênh thông tin mới để có thể tra cứu nhanh các câu hỏi về luật giao thông, xác định các nội dung trong quy định xử phạt liên quan đến hành vi vi phạm của bản thân.

Đề tài xây dựng hệ thống hỏi đáp luật giao thông của chúng tôi được xây dựng trên nền tảng chính là ngôn ngữ tự nhiên NPLvà mô hình nhúng từ của bert. Đề tài chỉ giới hạn trong việc hỏi đáp về luật giao thông đường bộ và quy định xử phạt đối với luật giao thông đường bộ.

Đề tài này đưa ra được sản phẩm có thể sử dụng tuy nhiên độ chính xác còn chưa cao. Vì vậy, trong tương lai chúng tôi sẽ ứng dụng thêm nhiều công cụ cũng như nâng cao bộ dữ liệu để năng cao độ chính xác cho sản phẩm.

Nhóm tác giả xin gửi lời cảm ơn đến *Thiếu tá.TS* ***Phan Việt Anh*** là những người hướng dẫn chính trong đề tài này.đề. Cuối cùng, chúng tôi xin cảm ơn tới các đảng ủy, chỉ huy đơn vị đại đội, tiểu đoàn, các ban ngành đã hỗ trợ chúng tôi để chúng tôi có thể hoàn thiện được đề tài này tốt nhất.

Trong quá trình xây dựng đề tài còn nhiều sai sót, mong độc giả đóng góp ý kiến để chúng tôi có thể cải thiện đề tài này tốt hơn.

# CHƯƠNG 1: CÔNG VIỆC CÓ LIÊN QUAN

1. **Phương pháp tiếp cận dựa trên tính năng không được giám sát ( Unsupervised Feature-based Approaches)**

Học các cách trình bày áp dụng rộng rãi của từ đã là một lĩnh vực nghiên cứu tích cực cho nhiều thập kỷ, bao gồm cả phi thần kinh và các phươn pháp thần kinh. Nhúng từ được đào tạo trước là một phần không thể thiếu của các hệ thống NLP hiện đại, mang đến những cải tiến đáng kể so với việc nhúng học từ đầu. Để đào tạo trước các vectơ nhúng từ, các mục tiêu mô hình ngôn ngữ từ trái sang phải đã được sử dụng, cũng như các mục tiêu để phân biệ từ đúng với các từ không chính xác ở bên trái và đúng ngữ cảnh. Những cách tiếp cận này đã được khái quát hóa thành chi tiết thô hơn, chẳng hạn như nhúng câu hoặc nhúng đoạn văn. Để đào tạo cách biểu diễn câu, trước công việc đã sử dụng các mục tiêu để xếp hạng ứng viên tiếp theo câu, thế hệ từ trái sang phải của các từ câu tiếp theo được đưa ra đại diện cho câu trước câu hoặc biểu thị các mục tiêu dẫn xuất của trình mã tự động. ELMo và người tiền nhiệm của nó khái quát hóa nghiên cứu nhúng từ truyền thống theo một khía cạnh khác. Họ trích xuất các tính năng nhạy cảm theo ngữ cảnh từ trái sang phải và mô hình ngôn ngữ từ phải sang trái. Biểu diễn theo ngữ cảnh của mỗi mã thông báo là sự kết hợp của các biểu diễn từ trái sang phải và từ phải sang trái. Khi tích hợp nhúng từ ngữ theo ngữ cảnh với các kiến ​​trúc dành riêng cho nhiệm vụ hiện có, ELMo nâng cao trạng thái của nghệ thuật cho một số NLP chính điểm chuẩn bao gồm trả lời câu hỏi, cảm xúc phân tích và thực thể được đặt tên sự công nhận . Melamud và cộng sự. (2016) đề xuất học tập biểu diễn theo ngữ cảnh thông qua nhiệm vụ dự đoán một từ duy nhất từ ​​cả ngữ cảnh bên trái và bên phải sử dụng LSTM. Tương tự như ELMo, mô hình của họ là dựa trên tính năng và không sâu về hai chiều. Feduset al. (2018) cho thấy rằng tác vụ cloze có thể được sử dụng để cải thiện tính mạnh mẽ của các mô hình tạo văn bản.

1. **Chuyển giao việc học dữ liệu được giám sát**

Cũng có những công việc cho thấy sự chuyển giao hiệu quả từ các nhiệm vụ được giám sát với các tập dữ liệu lớn chẳng hạn như suy luận ngôn ngữ tự nhiên và dịch máy. Nghiên cứu thị giác máy tính cũng đã chứng minh tầm quan trọng của việc học chuyển giao từ các mô hình lớn được đào tạo trước, nơi một công thức hiệu quả là tinh chỉnh các mô hình được đào tạo trước ImageNet.

1. **Phương pháp tiếp cận dựa trên tinh chỉnh không giám sát**

Đối với các phương pháp tiếp cận dựa trên tính năng, cách đầu tiên hoạt động theo hướng này chỉ là tham số nhúng từ được đào tạo trước từ văn bản không được gắn nhãn. Gần đây hơn, bộ mã hoá câu hoặc tài liệu tạo ra các biểu diễn mã thông báo theo ngữ cảnh đã được đào tạo trước từ văn bản không được gắn nhãn và được tinh chỉnh cho tác vụ hạ nguồn được giám sát. Ưu điểm của những cách tiếp cận này đó là một vài tham số cần phải học từ vết trầy. Ít nhất một phần lợi thế này, OpenAI GPT đã đạt được các kết quả hiện đại trước đây trên nhiều nhiệm vụ cấp độ giám sát từ điểm chuẩn GLUE. Các mục tiêu mô hình hoá ngôn ngữ từ trái sang phải và bộ mã hoá tự động đã được sử dụng để đào tạo trước các mô hình như vậy

# CHƯƠNG 2: TRANSFORMER

1. **Giới thiệu chung về Transformer architect**

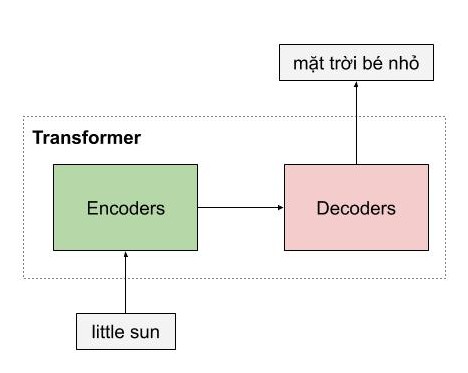
Transformer architect là một kiến trúc học sâu được giới thiệu vào năm 2017 mà sử dụng cơ chế attention. Đây là mô hình hiện đại nhất (state of the art) tính đến thời điểm hiện tại. Mô hình được sử dụng chủ yếu trong lĩnh vực xử lý ngôn ngữ tự nhiên(NLP) , những nghiên cứu gần đây cũng đã phát tiển ứng dụng của nó trong các tác vụ khác như video understanding. Giống như RNN , Transformer được thiết kế xử dụng dữ liệu tuần tự như ngôn ngữ tự nhiên cho các tác vụ như dịch, tóm tắt văn bản. Tuy nhiên ,không giống như RNN, Transformer không yêu cầu dữ liệu tuần tự phải được xử lý theo thứ tự. Ví dụ, nếu dữ liệu đầu vào là một câu ngôn ngữ tự nhiên, Transformer không cần xử lý phần đầu của nó trước phần cuối mà nó sử dụng cơ chế huấn luyện song song và do đó giảm thời gian đào tào so với RNN.

Trước khi ra đời Transformer, hầu hết các hệ thống NLP đều dựa trên “recurrent neural networks( RNN)” như LSTM, GRU với các cơ chế “attention” được bổ sung thêm vào. Transformer được xây dựng dựa trên các công nghệ “attention” mà không sử dụng cấu trúc RNN, làm nổi bật thực tế là chỉ các cơ chế “attention”, không cần xử lý tuần tự lặp lại, đủ mạnh để đạt được hiệu suất của RNN có bổ sung “attention”

1. **Tổng quan về mô hình**

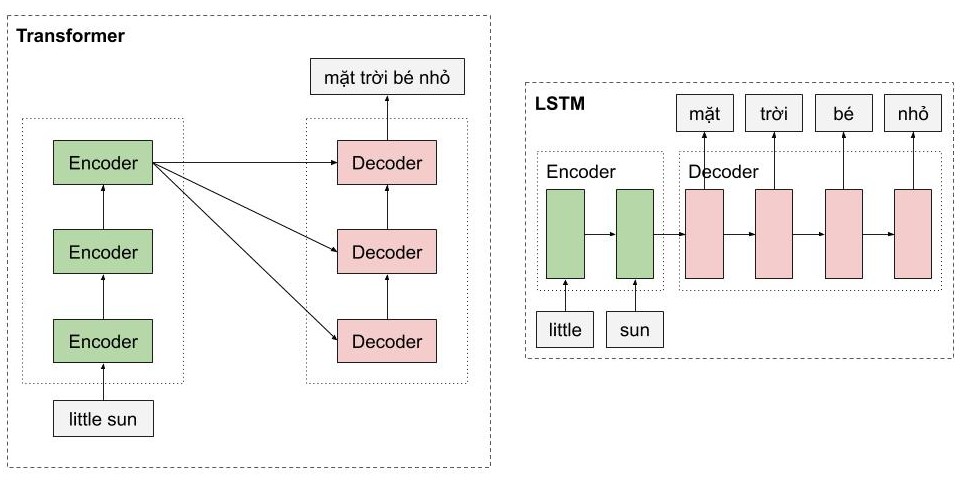
Giống như những mô hình dịch máy khác, kiến trúc tổng quan của mô hình Transformer bao gồm 2 phần lớn là encoder và decoder. Encoder dùng để học vector biểu của câu với mong muốn rằng vector này mang thông tin hoàn hảo của câu đó. Decoder thực hiện chức năng chuyển vector biểu diễn kia thành ngôn ngữ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

Trong ví dụ dưới đây, encoder của mô hình Transformer nhận một câu tiếng Anh, và encode thành một vector biểu diễn ngữ nghĩa của câu “little sun”, sau đó mô hình decoder nhận vector biểu diễn này, và dịch nó thành câu tiến việt “mặt trời bé nhỏ”



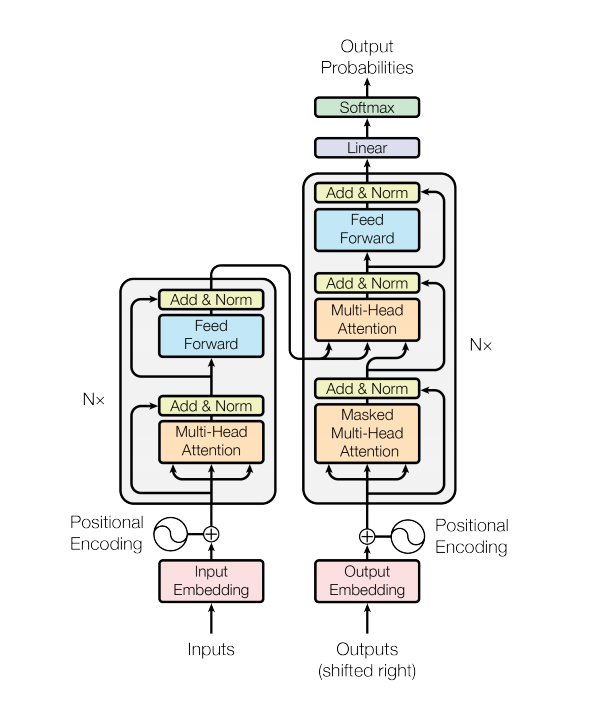
Hình ảnh 1. Mô hình kiến trúc Transformer

Một trong những ưu điểm của Transformer là mô hình này có khả năng xử lý song song cho các từ. Encoders của mô hình Transformer là một dạng feedforward neural nets, bao gồm nhiều encoder layer khác, mỗi encoder layer này xử lý đồng thời các từ. Trong khi đó, với mô hình LSTM, thì các từ được xử lý tuần tự, Ngoài ta, mô hình Transformer còn xử lý câu đầu vào theo hai hướng mà không cần phải stack thêm một hình LSTM nữa như trong kiến trúc Bidirectional LSTM

