TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Ngô Phù Hữu Đại Sơn

Sử dụng GNN biểu diễn các Embeddings cho mô hình Transformer

ĐỒ ÁN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH CHÍNH QUY

Tp. Hồ Chí Minh, tháng 07/2022

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Ngô Phù Hữu Đại Sơn- 18120078

Sử dụng GNN biểu diễn các Embeddings cho mô hình Transformer

ĐỒ ÁN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH CHÍNH QUY

NGƯỜI HƯỚNG DẪN
Nguyễn Ngọc Thảo

Tp. Hồ Chí Minh, tháng 07/2022

Lời cảm ơn

Trải qua thời gian dài học tập trong trường, đã đến lúc những kiến thức của em được vận dụng vào thực tiễn công việc. Em lựa chọn làm khóa luận tốt nghiệp để tổng hợp lại kiến thức của mình. Đề tài của em là: "Sử dụng GNN biểu diễn các Embeddings cho mô hình Transformer". Trong suốt quá trình làm khóa luận, em đã nhận được sự hướng dẫn, giúp đỡ quý báu của các thầy cô, các anh chị và các bạn. Em xin được bày tỏ lời cảm ơn chân thành tới:

Th.S Nguyễn Ngọc Thảođã hướng dẫn và truyền đạt những kinh nghiệm quý báu cho em trong suốt thời gian làm khóa luận tốt nghiệp của mình.

Em cũng cảm ơn gia đình và bạn bè đã giúp đỡ em hoàn thành tốt khóa luận.

Khóa luận của em còn những hạn chế về năng lực và những thiếu sót trong quá trình nghiên cứu. Em xin lắng nghe và tiếp thu những ý kiến của giáo viên phản biện để hoàn thiện, bổ sung kiến thức.

Em xin chân thành cảm ơn!

