MÔ HÌNH TRANSFORMERS VÀ ỨNG DỤNG TRONG XỬ LÝ NGÔN NGỮ TỰ NHIÊN

TRANSFORMERS MODEL AND APPLY IN NATURAL LANGUAGE PROCESSING

Trần Hồng Việt, Nguyễn Thu Hiền

Khoa Công nghệ thông tin, Trường Đại học Kinh tế - Kỹ thuật Công nghiệp

Đến tòa soạn ngày 20/07/2020, chấp nhận đăng ngày 12/08/2020

Tóm tắt: Xử lý ngôn ngữ tự nhiên là một lĩnh vực nghiên cứu đa dạng với nhiều nhiệm vụ riêng biệt, mỗi nhiệm vụ được đánh giá bởi bộ dữ liệu đặc thù. Trong thực tế, một số bài toán có ít dữ liệu gán nhãn chất lượng cao để huấn luyện mô hình, dẫn tới hiệu năng chưa cao. Giải quyết vấn đề này, một số mô hình sử dụng cơ chế tiền xử lý dữ liệu huấn luyện bằng việc sử dụng các mô hình biểu diễn từ chung, được huấn luyện từ một lượng lớn các dữ liệu không được gán nhãn như Word2vec, Glove hay FastText. Tuy nhiên, các mô hình đó không thể hiện được sự đại diện theo ngữ cảnh cụ thể của từ. Trong dịch máy thường sử dụng kiến trúc Recurrent Neural Networks. Mô hình này khó bắt được sự phụ thuộc xa giữa các từ trong câu và tốc độ huấn luyện chậm. Transformers là một kiến trúc mạng nơron nhân tạo được đề xuất để khắc phục các nhược điểm trên. Bài báo này, chúng tôi trình bày kiến trúc Transformers, đề xuất mô hình dịch máy sử dụng kiến trúc Transformers. Kết quả thực nghiệm trên cặp ngôn ngữ Anh - Việt và Việt - Anh chứng minh rằng, mô hình do chúng tôi đề xuất đạt hiệu năng vượt trội so với các mô hình trước đó.

Từ khóa: Trí tuệ nhân tạo, biến đổi, xử lý ngôn ngữ tự nhiên, nhúng từ, nhúng từ cảm ngữ cảnh, dịch máy.

Abstract: Natural language processing is a diverse field of research with many separate tasks, most of which are specific to each task. In fact, the lack of high quality labeling data to train the model, has a great impact on the performance and quality of natural language processing systems. To solve this problem, many models use a training data preprocessing mechanism by converting a trained generic model from large amounts of unlabeled data. For example, some models have implemented this task such as Word2vec, Glove or FastText. However, the above models do not represent the contextual representation of the word in a particular field or context. Natural language processing tasks, especially machine translation using Recurrent Neural Networks architecture. This method is difficult to capture the long dependence between words in a sentence and the training speed is slow due to sequential input processing. Transformers was born to solve these two problems. In this paper, we focus on the Transformers model and its application in natural language processing.

Keywords: Artificial intelligence, transformers, natural language processing, word embeddings, contextual word embedding, machine translation.

1. GIỚI THIỆU

Xử lý ngôn ngữ tự nhiên là một lĩnh vực nghiên cứu đa dạng với nhiều nhiệm vụ riêng biệt. Trong đó, mỗi nhiệm vụ được đánh giá bởi một bộ dữ liệu đặc thù. Để thực hiện tốt những nhiệm vụ này cần bộ dữ liệu rất lớn. Tuy

nhiên, trong thực tế hầu hết các tập dữ liệu hiện nay chỉ chứa số ít được gán nhãn bằng tay bởi con người. Trong thực tế, một số bài toán có ít dữ liệu gán nhãn có chất lượng cao để huấn luyện mô hình, dẫn tới hiệu năng của hệ thống xử lý ngôn ngữ tự nhiên tương ứng chưa cao.

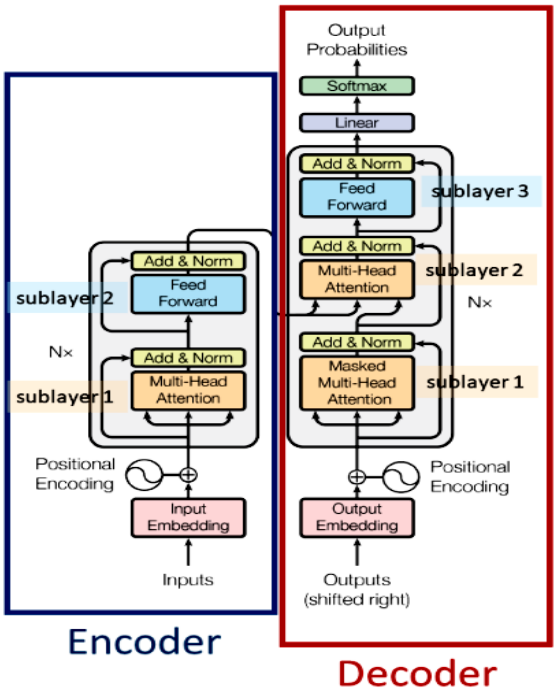
Giải quyết vấn đề này, nhiều mô hình xử lý ngôn ngữ tự nhiên sử dụng một cơ chế tiền xử lý dữ liệu huấn luyện bằng việc chuyển đổi từ một mô hình chung được huấn luyện từ một lượng lớn các dữ liệu không được gán nhãn. Ví dụ một số mô hình đã thực hiện nhiệm vụ này như *Word2vec, Glove* hay *FastText*.

