TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**CUỐI KỲ**

**XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

*Người hướng dẫn*: **PGS.TS. LÊ ANH CƯỜNG**

*Người thực hiện*: **HOÀNG ĐÌNH QUÝ VŨ – 521H0517**

**LÊ HUỲNH HUYỀN TRANG - 520C0156**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

A red and blue logo

Description automatically generated

**CUỐI KỲ**

**XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

*Người hướng dẫn*: **PGS.TS. LÊ ANH CƯỜNG**

*Người thực hiện*: **HOÀNG ĐÌNH QUÝ VŨ – 521H0517**

**LÊ HUỲNH HUYỀN TRANG - 520C0156**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

LỜI CẢM ƠN

Đây là phần tác giả **tự viết** ngắn gọn, thể hiện sự biết ơn của mình đối với những người đã giúp mình hoàn thành Luận văn/Luận án. Tuyệt đối không sao chép theo mẫu những “lời cảm ơn” đã có.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của **PGS.TS. Lê Anh Cường**. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Hoàng Đình Quý Vũ*

*Lê Huỳnh Huyền Trang*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

MỤC LỤC

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**CÁC KÝ HIỆU**

*f Tần số của dòng điện và điện áp (Hz)*

*p Mật độ điện tích khối (C/m3)*

**CÁC CHỮ VIẾT TẮT**

CSTD Công suất tác dụng

MF Máy phát điện

BER Tỷ lệ bít lỗi

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 2.1: Kiến trúc FTP 1](#_Toc387689394)

**DANH MỤC BẢNG**

[Bảng 3.1 Ví dụ cho chèn bảng 1](#_Toc387689363)

CHƯƠNG 1 – Kiến trúc và huấn luyện BERT

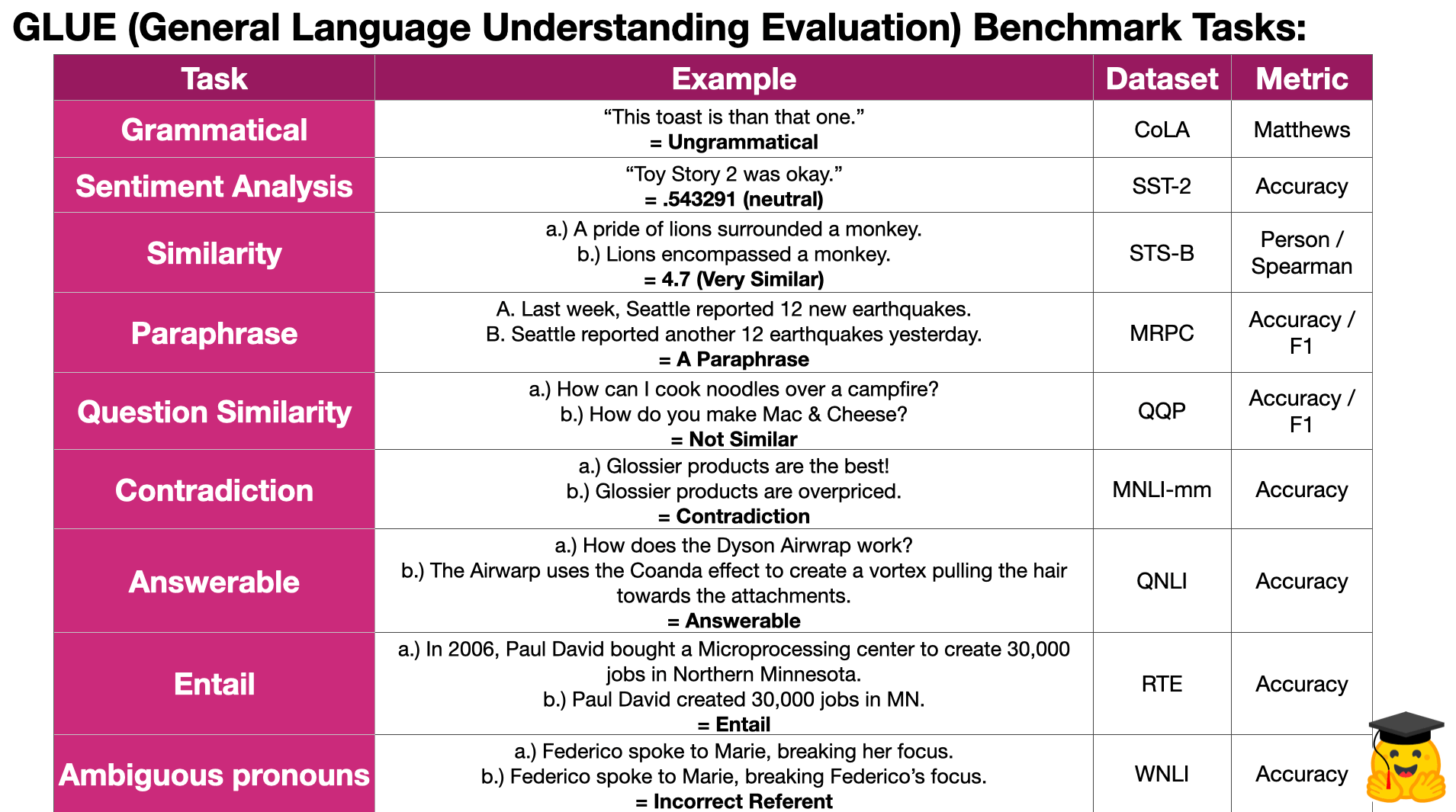
* 1. Giới thiệu

BERT là viết tắt của cụm từ **Bidirectional Encoder Representation from Transformer** có nghĩa là mô hình biểu diễn từ theo 2 chiều ứng dụng kỹ thuật Transformer. BERT là model biểu diễn ngôn ngữ được google giới thiệu vào năm 2018.

BERT được thiết kế để huấn luyện trước các biểu diễn từ (pre-train word embedding). Điểm đặc biệt ở BERT đó là nó có thể điều hòa cân bằng bối cảnh theo cả 2 chiều trái và phải.

Cơ chế attention của Transformer sẽ truyền toàn bộ các từ trong câu văn đồng thời vào mô hình một lúc mà không cần quan tâm đến chiều của câu. Do đó Transformer được xem như là huấn luyện hai chiều (bidirectional) mặc dù trên thực tế chính xác hơn chúng ta có thể nói rằng đó là huấn luyện không chiều (non-directional). Đặc điểm này cho phép mô hình học được bối cảnh của từ dựa trên toàn bộ các từ xung quanh nó bao gồm cả từ bên trái và từ bên phải.

Trong bài báo BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding các tác giả đã nêu ra những cải tiến của model BERT trong các tác vụ:

* Tăng GLUE score (General Language Understanding Evaluation score), một chỉ số tổng quát đánh giá mức độ hiểu ngôn ngữ lên 80.5% (tăng 7.7 %)
* 
* Tăng accuracy trên bộ dữ liệu MultiNLI đánh giá tác vụ quan hệ văn bản (text entailment) lên 86.7% (tăng 4.6 %)
* **Quan hệ văn bản (Textual Entailment)**: Là tác vụ đánh giá mối quan hệ định hướng giữa 2 văn bản? Nhãn output của các cặp câu được chia thành đối lập (contradiction), trung lập (neutral) hay có quan hệ đi kèm (textual entailment).
* Tăng accuracy F1 score trên bộ dữ liệu SQuAD v1.1(Stanford Question Answering Dataset) đánh giá tác vụ question and answering lên 93.2% (tăng 1.5) và SQuAD v2.0 Test F1 to 83.1 (tăng 5.1).