Mục lục

Danh sách hình

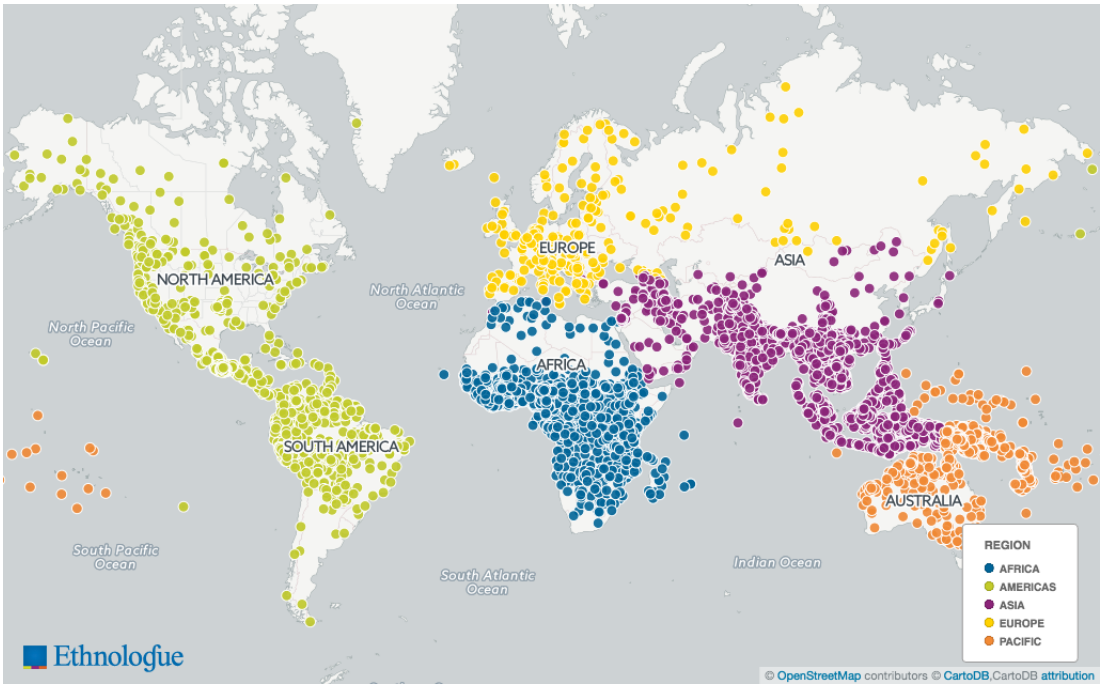
Danh sách bảng

Chương 1

Giới thiệu

1.1 Đặt vấn đề

Ngôn ngữ là một loại phương tiện giúp con người có thể giao tiếp và truyền đạt suy nghĩ, ý kiến của mình cho những người xung quanh. Theo trang Ethnologue.com, tính đến năm 2022, trên thế giới có 7151 ngôn ngữ. Vì vậy, một người không thể nào học và hiểu hết mọi ngôn ngữ trên thế giới. Từ đó, có thế độ cần thiết của việc dịch từ một ngôn ngữ sang một ngôn ngữ khác. Ngành khoa học máy tính, cụ thể hơn là xử lý ngôn ngữ tự nhiên, chúng ta cũng tâm đến bài toán trên.



Hình 1.1: Phân bố các ngôn ngữ trên thế giới

1.2 Bài toán dịch máy (Machine Translation)

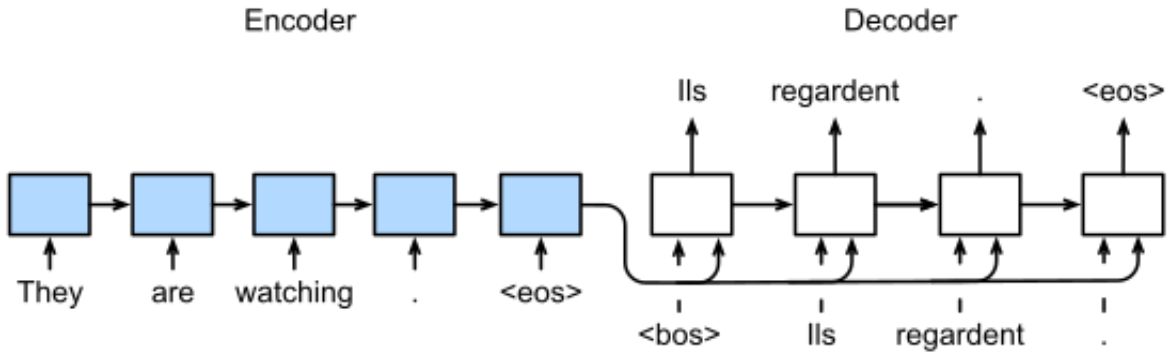
Bài toán dịch máy là một lĩnh vực trong ngành khoa học máy tính. Đầu vào được nhập vào máy tính là một đoạn văn bản từ một ngôn ngữ (ngôn ngữ nguồn). Và qua quá trình xử lý, đưa ra được đoạn văn bản tương ứng ở một ngôn ngữ khác (ngôn ngữ đích). Khác với các mô hình xử lý ngôn ngữ khác, khi chúng chỉ cần một bộ ngữ liệu của một ngôn ngữ. Các mô hình dịch máy cần ích nhất hai bộ ngữ liệu của ngôn ngữ nguồn và ngôn ngữ đích. Bộ ngữ liệu bao gồm tập các đoạn văn bản. Với mỗi đoạn văn bản từ ngôn ngữ nguồn sẽ được ánh xạ đến một đoạn văn bản có ý nghĩa tương ứng ở ngôn ngữ đích. Các bộ ngữ liệu này có thể tổng hợp từ các nguồn khác nhau: từ các bản dịch (subtitle) của các bộ phim, bản dịch sách và cả các bộ dữ liệu được thiết kế riêng cho bàn toàn dịch máy (các bản dịch từ các chuyên gia).

Để có thể hiểu hơn sâu hơn bài toán và tìm ra hướng giải quyết, ta cần phải hiểu được cách tự nhiên mà con người dịch một đoạn văn bản ngôn ngữ nguồn sang ngôn ngữ đích như thế nào. Quá trình này có thể chia làm hai bước:

- Trích xuất ngữ nghĩa (context) của ngôn ngữ nguồn thành thông tin.
- Chuyển hóa thông tin thu thập được thành ngôn ngữ đích.