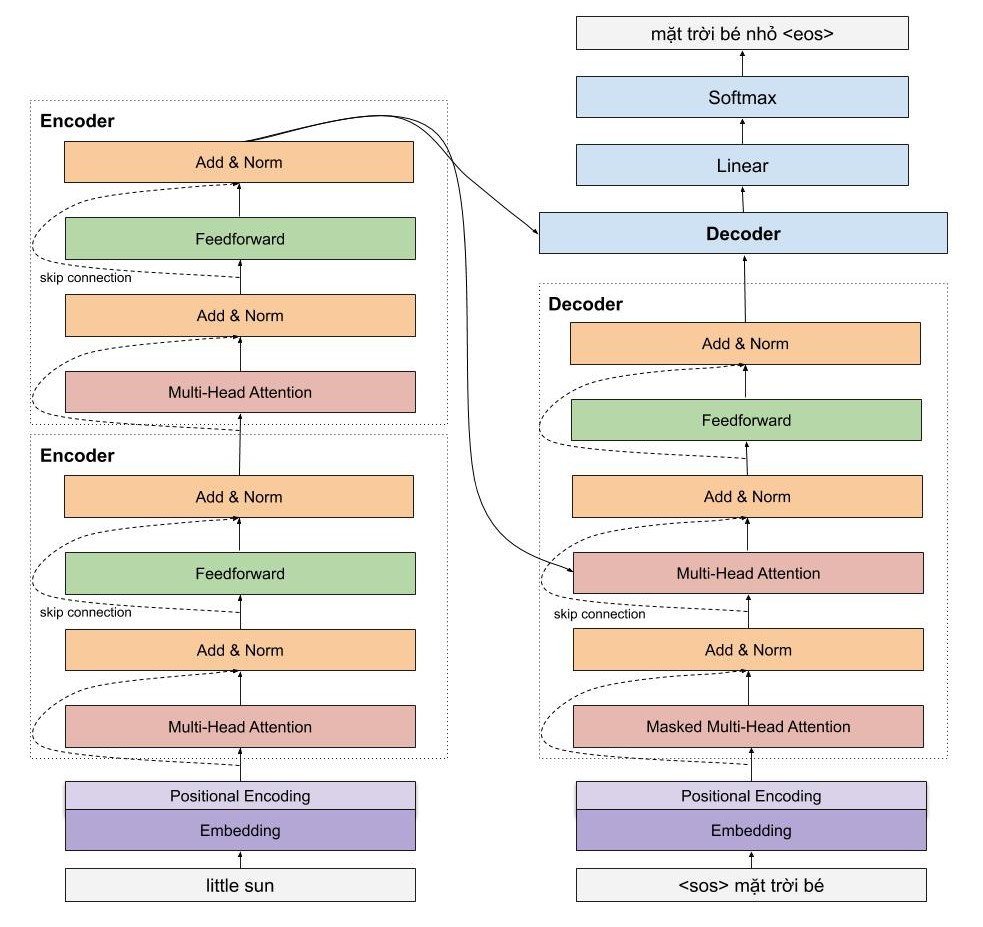


Hình ảnh 2. Hai mô hình kiến trúc BERT và LSTM

1. **Kiến trúc mô hình Transformer**



Hình ảnh 3 Kiến trúc mô hình Transformer

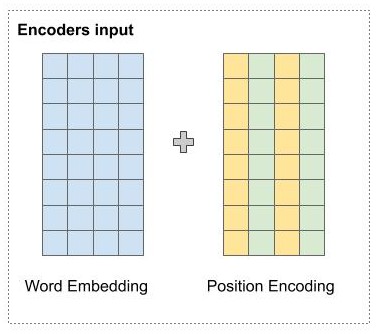


Hình ảnh 4 Kiến trúc chi tiết các lớp của mô hình transformer

* 1. **Embeding Layer với Position Encoding**

Đầu tiên, các từ được biểu diễn bằng một vector sử dụng một ma trận word embedding có số dòng bằng kích thước của tập từ vựng. Sau đó các từ trong câu được tìm kiếm trong ma trận này và được nối với nhau thành các dòng của một ma trận 2 chiều chứ ngữ nghĩa của từ riêng biệt. Transformer sử dụng xử lý các từ song song, do đó , với chỉ word embedding mô hình không thể nào biết được vị trí các từ. Như vậy, chúng cần bổ sung thêm thông tin về vị trí các từ vào trong vector đầu vào. Đó là lúc positional encoding xuất hiện và giải quyết vấn đề của chúng ta.

Vị trí của các từ được mã hoá bằng một vector có kích thước bằng word embedding và được cộng trực tiếp vào word embedding

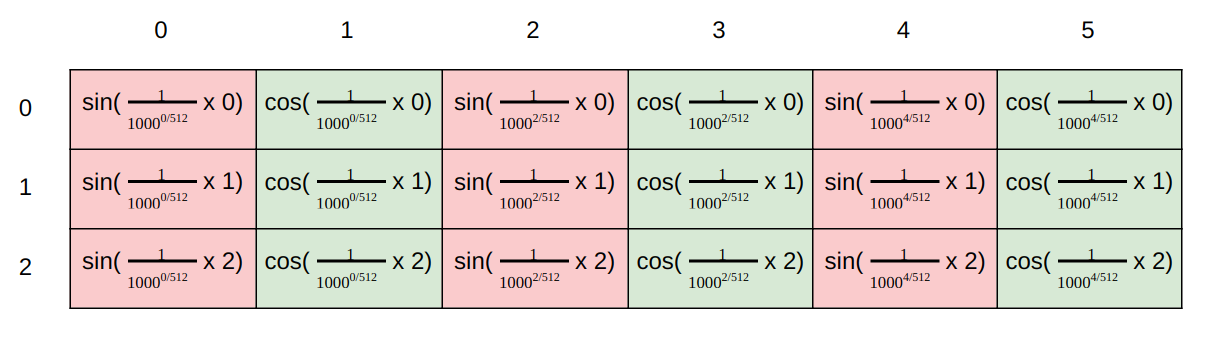


Hình ảnh 5. Word embedding cho đầu vào lớp encoder

Cụ thể, tại vị trí chẵn, ta sử dụng hàm sin và với vị trí lẻ ta sử dụng hàm cos để tính giá trị tại chiều đó

Trong đó

Trong hình dưới đây minh hoạ cho cách tính position encoding. Giả sử chúng ta có word embedding 6 chiều, thì position encoding cũng tương ứng là 6 chiều. Mỗi dòng tương ứng với 1 từ. Giá trị của các vector tại mỗi vị trí được tính toán theo công thức dưới đây

Hình ảnh 6. Minh hoạ cách tính position encoding

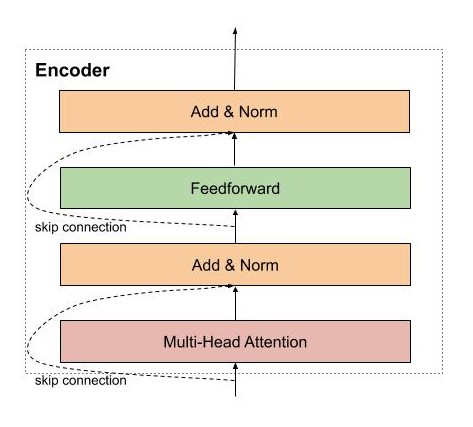
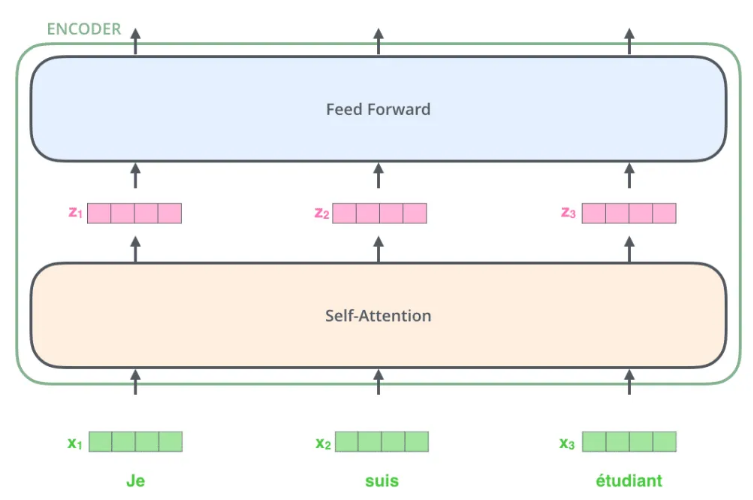
Do đó nó cho phép mô hình dễ dàng học được mối quan hệ tương đối giữa các từ. Cụ thể , biểu diễn vị trí của từ t + offset có thể chuyển thành biểu diễn vị trí của từ bằng một phép biến đổi tuyến tính dựa trên ma trận phép quay

* 1. **Encoder**

Encoder của mô hình Transformer có thể bao gồm nhiều encoder layer tương tự nhau. Mỗi encoder layer của Transformer lại bao gồm 2 thành phần chính là multi head attention và feedforward network, ngoài ra còn có cả skip connection và normalization layer

Trong 2 thành phần chính này, cần chú ý nhiều hơn đến multi-head attention vì đó là một layer mới , và nó tạo nên sự khác biệt giữa mô hình LSTM và Transformer

Encoder bao gồm một chồng N=6 lớp giống nhau. Mỗi lớp có 2 lớp con. Đầu tiên là cơ chế self-attention và thứ 2 là một mạng chuyển tiép kết nối đầy đủ theo vị trí ( feed-forward network). Chúng sử dụng một kết nối xung quanh mỗi 2 lớp con, sau đó là chuẩn hoá lớp. Nghĩa là đầu ra của mỗi lớp con là LayerNorm(x + Sublayer(x)), trong đó Sublayer(x) là hàm được thực hiện bởi lớp con của chính nó, để tạo điều kiện thuận lợi cho các kết nối còn lại này, tất cả các lớp con trong mô hình , cũng như việc nhúng các lớp tạo ra kết quả đầu ra có kích thước dmodel= 512



Hình ảnh 7. Cấu trúc của Encoder

Encoder đầu tiên sẽ nhận ma trận biểu diễn của các từ đã được cộng với thông tin vị trí thông qua positional encoding. Sau đó, ma trận này sẽ được xử lý bởi multi head attention. Multi Head Attention thực chất là self-attention, nhưng mà để mô hình có thể chú ý nhiều pattern khác nhau, tác giả đơn giản là sử dụng nhiều self-attention.

3.2.1.Lớp Seft -Attention

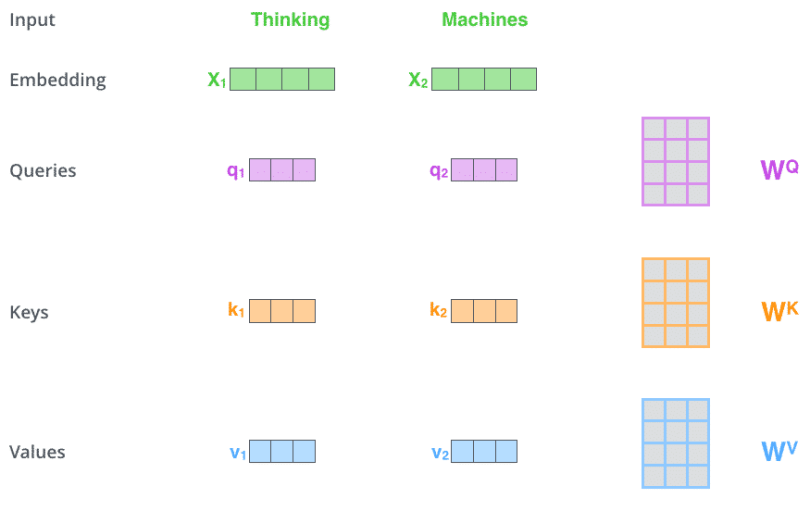
Self attention cho phép mô hình khi mã hoá một từ có thể sử dụng thông tin của những từ liên quan tới nó.

Ví dụ: “ The animal didn’t cross the street because it was too tired”

Khi mô hình xử lý từ “it” , self-attention cho phép nó liên kết “it” với “animal”.

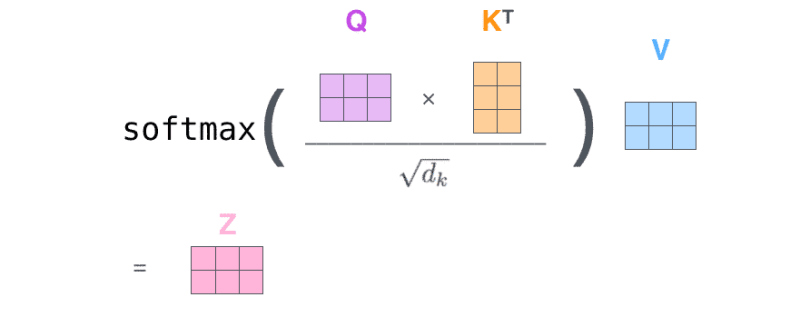
Khi mô hình xử lý từng từ ( từng vị trí trong câu đầu vào), self-attention cho phép nó quan sát các vị trí khác trong câu để tìm ra ý tưởng cho việc mã hoá từ hiện tại tốt hơn.