Việc nghiên cứu các mô hình này sẽ giúp thu hẹp khoảng cách giữa các tập dữ liệu chuyên biệt cho huấn luyện bằng việc xây dựng mô hình tìm ra đại diện chung của ngôn ngữ sử dụng một số lượng lớn các văn bản chưa được gán nhãn lấy từ các trang web. Các pre-train model khi được tinh chỉnh lại trên các nhiệm vụ khác nhau với các bộ dữ liệu nhỏ như *Question Answering, Sentiment Analysis*,... cải thiện đáng kể về độ chính xác cho so với các mô hình được huấn luyện trước với các bộ dữ liệu này.

Tuy nhiên, các mô hình Word2vec, Glove hay FastText có những yếu điểm riêng của nó, đặc biệt là không thể hiện được sự đại diện theo ngữ cảnh cụ thể của từ trong từng lĩnh vực hay văn cảnh cụ thể. Đối với các tác vụ xử lý ngôn ngữ tự nhiên có sử dụng thông tin ngữ cảnh, đặc biệt là dịch máy sử dụng kiến trúc *Recurrent Neural Networks*, có hai thách thức chính: thứ nhất, các mô hình khó nắm bắt sự phụ thuộc xa giữa các từ trong câu; thứ hai, tốc độ huấn luyện cũng như thực thi chậm do phải xử lý dữ liệu đầu vào (input) một cách tuần tự. Kiến trúc Transformers được đề xuất để vượt qua hai thách thức kể trên. Thêm nữa, các biến thể của Transformers như BERT, GPT-2 đạt được hiệu năng vượt trội so với các mô hình trước đây (state-of-the-art) trên một số tác vụ xử lý ngôn ngữ tự nhiên. Đây là kiến trúc được biết đến nhiều trong deep learning, là cơ sở của hàng loạt các mô hình BERT khác nhau sau này. Trong bài báo này, chúng tôi trình bày về mô hình Transformers và ứng dụng trong xử lý ngôn ngữ tự nhiên.

2. KIẾN TRÚC TRANSFORMERS

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.



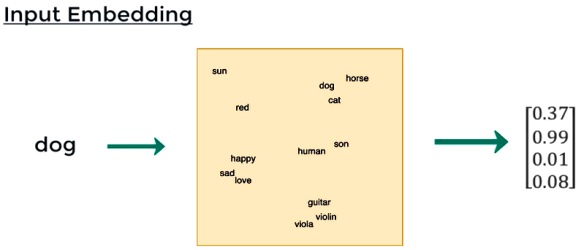
Hình 1. Kiến trúc mô hình Transformer

2.1. 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 tiếp cận 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 từ được đề xuất như GloVe, Fasttext, gensim Word2Vec.



Hình 2. Biểu diễn nhúng từ

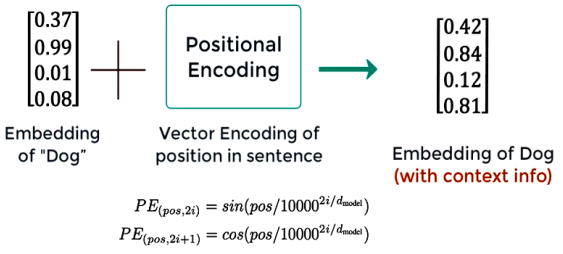
- Positional Encoding

Word embeddings giúp biểu diễn ngữ nghĩa của một từ, tuy nhiên cùng một từ ở vị trí khác nhau 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.

$$PE_{(pos,\,2i)} = \sin\left(pos/1000^{2i/d_{model}}\right)$$

$$PE_{(pos,\,2i+1)} = \cos\left(pos/10000^{2i/d_{model}}\right)$$

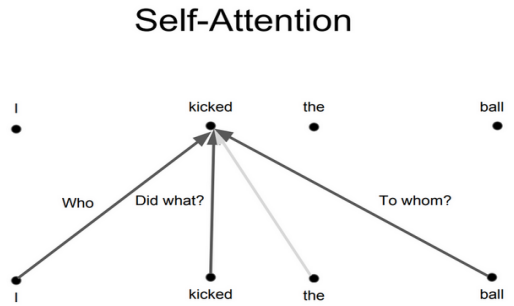
Trong đó pos là vị trí của từ trong câu, PE là giá trị phần tử thứ i trong embeddings có độ dài d_{model}. Sau đó cộng PE vector và Embedding vector.