1.3 Các cách tiếp cận trước

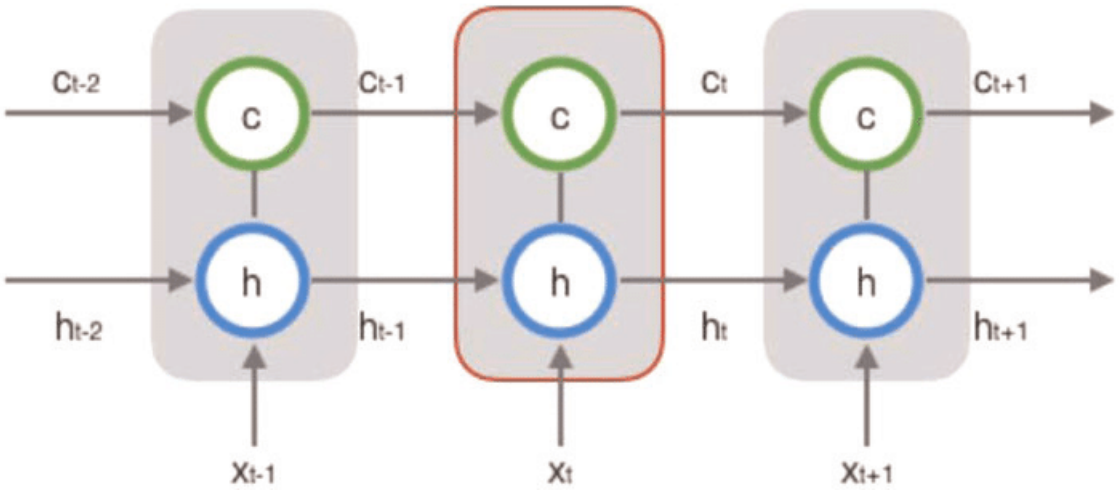
Với tính chất trên của bài toán, ta có thể thấy bài toán này là một bài toán sequence-to-sequence(Seq2Seq) và có thể giải quyết bằng kiến trúc encoder-decoder. Các mô hình sử dụng các mạng nơ ron hồi quy (Recursive Neural Network) sử dụng cơ chế long-short-term memory và cả cơ chế attention.



Hình 1.2: Mô hình Seq2Seq

Các mô hình hồi quy (Recurrent model) cho các các kết quả tốt trong bài toán dịch máy. Tuy nhiên, các mô hình này lại sử dụng cơ chế hồi quy. Ở mỗi bước tính toán, mô hình sử dụng thông tin ẩn (hiden state) được tổng hợp từ đầu văn bản đến hiện tại h_t để để làm đầu vào tính toán. Quá trình này lặp lại cho mỗi bước. Từ đó, ta có thể thấy mô hình toán toán bước tiếp theo phải phụ thuộc vào bước trước đó, dẫn đến không thể song

