TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO: DEVELOMENTAL NEGATION PROCESSING IN TRANSFORMER LANGUAGE MODELS

MÔN NHẬP MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN

GIẢNG VIÊN HƯỚNG DẪN: ThS NGUYỄN HỒNG BỬU LONG

SINH VIÊN THỰC HIỆN:

CÁI HỮU NGHĨA 20120335

NGUYỄN HOÀNG ANH KIỆT 19120266

Thành phố Hồ Chí Minh, tháng 3 năm 2022

Mục lục

I. Thông tin chung

a/ Thông tin sinh viên

Họ và tên	Mã số sinh viên
Cái Hữu Nghĩa	20120335
Nguyễn Hoàng Anh Kiệt	19120266

b/ Bảng phân công công việc và đánh giá mức độ hoàn thành

Mã số sinh viên	Họ và tên	Công việc được phân công	Mức độ hoàn thành
20120335	Cái Hữu Nghĩa	Tìm hiểu mô hình Transformer Language Model, viết báo cáo, đọc code, chạy thử code, làm slide	100%
19120266	Nguyễn Hoàng Anh Kiệt	Tìm hiểu về Developmental Negation Processing, viết báo cáo, đọc code, làm slide	100%

II. Chi tiết đồ án

1. Lời mở đầu:

Trong cuộc sống, "phủ định" là thứ không thể thiếu trong đời sống của con người. Bởi lẽ, phủ định là phương tiện giúp con người bày tỏ sự bác bỏ, sự từ chối, không hài lòng hay phủ định sự tồn tại của một sự vật sự việc gì đó, hay có thể nói là, phủ định là một công cụ nhận thức liên quan đến sự thật và sai lầm và là điều không thể thiếu để cho phép đạt được những thành tựu độc nhất của con người. Có phủ định, con người mới có cái nhìn tổng quát về môi trường và thế giới xung quanh, giúp con người có thể sống một cuộc sống tốt đẹp hơn. Chẳng hạn, từ thời xa xưa con người không mặc quần áo. Trải qua thời gian sau đó, khi con người đã biết về cái đẹp, họ đã mặc những bộ quần áo thô sơ, đơn giản vừa giúp làm đẹp, vừa giữ ấm cơ thể. Họ nhận ra rằng, họ sẽ tốt hơn khi thay đổi theo hướng tích cực như vậy. Do đó, phủ định có vai trò vô cùng quan trọng trong đời sống hằng ngày. Phủ định được biết là được thể hiện nhiều nhất qua ngôn ngữ. Chẳng hạn như câu "I am not a teacher" (tôi không phải là một giáo viên), mang ý nghĩa rằng, người nói phủ nhận việc đó (cụ thể đó là không phải là một giáo viên).

Xử lý negation (phủ định) được biết là khó đối với các mô hình **Transformer language models,** tức là rất khó để các mô hình học máy có thể xử lý tốt được. Mặc dù các nghiên cứu trước đây đã sử dụng các công cụ của ngôn ngữ học tâm lý để thăm dò khả năng lý luận về sự phủ định của một transformer, nhưng không có nghiên cứu nào tập trung vào các loại phủ định được nghiên cứu trong tâm lý học phát triển. Vì vậy, Antonio Laverghetta Jr. và John Licato (Advancing Machine and Human Reasoning (AMHR) Lab Department of Computer Science and Engineering University of South Florida Tampa, FL, USA) đã khám phá cách các máy biến áp có thể xử lý các loại phủ định như vậy tốt như thế nào, bằng cách đóng khung vấn đề dưới dạng nhiệm vụ suy luận ngôn ngữ tự nhiên (NLI). Họ sắp xếp một bộ câu hỏi chẩn đoán cho các danh mục mục tiêu của mình từ bộ dữ liệu NLI phổ biến và đánh giá mức độ hiệu quả của một bộ mô hình đối với chúng. Họ thấy rằng các mô hình chỉ hoạt động

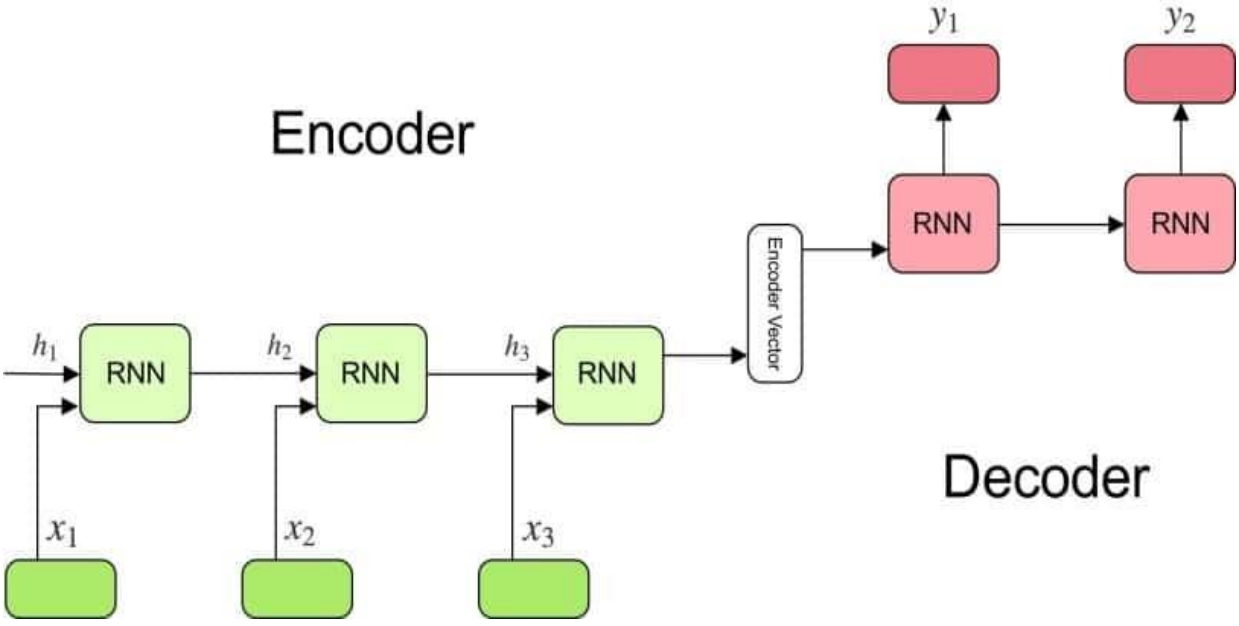
tốt hơn một cách nhất quán trên một số danh mục (category) nhất định, cho thấy sự khác biệt rõ ràng trong cách chúng được xử lý.
Trong bài báo cáo này, nhóm chúng em sẽ đi tìm hiểu các vấn đề liên quan của mô hình Transformer Language Models và tìm hiểu về phương pháp negation của nhóm tác giả Antonio Laverghetta Jr. và John Licato ở ACL năm 2022 trong việc xử lý phủ định trong tiếng Anh!

2. Tìm hiểu về mô hình Transformer Language Models:

Mô hình **Transformer Language Models** được dựa trên cơ chế **Attention (Attention-based)** nhưng ở mức phức tạp hơn.

a. Mô hình Seq2seq:

- Mô hình **Seq2seq** sử dụng **encoder** (bộ mã hóa) và **decoder** (bộ giải mã). Lớp các mô hình này đã mang đến những kết quả rõ nét trong các nhiệm vụ phức tạp như dịch máy, chú thích video,…
- Cách thức hoạt động của mô hình seq2seq:

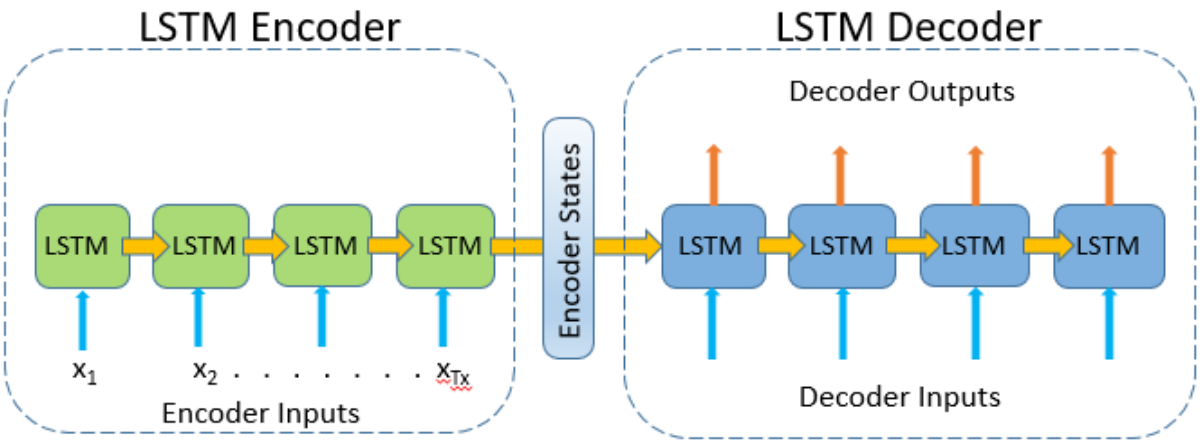


Hình 1: Kiến trúc của mô hình Seq2seq

- Mô hình Seq2seq bao gồm 3 phần: bộ mã hóa (Encoder), véc-tơ mã hóa trung gian (Encoder Vector) và bộ giải mã (Decoder) dùng để xây dựng một trình tóm tắt văn bản, với đầu vào là một chuỗi các từ (trong nội dung văn bản) và đầu ra là một bản tóm tắt ngắn (cũng là một chuỗi)
    - **Bộ mã hóa Encoder:**

- Một ngăn xếp chứa các mạng con là phần tử của RNN (hoặc các ô nhớ của LTSM hay GRU) nơi nhận tín hiệu vào của một phần tử của chuỗi đầu vào và truyền tiếp về phía cuối mạng.
- Trong bài toán hỏi đáp, chuỗi đầu vào là tập hợp tất cả các từ của câu hỏi. Mỗi từ được thể hiện bởi x_i với i là số thứ tự của từ đó.
- Các trạng thái ẩn h_i được tính theo công thức: $\boldsymbol{h_t} = \boldsymbol{f}(\boldsymbol{W^{(hh)}h_{t-1}} + \boldsymbol{W^{hx}x_t})$.
- Công thức này mô tả kết quả của một mang nơ ron hồi quy (RNN) thông thường. Như vậy ta có thể thấy, các trạng thái ẩn được tính bởi đầu vào tương ứng x_t và các trạng thái ẩn trước đó h_{t-1}.