Hình 3. 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 4. Cơ chế Self-Attention

Với kiến trúc chung, đầu vào của các môđun Multi-head Attention có ba mũi tên là ba vectors Querys (Q), Keys (K) và Values (V). Từ ba vectors này, tính vector attention Z cho một từ theo công thức sau:

$$Z = soft\max\left(\frac{Q \cdot K^T}{\sqrt{Dimension\ of\ \ vector\ \ Q,K\ \ or\ \ V}}\right) \cdot V$$

Thực hiện tính như sau:

– Bước 1: Tính ba vectơ Q, K, V, input embeddings được nhân với ba ma trận trọng số tương ứng WQ, WK, WV.

– Bước 2: Vectơ K đóng vai trò như một khóa đại diện cho từ, và Q sẽ truy vấn đến các vector K của các từ trong câu bằng cách nhân chập với những vectơ này. Nhân chập để tính toán độ liên quan giữa các từ với nhau (2 từ liên quan đến nhau sẽ có "Score" lớn).

Bước "Scale" chia "Score" cho căn bậc hai của số chiều của Q/K/V (trong hình chia 8 vì Q/K/V là 64-D vectơ) giúp giá trị "Score" không phụ thuộc vào độ dài của vectơ Q/K/V.

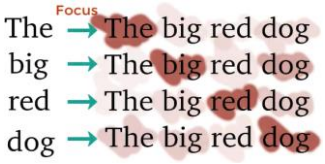
– Bước 3: Softmax các kết quả để đạt được một phân bố xác suất trên các từ.

– Bước 4: Nhân phân bố xác suất đó với vectơ V để loại bỏ những từ không cần thiết (xác suất nhỏ) và giữ lại những từ quan trọng (xác suất lớn).

– Bước 5: Cộng các vectơ V (đã được nhân với softmax output) tạo ra vector attention Z cho một từ. Lặp lại quá trình cho tất cả các từ để được ma trận attention cho 1 câu.

- Multi-head Attention

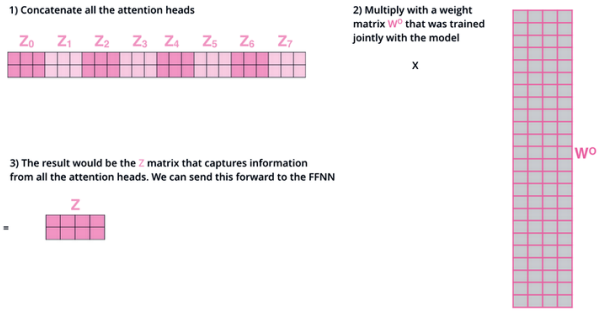
Vấn đề của Self-attention là attention của một từ sẽ luôn "chú ý" vào chính nó vì "nó" phải liên quan đến "nó" nhiều nhất. Ví dụ như sau:



Hình 5. 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.

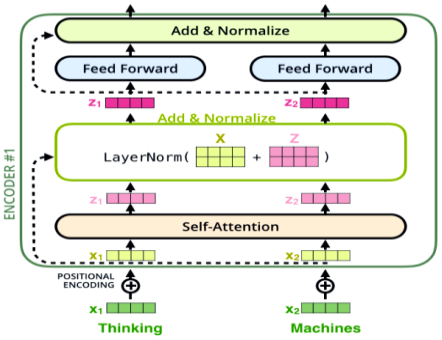
Mỗi "head" cho ra một ma trận attention riêng. Việc concat các ma trận này và nhân với ma trận trọng số WO sinh ra một ma trận attention duy nhất (weighted sum). Ma trận trọng số này được tune trong khi training.



Hình 6. Quá trình concat các Attention heads

- Residuals

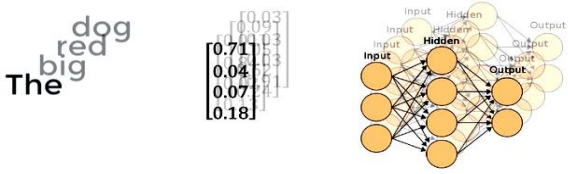
Trong mô hình tổng quát hình 6, mỗi sub-layer đều là một residual block. Skip connections trong Transformers cho phép thông tin đi qua sub-layer trực tiếp. Thông tin này (x) được cộng với attention (z) của nó và thực hiện Layer Normalization.



Hình 7. Quá trình cộng với attention (z) và thực hiện Layer Normalization

- Feed Forward

Sau khi được Normalize, các vectơ z được đưa qua mạng fully connected trước khi đẩy qua Decoder. Vì các vectơ này không phụ thuộc vào nhau nên có thể tận dụng được tính toán song song cho cả câu.