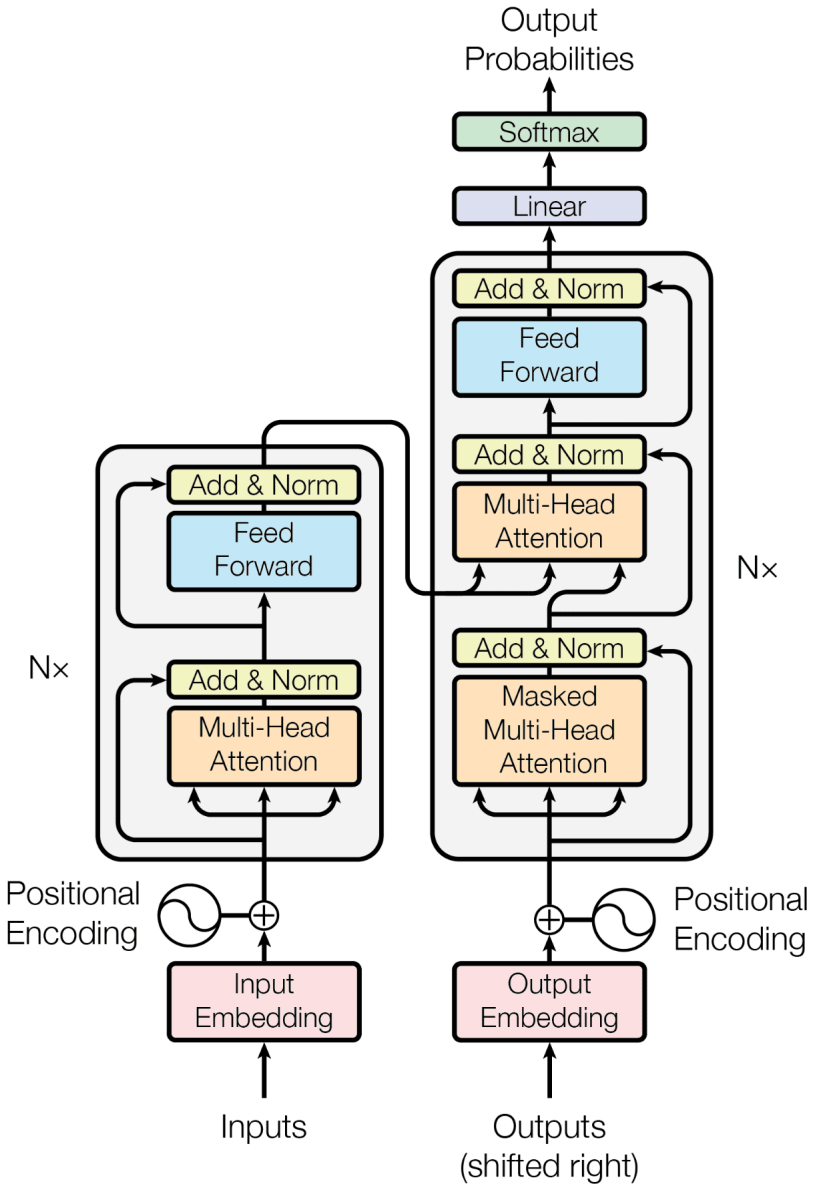
song quá trình tính toán này được. Điều này khiến cho việc tối ưu thời gian huấn luyện lẫn hiệu quả tính toán của mô hình trước nên khó khăn.



Hình 1.3: Kiến trúc RNN với các hidden state

1.4 Transformer

Transformer - một phương pháp dựa hoàn toàn trên cơ chế attention, bỏ qua các cấu trúc của mạng nơ ron hồi quy và mạng nơ ron tích chập phức tạp, giúp đơn giản hóa mô hình những vẫn thể hiện được độ hiệu quả của mô hình. Theo **Paper attention is all your need**, mô hình cho kết quả 41.8 BLEU khi huấn luyện trên tập WMT 2014, cao hơn tất cả các mô hình dịch máy trước đó.



Hình 1.4: Tổng quan kiến trúc của Transformer

Nhờ không sử dụng kiến trúc RNN, Transformer tránh được nhược điểm chí mạng của các mô hình loại này. Dựa hoàn toàn vào cơ chế attention, cụ thể hơn là giới thiệu kiến trúc multihead-attention giúp việc song song tính toán mô hình. Từ đó mà tăng độ hiệu quả huấn luyện cũng như hiệu quả tính toán.

1.5 Tokenization & Word Embeddings

Trong bài toán dịch máy, các đoạn văn bản thô cần phải được tiền xử lý, chuyển đổi thành các dạng dữ liệu mà máy tính của thể hiểu được. Quá trình đó được thực hiện như sau:

- Tokenization là quá trình tách đoạn văn bản ra thành các từ thành phần (token). Các token này đã được quy định sẵn trong một tập các từ đã biết trước (vocabulary). Với các từ lạ, không thuộc trong vocabulary sẽ được đánh dấu là UNK (unknown).