- **Véc-tơ mã hóa trung gian (Encoder Vector):**
  - Đây là trạng thái nằm ẩn ở cuối chuỗi, được tính bởi bộ mã hóa, nó cũng được tính bởi công thức của một mạng nơ ron hồi quy (RNN).
  - Vec-tơ này có chức năng gói gọn thông tin của toàn bộ các phần tử đầu vào để giúp cho bộ mã hóa dự đoán thông tin chính xác hơn.
  - Vec-tơ này sau đó hoạt động như trạng thái ẩn đầu tiên của bộ giải mã.
- **Bộ giải mã Decoder:**
  - Một ngăn xếp các mạng con là phần tử của RNN có nhiệm vụ dự đoán kết quả đầu ra y_t tại thời điểm t.
  - Mỗi phần tử này nhận đầu vào là một trạng thái ẩn trước đó và tạo ra kết quả đầu ra cũng như trạng thái ẩn của chính nó.
  - Trong bài toán hỏi đáp, chuỗi đầu ra là tập hợp các từ của câu trả lời. Mỗi từ được biểu diễn bởi y_i với i là thứ tự của từ.
  - Các trạng thái ẩn h_i được tính theo công thức: $\boldsymbol{h_t} = \boldsymbol{f}(\boldsymbol{W^{(hh)}h_{t-1}})$.
  - Đầu ra y_t tại thời điểm t được tính bởi công thức: $y_t = softmax(W^s h_t)$. Chúng ta tính đầu ra sử dụng trạng thái ẩn tương ứng tại thời điểm hiện tại nhân với trọng số tương ứng W(s). Softmax được sử dụng để tạo ra vec-tơ xác suất giúp chúng ta xác định đầu ra cuối cùng.



Hình 2: mô hình LSTM là một RNN cho mô hình Seq2seq

- **Hạn chế của mô hình Seq2seq:** vì bộ mã hóa chuyển đổi toàn bộ đầu vào thành một vec-tơ có độ dài cố định và sau đó bộ giải mã dự đoán chuỗi đầu ra nên:
    - Kiến trúc này chỉ hoạt động tốt với các chuỗi ngắn.
    - Khó để bộ mã hóa ghi nhớ các chuỗi dài thành một vec-tơ có độ dài cố định.

b. Mô hình Attention (Attention-based):

Như đã phát biểu, mô hình Seq2seq sẽ không mang lại kết quả tốt nhất nếu độ dài chuỗi tăng lên. Lí do là bởi đối với các chuỗi dài thì RNN gặp phải 2 vấn đề là hiện tượng tiêu biến gradient (vanishing gradient) và bùng nổ gradient (exploding gradient). LTSM mặc dù đã ra đời rất lâu nhưng lại có thể khắc phục được nhược điểm của RNN. Tuy vậy việc sử dụng LTSM vẫn có những hạn chế, đó là khó huấn luyện, thời gian huấn luyện lâu do gradient path rất dài (chuỗi 100 từ có gradient như là mạng 100 lớp), transfer learning không hoạt động với LTSM, điều đồng nghĩa với một bài toán mới thì ta cần huấn luyện lại mô hình với bộ dữ liệu riêng biệt cho nhiệm vụ đã được đề ra, điều này sẽ dẫn đến rất tốn kém tài nguyên. Quay lại mô hình seq2seq với RNN thì như vậy, encoder sẽ phải "nén" toàn bộ chuỗi đầu vào thành một vector duy nhất - việc này rất khó, khi mà chuỗi đủ dài và encoder buộc phải đưa toàn bộ thông tin vào 1 vector biểu diễn duy nhất này thì chắc chắn nó sẽ "quên" thông tin nào đó (bottleneck)! Ngoài ra, decoder chỉ nhìn thấy một vector biểu diễn đầu vào duy nhất, mặc dù tại mỗi time-step thì các phần khác nhau của chuỗi vào có thể có ích hơn các phần khác. Nhưng đối với mô hình hiện tại thì decoder sẽ phải trích các thông tin liên quan này từ một vector biểu diễn duy nhất - việc

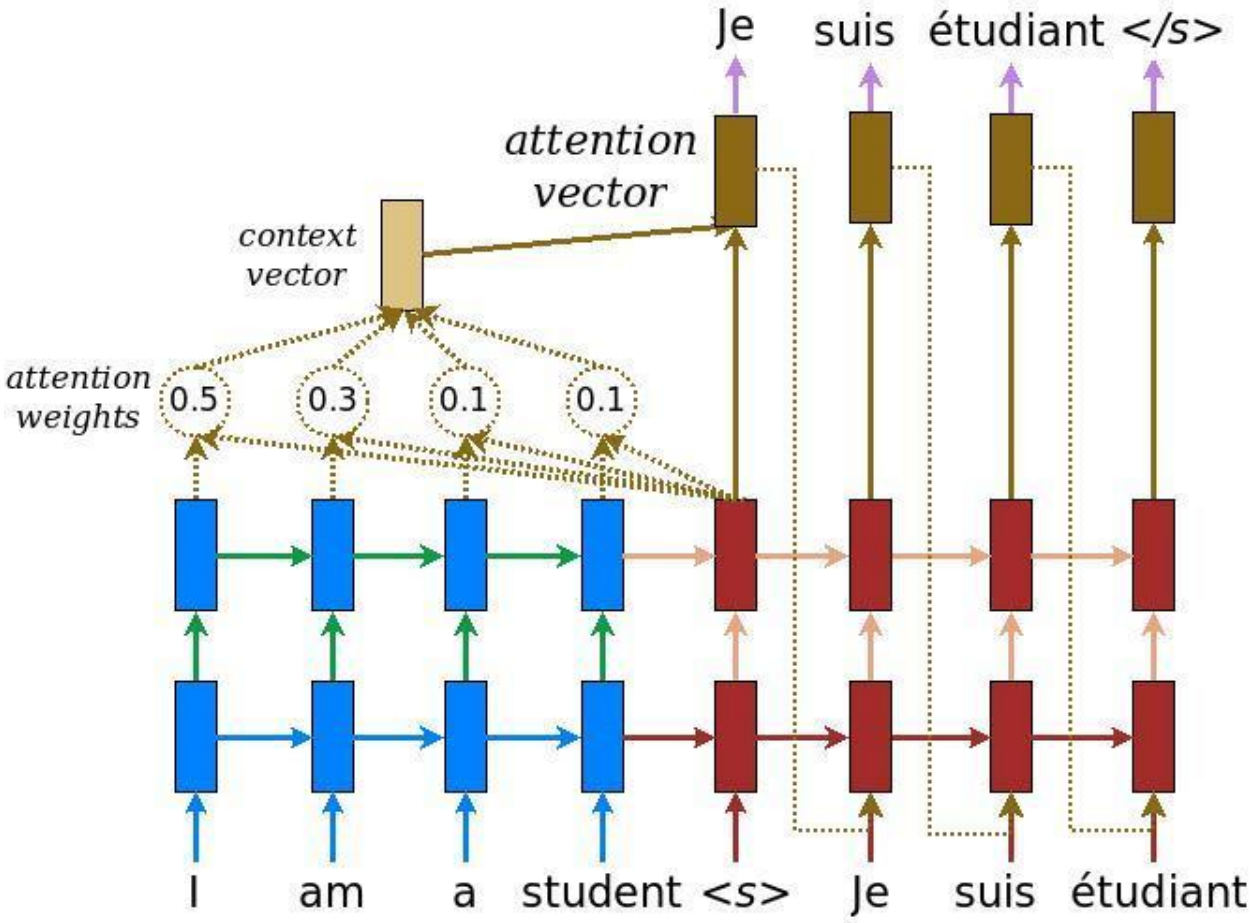
này cũng vô cùng khó. Và cơ chế Attention-based được ra đời để giải quyết những vấn đề đó.

Bây giờ hãy xét một ví dụ:

Input: "Which sport do you like most?"

Output: "I love cricket"

Từ "I" trong câu output có mối liên hệ với từ "you" trong input, tương tự từ "love" trong câu output lại có liên kết với từ "like" trong câu input. Vậy, thay vì nhìn vào tất cả những từ trong đầu vào thì chúng ta có thể tăng tầm quan trọng của một vài từ cụ thể của đầu vào có ý nghĩa đối với đầu ra. Đó là ý tưởng cơ bản của cơ chế Attention là sử dụng một vec-tơ bối cảnh có thể tương tác với toàn bộ vec-tơ trạng thái ẩn của encoder thay vì sử dụng trạng thái ẩn cuối cùng để tạo ra vec-tơ biểu diễn cho decoder.