Hình 8. Tính toán song song cho câu

2.2. Decoder

- 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 từng từ tiếng Pháp dựa trên NHỮNG TỪ TRƯỚC ĐÓ. Nếu sử dụng Multi-head attention trên cả câu như ở Encoder, Decoder sẽ "thấy" luôn từ tiếp theo mà nó cần dịch. Để ngăn 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 9. Tính toán song song cho câu

- Quá trình decode

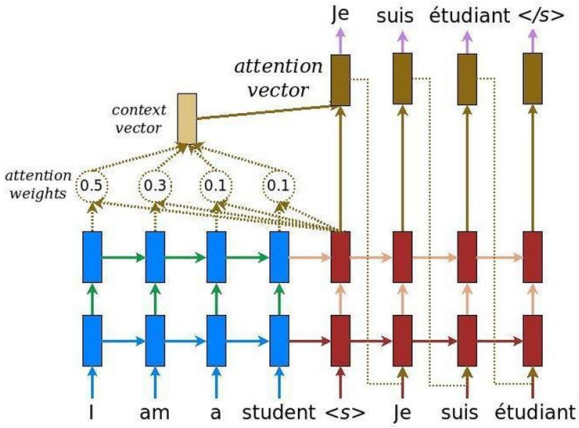
Quá trình decode cơ bản giống với encode, chỉ khác là Decoder decode từng từ một và input của Decoder (câu tiếng Pháp) bị masked. Sau khi masked input đưa qua sub-layer #1 của Decoder chỉ nhân với 1 ma trận trọng số WQ. K và V được lấy từ Encoder cùng với Q từ Masked multi-head attention đưa vào sub-layer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

Trên đây là kiến trúc về mô hình Transformers - một mô hình học sâu được sử dụng nhiều trong các ứng ụng NLP. Hiện Transformers có nhiều biến thể, cùng với pre-trained models đã được tích hợp trong rất nhiều packages hỗ trợ tensorflow, keras, pytorch đã được ứng dụng nhiều trong các bài toán xử lý ngôn ngữ tự nhiên có độ chính xác cao *(State-of-the-art)*.

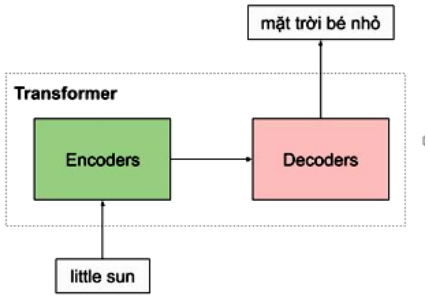
3. SỬ DỤNG TRANSFORMERS TRONG DỊCH MÁY

Kiến trúc mô hình dịch máy sử dụng Transformer giống như những mô hình dịch máy khác như hình 10. Gồm hai phần lớn là encoder và decoder. Encoder dùng để học vectơ biểu diễn của câu với mong muốn rằng vectơ này mang thông tin hoàn hảo của câu đó. Decoder thực hiện chức năng chuyển vector biểu diễn thành ngôn ngữ đích.

Trong ví dụ ở hình 11, encoder của mô hình transformer nhận một câu tiếng Anh, và encode thành một vectơ biểu diễn ngữ nghĩa của câu *little sun*, sau đó mô hình decoder nhận vector biểu diễn này, và dịch nó thành câu tiếng Việt *mặt trời bé nhỏ*.



Hình 10. Kiến trúc mô hình dịch máy seq2seq



Hình 11. Ví dụ mô hình dịch máy sử dụng Transformers

Ưu điểm của transformer là mô hình này có khả năng xử lý song song cho các từ. Như các bạn thấy, 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.

3.1. Embedding Layer với Position Encoding

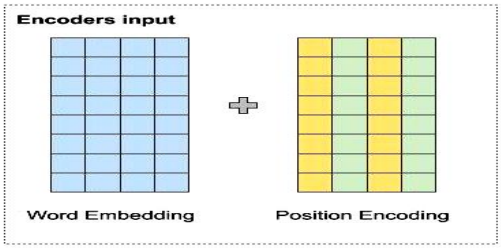
Position Encoding dùng để đưa thông tin về vị trí của các từ vào mô hình transformer. Đầu tiên, các từ được biểu diễn bằng một vector sử dụng một ma trận word embedding có số dòng bằng kích thước của tập từ vựng. Sau đó các từ trong câu được tìm kiếm trong ma trận này, và được nối nhau thành các dòng của một ma trận hai chiều chứa ngữ nghĩa của từng từ riêng biệt. Positional encoding giải quyết vấn đề đưa

thông tin vị trí các từ vào trong vector đầu vào.

Biểu diễn vị trí các từ bằng chuỗi các số liên tục từ 0,1,2,3…, n. Vấn đề là khi chuỗi dài thì số này có thể khá lớn, và mô hình sẽ gặp khó khăn khi dự đoán những câu có chiều dài lớn hơn tất cả các câu có trong tập huấn luyện. Để giải quyết vấn đề này, có thể chuẩn hóa lại cho chuỗi số này nằm trong đoạn từ 0-1 bằng cách chia cho n nhưng gặp vấn đề khác là khoảng cách giữa hai từ liên tiếp sẽ phụ thuộc vào chiều dài của chuỗi, và trong một khoảng cố định, không hình dùng được khoảng đó chứa bao nhiêu từ. Nghĩa là position encoding sẽ khác nhau tùy thuộc vào độ dài của câu đó.