1.2 Cấu trúc Transformer

1.2.1 Encoder và decoder

A diagram of a sublaying process

Description automatically generated

* Kiến trúc máy biến áp bao gồm một bộ mã hóa (encoder) và một bộ giải mã(dencoder), mỗi bộ được tạo thành từ nhiều mulit-head attention và feed forward

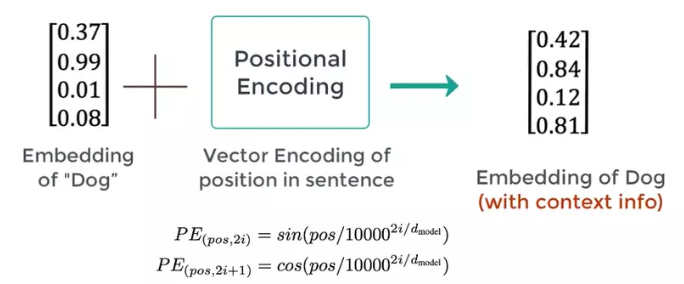
**Encoder**: là tổng hợp xếp chồng lên nhau của 6 layers xác định. Mỗi layer bao gồm 2 layers con (sub-layer) trong nó. Sub-layer đầu tiên là multi-head self-attention mà lát nữa chúng ta sẽ tìm hiểu. Layer thứ 2 đơn thuần chỉ là các fully connected feed-forward layer

**Decoder:** cũng là tổng hợp xếp chồng của 6 layers. Kiến trúc tương tự như các sub-layer ở Encoder ngoại trừ thêm 1 sub-layer thể hiện phân phối attention ở vị trí đầu tiên.

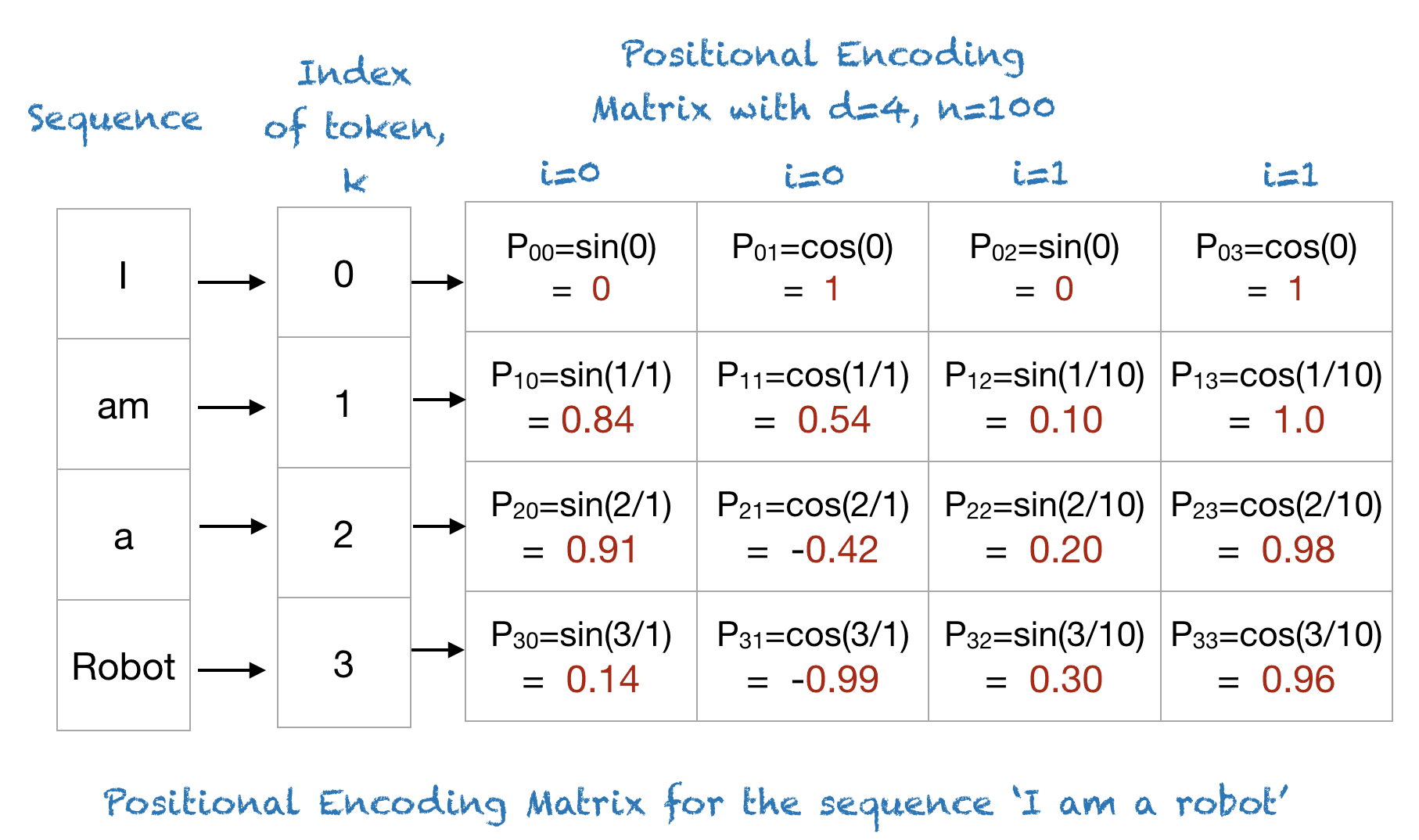
1.2.2 Positional encoding

Việc nhúng từ một phần giúp chúng ta thể hiện ngữ nghĩa của một từ, nhưng cùng một từ ở các vị trí khác nhau của câu có ý nghĩa khác nhau. Đó là lý do tại sao Transformers có thêm phần **Positional encoding** để đưa thêm thông tin. Về vị trí của một từ

Pos là vị trí của từ này trong câu, PE là giá trị của phần tử thứ được nhúng và có độ dài . Sau đó chúng ta tổng hợp vectơ PE và vectơ nhúng



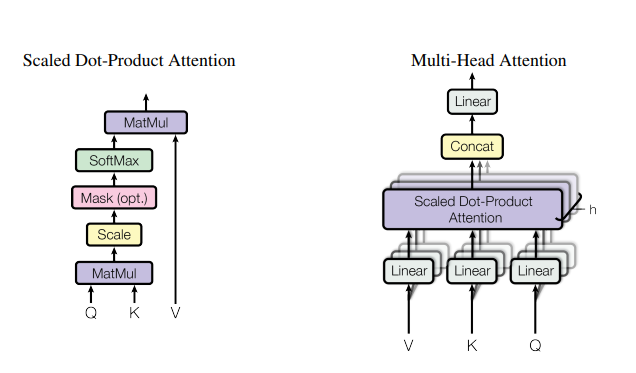
Để hiểu cách diễn đạt trên, hãy lấy ví dụ về cụm từ “I am a robot,” với n=100 và d=4. Bảng sau đây hiển thị ma trận mã hóa vị trí cho cụm từ này.