Hình 3: Mô hình Seq2seq khi áp dụng cơ chế attention.

- Như trong hình 3, từ "I" trong tiếng Anh tương ứng với từ "je" trong tiếng Pháp. Do đó, attention layer điều chỉnh một trọng số α lớn hơn ở context vector so với các từ khác.

- Phần màu xanh đại diện cho encoder, phần màu đỏ đại diện cho decoder. Các thẻ màu xanh chính là các hidden state ht được trả ra ở mỗi unit (trong keras, khi khởi tạo RNN chúng ta sử dụng tham số return_sequences = True để trả ra các hidden state, trái lại chỉ trả ra hidden state ở unit cuối cùng). Chúng ta thấy context vector chính là tổ hợp tuyến tính của các output theo trọng số attention. Ở vị trí thứ nhất của phase decoder thì context véc tơ sẽ phân bố trọng số attention cao hơn so với các vị trí còn lại. Điều đó thể hiện rằng véc tơ context tại mỗi time step sẽ ưu tiên bằng cách đánh trọng số cao hơn cho các từ ở cùng vị trí time step. Ưu điểm khi sử dụng attention đó là mô hình lấy được toàn bộ bối cảnh của câu thay vì chỉ một từ input so với model ở hình 1 khi không có attention layer. Cơ chế xây dựng attention layer là một qui trình khá đơn giản. Chúng ta sẽ trải qua các bước:

  - Đầu tiên tại time step thứ t ta tính ra list các điểm số, mỗi điểm tương ứng với một cặp vị trí input t và các vị trí còn lại theo công thức: $\boldsymbol{score}(\boldsymbol{h_t}, \overline{\boldsymbol{h_s}})$. Ở đây $\boldsymbol{h_t}$ cố định tại time step t và hidden state của từ mục tiêu thứ t ở phase decoder, $\overline{\boldsymbol{h_s}}$ là hidden state của từ thứ s trong phase encoder.

Here, score is referred as a *content-based* function for which we consider three different alternatives:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

Besides, in our early attempts to build attention-based models, we use a *location-based* function in which the alignment scores are computed from solely the target hidden state $\boldsymbol{h}_t$ as follows:

$$\boldsymbol{a}_t = \text{softmax}(\boldsymbol{W_a} \boldsymbol{h}_t) \qquad \textit{location} \qquad (8)$$

Given the alignment vector as weights, the context vector c_t is computed as the weighted average over all the source hidden states.[6]

Hình 4: công thức tính score bằng các phương pháp khác nhau. (dot, general, concat).

- Các score được tạo ra sau bước đầu tiên chưa được chuẩn hóa. Để tạo thành một phân phối xác suất chúng ta đi qua hàm softmax khi đó chúng ta sẽ thu được trọng số attention weight. ($\boldsymbol{\alpha_{ts}}$ là phân phối attention weight của các từ trong input tới các từ ở vị trí t trong output hoặc target).

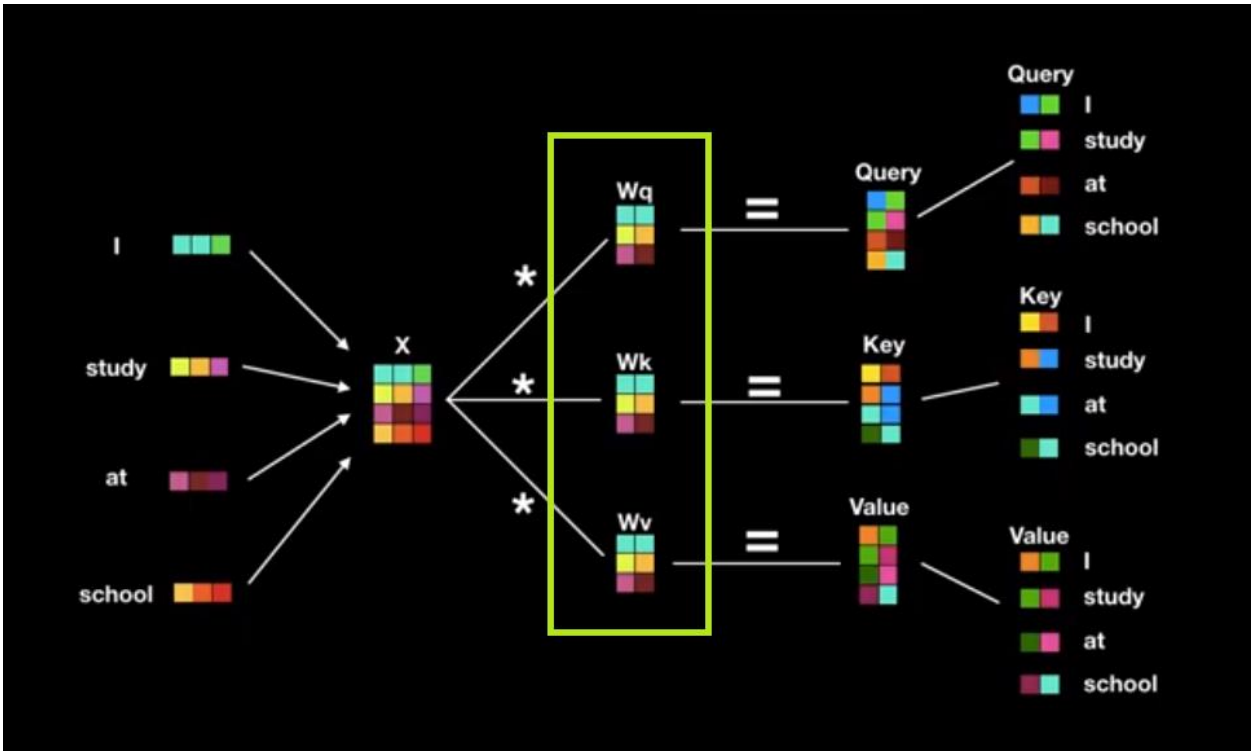
$$\boldsymbol{\alpha_{ts}} = \frac{\exp\left(score\left(h_t, \overline{h_s}\right)\right)}{\sum_{s'=1}^{S} \exp\left(score\left(h_t, \overline{h_s}\right)\right)}$$

- Kết hợp vector phân phối xác suất $\boldsymbol{\alpha_{ts}}$ với các vector hidden state để thu được context vector.

$$\boldsymbol{c_t} = \sum_{s'=1}^{S} \boldsymbol{\alpha_{ts}} \overline{\boldsymbol{h_{s'}}}$$

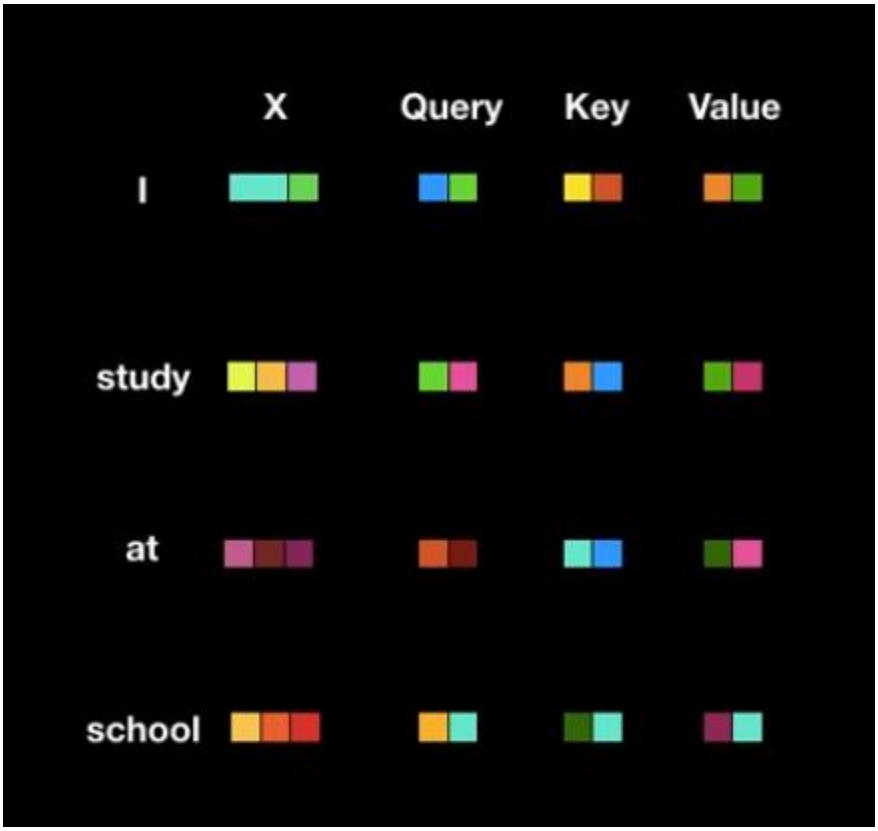
- Tính attention vector để decode ra từ tương ứng với ngôn ngữ đích. Attention vector sẽ là kết hợp của context vector và các hidden state ở decoder. Theo cách này attention vector sẽ không chỉ được học từ chỉ hidden state ở unit cuối cùng mà còn được học từ toàn bộ các vị trí khác thông qua context vector.