- Phương pháp sinusoidal position encoding

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



Hình 12. Encoders input sử dụng Position Encoding

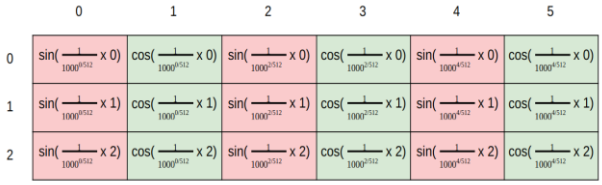
Tại vị trí chẵn, sử dụng hàm sin; vị trí lẻ sử dụng hàm cos để tính giá trị tại chiều đó

$$p_t^{'} = f(t)^i = \begin{cases} \sin(w_k * t) & if \quad i = 2k \\ \cos(w_k * t) & if \quad i = 2k+1 \end{cases}$$

Trong đó:

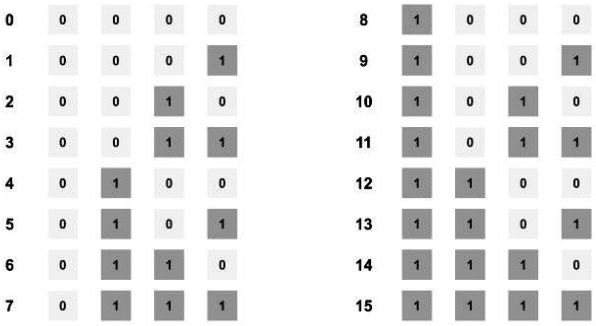
$$w_k = \frac{1}{10000^{2k/d}}$$

Hình 14 minh họa cho cách tính position encoding. Với word embedding có 6 chiều, position encoding cũng có tương ứng là 6 chiều. Mỗi dòng tương ứng với một từ. Giá trị của các vectơ tại mỗi vị trí được tính toán theo công thức ở hình 13.



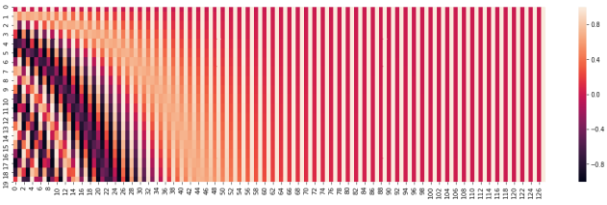
Hình 13. Tính giá trị các vectơ theo vị trí

Ví dụ: có các số từ 0-15, bit ngoài cùng bên phải thay đổi nhanh nhất mỗi 1 số, và sau đó là bit bên phải thứ 2, thay đổi mỗi 2 số, tương tự cho các bit khác.



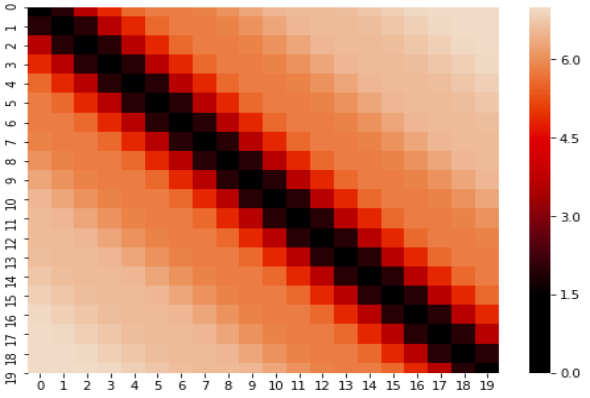
Hình 14. Cách tính position encoding

Trong công thức tính, hàm sin và cos có dạng đồ thị tần số và tần số này giảm dần ở các chiều lớn dần. Hình 15 ở chiều 0, giá trị thay đổi liên tục tương ứng với màu sắc thay đổi liên tục, và tần số thay đổi này giảm dần ở các chiều lớn hơn.



Hình 15. Thay đổi giá trị theo cách tính position encoding

Biểu diễn giá trị tương tự như cách biểu diễn các số nguyên trong hệ nhị phân, nên chúng ta có thể biểu diễn được vị trí các từ. Xem ma trận khoảng cách của các vectơ biểu diễn vị trí như hình 16. Các vector biểu diễn thể hiện được tính chất khoảng cách giữa hai từ. Hai từ cách càng xa nhau thì khoảng cách càng lớn hơn.