1.2.3 Attention

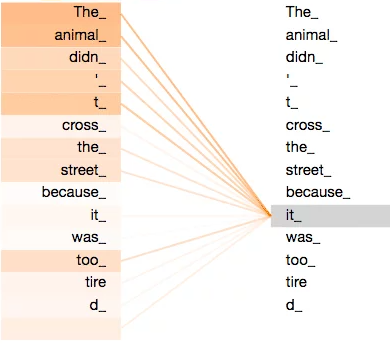
Sự chú ý (attention) trong mô hình Transformer cho phép mô hình tập trung có chọn lọc vào các phần cụ thể của chuỗi đầu vào, thay vì xử lý toàn bộ chuỗi đồng đều. Điều này giúp cải thiện hiệu quả và độ chính xác trong việc xử lý thông tin.

Trong Transformer, sự chú ý được sử dụng để ánh xạ giữa một truy vấn (query) và một tập hợp các cặp khóa-giá trị (key-value) để tạo ra một đầu ra. Cụ thể, truy vấn, khóa và giá trị đều là các vector. Đầu ra của cơ chế attention được tính như một tổng trọng số của các giá trị, trong đó trọng số được xác định bởi độ tương thích giữa truy vấn và các khóa tương ứng.



Hình 2: (trái) Scaled Dot-Product Attention. (phải) Chú ý nhiều đầu bao gồm một số

Giả sử câu sau là câu đầu vào mà chúng ta cần dịch: ***”The animal didn't cross the street because it was too tired”.*** Từ "it" trong câu trên đại diện cho điều gì? "**animal**" hay "**street**"?

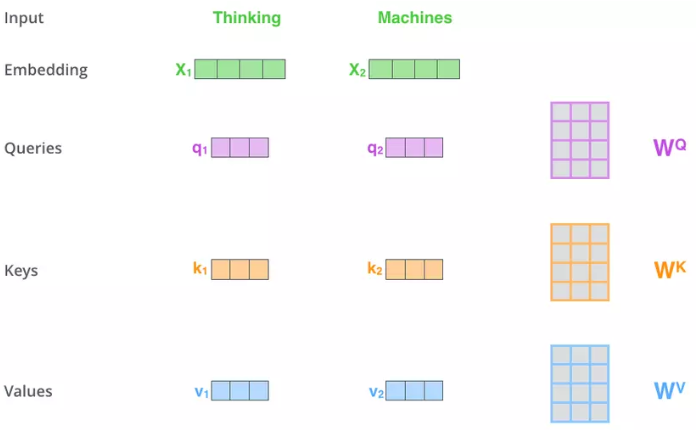
****

Câu hỏi này đơn giản đối với con người nhưng không đơn giản đối với các thuật toán. Khi mô hình xử lý từ "it", sự tự chú ý cho phép nó liên kết "it" với "animal ". Khi mô hình xử lý từng từ (mỗi vị trí trong cầu nối đầu vào), sự tự chú ý cho phép nó quan sát các vị trí khác trong câu để tìm ý tưởng để mã hóa tốt hơn từ hiện tại. Tự chú ý là những gì Transformer sử dụng để duy trì kiến thức về các từ khác có liên quan đến từ hiện tại.

1.2.3.1 Công thức self-attention

* **Trong đó:**
* Q (Query) là vector truy vấn.
* K (Key) là vector khóa.
* V (Value) là vector giá trị.
* là kích thước của vector khóa (Key).

Công thức này khá đơn giản, nó được thực hiện như sau. Đầu tiên, để có được 3 vectơ Q, K, V, các nhúng đầu vào được nhân với 3 ma trận trọng lượng tương ứng (được điều chỉnh trong quá trình đào tạo) .



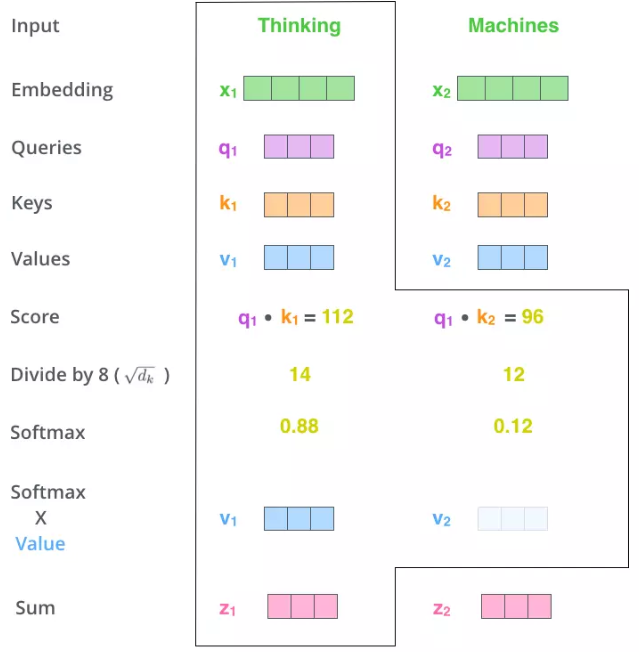
Tại thời điểm này, vectơ K hoạt động như một khóa đại diện cho từ và Q sẽ truy vấn các vectơ K của các từ trong câu bằng cách tích chập với các vectơ này. Mục đích của phép chập là để tính toán mối quan hệ giữa các từ. Theo đó, hai từ liên quan đến nhau sẽ có kết quả lớn và ngược lại.

**Bước thứ hai** là bước "Scale ", chỉ đơn giản là chia "Score" cho căn bậc hai của chiều Q / K / V (trong hình chia cho 8 vì Q / K / V là vectơ 64-D). Điều này làm cho giá trị "Score" không phụ thuộc với độ dài của vectơ Q/K/V

**Bước thứ ba** là softmax các kết quả trước đó để có được phân phối xác suất trên các từ.

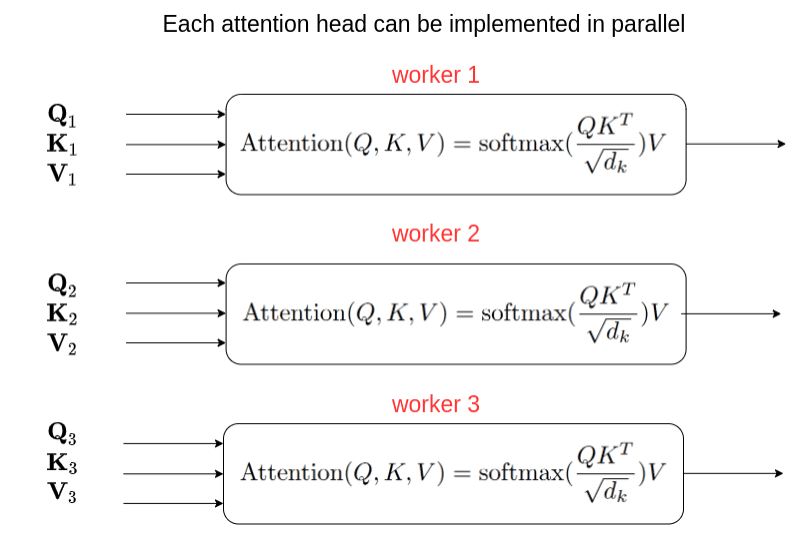
**Bước thứ tư** là nhân phân bố xác suất đó với vectơ V để loại bỏ các từ không cần thiết (xác suất nhỏ) và giữ lại các từ quan trọng (xác suất lớn).