Tính toán Self-attention là tạo ra 3 vector từ mỗi vector đầu vào của encoder( trong trường hợp này là embedding của mỗi từ). Với mỗi từ, ta sẽ ạo một vector truy vấn ( Query, một vector khoá ( 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.



Hình ảnh 8. Tính toán self-attention

Self – attention sẽ được tính toán thông qua công thức:



Hình ảnh 9. Minh hoạ công thức tính attention

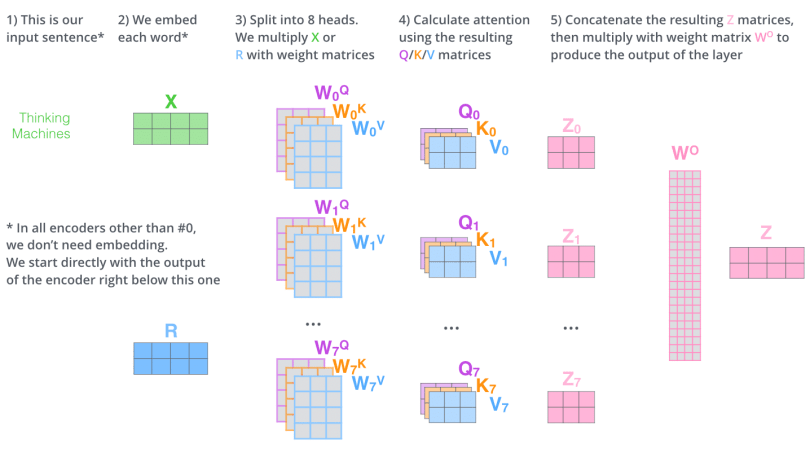
3.2.2 Multi – Head Attention

Thay vì thực hiện một chức năng attention duy nhất với dmodel – các key, value và query theo chiều mô hình, chúng ta thấy có lợi khi chiếu tuyến tính các query, key, value h lần với các phép chiếu tuyến tính tương ứng với các kích thước dk, dk, dv. Trên mỗi phiên bản dự kiến này của các query, key, value, sau đó chúng ta thục hiện chức năng attention song song, mang lại nhiều giá trị đầu ra dv. Chúng được nối và một lần nữa được chiếu, dẫn đến các giá trị cuối cùng như mô tả

Multi-Head Attention cho phứ model cùng tham gia vào thông tin từ các đại diện khác nhau các không gian con ở các vị trí khác nhau. Với một đầu attention duy nhất, tính trung bình hạn chế điều này.

Trong đó các phép chiếu là ma trận tham số , , và

Trong công việc này, sử dụng h= 8 lớp attention song song hoặc head. Cho mỗi cái này, chúng ta sử dk = dv = . Do kích thước của mỗi head giảm, tổng chi phí tính toán tương tự như single-head attention với đầy đủ các chiều



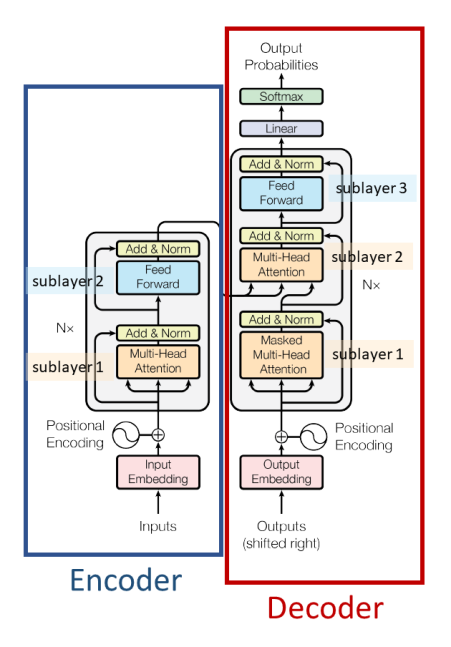
Hình ảnh 10. Tính toán multi-head attention

3.2.3 Residuals Connection và Normalixation layer

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

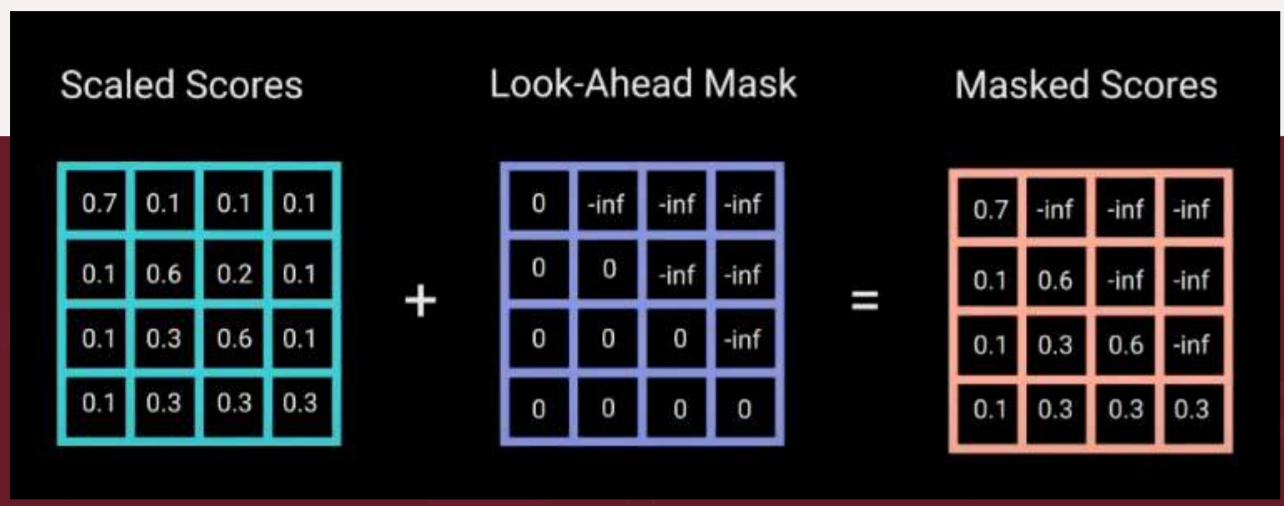
* 1. **Decoder**

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



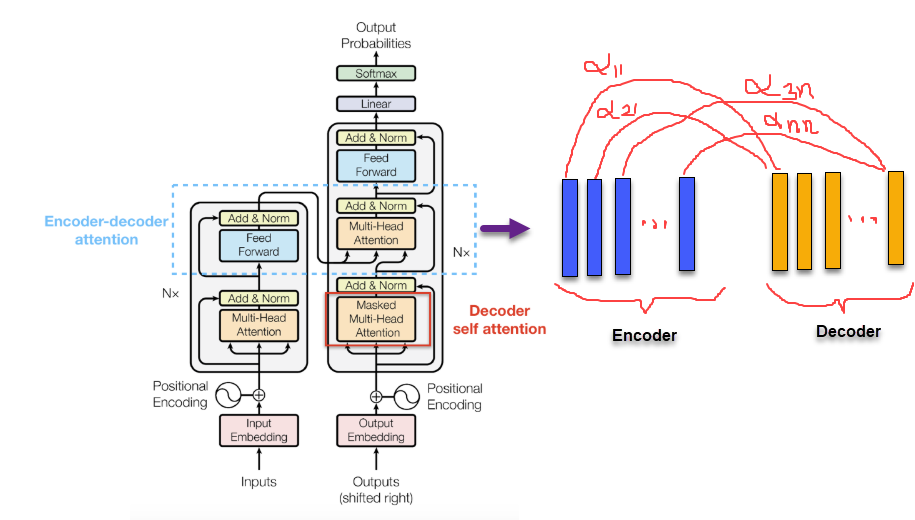
Hình ảnh 11. Minh hoạ cấu trúc decoder

* Masked multi head attention chính là multi head attention , có chức năng dùng để encode các từ của câu đích trong quá trình dịch. Trong decoder , sau khi câu đã được vector hoá sẽ tham dự vào lớp masked attention. Ma trận các vector sẽ được nhân với một ma trận đường chéo để tạo thành các mask, để che dấu đi các từ ở tương lại chưa được mô hình dịch đến. Seft-attention layer trong decoder cho phép mỗi vị trí trong decoder tham gia tới tất cả các vị trí trong decoder trên nó bao gồm cả vị trí đó. Chúng ta cần ngăng chặn bên trái luồng thông tin trong bộ decoder để bảo toàn thuộc tính tự động hồi quy, chúng ta thực hiện điều này bên trong dot-product attention được chia tỉ lệ bằng cách che dấu tất cả các giá trị đầu vào của hàm softmax tương ứng với các kết nối bật hợp pháp.



H12.

Hình ảnh 12. Tạo input mask trước khi đưa vào decoder



Hình ảnh 13**.** Minh hoạ Encoder-decoder attention

Đây là kiến trúc attention tương tác giữa các vector embedding của encode và decoder. Lớp Encoder-Decoder Attention” hoạt động như một self-attention nhiều đầu, ngoại trừ việc nó tạo ra các ma trận Q từ lớp phía dưới và lấy các ma trận K và V từ đầu ra của ngăn xếp encoder. Vector context được tính toán trên encoder đã được tính tương qua với vector decoder nên sẽ có ý nghĩa giải thích bối cảnh của từ tại vị trí time step decoder tương ứng. Sau khi kết hợp giữa vector context và vector decoder ta sẽ project tiếp qua một fully connected layer để tính phân phối xác suất cho output.

**Final Fully Connected layer, Softmax và Loss function**

Giống như các mô hình khác, chúng ta cần có thêm một fully connected layer để chuyển output từ layer phía trước thành ma trận có chiều băng số từ cần dự đoán. Sau đó đến softmax để tính được xác suất của từ xuất hiện tiếp theo là bao nhiêu

Loss function thì là cross-entropy

# CHƯƠNG 3: BERT

1. **Giới thiệu về Bert model**

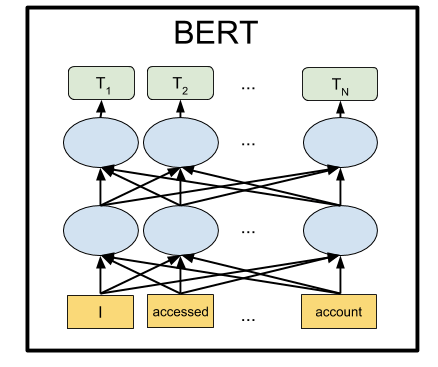
BERT được viết tắt của Bidirectional Encoder Representations from Transformer, một kiến trúc mới cho lớp bài toán Language Representation được Google công bố bào 2/11/2018. Không giống như các mô hình trước đó, BERT được thiết kế để đào tạo ra các vector đại diện cho ngôn ngữ văn bản thông qua ngữ cảnh 2 chiều ( trái và phải) của chúng. Kết quả là, vector đại diện được sinh ra từ mô hình BERT được tinh 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.

BERT là một khái niệm đơn giản nhưng lại mang lại hiệu quả cực lớn trong thực tế. Nó đã thu được kết quả tối ưu mới nhất cho 11 nhiệm vụ xử lý ngôn ngữ tự nhiên, bao gồm việc đẩy kết quả của nhiệm vụ GLUE benchmark lên 80.4% ( cải tiến thêm 7.6%) và SQuAD v1.1 với F1 score trên tập test đạt 93.2% ( cải tiến thêm 1.5%), tốt hơn con người 2%.