- Mô hình attention có 2 cơ chế quan trọng để phát triển Transformer:
  - Self-Attention:



Hình 5: Self Attention. Trong khung màu vàng là 3 ma trận $\boldsymbol{W_q, W_k, W_v}$ *chính là những hệ số mà model cần huấn luyện. Sau khi nhân các ma trận này với ma trận đầu vào X ta thu được ma trận* ***Q,K,V*** *(tương ứng với trong hình là ma trận* ***Query, Key và Value****). Ma trận* ***Query và Key*** *có tác dụng tính toán ra phân phối* ***score*** *cho các cặp từ (giải thích ở hình 7). Ma trận* ***Value*** *sẽ dựa trên phân phối* ***score*** *để tính ra véc tơ phân phối xác suất output.*

- Như vậy mỗi từ sẽ được gán bởi 3 vector **query, key và value** là các dòng của **Q, K, V**.

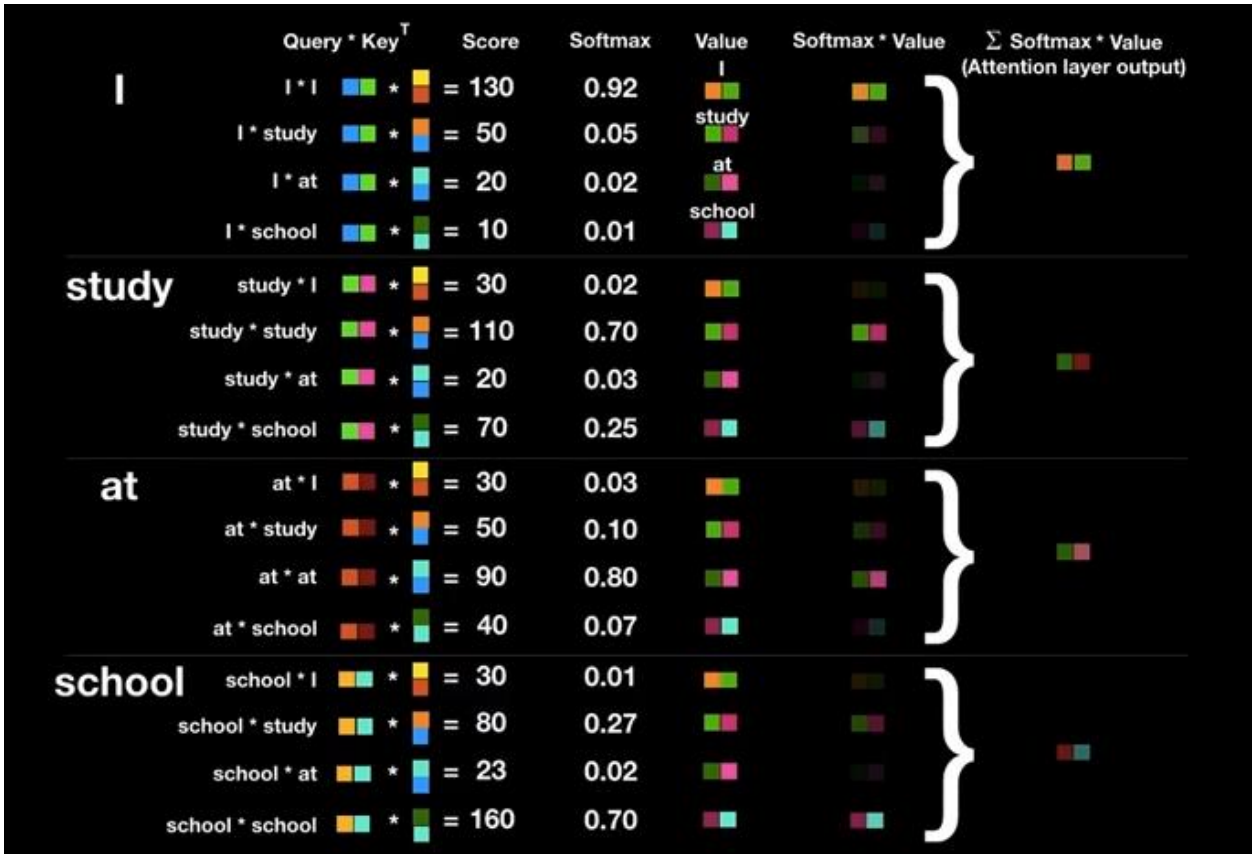


*Hình 6: Các từ input tương ứng với vector **key, query và value**.*

- Để tính ra score cho mỗi cặp từ $(\boldsymbol{W_i}, \boldsymbol{W_j})$, chúng ta sẽ tính toán dot-product giữa query với key, phép tính này nhằm tìm ra mối liên hệ trọng số của các cặp từ. Tuy nhiên điểm số sau cùng là điểm số chưa được chuẩn hóa. Do đó chúng ta chuẩn hóa bằng một hàm **softmax** để đưa về một phân phối xác xuất mà độ lớn sẽ đại diện cho mức độ attention của từ query tới từ key. Trọng số càng lớn càng chứng tỏ từ $\boldsymbol{W_i}$ trả về một sự chú ý lớn hơn đối với từ $\boldsymbol{W_j}$. Sau đó chúng ta nhân hàm softmax với các vector giá trị của từ hay còn gọi là value vector để tìm ra vector đại diện (attention vetor) sau khi đã học trên toàn bộ câu input.

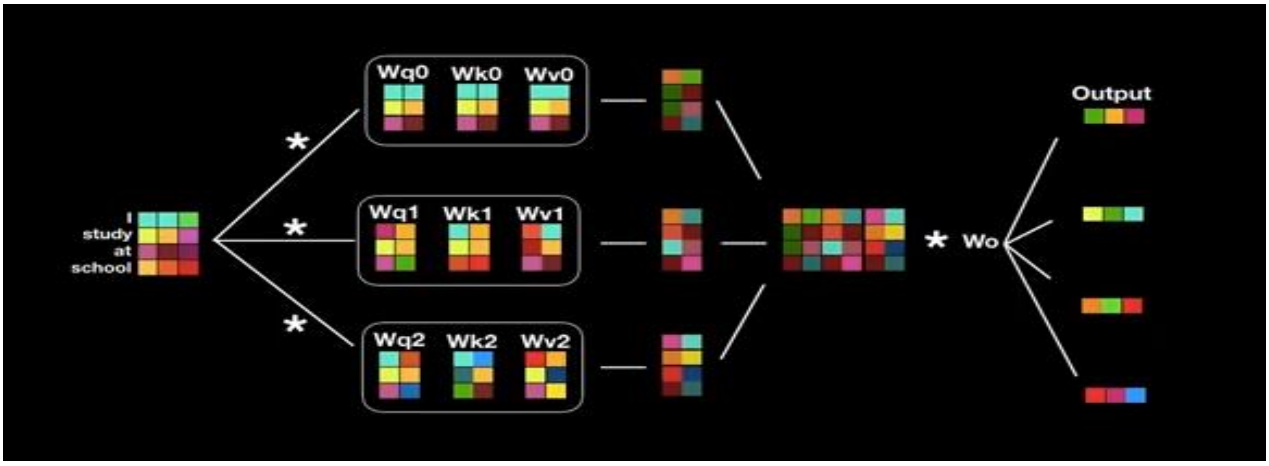
Hình 7: Quá trình tính toán trọng số attentiion và attention vector cho từ I trong câu "I study at school".

- Cuối cùng ta thu được kết quả sau quá trình tính toán tương tự cho các từ khác.



Hình 8: kết quả tính attention vector cho toàn bộ các từ trong câu.

- Multi head Attention: Như vậy sau quá trình Scale dot production chúng ta sẽ thu được 1 ma trận attention. Các tham số mà model cần tinh chỉnh chính là các ma trận $\boldsymbol{W_q}, \boldsymbol{W_k}, \boldsymbol{W_v}$. Mỗi quá trình như vậy được gọi là 1 head của attention. Khi lặp lại quá trình này nhiều lần (trong bài báo là 3 heads) ta sẽ thu được quá trình Multi-head Attention như biến đổi bên dưới:



Hình 9: Sơ đồ cấu trúc Multi-head Attention. Mỗi một nhánh của input là một đầu của attention. Ở nhánh này ta thực hiện Scale dot production và output là các matrix attention.

- Sau khi thu được 3 matrix attention ở đầu ra chúng ta sẽ concatenate các matrix này theo các cột để thu được ma trận tổng hợp multi-head matrix có chiều cao trùng với chiều cao của ma trận input.