Trong bước **cuối cùng**, các vectơ V (nhân với đầu ra softmax) được cộng lại với nhau, tạo ra vectơ chú ý Z cho một từ. Lặp lại quy trình trên cho tất cả các từ, chúng ta có được ma trận chú ý cho 1 câu.



1.2.3.2 Multi-head attention

Multi-Head Attention trong Transformer cho phép mô hình học và tích hợp thông tin từ nhiều khía cạnh khác nhau của chuỗi đầu vào một cách hiệu quả. Thay vì thực hiện một phép tính attention duy nhất, Multi-Head Attention chia quá trình này thành nhiều **"đầu" attention song song.** Các truy vấn (query), khóa (key) và giá trị (value) được chiếu tuyến tính thành nhiều không gian khác nhau bằng các trọng số khác nhau, tạo ra nhiều phiên bản. Trên mỗi phiên bản này, hàm Attention được thực hiện song song, thu được các giá trị đầu ra. Các đầu ra này sau đó được nối lại và chiếu tuyến tính một lần nữa để tạo ra giá trị đầu ra cuối cùng.



1.2.4 Residuals

Residuals, hay còn gọi là kết nối bỏ qua (skip connections), là một kỹ thuật được sử dụng trong nhiều mô hình học sâu, bao gồm cả Transformer. Ý tưởng chính của kỹ thuật này là cho phép thông tin từ đầu vào đi qua các lớp (layers) mà không bị biến đổi, bằng cách thêm đầu vào trực tiếp vào đầu ra của lớp đó.

Cụ thể, đầu vào (x) được thêm vào đầu ra của lớp attention hoặc lớp feed-forward network, và kết quả này được đưa qua một lớp Chuẩn hóa lớp (Layer Normalization). Công thức tổng quát cho quá trình này như sau:

A diagram of a process

Description automatically generated

Output=LayerNorm (x + SubLayer(x))

* Trong đó:
* x là đầu vào của lớp con
* SubLayer(x) là đầu ra của lớp con (có thể là attention hoặc feed-forward network).
* LayerNorm là lớp Chuẩn hóa lớp được áp dụng sau khi thêm đầu vào x và đầu ra SubLayer(x)
* Lợi ích:
* Giữ Lại Thông Tin Gốc: Bằng cách thêm trực tiếp đầu vào x vào đầu ra SubLayer(x), thông tin gốc từ đầu vào được giữ lại và truyền qua các lớp, giúp mô hình học được các đặc trưng quan trọng từ đầu vào một cách hiệu quả hơn.
* Giảm Gradient Vanishing: Residuals giúp giảm vấn đề biến mất gradient trong quá trình huấn luyện các mô hình sâu, giúp cải thiện hiệu quả học tập và tăng tốc độ hội tụ.

1.2.5 Feed forward

* Ngoài các lớp con chú ý, mỗi lớp trong bộ mã hóa và bộ giải mã của chúng tôi chứa một mạng chuyển tiếp nguồn cấp dữ liệu được kết nối đầy đủ, được áp dụng cho từng vị trí riêng biệt và giống hệt nhau. Điều này bao gồm hai phép biến đổi tuyến tính với kích hoạt ReLU ở giữa.
* Mặc dù các phép biến đổi tuyến tính giống nhau ở các vị trí khác nhau, nhưng chúng sử dụng các tham số khác nhau từ lớp này sang lớp khác. Một cách khác để mô tả điều này là hai kết cấu với kích thước hạt nhân 1. Kích thước của đầu vào và đầu ra là = 512 và lớp bên trong có chiều = 2048.
* A screenshot of a computer

  Description automatically generated
* Vì các vectors này không phụ thuộc vào nhau nên ta có thể tận dụng được tính toán song song cho cả câu.

1.2.6 Decoder

1.2.6.1 Masked Multi-head Attention

Giả sử bạn muốn Transformers thực hiện bài toán dịch Anh-Pháp, thì công việc của Bộ giải mã là giải mã thông tin từ Bộ mã hóa và tạo ra từng từ tiếng Pháp dựa trên PREVIOUS WORDS. Vì vậy, nếu chúng ta sử dụng Multi-head attention trên toàn bộ câu như trong Encoder, Decoder sẽ "nhìn thấy" từ tiếp theo mà nó cần dịch (Nó sẽ nhìn thấy từ tiếp theo mà không có học học để dịch ra). Để ngăn chặn điều đó, khi Bộ giải mã dịch sang từ từ tôi, phần sau của câu tiếng Pháp sẽ bị che và Bộ giải mã sẽ chỉ được phép "xem" phần mà nó đã dịch trước đó.A diagram of a group of words

Description automatically generated with medium confidence

1.2.6.2 Decode proccess

Quá trình giải mã về cơ bản giống như mã hóa, ngoại trừ Bộ giải mã giải mã từng từ một và đầu vào của Bộ giải mã (câu tiếng Pháp) được che giấu. Sau khi đầu vào ẩn được truyền qua **sub-layer#1** của Bộ giải mã, nó sẽ không còn nhân với 3 ma trận trọng số để tạo ra Q, K, V mà sẽ chỉ nhân với 1 ma trận trọng lượng WQ. K và V được lấy từ Bộ mã hóa cùng với Q từ Masked multi-head attention và được đưa vào các sub-layers # 2 và # 3 tương tự như Bộ mã hóa. Cuối cùng, các vectơ được đẩy vào lớp tuyến tính (là một mạng được kết nối đầy đủ) tiếp theo là Softmax để tạo ra xác suất của từ tiếp theo.

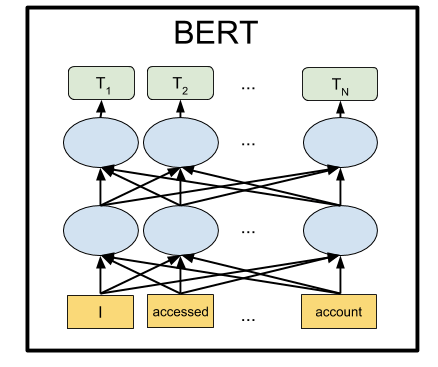
Hai hình dưới đây mô tả trực quan quá trình mã hóa và giải mã Transformers

A screenshot of a computer

Description automatically generated

1.3 Cấu trúc Bert

Khác với cấu trúc của Transformer thì cấu trúc của Bert chỉ lấy phần encoder bên trái và bỏ đi phần decoder bên phải.