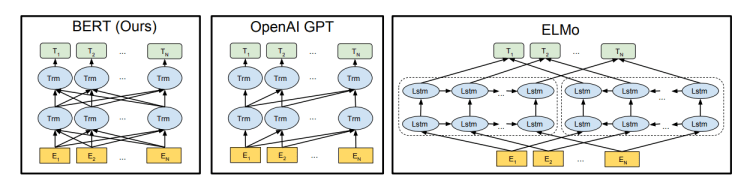
Lớp bài toán Representation cho mô hình ngôn ngữ đã cho thấy hiệu quả trong việc cải thiện nhiều nhiệm vụ trong lĩnh vực xử lý ngôn ngữ tự nhiên. Những nhiệm vụ này có thể là những nhiệm vụ cấp câu như Natural language inference, Paraphrasing nhằm dự đoán mối quan hệ giữa các câu bằng cách phân tích 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 tả kết quả chính xác cho câu hỏi ở dạng từ,…

Có hai kiến trúc để sử dụng các biểu diễn ngôn ngữ được huấn luyện trước ngày có các nhiệm vụ về sau gồm feature-based và fine-tuning. Cách tiếp cận feature-based ví dụ như ELMo sử dụng các kiến trúc theo nhiệm vụ cụ thể bao gồm các đại diện được đào tạo trước làm các tính năng bổ sung. Các tiến cận fine-tuning ví dụ như Generative Pre-trained Transformer giới thiệt tham số nhiệm vụ cụ thể, được đào tạo về các tác vụ xuôi dòng đơn giản bằng cách tinh chỉnh tất cả các tham số được đào tạo trước. Hai các tiếp cận chia sẻ cùng một chức năng mục tiêu trong quá trình đào tạo trước đó là họ sử dụng mô hình ngôn ngữ đơn bước để học biểu diễn ngôn ngữ. Điều này gây hạn chế cho sự lựa chọn kiến trúc có thể được sử dụng trong quá trình đạo tạo trước. Mô hình ngôn ngữ một chiều sẽ học học được ngôn ngữ dựa trên ngữ cảnh của chúng mà chỉ học các ngôn ngữ đơn, đứng đọc lập.Khác với các mô hình ngôn ngữ trước đó, BERT sử dụng kiến trúc hai chiều biểu diễn bộ mã hoá từ Transformer, giảm bớt hạn chế về tính đơn hướng bằng cách sử dụng mục tiêu đào tạo trước MLM.

1. **Kiến trúc của BERT**



Hình ảnh 14. Neural của BERT



Hình ảnh 15. Neural của các mô hình BERT, OpenAI GPT, ELMo

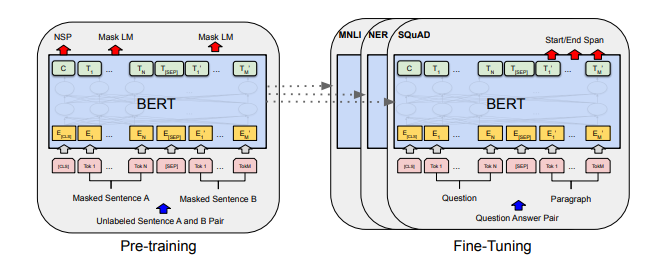
Kiến ​​trúc mạng nơ-ron của BERT so với các phương pháp đào tạo trước theo ngữ cảnh hiện đại trước đây được hiển thị bên dưới. Các mũi tên chỉ ra luồng thông tin từ lớp này sang lớp tiếp theo. Các hộp màu xanh lá cây ở trên cùng biểu thị sự trình bày theo ngữ cảnh cuối cùng của mỗi từ đầu vào.

BERT dựa trên kiến trúc model Transformer. Một Transformer cơ bản bao gồm một bộ Encoder để đọc đầu vào văn bản và một bộ Decoder để đưa ra dự đoán cho task. Vì mục tiêu của BERT là tạo ra một mô hình biểu diễn ngôn ngữ, nó chỉ cần phần bộ Encoder. Đầu vào cho bộ Encoder cho BERT là một chuỗi mã thông báo, đầu tiên được chuyển đổi thành vector và sau đó được xử lý trong mạng nơ-ron

BERT sử dụng đồng thời hai chiến lược là Masked LM ( để dự đoán từ thiếu trong câu - Masked language Model) và Next Sentence Prediction ( NSP – dự đoán câu tiếp theo câu hiện tại). Hai kĩ thuật này được train đồng thời và loss tổng sẽ kết hợp loss cảu 2 task và model sẽ cố gắng minimize loss tổng này.

### **2.1 Fine-tuning BERT ( tinh chỉnh BERT)**

Một đặc điểm ở BERT mà các model embedding trước đây chưa tưng có đó là kết quả huấn luyện có thể fine-tuning được. Chúng ta sẽ thêm vào kiến trúc model một output layer để tuỳ biến theo tác vụ huấn luyện



Hình ảnh 16. Toàn bộ tiến trình pre-traning và fine-tuning của BERT. Một kiến trúc tương tự được sử dụng cho cả pretrain-model và fine-tuning model.

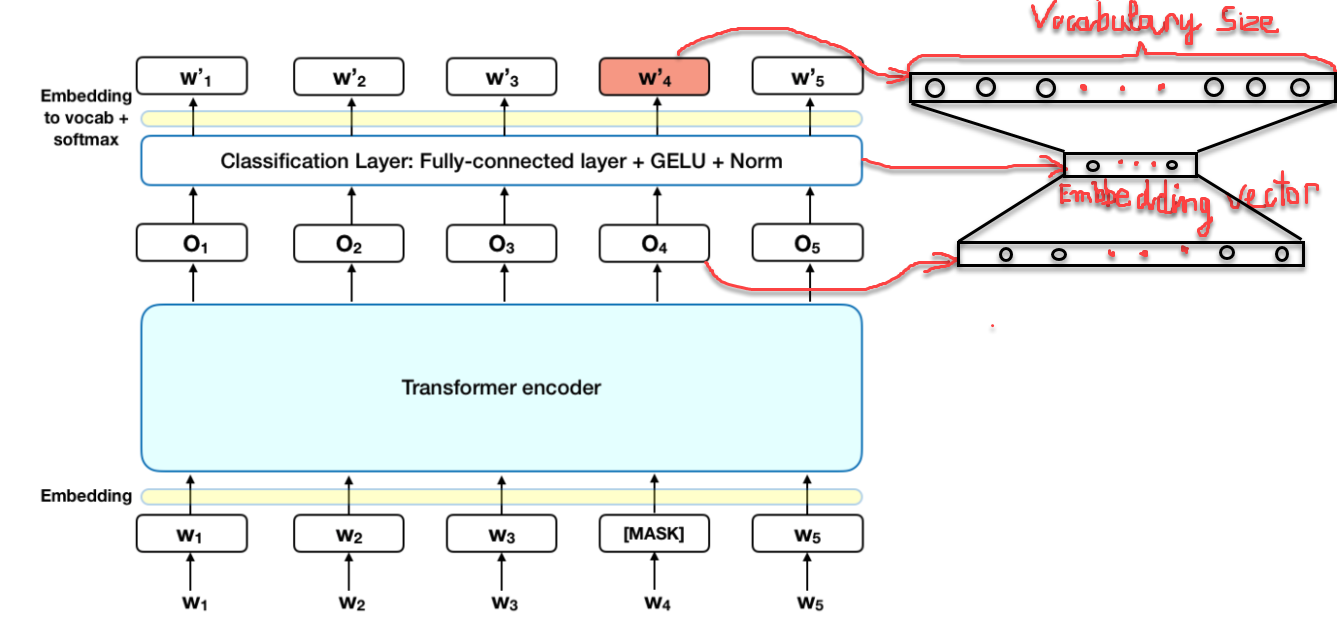
Tiến trình áp dụng fine-tuning sẽ như sau:

* Bước 1: Embedding toàn bộ các token của cặp câu bằng các vector nhúng từ pretrain model. Các token embedding bao gồm cả 2 token là [CLS] và [SEP] để đánh dấu vị trí bắt đầu của câu hỏi và vị trí ngăn cách giữa hai câu. Hai token này sẽ được dự báo ở output để xác định các phần Start/ End spand của câu output.
* Bước 2: Các Embedding vector sau đó sẽ được truyền vào kiến trúc multi-head attention với nhiều block code ( thường là 6, 12, 24 blocks tuỳ theo kiến trúc BERT). Ta thu được một vector output ở encoder.
* Bước 3: để dự báo phân phối xác suất cho từng vị trí từ ở decoder ở mỗi time step chúng ta sẽ truyền vào decoder vector output của encoder và vector embedding input của decoder để tính encoder-decoder attention. Sau đó projection qua liner layer và softmax để thu được phân phối xác suất cho output tương ứng ở time step t.
* Bước 4: Trong kết quả trả ra ở output của Transformer ta sẽ cố định kết quả của câu Question sao cho trùng với câu Question ở input. Các vị trí còn lại sẽ là thành phần mở rộng Start/End Span tương ứng với câu trả lời tìm được từ câu input.

**2.2 Masked Language Model (MLM)**

Masked ML là một tác vụ cho phép chúng ta fine-tuning lại các biểu diễn từ trên các bộ dữ liệu unsupervised-text bất kỳ. Chúng ta có thể áp dụng Masked ML cho những ngôn ngữ khác nhau để tạo ra biểu diễn embedding cho chúng. Các bộ dữ liệu của tiếng anh có kích thước lên tới vài trăm tới vài nghìn GB được huấn luyện trên BERT đã tạo ra những kết ủa khá ấn tưởng

Bên dưới là sơ đồ huấn luyện BERT theo tác vụ Masked ML



Hình ảnh 17. Sơ đồ kiến trúc BERT cho tác vụ Masked ML

Theo đó:

* Khoảng 15% các token của câu input được thay thế bởi [MASK] token trước khi truyền vào model đại diện cho những từ bị che dấu (masked). Mô hình sẽ dựa trên các từ không được che (non-masked) dấu xung quanh [MASK] và đồng thời là bối cảnh của [MASK] để dự báo giá trị gốc của từ được che dấu. Số lượng từ được che dấu được lựa chọn là một số ít ( 15%) để tỉ lệ bối cảnh chiếm nhiều hơn 85%
* Bản chất của kiến trúc BERT vẫn là một mô hình seq2seq gồn 2 phase encoder giúp embedding các từ input vào decoder giúp tìm ra phân phối xác suất của các từ ở output. Kiến trúc Transformer encoder được giữ lại trong tác vụ Masked ML. Sau khi thực hiện self-attention và feed forward ta sẽ thu được vector embedding là O1, O2,…, O5
* Để tính toán phân phối xác suất cho từ output, chúng ta thêm một Fully connect layer ngay sau Transformer Encoder. Hàm softmax có tác dụng tính toán phân phối xác suất. Số lượng units của fully connected layer phải bằng với kích thước của từ điển
* Cuối cùng ta thu được vector nhúng của mỗi một từ tại vị trí MASK sẽ là embedding vector giảm chiều của vector Oi sau khi đi qua fully connected layer như mô tả hình bên phải.

Ví dụ: với câu :con\_chó của tôi đẹp quá”. Từ được chọn để mask là “đẹp”

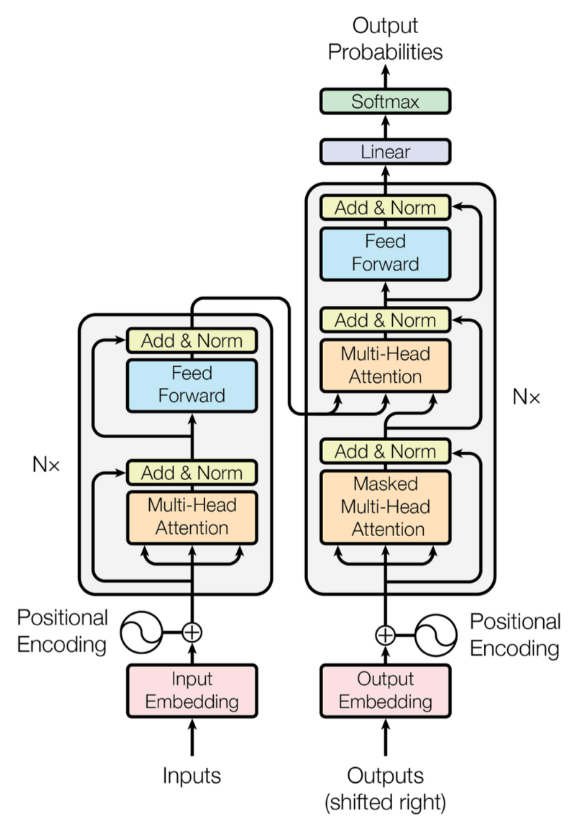
* Thay thế 80% từ được chọn trong dữ liệu huấn luyện thành token [MASK] => con chó của tôi [MASK] quá”
* 10% các từ được chọn sẽ được thay thế bởi 1 từ ngẫu nhiên => “con chó của tôi máy tính quá”.
* 10% còn lại được giữ không thay đổi => “con chó của tôi đẹp quá”

Hàm loss function của BERT sẽ bỏ qua mất mát từ những từ không bị che dấu và chỉ đưa vào mất mát của những từ bị che dấu. Do đó mô hình sẽ hội tụ lâu hơn nhưng đây là đặc tính bù trừ cho sự gia tăng ý thức về bối cảnh. Việc lựa chọn ngẫu nhiên 15% số lượng các từ bị che dấu cũng tạo ra vô số các kịch bản input cho mô hình huấn luyện nên mô hình sẽ cần phải huấn luyện rất lâu mới học được toàn diện các khả năng.