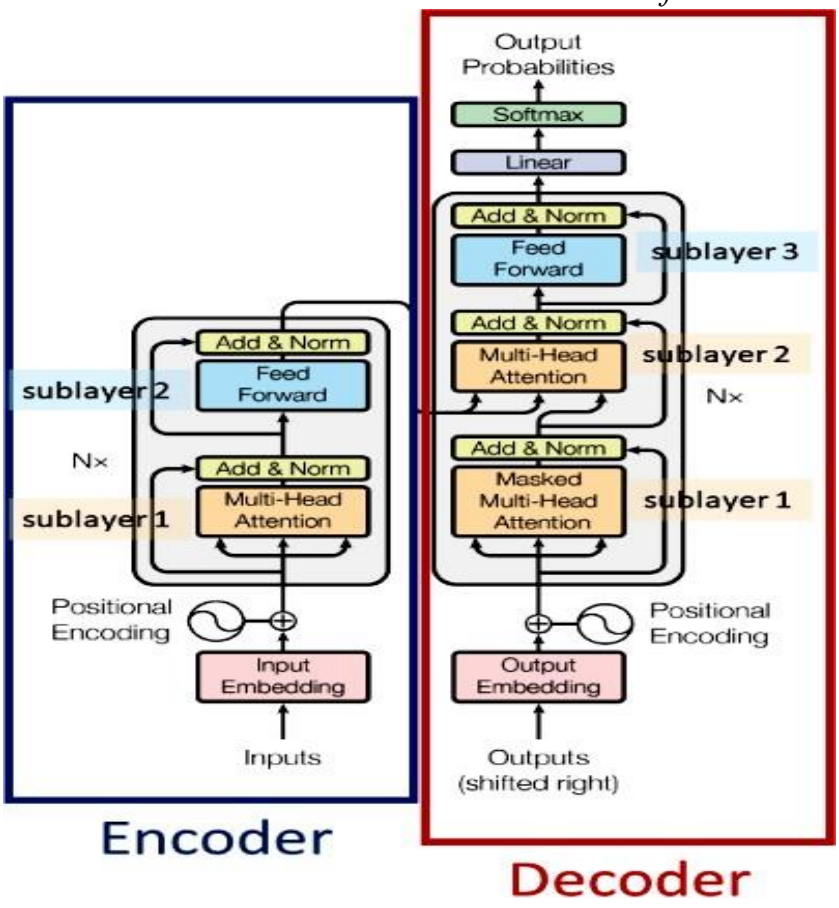
$$\boldsymbol{MultiHead(Q,K,V)} = \boldsymbol{concatenate(head_1, head_2, \dots head_h)W_0}$$

  ở đây $\boldsymbol{head_i = Attention(Q_i, K_i, V_i)}$.
- Để trả về output có cùng kích thước với ma trận input chúng ta chỉ cần nhân với ma trận W_0 chiều rộng bằng với chiều rộng của ma trận input.

c. Mô hình Transformer:

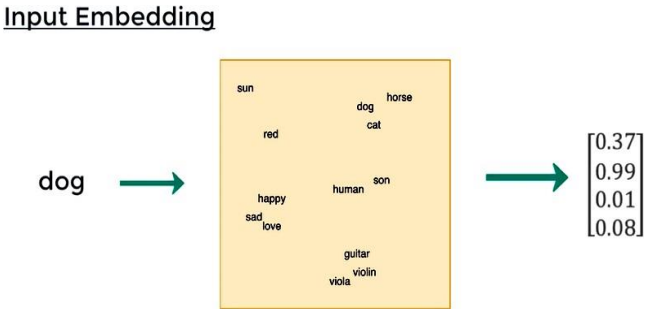
- Kiến trúc Transformers cũng sử dụng hai phần Encoder và Decoder khá giống RNNs. Điểm khác biệt là input được đẩy vào cùng một lúc, không còn khái niệm time-step trong Transformers nữa. Cơ chế Self-Attention thay thế cho "recurrent" của RNNs.

Hình 10: Kiến trúc mô hình Transformer



- **Encoder layer:**
  - **Input Embedding:** Trong các mô hình học máy, học sâu (deep

learning), dữ liệu đầu vào phải được mã hóa dưới dạng số thực hoặc phức, cũng như được biểu diễn bởi các cấu trúc toán học như vector, ma trận. Do vậy, cùng với sự phát triển của tiếpcận deep learning, các phương pháp học biểu diễn là một hướng nghiên cứu được quan tâm. Gần đây, một số mô hình học biểu diễn cho từđược đề xuất như GloVe, Fasttext, gensimWord2Vec.



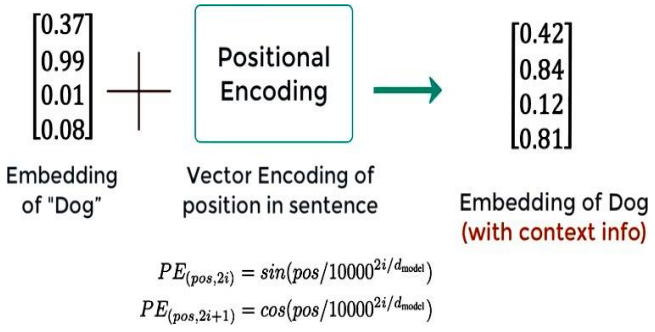
Hình 11: biểu diễn nhúng từ

- **Positional Encoding:** Word embeddings giúp biểu diễn ngữ nghĩa của một từ, tuy nhiên cùng một từ ở vị trí khácnhau của câu lại mang ý nghĩa khác nhau. Do đó Transformers có thêm một phần Positional Encoding để đưa thêm thông tin về vị trí của một từ.

$$\boldsymbol{PE_{(pos,2i)} = \sin\left(\frac{pos}{1000^{\frac{2i}{d}mod\ el}}\right)}$$

$$\boldsymbol{PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}mod\ el}}\right)}$$

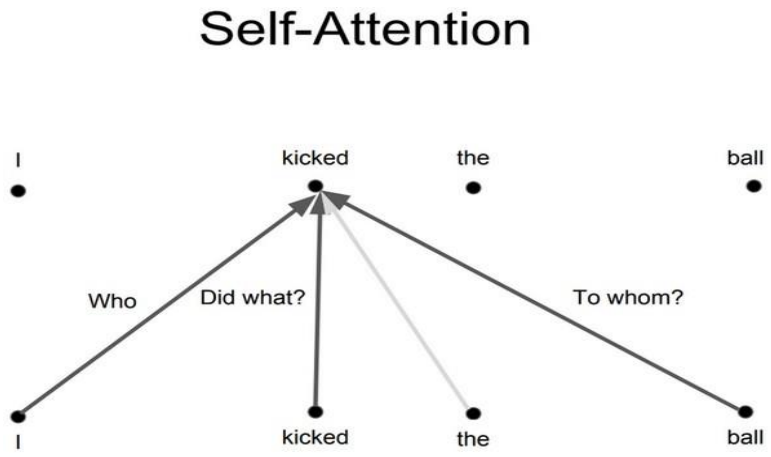
Trong đó pos là vị trí của từ trong câu, PE là giá trị phần tử thứ i trong embeddings có độ dài d. Sau đó cộng PE vector và Embedding vector.



Hình 12: mã hóa vị trí từ nhúng.

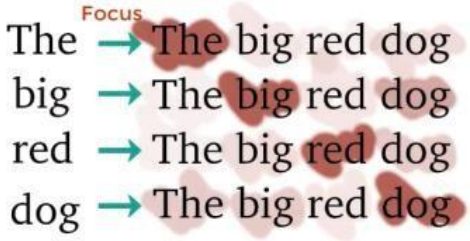
- **Self-Attention**: Self-Attention là cơ chế giúp Transformers "hiểu"

được sự liên quan giữa các từ trong một câu. Ví dụ như từ “kicked” trong câu “I kicked the ball” (Tôi đã đá quả bóng) liên quan như thế nào đến các từ khác? Liên quan mật thiết đến từ “I” (chủ ngữ), “kicked” là chính nó lên sẽ luôn “liên quan mạnh” và “ball” (vị ngữ). Ngoài ra từ “the” là giới từ nên sự liên kết với từ “kicked” gần như không có.



Hình 13: cơ chế self-attention.

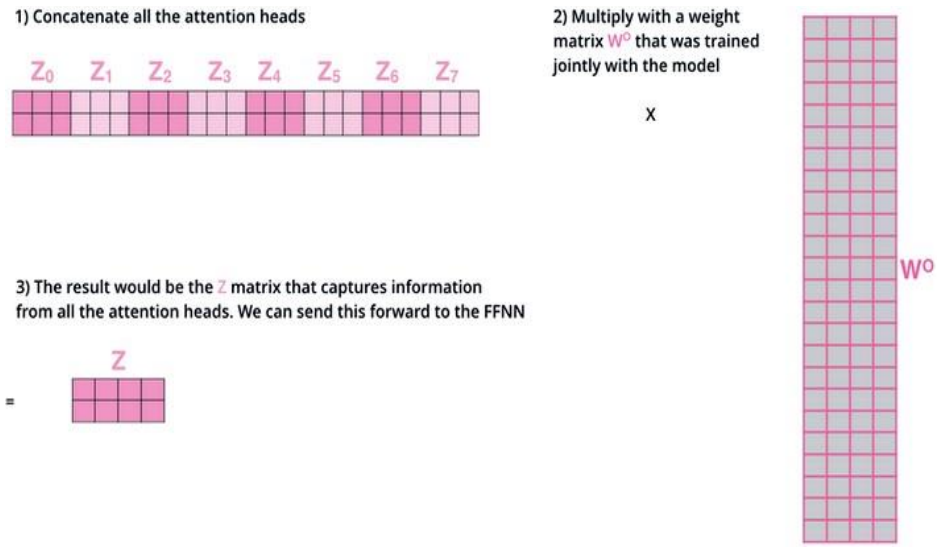
- **Multi Head Attention:** Vấn đề của Self-attention là attention của một từ sẽ luôn "chú ý" vào chính nó vì "nó" phải liên quan đến "nó" nhiều nhất. Ví dụ như sau:



Hình 14: Multi-head Attention cho câu.