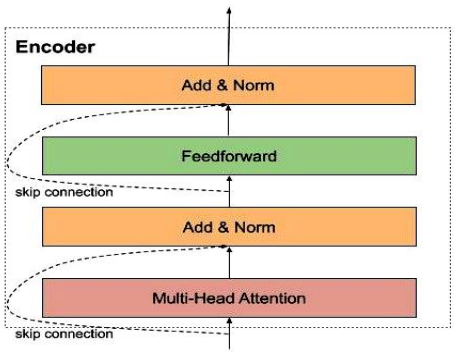


Hình 16. Biểu diễn ma trận khoảng cách của các vectơ biểu diễn vị trí

Với biểu diễn này, 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 t bằng một phép biến đổi tuyến tính dựa trên ma trận phép quay.

3.2. 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 skip connection và normalization layer. Trong đó multi-head attention là một layer mới tạo nên sự khác biệt giữa mô hình LSTM và mô hình Transformer đang được đưa ra.

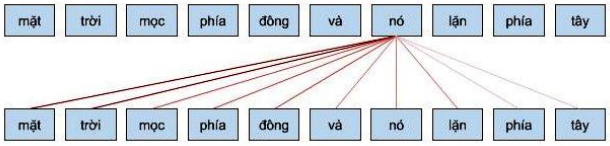


Hình 17. Biểu diễn ma trận khoảng cách của các vectơ biểu diễn vị trí

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 thật chất là self-attention, nhưng để mô hình có thể có chú ý nhiều pattern khác nhau, đơn giản là sử dụng nhiều self-attention.

- Self Attention Layer

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



Hình 18. Lớp Self Attention khi mã hóa từ

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

Với mỗi từ, cần tạo ra 3 vectơ: query, key, value vector bằng cách nhân ma trận biểu diễn các từ đầu vào với ma trận học tương ứng. Trong đó:

– Query vectơ: chứa thông tin của từ được tìm kiếm, so sánh. Giống như là câu query của google search.

– Key vectơ: biểu diễn thông tin các từ được so sánh với từ cần tìm kiếm ở trên. Ví dụ, như các trang web mà google sẽ so sánh với từ khóa tìm kiếm.

– Value vectơ: biểu diễn nội dung, ý nghĩa của các từ. Ví dụ: nội dung trang web được hiển thị cho người dùng sau khi tìm kiếm.

Để tính tương quan, cần tính tích vô hướng dựa các vectơ query và key. Sau đó dùng hàm

softmax để chuẩn hóa chỉ số tương quan trong đoạn 0-1 và tính trung bình cộng có trọng số giữa các vector values sử dụng chỉ số tương quan để tính.

Quá trình tính toán attention vector gồm:

– Bước 1: Tính ma trận query, key, value bằng cách khởi tạo 3 ma trận trọng số query, key, vectơ. Và nhân input với các ma trận trọng số này để tạo thành 3 ma trận tương ứng.

– Bước 2: Tính attention weights. Nhân 2 ma trận key, query đã tính ở bước 1 với nhau để so sánh giữa câu query và key cho việc học mối tương quan. Chuẩn hóa về đoạn [0-1] bằng hàm softmax (1 nghĩa là câu query giống với key, 0 có nghĩa là không giống).

– Bước 3: Tính output. Nhân attention weights với ma trận value. Nghĩa là biểu diễn một từ bằng trung bình có trọng số (attention weights) của ma trận value.

- Multi Head Attention

Để mô hình có thể học nhiều kiểu mối quan hệ giữa các từ với nhau, mỗi self-attention học được một kiểu pattern. Do đó để có thể mở rộng khả năng này, đơn giản là thêm nhiều self-attention (cần nhiều ma trận query, key, value). Ma trận trọng số key, query, value có thêm 1 chiều depth.

Multi head attention cho phép mô hình chú ý đến đồng thời những pattern dễ quan sát được như sau:

– Chú ý đến từ kế trước của một từ;

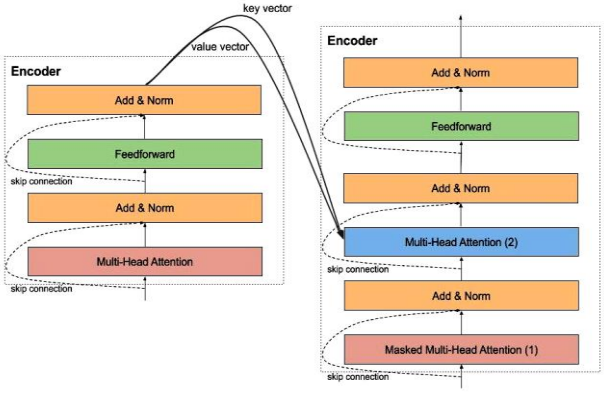
– Chú ý đến từ kế sau của một từ;

– Chú ý đến những từ liên quan của một từ.

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

3.3. Decoder

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



Hình 19. Quá trình decode

- Masked Multi Head Attention

Masked Multi Head Attention là multi head attention, có chức năng dùng để encode các từ câu đích trong quá trình dịch. Lúc cài đặt cần phải che đi các từ ở tương lai chưa được mô hình dịch đến, bằng cách nhân với một vector chứa các giá trị 0,1.