* 1. **Next Sentence Prediction (NSP)**

Đây là một bài toán học có giám sát với 2 nhãn ( hay còn gọi là phân loại nhị phân). Input đầu vào của mô hình là một cặp câu pair-sequence) sao cho 50% câu thứ 2 được lựa chọn là câu tiếp theo của câu thứ nhất và 50% được lựa chọn một cách ngẫu nhiên từ bộ văn bản mà không có mối liên hệ gì với câu thứ nhất. Nhãn của mô hình sẽ tương ứng với IsNext khi cặp câu là liên tiếp hoặc NotNext nếu cặp câu không liên tiếp.

Cũng tương tự như mô hình Question and Answering, chúng ta cần xác định các vị trí đầu câu thứ nhất bằng token [CLS] và vị trí cuối cùng các câu bằng token [SEP]. Các token này có tác dụng nhận biết các vị trí bắt đầu và kết thúc của từng câu thứ nhất và thứ hai.



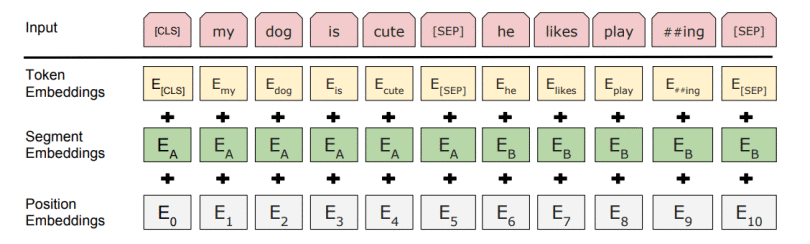
Hình ảnh 18. Sơ đồ kiến trúc BERT cho tác vụ NSP

Thông tin input được preprocessing trước khi đưa vào mô hình huấn luyện bao gồm:

* Ngữ nghĩa của từ ( token embeddings): thông qua các embedding vector cho từng từ. Các vector được khởi tạo từ pretrain model.

Ngoài embedding biểu diễn từ của các từ trong câu, mô hình còn embedding thêm một số thông tin:

* Loại câu( segment embeddings): gồm hai vector là EA nếu từ thuộc câu thứ nhất và EB nếu từ thuộc câu thứ 2
* Vị trí của từ trong câu ( position embedding) : là các vector E0, …, E10. Tương tự như position embedding trong transformer.



Hình ảnh 19. Minh hoạ cách tính vector đầu vào cho BERT

Vector output sẽ bằng tổng của cả ba thành phần embedding theo từ, câu và vị trí.

Ví dụ:

**Input**: [CLS] người đàn\_ông làm [MASK] tại cửa\_hàng [SEP] anh\_ta rất [MASK] và thân\_thiện [SEP]

**Label**: isNex

**Input**: [CLS] người đàn\_ông làm [MASK] tại cửa\_hàng [SEP] cô\_ta đang cầm súng [SEP]

**Label:** notNext

1. **Các kiến trúc model BERT**

Hiện tại có nhiều phiên bản khác nhau của Model BERT. Các phiên bản đều dựa trên việc thay đổi kiến trúc của Transformer tập trung ở 3 tham số. L: số lượng các Block sub-layer trong Transformer, H: kích thước của embedding vector ( hay còn gọi là hidden size), A: số lượng head trong multi-head layer, mỗi một head sẽ thực hiện một self-attention. Tên của của 2 kiến trúc bao gồm:

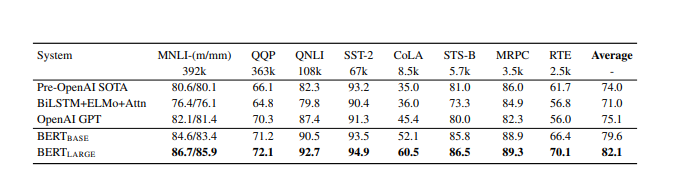
* BERTBASE ( L = 12, H = 768, A = 12) : tổng tham số 110 triệu.
* BERTLARGE ( L = 24, H = 1024, A = 16): tổng tham số 340 triệu.

Như vậy ở kiến trúc BERT Large chúng ta tăng gấp đôi số layer, tăng kích thước Hidden size của embedding vector gấp 1,33 lần và tăng số lương head multi-head layer gấp 1.33 lần.

1. **Kết quả các nhiệm vụ NLP sử dụng BERT**

*4.1 GLUE (General Language Understanding Evaluation – nhiệm vụ đánh giá thông hiểu ngôn ngữ chung)*

GLUE là một tập hợp của các nhiệm vụ thông hiểu ngôn ngữ khác nhau. Chúng bao gồm MNLI ( Multi-Genre Natural Language Interence – suy luận ngôn ngữ tự nhiên đa thể loại), QQP( Quora Question Pairs – cặp câu hỏi Quora), QNLI ( Question Natural Laguage Inference – suy luận câu hỏi ngôn ngữ tự nhiên), SST-2 ( the Stanford Sentiment Treebank), CoLA ,…. Cả hai BERTbase và BERTlarge đều vượt trội hơn hẳn so với các mô hình trước đó.



h. kết quả so sánh của BERT với các model trước đó

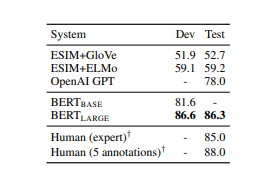
*4.2 SQuAD v1.1 Dataset ( Stanford Question Answer Dataset – bộ dữ liệu câu hỏi- câu trả lời tiêu chuẩn)*

SQuAD là một tập hợp gồm 100k cặp câu hỏi – đáp án được lấy từ nhiều nguồn khác nhau. Một điểm dữ liệu chứa một câu hỏi và một đoạn văn lấy từ Wikipedia nó chứa đáp án. Nhiện vụ là dự đoán câu trả lời từ đoạn văn bản. BERT hoạt động tốt nhất ( với tổ hộ và TriviaQA) hoạt động tốt hơn các hệ thống hàng đầu với 1,5 điểm F1 trong tổ hợp và 1.3 điểm F khi là một hệ thống đơn lẻ. Trong thực tế, BERTbase  đơn lẻ tốt hơn hệ thống tổng hợp hàng đầu về điểm F1

* 1. *SWAG ( Sitiuation With Adversarial Generations – các tình huống với các thế hệ đối nghịch*)

Bộ dữ liệu SWAG chứa 113k nhiệm vụ câi đánh giá câu trả lời phù hợp nhất bằng cách sử dụng suy luận chung có căn cứ. Cho một câu, nhiệm vụ là chọn cách tiếp diễn hợp lý nhất trong số bốn lựa chọn.

BERTlarge hoạt động tốt hơn OpentAI GPT 8.5%. Nó thậm chí còn hoạt động tốt hơ một con người chuyên nghiệp. kết quả của tập dữ liệu SWAG được đưa ra sau đây:



# CHƯƠNG 4: BÀI TOÁN

1. **Tổng qua về hệ thống Question Answering**

Question Answering là một ngành khoa học máy tính trong các lĩnh vực của truy vấn thông tinh và xử lý ngôn ngữ tự nhiên (NLP), nó được kết nối với xây dựng hệ thống tự động trả lời câu hỏi đặt ra bởi con người trong một ngôn ngữ tự nhiên

Triển khai trả lời câu hỏi, thường là một chương trình máy tính, có thể xây dựng câu trả lời của nó bằng cách truy vaans cơ sở dữ liệu đã biết hoặc thông tin có cấu trúc, thường là kiến thức cơ sở. thông thường, hệ thống QA có thể kéo các câu trả lời từ một bộ sưu tập các tài liệu ngôn ngữ tự nhiên không có cấu trúc.

Nghiên cứu trả lời câu hỏi đang cố gắng giải quyết nhiều loại câu hỏi bao gồm: thực tế, danh sách, định nghĩa, như thế nào, tại sao, giả thuyết, hạn chế ngữ nghĩa và câu hỏi đa ngôn ngữ

Có hai dạng QA là Closed-domain QA và Open-domain QA

* Closed-domain question answering: xử lý những câu hỏi trong một miền xác định ( ví dụ : y tế, giáo dục, …) và có thể khai thác kiến thức về miền cụ thể thường được chính thức hoá trong các bản thể học.
* Open-domain question anwering: giải quyết các câu hỏi về hầu hết mọi thứ và chỉ có thể dự vào các bản thể luận chung và kiến thức ths giới. Mặt khác, các hệ thống này thường có sẵn nhiều dữ liệu hơn để lấy ra câu trả lời

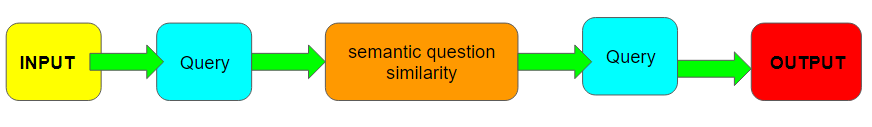
1. **Bài toán “ Xây dựng hệ thống trả lời câu hỏi luật”**
   1. **Các nghiên cứu trước**

Đã có rất nhiều nghiên cứu về hệ thống trả lời câu hỏi tự động (Question Answering) sử dụng các model như RNN, LSTM hay gần đây nhất là sử dụng BERT. Các bài toán đã đạt được những kết quả tương đối, tuy nhiên về một hệ thống trả lời câu hỏi cho tiếng Việt thì còn khá là hạn chế và đặc biệt là hệ thống trả lời câu hỏi văn bản luật cho tiếng Viêt. Do đó chúng tôi đã nghiên cứu và xây dựng đề tài với chủ đề : “Xây dựng hệ thống hỏi đáp luật” tập trung vảo luật giao thông đường bộ , áp dụng cho tiếng Việt.

* 1. **Xây dựng hệ thống trả lời câu hỏi luật ( BERT)**
     1. **Bộ dữ liệu**

Xây dựng hai bộ dữ liệu bao gồm: bộ dữ liệu các câu hỏi tương tự ( similarity question) có …. và bộ dữ liệu câu hỏi – câu trả lời ( question – answer) có….