Các phiên bản đều dựa trên việc thay đổi kiến trúc của Transformer tập trung ở 3 tham số:

* Transformer layers: số lượng các block sub-layers trong transformer,
* Hidden size: kích thước của embedding véc tơ
* Attention heads: Số lượng head trong multi-head layer, mỗi một head sẽ thực hiện một self-attention.
* Với 1 TPUs = 4 chíp TPU

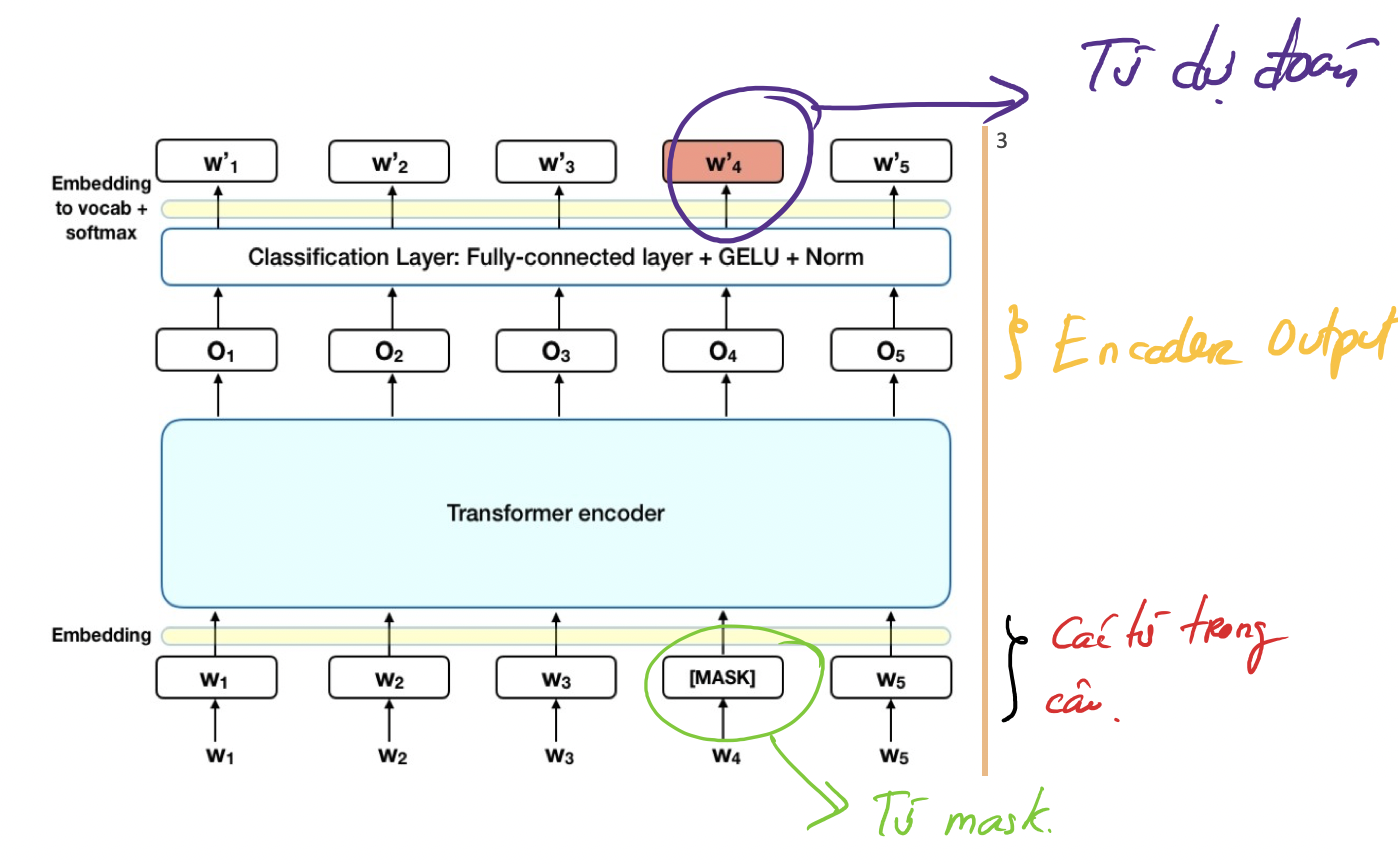
A white background with black text

Description automatically generated

BERT được train đồng thời 2 task gọi là Masked LM (để dự đoán từ thiếu trong câu) và Next Sentence Prediction (NSP – dự đoán câu tiếp theo câu hiện tại). Hai món này được train đồng thời và loss tổng sẽ là kết hợp loss của 2 task và model sẽ cố gắng minimize loss tổng này. Chi tiết 2 task này như sau:

1.3.1 Masked ML

Trong task này, quá trình huấn luyện sẽ che đi khoảng 15% số từ trong câu và đưa vào mô hình. Mục tiêu của quá trình huấn luyện là để mô hình dự đoán các từ bị che dựa vào các từ còn lại trong câu.



* Cụ thể:
* Thêm một lớp classification lên trên encoder output
* Đưa các vector trong encoder ouput về vector bằng với vocab size, sau đó softmax để chọn ra từ tương ứng tại mỗi vị trí trong câu.
* Loss sẽ được tính tại vị trí masked và bỏ qua các vị trí khác (để đánh giá xem model dự đoán từ mask đúng/sai).

Ví dụ với câu: “Hôm nay Nam đưa bạn gái đi chơi”.

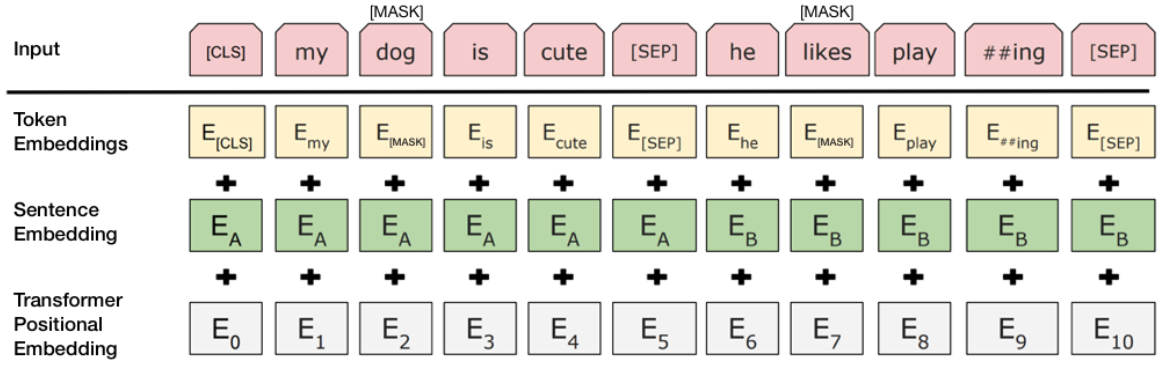
Câu sẽ bị che (masked): Hôm nay Nam đưa [mask] đi chơi.