Trong decoder còn có một multi head attention khác có chức năng chú ý các từ ở mô hình encoder, layer này nhận vector key và value từ mô hình encoder, và output từ layer phía dưới. Do muốn so sánh sự tương quan giữa từ đang được dịch với các từ nguồn.

- Final Fully Connected Layer, Softmax và Loss function

Giống như nhiều mô hình khác, cần thêm một fully connected layer để chuyển output từ layer phía trước thành ma trận có chiều bằng số từ cần dự đoán. Sau đó chuyển đến softmax để

tính được xác suất của từ xuất hiện tiếp theo là bao nhiêu. Loss function là cross-entropy, giống như các mô hình phân loại khác thường sử dụng.

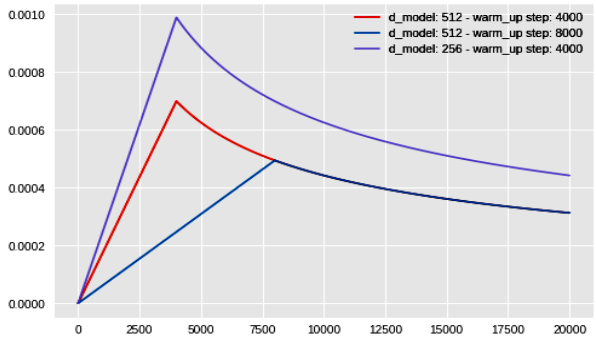
- Kỹ thuật để huấn luyện Transformer

Để huấn luyện mô hình Transformers, cần Optimizer và Label Smoothing để mô hình Transformer hội tụ được.

– *Optimizer*

Để huấn luyện mô hình transformer, sử dụng Adam, learning rate cần phải được điều chỉnh trong suốt quá trình học theo công thức sau:

$$lr\_rate = d_{d\_model}^{-0.5} * min(step\_num^{-0.5}, step\_num * warmup\_steps^{-1.5})$$

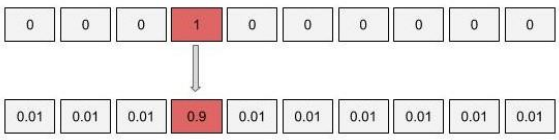


Hình 20. Learning rate

Learning rate sẽ tăng dần trong các lần cập nhật đầu tiên, các bước này được gọi là warmup step, lúc này mô hình sẽ 'chạy'. Sau đó learning rate giảm dần, để mô hình hội tụ.

– *Label Smoothing*

Với mô hình nhiều triệu tham số của Transformers, để hạn chế hiện tượng overfit, có thể sử dụng kỹ thuật label smoothing. Thay vì mã hóa nhãn là một one-hot vector, sẽ thay đổi nhãn một chút bằng cách phân bố thêm xác suất vào các trường hợp còn lại.



Hình 21. Label Smoothing

Phạt mô hình khi có thể để số epoch lớn mô hình sẽ không overfit.

- Dữ liệu thực nghiệm

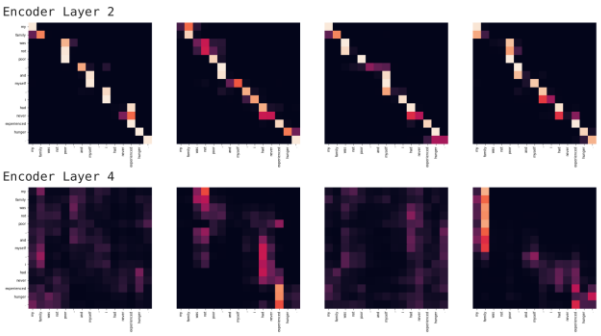
Dataset song ngữ Anh - Việt: bộ dữ liệu song ngữ được thu thập trên TEDTALK bao gồm hơn 132k câu song ngữ từ bộ dữ liệu của hội nghị IWSLT2015 và bộ 1.100.000 cặp song ngữ thu thập được từ các nguồn dữ liệu tổng hợp khác nhau.

– *Visualization*

Visualize trọng số của các mô hình sử dụng cơ chế attention. Trong mô hình transformer, visualize tại encoder và tại decoder. Có thể visualize đồng thời tại các heads của multi-head attentions, và tại layers khác nhau.

– *Encoder Visualize*

Dùng heatmap để visualize giá trị attention, sẽ cho biết khi encode một câu mô hình chú ý từ ở lân cận.