* + 1. **Kiến trúc hệ thống**



Hình ảnh 20. Tiến trình của hệ thống “Hỏi đáp văn bản luật”

Trong đó:

* Khi đưa câu hỏi vào INPUT, hệ thống sẽ nhận câu hỏi, truy vấn câu hỏi trong bộ dữ liệu similarity question sử dụng công cụ elasticsearch để tìm ra những câu hỏi gần giống nhau
* Sau đó đưa vào hệ thống “semantic question similary” để tính toán , tìm ra câu hỏi có ngữ nghĩa đúng nhất với câu hỏi trong bộ dữ liệu question – answer.
* Sau khi đã tìm ra được câu hỏi có ngữ nghĩa tương tự với câu hỏi trong input thì đưa ra câu trả lời trong output.
  + 1. **Công cụ sử dụng Elasticsearch**

Chúng tôi sử dụng công cụ Elasticsearch để lưu trữ dữ liệu và truy vấn dữ liệu được dễ dàng và nhanh chóng

Elastichsearch là một công cụ tìm kiếm và tìm kiếm miễn phí và mở cho tất cả loại dữ liệu bao gồm văn bản số , không gian địa lý, có cấu trúc và không có cấu trúc. Elasticsearch được xây dựng trên Apache Lucene và được pát hành lần đầu tiên vào năm 2010 bởi Elasticsearch N.v. Được biết đến với các API còn lại đơn giản, bản chất phân tán, tốc độ và khả năng mở rộng. Elasticsearch giống như một server lưu trữ giữ liệu dưới dạng json để ta dễ dàng lưu trữ và truy vấn được một cách nhanh chóng và dễ dàng nhất

* + 1. **Mô hình BERT và PhoBERT**

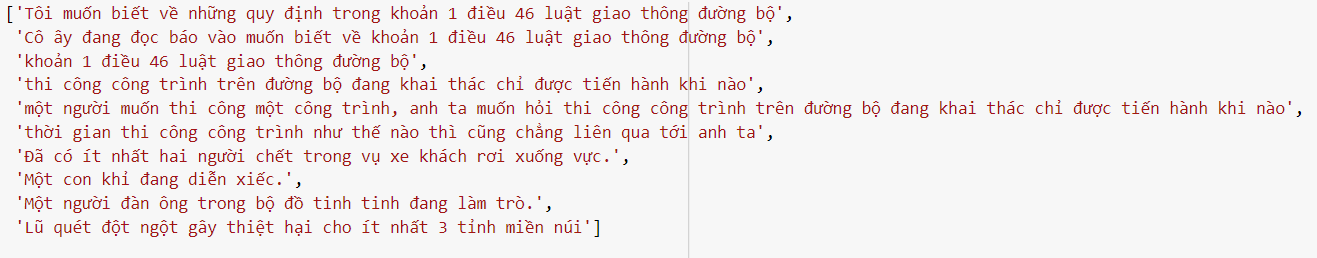
Chúng tôi sử dụng mô hình BERTBASE và PhoBERTBASE để thực hiện bài toàn đặt ra với kiến trúc như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bert-base-uncased | Key | Shape | Count |  |
| Embedding | embeddings.word\_embeddings.weight | [30522, 768] | 23,440,896 | 23,837,184 |
|  | embeddings.postion\_embeddings.weight | [512, 768] | 393,216 |  |
|  | embeddings.token\_type\_embeddings.weight | [2, 768] | 1,536 |  |
|  | embedding.LayerNorm.weight | [768] | 768 |  |
|  | embedding.LayerNorm.bias | [768] | 768 |  |
| Transformer\*12 | encoder.layer.0.attetion.self.query.weight | [768, 768] | 589,824 | 7,087,872\*12 = 85,054,464 |
|  | encoder.layer.0.attetion.self.query.bias | [768] | 768 |  |
|  | encoder.layer.0.attetion.self.key.weight | [768, 768] | 589,824 |  |
|  | encoder.layer.0.attetion.self.key.bias | [768] | 768 |  |
|  | encoder.layer.0.attetion.self.value.weight | [768, 768] | 589,824 |  |
|  | encoder.layer.0.attetion.self.value.bias | [768] | 768 |  |
|  | encoder.layer.0.attetion.output.dense.weight | [768, 768] | 589,824 |  |
|  | encoder.layer.0.attetion.output.dense.bias | [768] | 768 |  |
|  | encoder.layer.0.attetion.output.LayerNorm.weight | [768] | 768 |  |
|  | encoder.layer.0.attetion.output.LayerNorm.bias | [768] | 768 |  |
|  | encoder.layer.0.intermediate.dense.weight | [3072, 768] | 2,359,296 |  |
|  | encoder.layer.0.intermediate.dense.bias | [3072] | 3072 |  |
|  | encoder.layer.0.output.dense.weight | [768, 3072] | 2,359,296 |  |
|  | encoder.layer.0.output.dense.bias | [768] | 768 |  |
|  | encoder.layer.0.output.LayerNorm.weight | [768] | 768 |  |
|  | encoder.layer.0.output.LayerNorm.bias | [768] | 768 |  |
| Polder | polder.dense.weight | [768, 768] | 589,824 | 590,592 |
|  | polder.dense.bias | [768] | 768 |  |
|  |  |  |  | 109,482,240 |

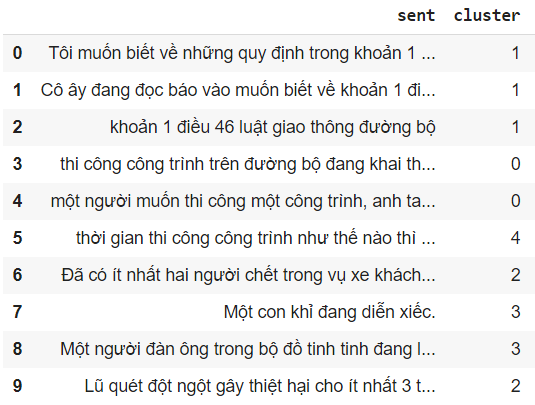
Bộ dữ liệu hiện có : 50 000 cặp câu tương tự ( semamtic similarity question ) và 1500 cặp câu hỏi câu trả lời ( question – answer)

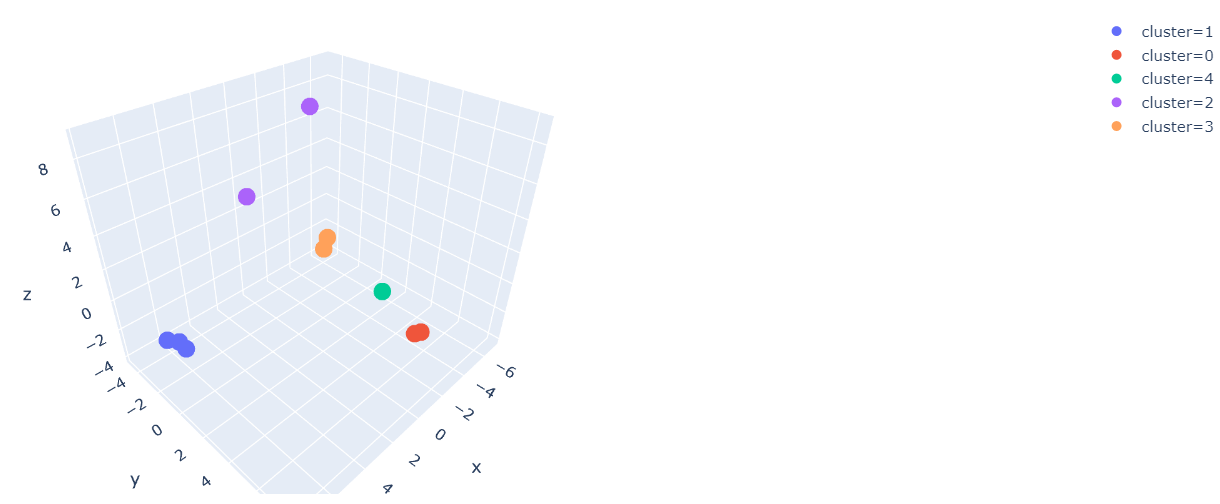
Sau khi tiến hành chạy thử kết quả , nhóm chúng tôi đã thu được kết quả như sau:

Dữ liệu đầu vào :



Chúng tôi muốn chia dữ liệu thành 5 nhóm , tìm xem những câu nào được ghép vào 5 nhóm thì kết quả như sau:



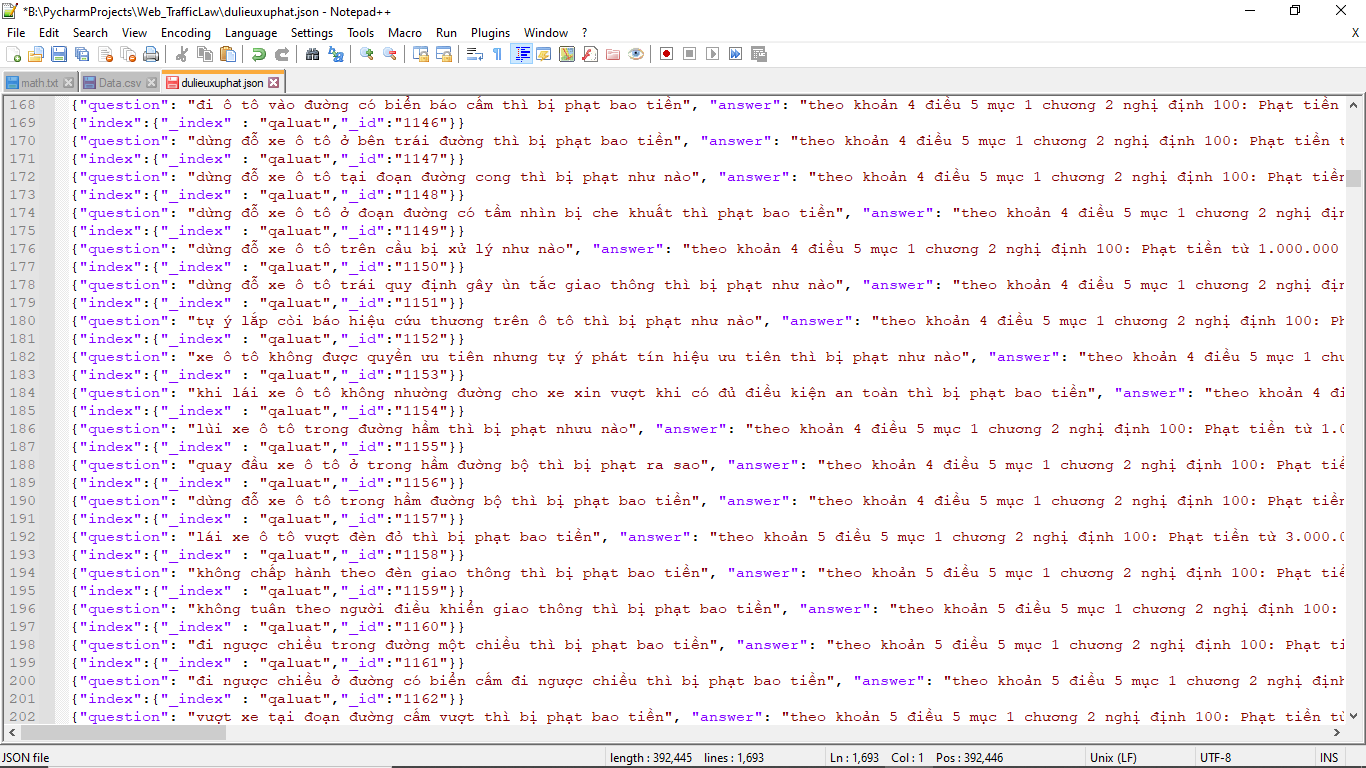


1. **Xây dựng chương trình**

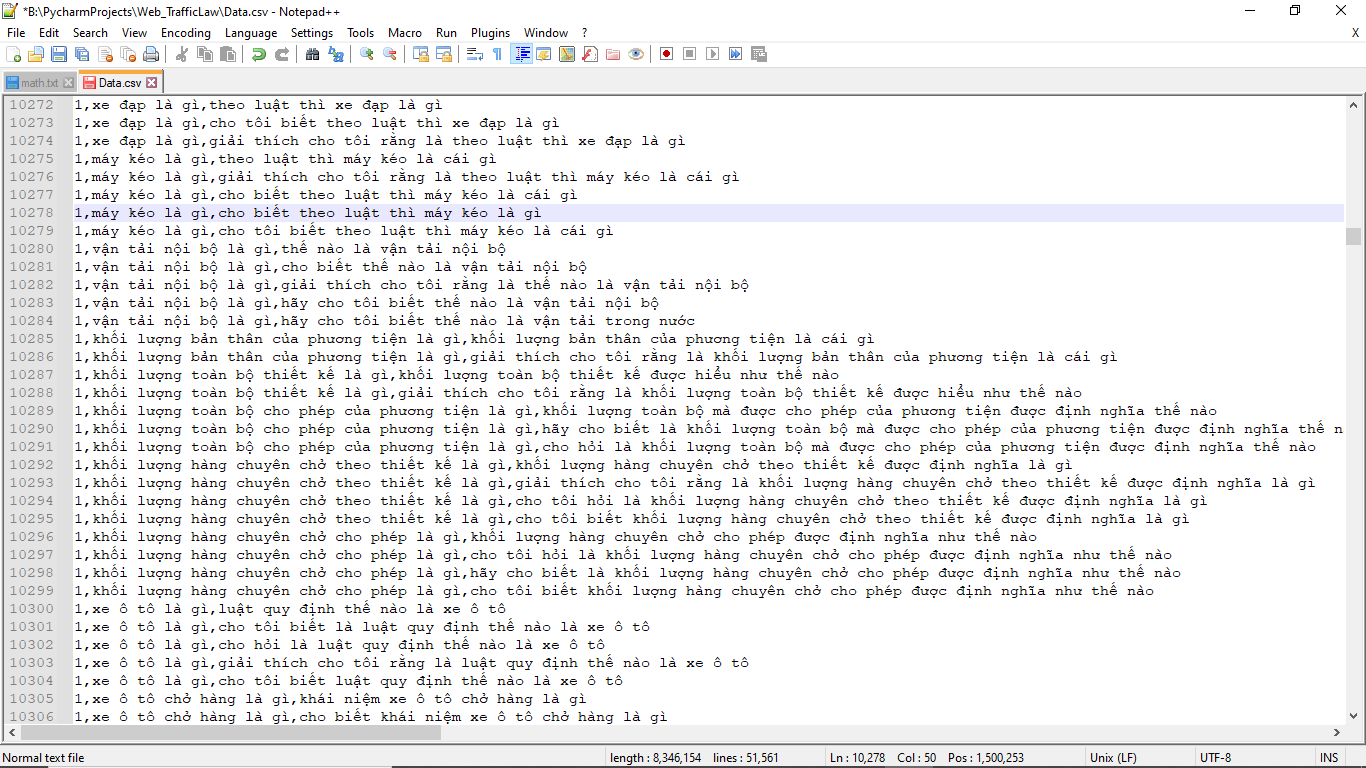
3.1 Chuẩn bị dữ liệu

- Chúng tôi xây dựng 2 bộ dữ liệu:

. Bộ thứ nhất là bộ Question-Answer: gồm 1500 cặp câu hỏi và câu trả lời, đồng thời chúng tôi cũng móc nối mỗi câu hỏi với các câu hỏi khác để khi người dùng hỏi 1 câu thì các câu tương tự khác sẽ được đưa ra. Dựa trên Luật giao thông đường bộ năm 2008, và Luật xử phạt vi phạm hành chính với lĩnh vực giao thông đường bộ năm 2019. Dữ liệu được đẩy lên công cụ Elasticsearch.