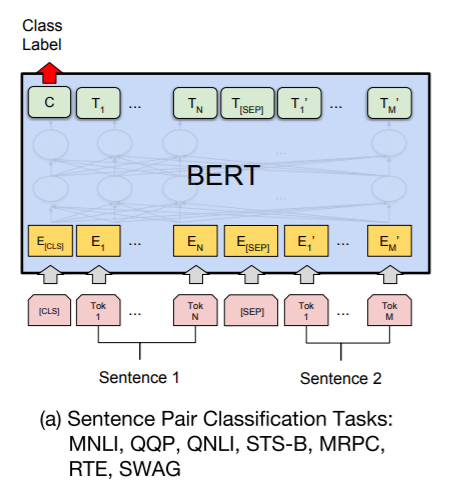
1.3.2 Next Sentence Prediction (NSP)

Trong task này, model sẽ nhận vào một cặp câu và phải đưa ra kết quả là 1 nếu câu thứ hai là câu tiếp theo của câu thứ nhất và 0 nếu không phải. Trong quá trình huấn luyện, chúng ta sẽ chọn ngẫu nhiên 50% mẫu là Positive (kết quả là 1) và 50% còn lại là Negative được tạo ra từ việc ghép linh tinh (kết quả là 0).

Các train cụ thể như sau:

* Bước 1: Ghép 2 câu vào nhau và thêm 1 số token đặc biệt để phân tách các câu. Token [CLS] thêm vào đầu cầu Thứ nhất, token [SEP] thêm vào cuối mỗi câu.
* Ví dụ ghép 2 câu “Hôm nay em đi học” và “Học ở trường rất hay” thì sẽ thành [CLS] Hôm nay em đi học [SEP] Học ở trường rất vui [SEP]
* Bước 2. Khác với transformer, mỗi token trong câu sẽ được cộng thêm một vector gọi là Sentence Embedding, thực ra là đánh dấu xem từ đó thuộc câu Thứ nhất hay câu thứ 2 thôi.
* Ví dụ nếu thuộc câu Thứ nhất thì cộng thêm 1 vector toàn số “0” có kích thước bằng Word Embedding, và nếu thuộc câu thứ 2 thì cộng thêm một vector toàn số “1”.
* Bước 3. Sau đó các từ trong câu đã ghép sẽ được thêm vector Positional Encoding vào để đánh dấu vị trí từng từ trong câu đã ghép
* Bước 4. Đưa chuỗi sau bước 3 vào mạng (Tổng của token embedding, segment embedding và positional embedding)
* Bước 5. Lấy encoder output tại vị trí token [CLS] được transform sang một vector có 2 phần tử [c1 c2].
* Bước 6. Tính softmax trên vector đó và output ra probality của 2 class: Đi sau và Không đi sau.





|  |  |
| --- | --- |
| Input | Label |
| [CLS] người đàn\_ông làm [MASK] tại cửa\_hàng [SEP] anh\_ta rất [MASK] và thân\_thiện [SEP] | isNext |
| [CLS] người đàn\_ông làm [MASK] tại cửa\_hàng [SEP] cô\_ta đang cầm súng [SEP] | notNext |

1.4 Quy trình tiền huấn luyện

Chúng tôi đã sử dụng 2 nguồn dữ liệu để huấn luyện mô hình, đó là BooksCorpus **(800 triệu từ) (Zhu et al., 2015)** và **English Wikipedia (2,500 triệu từ).** Đối với Wikipedia, chúng tôi chỉ thu thập các đoạn văn bản và loại bỏ danh sách, bảng và tiêu đề. Quan trọng nhất là chúng tôi sử dụng văn bản ở mức độ đoạn văn thay vì tập hợp các câu bị xáo trộn.

* Để tạo dữ liệu huấn luyện, chúng ta sẽ lấy mẫu từ corpus bằng cách chọn 2 đoạn văn liên tiếp, mặc dù chúng thường dài hơn nhiều so với các câu thông thường (hoặc có thể ngắn hơn).
* Chúng ta sẽ lấy mẫu sao cho sau khi kết hợp, chiều dài của mẫu kết hợp không vượt quá 512 tokens.
* Sau khi áp dụng WordPiece tokenization, chúng ta sẽ sử dụng mask cho MLM với tỷ lệ 15% như thông thường.

1.5 Quy trình Fine-tuning

Một kiến trúc tương tự được sử dụng cho cả pretrain-model và fine-tuning model. Chúng ta sử dụng cùng một tham số pretrain để khởi tạo mô hình cho các tác vụ down stream khác nhau. Trong suốt quá trình fine-tuning thì toàn bộ các tham số của layers học chuyển giao sẽ được fine-tune.

A diagram of a diagram

Description automatically generated

Đối với các tác vụ sử dụng input là một cặp sequence (pair-sequence) ví dụ như question answering thì ta sẽ thêm token khởi tạo là [CLS] ở đầu câu, token [SEP] ở giữa để ngăn cách 2 câu.

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 2 câu. 2 token này sẽ được dự báo ở output để xác định các phần Start/End Span của câu output.
* Bước 2: Các embedding vector sau đó sẽ được truyền vào kiến trúc multi-head attention trong Encoder của BERT với nhiều block (thường là 6, 12 hoặc 24 blocks tùy theo kiến trúc BERT). Ta thu được một vector output từ Encoder.
* Bước 3: Để dự báo phân phối xác suất cho từng vị trí từ trong câu, vector output của Encoder tại vị trí token [CLS] được đưa qua một lớp Fully Connected Layer, sau đó áp dụng hàm softmax để tính toán phân phối xác suất.
* Bước 4: Trong kết quả trả ra ở output của BERT, ta sẽ xác định các vị trí bắt đầu và kết thúc (Start/End Span) cho câu trả lời từ câu input.

CHƯƠNG 2 – SO SÁNH BERT VỚI CÁC BIẾN THỂ

* 1. Giới thiệu

Trong chương này, chúng ta sẽ so sánh BERT với hai biến thể nổi bật là RoBERTa và PhoBERT, đồng thời trình bày chi tiết kiến trúc, ưu điểm và nhược điểm của một trong những mô hình biến thể này.

Tóm tắt lại Bert:

* Sử dụng mô hình Transformer Encoder với 12 hoặc 24 layers.
* Training dựa trên hai nhiệm vụ chính: Masked Language Model (MLM) và Next Sentence Prediction (NSP).
* Sử dụng token embedding, segment embedding, và positional embedding.
  1. RoBERTa

Mô hình RoBERTa (A Robustly Optimized BERT) là phiên bản cải tiến của BERT. Kiến trúc RoBerta được đào tạo theo Kiến trúc của BERT (L = 24, h = 1024, A = 355M) và khác ở số tham số.

A white background with black text

Description automatically generated

So với BERT, về quy mô mô hình, khả năng tính toán và dữ liệu, đã có những cải tiến sau:

* Thông số mô hình lớn hơn (dựa trên thời gian huấn luyện được cung cấp trong bài báo, mô hình sử dụng 1024 GPU V100 để huấn luyện trong thời gian 1 ngày).
* Kích thước Batch size lớn hơn. ReBERTa sử dụng kích thước batch size lớn hơn trong quá trình đào tạo, đã thử với kích thước batch từ 256 đến 8000. Ppl là độ phức tạp(perlexity)

A black and white text

Description automatically generated

* A screenshot of a computer

  Description automatically generatedNhiều training data hơn (bao gồm 160GB plain text CC-NEWS. Trước đó BERT sử dụng 16GB dataset BookCorpus và Wikipedia tiếng Anh để training).