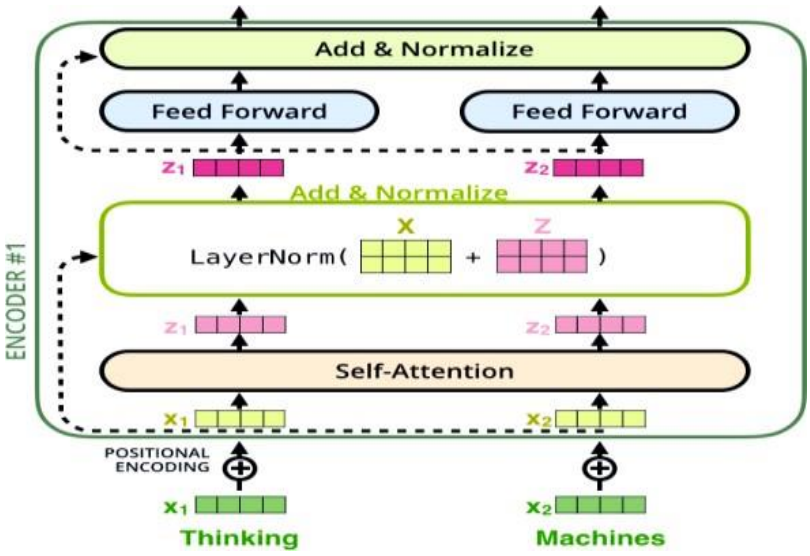
Sự tương tác giữa các từ KHÁC NHAU trong câu được thực hiện bởi Multi-head attention: thay vì sử dụng 1 Self-attention (1 head) bằng cách sử dụng nhiều Attention khác nhau (multi-head), mỗi Attention sẽ chú ý đến một phần khác nhau trong câu.

Mỗi "head" cho ra một ma trận attention riêng. Việc concat các ma trận này và nhân với ma trận trọng số WO sinh ra một ma trận attention duy nhất (weighted sum). Ma trận trọng số này được tune trong khi training.



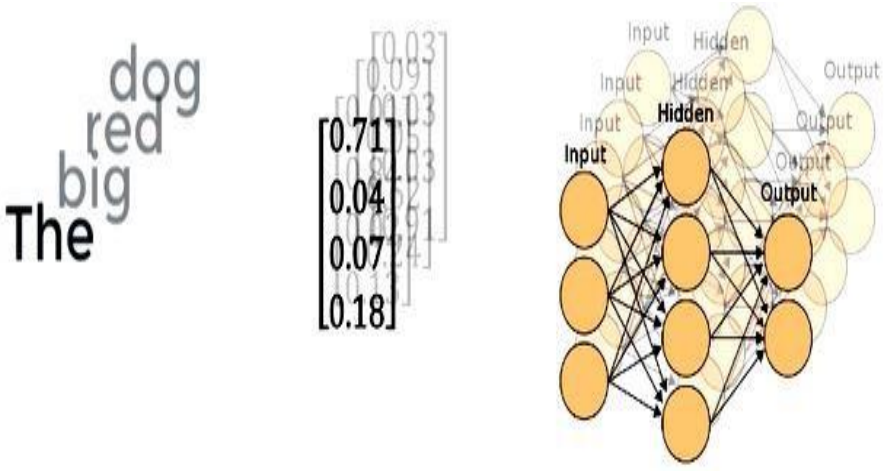
Hình 15: Quá trình concat các Attention heads.

- **Residuals:** Trong mô hình tổng quát hình 15, mỗi sub-layer đều là một residual block. Skip connections trong Transformers cho phép thông tin đi qua sub-layer trực tiếp. Thông tin này (x) được cộng với attention (z) của nó và thực hiện Layer Normalization.



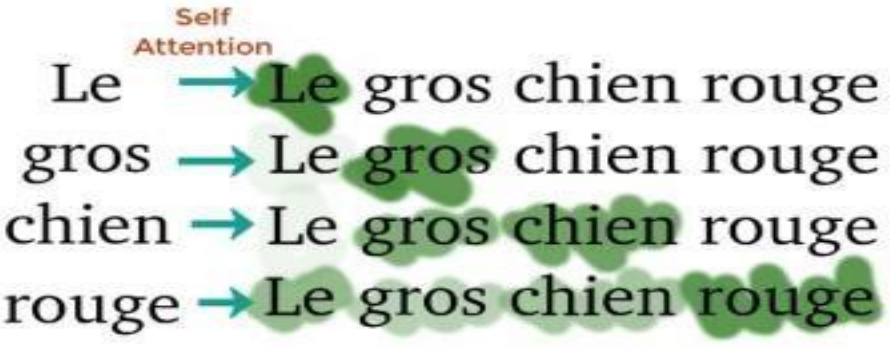
Hình 16: quá trình cộng với attention (z) và thực hiện Layer Normalization.

- **Feed forward:** sau khi được normalize, các vector z được đưa qua mạng fully connected trước khi đẩy qua Decoder. Vì các vector này không phụ thuộc vào nhau nên có thể tận dụng được tính toán song song cho cả câu.



Hình 17: tính toán song song cho cả câu.

- **Decoder layer:**
  - **Masked Multi-Head Attention:** Trong việc thực hiện bài toán English-France translation với Transformers, công việc của Decoder là giải mã thông tin từ Encoder và sinh ra những từ tiếng Pháp dựa trên những từ trước đó. Nếu sử dụng Multi-head Attention trên cả câu như ở trên Encoder, Decoder sẽ "thấy" luôn từ tiếp theo mà nó cần dịch. Để ngăn cản điều đó, khi Decoder dịch đến từ thứ i, phần sau của câu tiếng Pháp sẽ bị che lại (masked) và Decoder chỉ được phép "nhìn" thấy phần nó đã dịch trước đó.



Hình 18: quá trình tính toán song song cho câu.

  - **Quá trình Decode:** Quá trình Decode cơ bản giống với quá trình Encode, chỉ khác là Decoder sẽ decode từng từ một và input của Decoder (câu tiếng Pháp) bị masked. Sau khi masked input đưa qua sub-layer #1 của Decoder chỉ nhân với một ma trận trọng số WQ. K và V được lấy từ Encoder cùng với Q từ Masked Multi-head attention đưa vào sub-layer #2 và #3 tương tự như Encoder. Cuối cùng, các vector được đẩy vào lớp Linear (là mạng Fully Connected) theo sau bởi Softmax để cho ra xác suất của từ tiếp theo.

3. Nội dung bài báo Developmental negation processing in transformer language model

➢ Giới thiệu về xử lý phủ định

Trẻ em đã tập nói "không" để bày tỏ những điều mình không muốn, để thể hiện suy nghĩ, thái độ của bản thân từ những tuổi đầu. Do đó các nhà khoa học muốn máy tính có "tư duy" như một đứa trẻ để xử lý phủ định.

Trong bài báo ngắn này, các nhà nghiên cứu đã điều tra xem một bộ TLM có thể xử lý phủ định bằng cách đóng khung vấn đề thành một task NLI hay không.

Họ thấy rằng các model chỉ có thể đạt hiệu quả cao một cách nhất quán trên một số category nhất định và việc train trên bộ kết hợp hoặc kế thừa từ các category này cơ bản không cải thiện thêm hiệu suất.

➢ Các nghiên cứu có liên quan

Sự phủ định xuất hiện sớm trong suốt quá trình phát triển của một đứa trẻ. Từ sơ sinh đến năm lên 9, trẻ có thể hiểu và sử dụng các dạng phủ định đơn giản để thể hiện sự từ chối, sự thiếu vắng của một người/đồ vật/sự việc.

Các loại phủ định có chức năng chân lý xuất hiện muộn hơn.

Không rõ thứ tự này có phản ánh sự phụ thuộc giữa các category hay không, hay các thứ tự này có phản ánh các có mô hình ngôn ngữ (LM) nhân tạo học về phủ định.

Trong NLP, các phương pháp từ ngôn ngữ học tâm lý được dùng để thăm dò khả năng lập luận của các LM

➔ Kết quả: TLM không giống con người trong quá trình chúng xử lý phủ định.

Các TLM thường thay đổi dự đoán của chúng khi phủ định được được thêm vào hoặc xoá, thậm chí ngay cả khi phủ định không làm thay đổi mối quan hệ kế thừa.

Theo lập luận của Kruszewski và cộng sự, một phần thách thức của việc lập mô hình logic thuần tuý là một predicate thường xảy ra trong các ngữ cảnh tương tự bất kể nó có bị phủ định hay không.

Họ lập luận rằng chúng ta nên xem phủ định như là một "hàm tương tự

được phân loại", và chỉ ra rằng các mô hình phân phối có thể dự đoán các đánh giá hợp lý của con người khá tốt, ngay cả khi có sự hiện diện của phủ định.

Những nghiên cứu trên cho thấy không rõ các làm cách nào các mô hình phân phối, đặc biệt là TLM, thực sự xử lý phủ định.

➢ Thể phủ định phát triển

Các vấn đề NLI bao gồm hai câu: một tiền đề (p) và giả thuyết (h), và giải quyết một vấn đề như vậy liên quan đến việc đánh giá liệu p về mặt văn bản có dẫn đến h.

Cách làm: Sử dụng gói diaparser kiểm tra xem nội dung có chứa các dấu hiệu phủ định ("no", "not", "n't") hay không. Nếu có, kiểm tra xem cấu trúc cú pháp có tuân theo quy tắc của bất kỳ category nào hay không. Nếu có, đánh dấu nó thuộc về category đó.