Hình 22. Encoder layer 2, layer 4

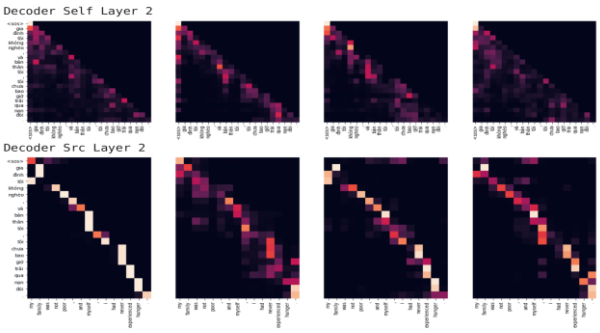
Ví dụ visualize giá trị attention của encoder layer số 2 và 4, tại các head 0,1,2,3. Nhìn vào các heatmaps ở trên, có thể thấy được rằng khi encode một từ mô hình sẽ nhìn vào các từ liên quan xung quanh. Ví dụ từ *family* có thể được mã hóa bằng 2 từ liên quan như *my* và *family*.

– *Decoder Visualize*

Ở decoder, có hai loại visualization

✓ self attention: giá trị attention khi mô hình decoder mã hóa câu đích lúc dịch;

✓ src attention: giá trị attention khi mô hình decoder sử dụng câu nguồn lúc dịch.



Hình 23. Decoder self layer 2

Ở ví dụ trên visualize decoder layer số 2, tại bốn heads 0,1,2,3. Có thể quan sát được khi encode từ đỉnh mô hình sẽ nhìn vào các từ.

4. KẾT QUẢ THỰC NGHIỆM

Qua tìm hiểu nghiên cứu chạy thực nghiệm thu được một số kết quả sau:

Bảng 1. Kết quả thực nghiệm trên kho ngữ liệu Anh - Việt

Language Pairs	Sequence2 Sequence	Transformer	
		4 heads	*1 head*
En – Vi (1.100.000)	27.7	29.17 (+1.47)	28.6 (+0.9)
En – Vi (133.000 - iwslt 15)	25.5	26.54 (+1.04)	26.47 (+0.97)

Cấu hình: d_model = 512, layers = 4, batch_size: 64, d_ff = 2048, dropout = 0.1.

– Mô hình chạy nhanh và cho kết quả tốt trên cặp ngôn ngữ Anh - Việt.

– Khó xử lý unknown dựa trên attention. Do attention vector bị chia thành n phần nhỏ.

5. KẾT LUẬN

- Mô hình Transformers có ưu điểm:

– Có khả năng thực hiện song song trong quá trình encoder.

– Cho kết quả tốt với dữ liệu câu dài.

– Hệ thống cho kết quả cao nhất *(State-of-the-art)* đối với nhiều cặp ngôn ngữ trong đó có cặp ngôn ngữ Anh - Việt.

- Nhược điểm: Khó xử lý unknown words.

Trên đây là những nội dung tìm hiểu về mô hình Transformers và ứng dụng trong xử lý ngôn ngữ tự nhiên. Ứng dụng mô hình Transformers cho bài toán dịch máy - một trong các bài toán hay và thú vị trong xử lý ngôn ngữ tự nhiên.

Kết quả đã tìm hiểu và nắm bắt được các vấn đề về mặt mô hình, kỹ thuật trong bài toán dịch máy cho ngữ liệu song ngữ Anh - Việt. Thực nghiệm với model đã xây dựng và trong thực tế đã cho kết quả và là cơ sở áp dụng cho các bài toán xử lý ngôn ngữ tự nhiên có tính phức tạp và đòi hỏi áp dụng các kỹ thuật mới trong việc xây dựng mô hình.

TÀI LIỆU THAM KHẢO

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning,* MIT Press, (2016).

[2] Mohit Sewak, Md. Rezaul Karim, Pradeep Pujari, *"Practical Convolutional Neural Networks", (2018).*

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, *Attention Is All You Need*, Submitted on 12 Jun 2017 (v1), last revised 6 Dec 2017 (this version, v5).

[4] Shaoqing Ren et al. "Faster R-CNN: *Towards Real-Time Object Detection with Region Proposal Networks*". In: IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (June 2015). DOI: 10.1109/TPAMI.2016.2577031 (cited on page 183).

[5] A Gentle Introduction to Transfer Learning for Deep Learning.

[6] URL: https://machinelearningmastery.com/transfer-learning-for-deep-learning/(cited on page 148).

[7] https://viblo.asia/p/transformers-nguoi-may-bien-hinh-bien-doi-the-gioi-nlp-924lJPOXKPM

[8] https://www.tensorflow.org/tutorials/text/transformer

[9] https://viblo.asia/p/bert-buoc-dot-pha-moi-trong-cong-nghe-xu-ly-ngon-ngu-tu-nhien-cua-google-RnB5pGV7lPG

[10] http://jalammar.github.io/illustrated-transformer

[11] https://pbcquoc.github.io/transformer/

Thông tin liên hệ: **Trần Hồng Việt**

Điện thoại: 0975486888 - Email: thviet@uneti.edu.vn
Khoa Công nghệ thông tin, Trường Đại học Kinh tế - Kỹ thuật Công nghiệp.

Nguyễn Thu Hiền

Điện thoại: 0936774362 - Email: nthien@uneti.edu.vn
Khoa Công nghệ thông tin, Trường Đại học Kinh tế - Kỹ thuật Công nghiệp.