- Các token sau khi được tách ra vẫn chưa thể đưa vào mô hình do chúng vẫn ở dạng dữ liệu mà các mô hình chưa thể hiểu. Do đó, với mỗi token, ta cần chuyển chúng thành các vector N chiều (N chọn trước) mang tính chất và đại diện cho từ đó. Với mỗi chiều của vector được biểu diễn bằng một số thực. Các vector này được gọi là các Word Embeddings. Các Word Embeddings sẽ là đầu vào của mô hình. Một số phương pháp để tính toán các Word Embeddings trước đây gồm: ...

1.6 Graph Convolution Network (GCN)

Khai thác quan hệ giữa các điểm dữ liệu với nhau là một đề tài được nhiều sự quan tâm trong học máy. Trong các mô hình học sâu trước đây, ta chỉ có thể trích xuâts được các mối quan hệ thông thường từ các bộ dữ liệu Euclidian.

Trong thực tế, không phải mọi dữ liệu đều biểu diễn ở dạng Euclidian. Do đó, để khai thác được thông tin từ các bộ dữ liệu đồ thị (non-Euclidian), ta cần sử dụng các mô hình Graph Neural Network (GNN). Trong vài năm qua, nhiều biến thể của đã được phát triển. Trong đó, Graph Convolution Network (GCN) là một biến thể quan trọng được xem như là biến thể cơ bản nhất của GNN.

GCN ứng dụng phép tích chập được giới thiệu trong convolution layer của CNN:

- Đối với phép tích chập trên CNN, một đoạn dữ liệu của lớp hiện tại sẽ được nhân vô hướng (tích chập) với một bộ lọc (*filter* hoặc *kernel*). Bộ lọc sẽ trượt trên bộ dữ liệu và các đầu ra của phép tích chập sẽ được truyền vào lớp tiếp theo của mạng.

- Với phép tích chập trên GCN. Ta xét từng nút(node) của đồ thị. Với mỗi nút, ta nhóm nút này với các nút kề của chính nó lại và thực hiền phép tích chập tương tự với một màn lọc. Do số lượng nút kề của mỗi nút là khác nhau, nên kích thước của filter sẽ không cố định. Phép tích chập sẽ được thực hiện trên tất cả các nút của đồ thị.

1.7 Word GCN

Ngôn ngữ là một dạng dữ liệu non-Euclidian. Do đó, trích xuất thông tin của chúng bằng các mô hình GCN được kì vọng là có kết quả tốt hơn các phương pháp trước đó. Các loại thông tin có thể khai thác bao gồm:

- Thông tin về cú pháp (syntactic) của ngôn ngữ. Các từ trong một câu sẽ sẽ có các quy định về các mối quan hệ của chúng trong câu. Biểu diễn các mối quan hệ này là các cạnh còn các từ trong câu là các nút của đồ thị. Từ đó mà ta có thể khai thác được các đặc trưng cú pháp của các từ. Mô hình SynGCN sẽ giúp ta huấn luyện được các embeddings mang các thông tin trên

- Thông tin về ngữ nghĩa (semantic) của ngôn ngữ. Ý nghĩa của các từ sẽ có các quan hệ với nhau như: đồng nghĩa, trái nghĩa, kế thừa,... Nhờ các mối quan hệ đó, ta có thể biểu diễn được đồ thị ngữ nghĩa của các từ trong bộ từ vựng. Từ đó mà ta có thể khai thác được các đặc trưng ngữ nghĩa của các từ. Mô hình SemGCN sẽ giúp ta huấn luyện được các embeddings mang thông tin ngữ nghĩa.

1.8 Mô hình đề xuất

Ở mô hình đề xuất, tôi muốn tích hợp các embeddings được huấn luyện thông qua WordGCN để đưa vào mô hình các mô hình dịch máy. Mô hình dịch máy để tích hợp vào là transformer do tính hiệu quả cao của mô hình transformer và khả năng tính toán song song của nó.