Ngoài ra, RoBERTa còn có những cải tiến về phương pháp training như sau:

* Loại bỏ dự đoán câu tiếp theo (Next Sentence Prediction – NSP).

A table with numbers and a number of objects

Description automatically generated with medium confidence

* Mặt nạ động (Dynamic Masking): BERT dựa trên masking ngẫu nhiên (randomly masking) và dự đoán token (predicting tokens). Khi triển khai nguyên bản BERT, trong quá trình tiền xử lý dữ liệu, kết quả sẽ thu được một static mask. Trong khi đó RoBERTa sử dụng dynamic mask, mỗi khi một sequence được đưa vào mô hình thì một mẫu mặt nạ (masking pattern) mới sẽ được tạo ra. Bằng cách này, trong quá trình nhập liên tục một lượng lớn dữ liệu, mô hình sẽ dần thích ứng với các chiến lược masking khác nhau, và học được cách biểu diễn ngôn ngữ khác nhau.
* A table with numbers and text

  Description automatically generated
* Mã hóa văn bản (text encoding): Byte-Pair Encoding (BPE) là sự kết hợp giữa các biểu diễn cấp độ ký tự và cấp độ từ, đồng thời hỗ trợ xử lý nhiều từ thông dụng trong kho dữ liệu ngôn ngữ tự nhiên. Hiện thực nguyên bản BERT sử dụng từ vựng BPE cấp độ ký tự với kích thước 30K, được học sau khi xử lý trước đầu vào bằng quy tắc mã hóa heuristic (heuristic tokenization rule). Các nhà nghiên cứu của Facebook đã không áp dụng phương pháp này, nhưng đã cân nhắc sử dụng từ vựng BPE cấp byte lớn hơn để đào tạo BERT. Từ vựng này 50K đơn vị subword, mà không cần bất kỳ xử lý trước nào khác về mã hóa (tokenization) đầu vào nào.

Dưới đây tóm tắt hai phương pháp triển khai BPE:

* Dựa trên char-level: phương pháp BERT nguyên thủy, có dược bằng cách lấy văn bản đầu vào theo phương pháp heuristically (heuristically stemming).
* Dựa trên bytes-level: sự khác biệt với char-level là bytes-level sử dụng byte thay vì ký tự unicode làm đơn vị cơ bản của từ, việc này cải thiện mã hóa mà không cần sử dụng label UNKOWN.
* Khi BPE sử dụng bytes-level, kích thước từ vựng được tăng lên từ 30,000 (char-level của BERT nguyên thủy) lên 50,000. Điều này bổ sung thêm 15 triệu tham số cho BERT-base và 20 triệu tham số cho BERT-large.
* Các nghiên cứu trước đây đã chỉ ra rằng phương pháp tiếp cận này có thể dẫn đến sự giảm hiệu suất trong việc thực hiện các nhiệm vụ phụ thuộc. Tuy nhiên, tác giả của bài báo này tin rằng những lợi ích của việc học mã hóa này sẽ vượt trội hơn sự giảm hiệu suất, và tác giả sẽ tiếp tục so sánh các phương pháp mã hóa khác nhau trong các nhiệm vụ sau này.

Kết quả dưới đây thể kết quả cải tiến hơn so với BERT-large:

A screenshot of a computer

Description automatically generated

Kết quả GLUE:

A screenshot of a report

Description automatically generated

Kết quả SquaD:

A table with numbers and text

Description automatically generated

Kết quả RACE:

A black and white text on a white background

Description automatically generated

2.3 PhoBERT

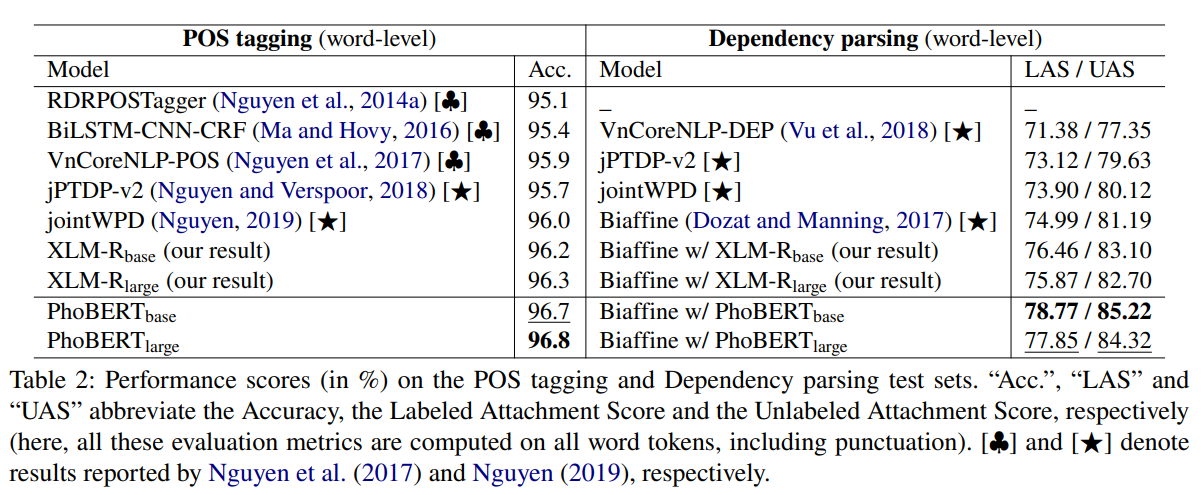
PhoBERT: Các mô hình ngôn ngữ được huấn luyện trước cho tiếng Việt Các mô hình PhoBERT được huấn luyện trước là những mô hình ngôn ngữ tiên tiến nhất cho tiếng Việt (Pho, tức là "Phở", là một món ăn phổ biến ở Việt Nam):

Hai phiên bản PhoBERT "base" và "large" là những mô hình ngôn ngữ đơn ngữ quy mô lớn đầu tiên được công khai và huấn luyện trước cho tiếng Việt. Phương pháp huấn luyện trước của PhoBERT dựa trên RoBERTa, tối ưu hóa quy trình huấn luyện trước của BERT để đạt được hiệu suất mạnh mẽ hơn.

PhoBERT vượt trội hơn so với các phương pháp đơn ngữ và đa ngữ trước đó, đạt được các thành tích tiên tiến nhất mới trên bốn tác vụ NLP tiếng Việt bao gồm: Gán nhãn từ loại (**Part-of-speech tagging - POS)**, Phân tích phụ thuộc (**Dependency parsing - Dep**), Nhận diện thực thể có tên (**Named-entity recognition – NER**) và Suy luận ngôn ngữ tự nhiên **(Natural language inference - NLI**). Kiến trúc tổng quát và kết quả thực nghiệm của PhoBERT có thể được tìm thấy trong bài báo: “**PhoBERT: Pre-trained language models for Vietnamese**” là của nhóm nghiên cứu của Nguyễn Tuấn Anh và Nguyễn Quốc Đạt tại Viện Nghiên cứu Dữ liệu lớn VinAI Research, Việt Nam.