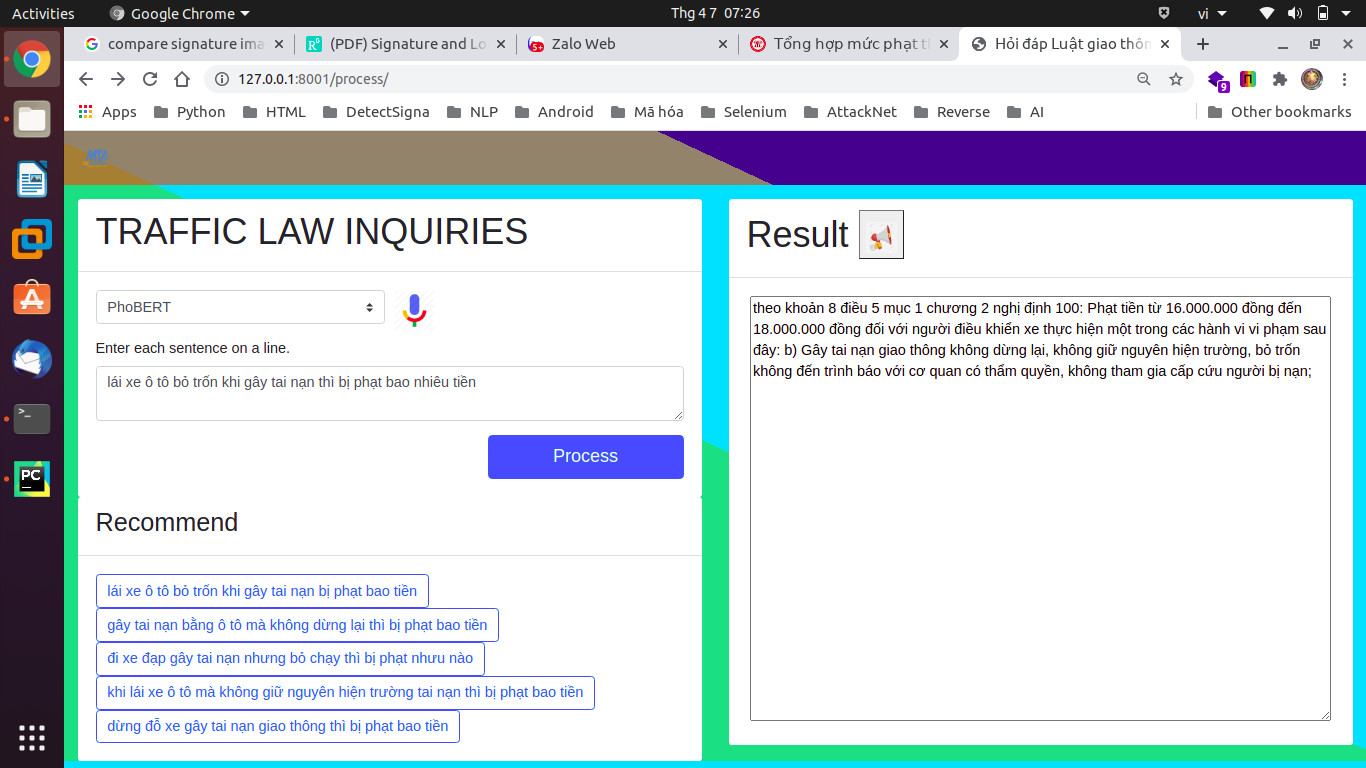
. Bộ thứ hai là bộ dữ liệu huấn luyện cho 2 mô hình BERT và PhoBERT: gồm 50 000 cặp câu tương đồng và không tương đồng về ngữ nghĩa được đánh nhãn 1 (tương đồng) và 0 (không tương đồng), trong đó có 29 000 cặp câu tương đồng, 21 000 cặp câu không tương đồng



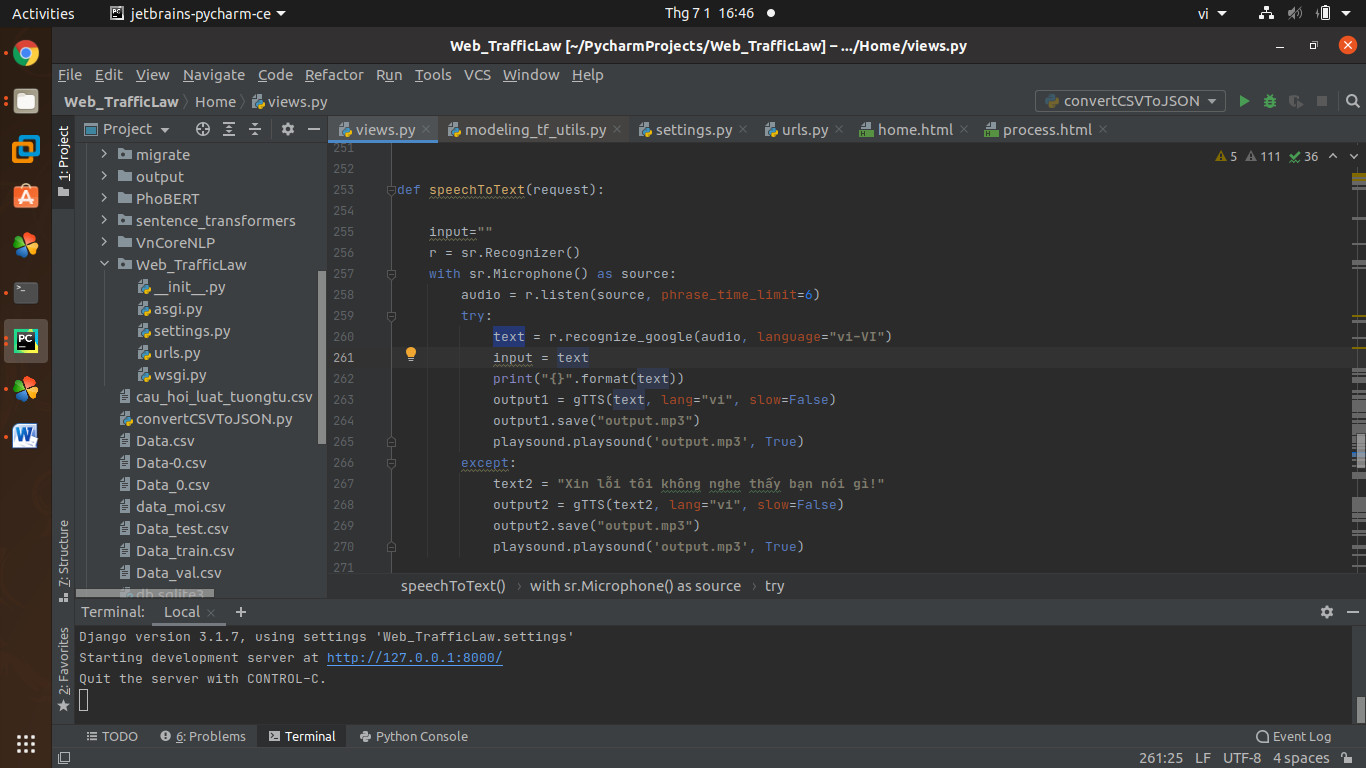
Accuracy trong quá trình train và test là 0.79 đối với PhoBERT và 0.71 đối với BERT.

* 1. Xây dựng chương trình

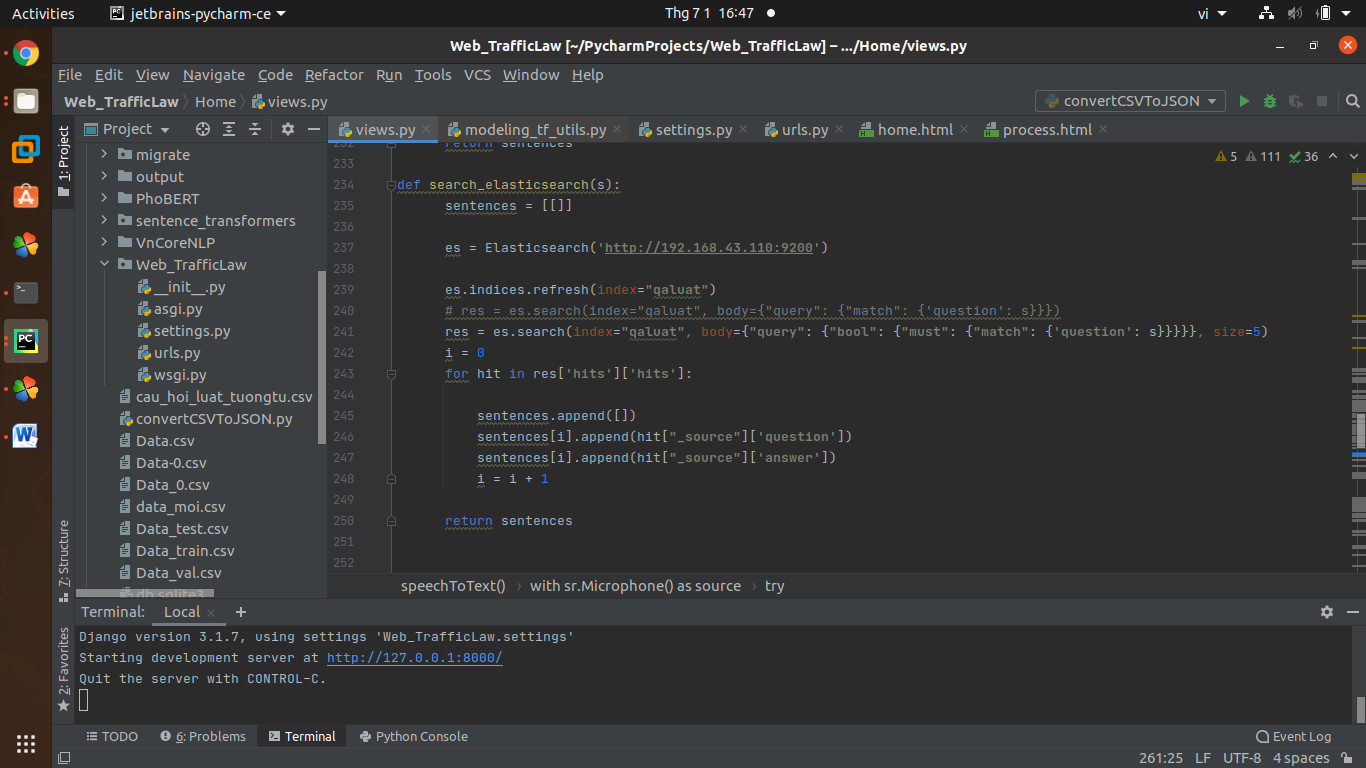
Chúng tôi xây dựng trên framework Django viết bằng ngôn ngữ Python:

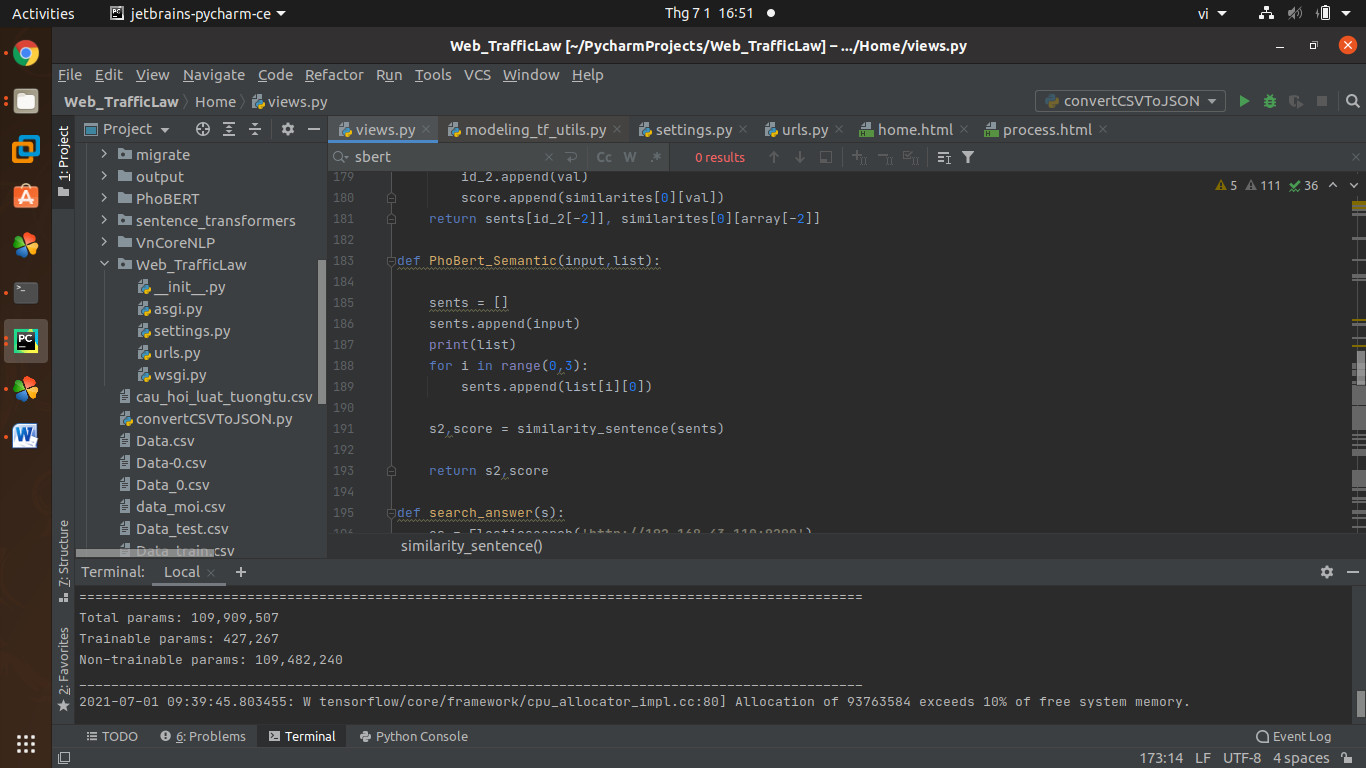


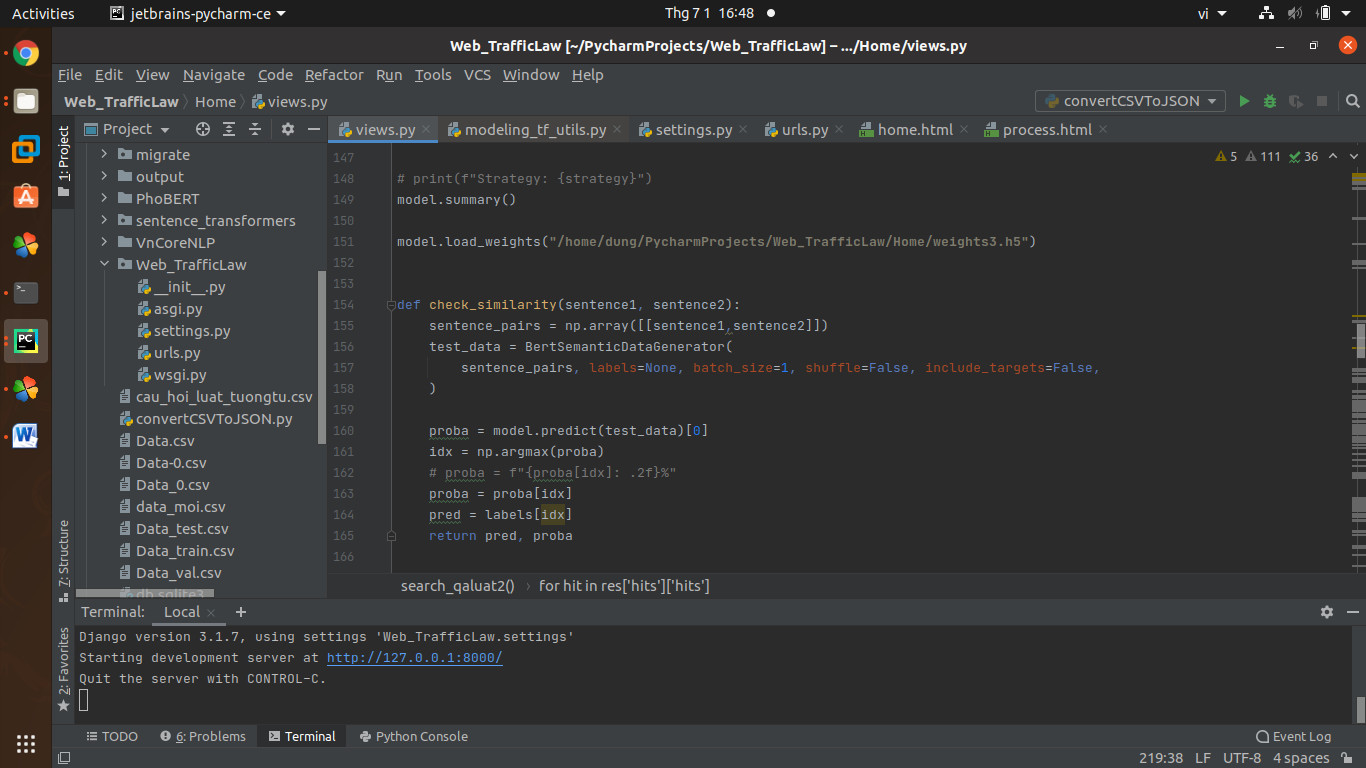
- Input: Nhập text hoặc là dùng Speech to Text (sử dụng speech recognition của Google)



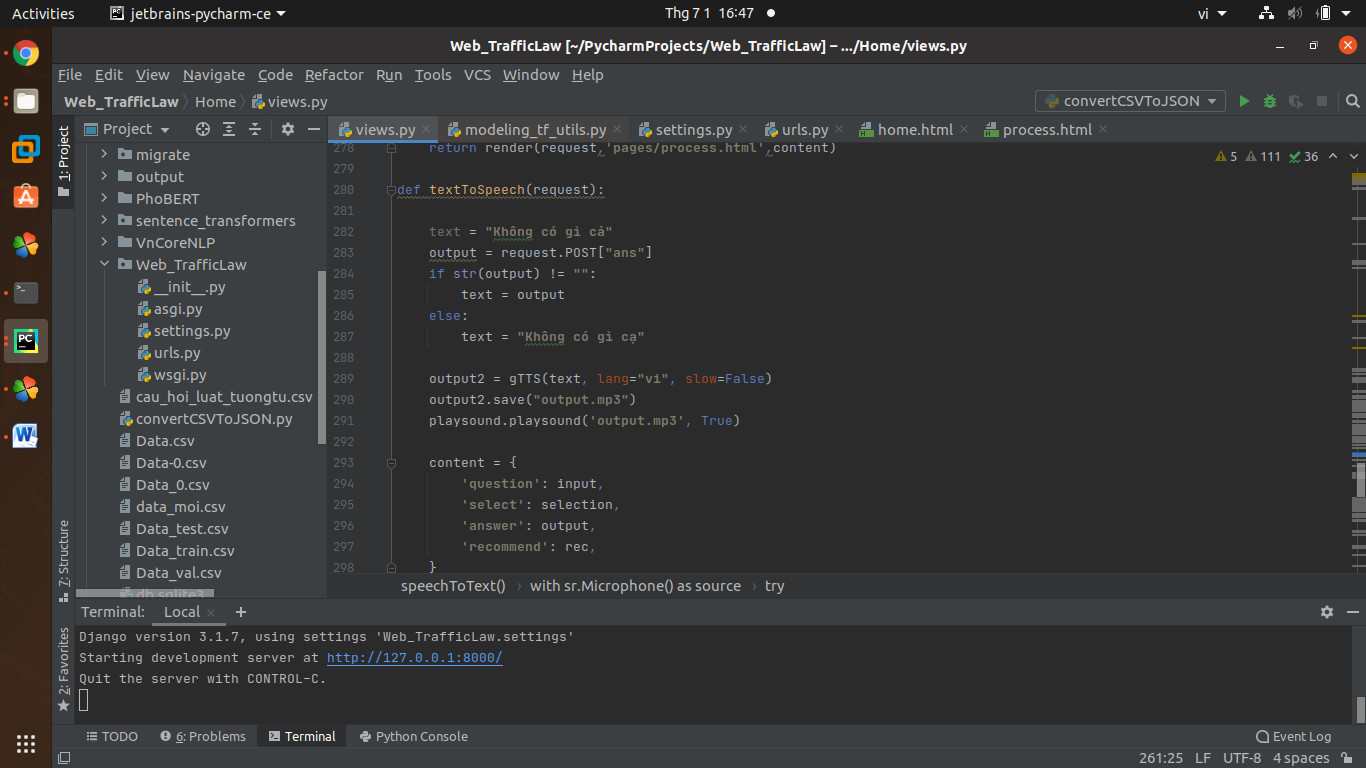
- Xử lí: Sử dụng Elasticsearch để đưa ra 5 câu hỏi có sự tương đồng về kí tự nhất, sau đó sử dụng BERT hoặc PhoBERT để lấy ra câu có sự tương đồng về ngữ nghĩa nhất, đồng thời dùng Elasticsearch lần 2 để đưa ra những câu tương tự với câu hỏi ban đầu ở phần Recommend. Nếu câu hỏi của người dùng không có trong bộ Q-A thì Elasticsearch vẫn sẽ đưa ra những câu có liên quan nhất đến câu hỏi của người dùng có trong bọ Q-A







- Output: Câu trả lời bên Result, người dùng có thể Text to Speech (Google Voice) và những câu hỏi liên quan phần Recommend.



Mọi người có thể tham khảo quá trình train và test:

Link Colab train, test PhoBERT:

<https://colab.research.google.com/drive/1RJBpYiAbGMYIj86FjyVux5-DaRhMemjd>

Link Colab train, test BERT:

<https://colab.research.google.com/drive/1ZIuUYdjycWjJo3m10vVUZjSYtAbvdqEW>

# CHƯƠNG 5: PHÂN CÔNG CÔNG VIỆC NHÓM 10

|  |  |
| --- | --- |
| **Thành viên** | **Công việc** |
| Lê Duy Dũng (Trưởng nhóm) | Nghiên cứu lí thuyết, sử dụng model BERT, PhoBERT, train và test  Xây dựng Back-End cho chương trình |
| Trần Sang Anh | Nghiên cứu lí thuyết, sử dụng công cụ Elasticsearch  Xây dựng Front-End cho chương trình |
| Phạm Mạnh Cường | Chuẩn bị và hiệu chỉnh dữ liệu  Kiểm thử chương trình |
| Hồ Thị Vân | Chuẩn bị và hiệu chỉnh dữ liệu  Kiểm thử chương trình |

**Tài liệu tham khảo**

1. Attention is all you need (Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit)
2. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding(Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova)
3. Bert model – Khoa học dữ liệu – Khanh’s blog
4. The illustrated Transformer – Jay Alammar
5. Exlanation of BERT Model – NLP ( GeeksforGeeks)
6. BERT, RoBERTa, PhoBERT, BERTweet: Ứng dụng state-of-the-art pre-trained model cho bài toán phân loại văn bản – Viblo.asia