Các nhà nghiên cứu chia 9000 câu hỏi từ bộ train một cách ngẫu nhiên, chia đều cho mỗi nhãn (label). Sau đó họ mô tả cách quy tắc chính xác được dung để xác định mỗi ví dụ phủ định nên được gán cho nhãn nào:

- **Possession** (PO): lemma của từ gốc là ***have***, ***has***, hay ***had***, và gốc đó là được sửa đổi trực tiếp bởi cả phủ định và động từ ***do***.
- **Existence** (EX): ***there*** xuất hiện trong văn bản và đứng trước dấu phủ định và dấu phủ định đó trực tiếp sửa đổi một cụm danh từ, hạn định, hoặc một trạng từ.
- **Labeling** (L): câu phải được bắt đầu bằng ***That*** hoặc ***It***, và gốc của câu là một danh từ được sửa đổi bởi **is** hoặc ***'s***.
- **Prohibition** (PR): câu không chứa chủ ngữ và phủ định có do đứng trước. Để không nhầm lẫn category này với category khác, chúng tôi lọc ra các trường hợp trong đó gốc chứa một trong những điểm đánh dấu rõ ràng của danh mục khác (ví dụ: thích hoặc muốn trong trường hợp bị từ chối).
- **Inability** (I): phủ định trực tiếp sửa đổi gốc của câu và ngay trước ***can*** hay ***could*** (ví dụ: ***can not do***).
- **Epistemic** (EP): gốc phải là ***remember***, ***know*** hay ***think*** và gốc phải trực tiếp được sửa đổi bởi động từ ***do***.
- **Rejection** (R): lemma của từ gốc là ***like*** hoặc ***want***, và từ gốc đó được sửa đổi bởi điểm đánh dấu phủ định.

Kết quả: Do L và PR có ít hơn 1000 ví dụ (không đủ, ít nhất phải 1500)

→ Sử dụng phương pháp tăng cường dữ liệu đơn giản sử dụng CSDL Wordnet

Cách làm: Kiểm tra xem gốc của từ có xuất hiện ở cả 2 span hay không, nếu có thì thay thế từ gốc ở cả 2 span bằng từ đồng nghĩa.

Bảng 1 sau đây hiển thị số liệu thống kê sau khi tăng cường.

Category	# Train	# Test
Possession (PO)	1053	520
Existence (EX)	5528	2723
Labeling (L)	2241	1104
Prohibition (PR)	814	400
Inability (I)	1384	682
Epistemic (EP)	1903	936
Rejection (R)	1737	856

Table 1: Summary statistics for the curated dataset.

Bảng 2 cho thấy các ví dụ được trích xuất, cùng với category của chúng. Họ thấy rằng các ví dụ được trích xuất phù hợp khá tốt với category nguyên mẫu, mặc dù trong một số trường hợp ngữ nghĩa của chúng hơi khác nhau.

Category	Premise	Hypothesis
PO	yeah you probably don't have the right temperatures...	You probably have ideal temperatures...
EX	This analysis pooled estimates...	The analysis proves that there is no link...
L	Not orders, no.	It is not orders.
PR	Two people are sitting against a building near shopping carts.	Run that way but don't run into the...
I	His manner was unfortunate, I observed thoughtfully.	I could not pick out what kind of manner he...
EP	yeah i don't know why	I know why
R	I lowered my voice...	I didn't want to be overheard.

Table 2: NLI examples extracted from each category, long examples have been trimmed to fit on one line.

VD: Xét một PR với p = ***don't miss having a flick through the albums*** và h = ***The pictures of old Madeira show a more interesting city than now***.

Mặc dù về mặt kỹ thuật đây được tính là PR, nhưng nó không hoàn toàn có cùng ngữ nghĩa với một lệnh thực tế.

Những sự mơ hồ này không dễ giải quyết, do phủ định có nhiều hình thức và có thể xảy ra ở bất kỳ đâu trong câu.

→ Các nhà nghiên cứu đã quyết định tập trung vào các hình thức phủ định dễ dàng trích xuất.

➢ Các thử nghiệm

❖ **Thử nghiệm 1**

- Đào tạo các model trên NLI_{train} trong 10 epoch, sử dụng learning rate là 1e - 5, trọng số phân rã 0,01, kích thước batch là 16 và chiều dài tối đa 175.
- Các model trên NLI_{dev} đều đạt được Tương quan Matthews ít nhất 0.6.
- Các category có thứ hạng tương tự nhau về độ chính xác. Ví dụ: L và PO nằm trong số hai category hoạt động tốt nhất, trong khi R là một trong những category hoạt động kém nhất.

➔ Cho thấy sự khác biệt rõ ràng trong cách các LM xử lý các category.

- BabyBERTa, không giống như các model khác, cũng cho thấy những điểm tương đồng mạnh mẽ hơn về cách trẻ em tiếp nhận phủ định. Trong khi R được cho là một trong những loại đầu tiên mà trẻ em tiếp nhận, BabyBERTa là mô hình duy nhất mà R là một trong các category xếp hạng cao nhất về độ chính xác.

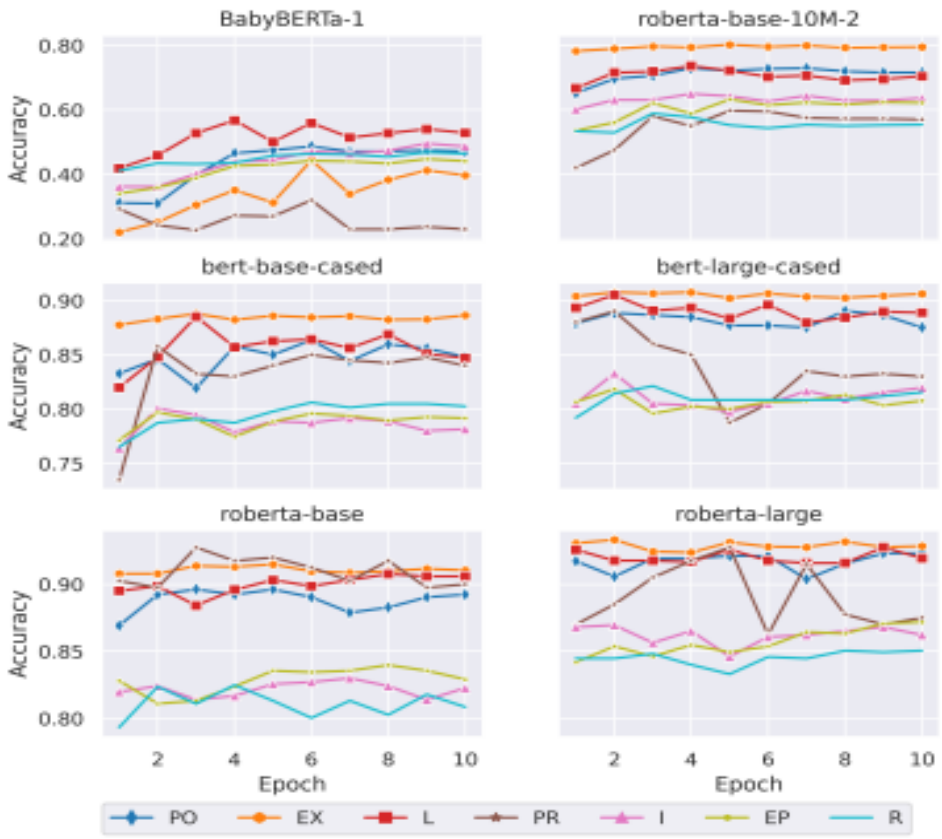


Figure 1: Performance of models finetuned on NLI_{train} for each diagnostic test set. We refer to MiniBERTa using its Huggingface model ID (*roberta-base-10M-2*).

❖ **<u>Thử nghiệm 2</u>**

- Trẻ em được mong đợi rằng sẽ phát triển một sự hiểu biết trừu tượng hơn về phủ định khi chúng được tiếp xúc với các category khác nhau.
- Được đề xuất bởi Pea (1978), người cho rằng nhiều hình thức phủ định trừu tượng phát triển từ những hình thức ít trừu tượng hơn, gợi ý rằng việc nắm vững một hình thức phủ định có thể dẫn đến chuyển đổi tích cực trên những hình thức khác.
- Kiểm tra mức độ chuyển đổi tích cực có thể thu được từ việc đào tạo trên một category phủ định, và sau đó thử nghiệm trên các loại khác.
- Sử dụng các model được train trong thử nghiệm (TN) 1, tiếp tục hoàn thiện các model này trong 25 epoch trên mỗi train set riêng biệt.
- Đánh giá các model hoàn thiện trên mỗi train set, cho phép kiểm tra tất cả các tương tác theo cặp có thể có giữa các category
- Thấy rằng chuyển đổi tích cực thường chỉ xảy ra khi một model được train trên cùng một category mà nó đang được thử nghiệm. Train trên một category khác sẽ ít hoặc không ảnh hưởng đến category mục tiêu