Thông qua bài báo trên, có những điểm chú ý sau:

* Đây là một pre-trained được huấn luyện monolingual language, tức là chỉ huấn luyện dành riêng cho tiếng Việt. Việc huấn luyện dựa trên kiến trúc và **cách tiếp cận giống** **RoBERTa** của Facebook được Facebook giới thiệu giữa năm 2019.
* Tương tự như BERT, PhoBERT cũng có 2 phiên bản là "base" và "large" với 24 transformers block.
* PhoBERT được train trên khoảng 20GB dữ liệu bao gồm khoảng 1GB Vietnamese Wikipedia corpus và 19GB còn lại lấy từ Vietnamese news corpus. Đây là một lượng dữ liệu khả ổn để train một mô hình như BERT.
* PhoBERT sử dụng RDRSegmenter của VnCoreNLP để tách từ cho dữ liệu đầu vào trước khi qua BPE encoder.
* Như đã nói ở trên, do tiếp cận theo tư tưởng của RoBERTa, PhoBERT chỉ sử dụng task Masked Language Model để train, bỏ đi task Next Sentence Prediction.
* 05/2020 là BERTweet: [A pre-trained language model for English Tweets](https://github.com/VinAIResearch/BERTweet), cùng 2 tác giả PhoBert. Khá là giống với PhoBert chỉ khác là dữ liệu train thôi.
* Các thư viên hỗ trợ để triển khai PhoBert:
* fairseq: Là project của Facebook chuyên hỗ trợ các nghiên cứu và dự án liên quan đến model seq2seq. Và giúp đơn giản hóa quá trình tiền xử lý dữ liệu, định nghĩa mô hình, huấn luyện mô hình và triển khai mô hình
* fastBPE: Là package hỗ trợ tokenize từ (word) thành các từ phụ (subwords) theo phương pháp mới nhất được áp dụng cho các pretrain model NLP hiện đại như BERT và các biến thể của BERT.
* vncorenlp: Là một package NLP trong Tiếng Việt, hỗ trợ tokenize và các tác vụ NLP khác.
* transformers: Là một project của Huggingface hỗ trợ huấn luyện các model dựa trên kiến trúc transformer như BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet, T5, CTRL… phục vụ cho các tác vụ NLP trên cả nền tảng Pytorch và TensorFlow.
* Họ đã sử dụng kích thước lô (batch size) là 1024 trên 4 GPU V100, mỗi GPU có dung lượng 16GB, và tốc độ học tối đa (peak learning rate) là 0.0004 cho PhoBERT base. Đối với PhoBERT large, họ đã sử dụng kích thước lô là 512 và tốc độ học tối đa là 0.0002. Quá trình huấn luyện đã chạy trong 40 epoch, trong đó tốc độ học được điều chỉnh trong 2 epoch, tổng cộng khoảng 540K bước huấn luyện cho PhoBERTbase và 1.08M bước huấn luyện cho PhoBERT large. Chúng tôi đã huấn luyện trước PhoBERT base trong 3 tuần và trong 5 tuần.



A screenshot of a computer

Description automatically generated

2.4 Ưu và nhược điểm

|  |  |  |
| --- | --- | --- |
|  | Ưu điểm | Nhược điểm |
| BERT | Hiệu suất cao trên nhiều tác vụ NLP nhờ kiến trúc Transformer hai chiều, có khả năng xử lý tốt ngữ cảnh.  Được cộng đồng sử dụng rộng rãi, nhiều tài nguyên và công cụ hỗ trợ.  Có nhiều phiên bản tiền huấn luyện có sẵn, dễ dàng fine-tune cho các tác vụ cụ thể. | Yêu cầu tài nguyên tính toán lớn, thời gian huấn luyện dài.  Hiệu suất có thể không tốt bằng các mô hình mới hơn như RoBERTa trong một số tác vụ.  Có thể gặp khó khăn với ngữ cảnh dài và ngữ cảnh phức tạp. |
| RoBERTa | Cải tiến từ BERT với việc huấn luyện lâu hơn và trên lượng dữ liệu lớn hơn, không sử dụng nhiệm vụ NSP (Next Sentence Prediction), tối ưu hóa quá trình huấn luyện.  Hiệu suất vượt trội trên các tác vụ NLP, đặc biệt là khi so sánh với BERT.  Khả năng xử lý ngữ cảnh tốt hơn nhờ vào các cải tiến trong quy trình huấn luyện.  Được train với data lớn hơn nên mô hình có khả năng tổng quát tốt hơn và đạt được kết quả cao. | Yêu cầu tài nguyên tính toán lớn hơn BERT, thời gian huấn luyện dài hơn.  Sử dụng lượng dữ liệu và thời gian huấn luyện lớn, không phù hợp với những hệ thống có tài nguyên hạn chế.  Không sử dụng nhiệm vụ NSP, có thể là một nhược điểm trong một số tác vụ yêu cầu hiểu biết về quan hệ câu. |
| PhoBERT | Được huấn luyện đặc biệt cho tiếng Việt, đạt hiệu suất tốt hơn các mô hình đa ngữ như XLM-R trên các tác vụ NLP tiếng Việt.  Công khai phát hành mô hình giúp thúc đẩy nghiên cứu và ứng dụng trong lĩnh vực NLP tiếng Việt.  Được tối ưu hóa cho tiếng Việt, phù hợp với các yêu cầu ngôn ngữ đặc thù. | Cần nhiều tài nguyên tính toán và thời gian huấn luyện dài, đặc biệt là với mô hình PhoBERTlarge.  Chỉ tối ưu cho tiếng Việt, không sử dụng tốt cho các ngôn ngữ khác.  Chưa được ứng dụng rộng rãi, ít tài nguyên và công cụ hỗ trợ hơn so với BERT và RoBERTa. |

TÀI LIỆU THAM KHẢO

Vaswani, A., Shazeer, N., Parmar, N., & Jones, J. (2017). Attention is All You Need. arXiv. preprint **arXiv:1706.03762**.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT Pre-training of Deep Bidirectional Transformersfor Language Understanding. arXiv preprint ***arXiv:1810.04805****.*

Dat Q. Nguyen, Anh T. Nguyen (2020). PhoBERT: Pre-trained language models for Vietnamese. arXiv preprint **arXiv: 2003.00744**

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov (2019) RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint, ***arXiv:1907.11692***

Top of Form

**PHỤ LỤC**

Phần này bao gồm những nội dung cần thiết nhằm minh họa hoặc hỗ trợ cho nội dung luận văn như số liệu, biểu mẫu, tranh ảnh. . . . nếu sử dụng những câu trả lời cho một *bảng câu hỏi thì bảng câu hỏi mẫu này phải được đưa vào phần Phụ lục ở dạng nguyên bản* đã dùng để điều tra, thăm dò ý kiến; **không được tóm tắt hoặc sửa đổi**. Các tính toán mẫu trình bày tóm tắt trong các biểu mẫu cũng cần nêu trong Phụ lục của luận văn. Phụ lục không được dày hơn phần chính của luận văn