• BabyBERTa lại là một ngoại lệ, chuyển đổi tích cực cho hầu hết các cặp,

cho thấy mô hình đang khái quát hóa trên khắp các category

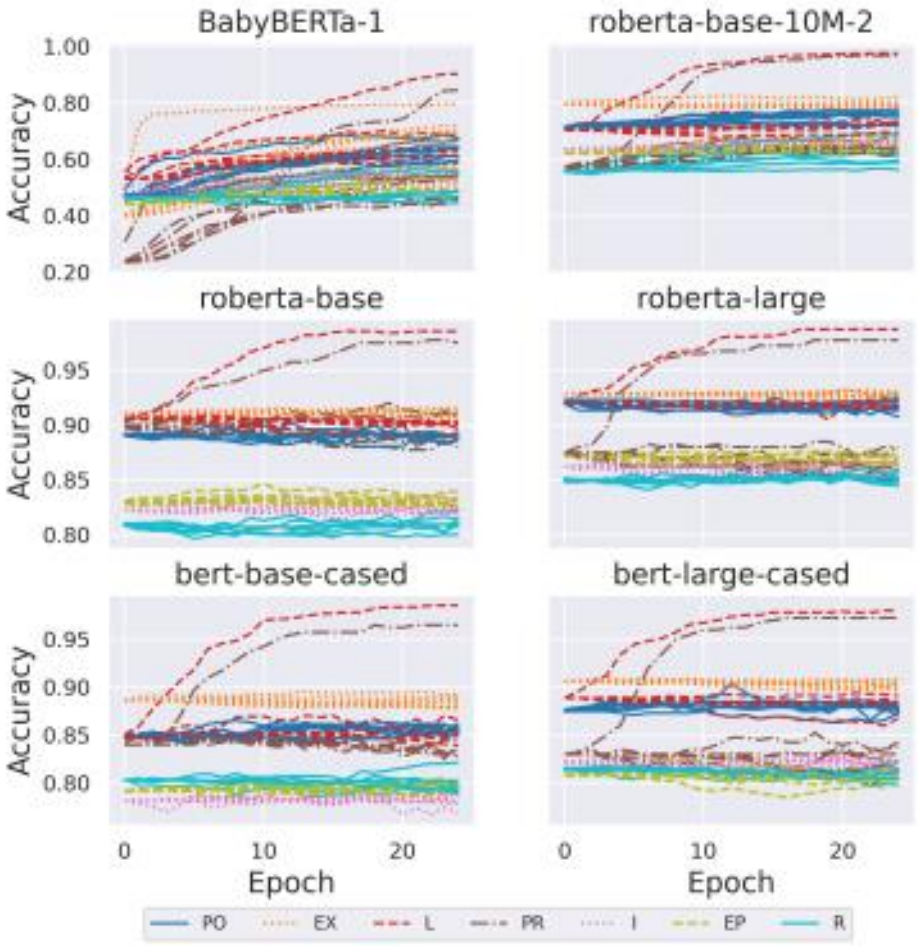


Figure 2: Accuracy of each model on every diagnostic test set, after being finetuned on every diagnostic train set. Plots are color-coded based on the target category.

❖ **<u>Thử nghiệm 3</u>**

- Dựa trên TN 2, kiểm tra hiệu suất của các model bị ảnh hưởng như thế nào khi được train trên tất cả các category theo thứ tự.
- Giả sử không có chuyển đổi tích cực tồn tại giữa các category, các nhà khoa học mong muốn xem hiệu suất của một model trên một category cụ thể chỉ cải thiện sau khi nó đã được train trên cùng category, và thậm chí train trên nhiều category khác không nên cải thiện đáng kể hiệu suất trên mục tiêu.
- Sử dụng các model từ TN 1, tinh chỉnh từng model trong 10 epoch trên mọi train set, sử dụng trình tự các category được hiển thị trong trục x của Hình 3.
- Lấy mẫu dưới tất cả các train set để có cùng số câu hỏi với PR, do đó tất cả các category đóng góp cùng một lượng dữ liệu.

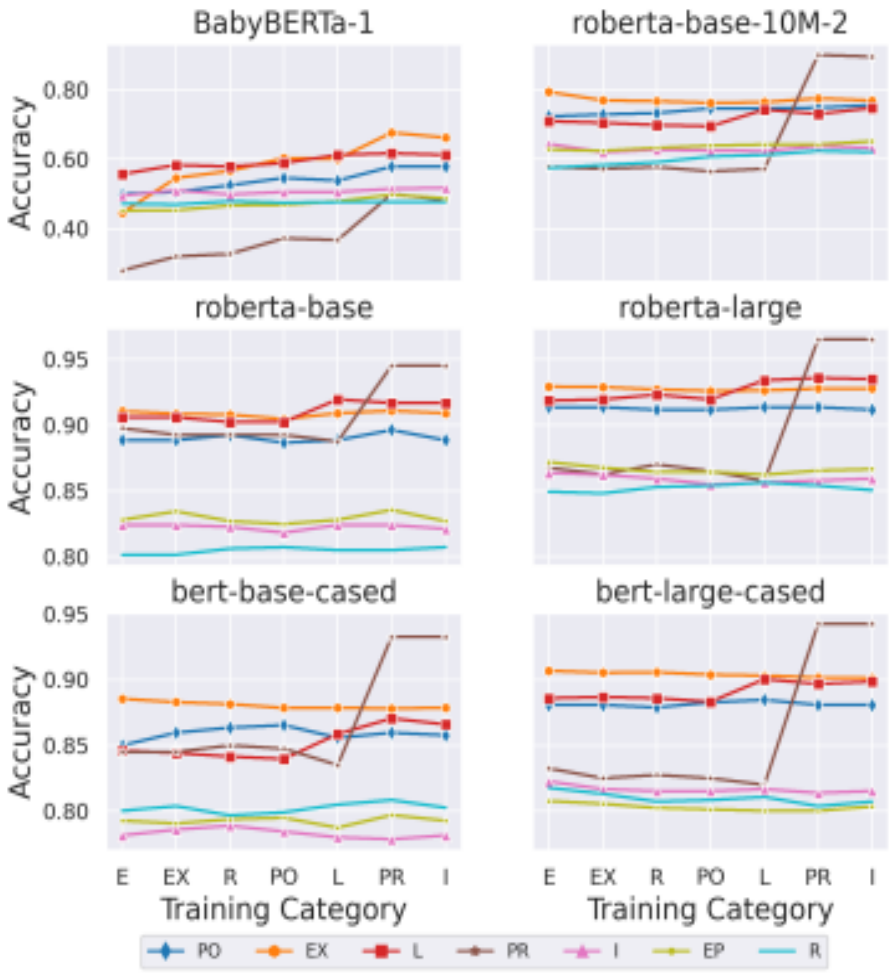


Figure 3: Results from Experiment 3. The x-axis shows the sequence of categories on which all models were trained, while the y-axis shows the accuracy obtained after being trained on a category.

- Đối với một số category, chẳng hạn như L và PR, chúng ta thấy xu hướng dự kiến. Độ chính xác tăng nhiều nhất trên các category này xảy ra bất cứ khi nào model được train trên cùng một category mà nó đang được thử nghiệm và hiệu suất giảm xuống một chút sau khi được train trên những category khác.
- Tuy nhiên, đối với các category như R, mức tăng hiệu suất tốt nhất không phải lúc nào cũng sau khi được train trên cùng một category. Đôi khi chúng ta thấy model tiếp tục cải thiện trên R sau khi được train trên R, và trong một số trường hợp, train trên R giảm hiệu suất trên R.

4. Xử lý tiền tố

- Trong bài báo Developmental negation processing in Transformer Language Models, nhóm tác giả không đề cập đến việc xử lý các tiền tố trong tiếng Anh. Các tiền tố phủ định trong tiếng Anh (negative prefixes) là phụ từ đứng trước từ gốc để làm đảo ngược ý nghĩa của từ gốc mà không cần phải thêm những từ phủ định như "not" hay "no",… Các tiền tố đó thường là UN-, IM-, IN-, IL-, IR-, DIS-,… Ví dụ như "lucky" có nghĩa là may mắn, "**un**lucky" có nghĩa là không may mắn. Một ví dụ khác, "I do **not** agree" (tôi không đồng ý) sẽ tương đương với "I **dis**agree".
- Các tiền tố khoa học thường không có quy tắc để thêm vào, nhưng đa số những tiền tố được thêm vào trước các tính từ, danh từ để làm phủ định chúng, một số khác được thêm vào động từ.
- Việc xử lý các từ phủ định có tiền tố trong việc xử lý ngôn ngữ là việc vô cùng khó, bởi lẽ ta sẽ có vô số từ phủ định khác nhau mà chúng không có nguyên tắc nhất định nào. Cách tốt nhất để xử lý đó chính là:
    - Chuyển các từ loại có tiền tố phủ định về trạng thái gốc không có tiền tố và gán nhãn các từ loại trước khi chuyển.
    - Thêm no/not/n't vào câu sao cho câu đó không thay đổi nghĩa. Ví dụ: It's **un**necessary to call the police → It's **not** necessary to call the police.
    - Xử lý như các dạng đã được đề cập.
    - Tổng hợp các loại từ từ các nhãn cho trước và đưa ra kết quả.
- Giải pháp đưa ra ở trên là giải pháp chủ quan và nhóm đang tiếp tục nghiên cứu và tìm hiểu.

III. Tài liệu tham khảo

- https://pbcquoc.github.io/transformer/
- https://trituenhantao.io/kien-thuc/hieu-ve-mo-hinh-encoder-decoder-seq2seq/
- https://viblo.asia/p/gioi-thieu-bai-toan-tom-tat-van-ban-su-dung-mo-hinh-lstm-based-seq2seq-va-co-che-attention-6J3ZgWeRZmB
- https://arxiv.org/abs/1508.04025
- https://phamdinhkhanh.github.io/2019/06/18/AttentionLayer.html
- https://sti.vista.gov.vn/tw/Lists/TaiLieuKHCN/Attachments/341919/CVv15S272022032.pdf
- https://aclanthology.org/2022.acl-short.60/
- https://viblo.asia/p/recurrent-neural-networkphan-1-tong-quan-va-ung-dung-jvElaB4m5kw
- https://nttuan8.com/bai-13-recurrent-neural-network/