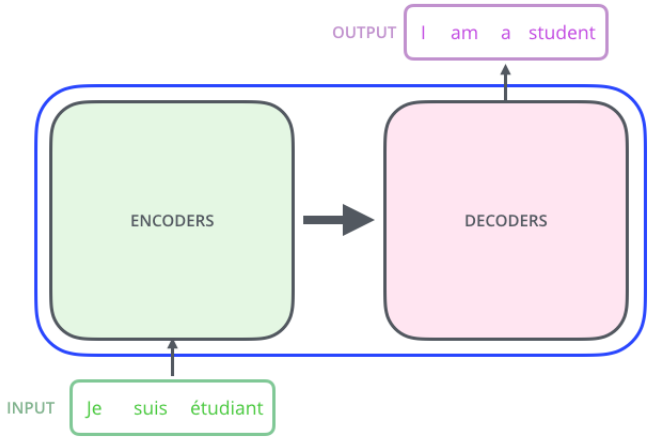
Chương 2

Tổng quan lý thuyết

2.1 Mô hình Transformer dịch máy

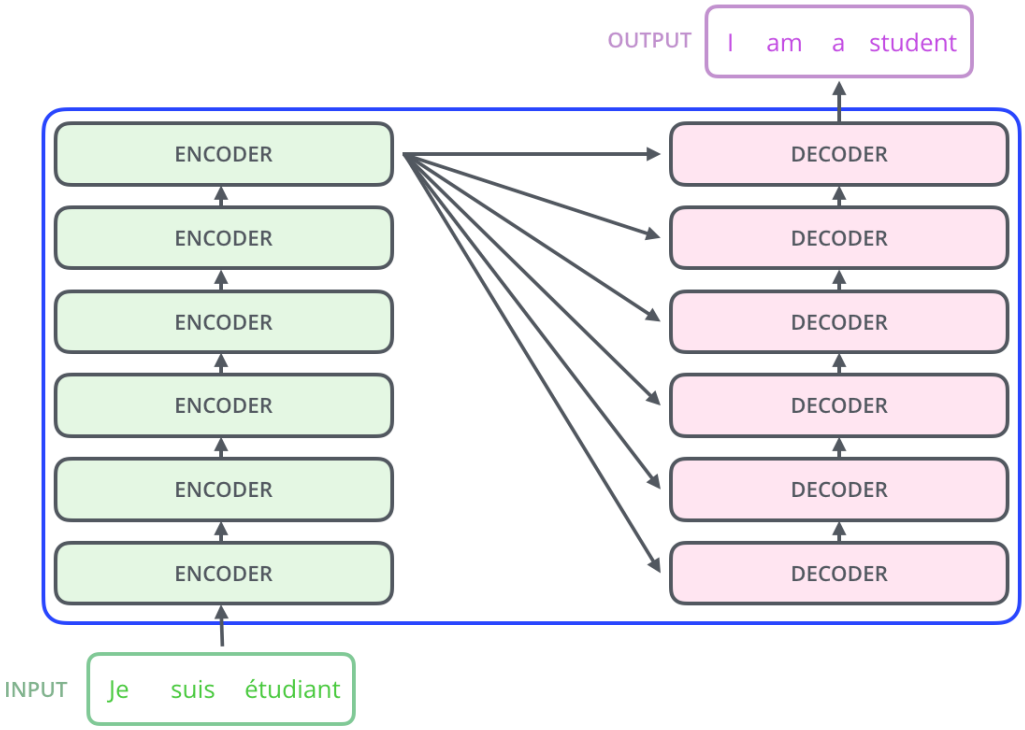
2.1.1 Tổng quan mô hình

Transformer là một mô hình có kiến trúc encoder-decoder. Mô hình bao gồm 2 thành phần chính là Encoder(bộ mã hóa) và Decoder(bộ giải mã). Khi đưa một đoạn văn bản nguồn vào, mô hình sẽ xử lý vào đưa ra đoạn văn bản có ngữ nghĩa tương ứng ở ngôn ngữ đích.



Hình 2.1: Encoder-decoder

Cụ thể hơn, Bộ mã hóa sẽ bao gồm nhiều lớp mã hóa xếp chồng lên nhau. Theo paper transformer, họ sử dụng 6 lớp mã hóa để tạo thành một bộ mã hóa. Bộ giải mã, tương tự cũng được xếp chồng bởi các lớp giải mã. Số lượng lớp của 2 thành phần phải bằng nhau.



Hình 2.2: Encoder-decoder

2.1.2 Encoder

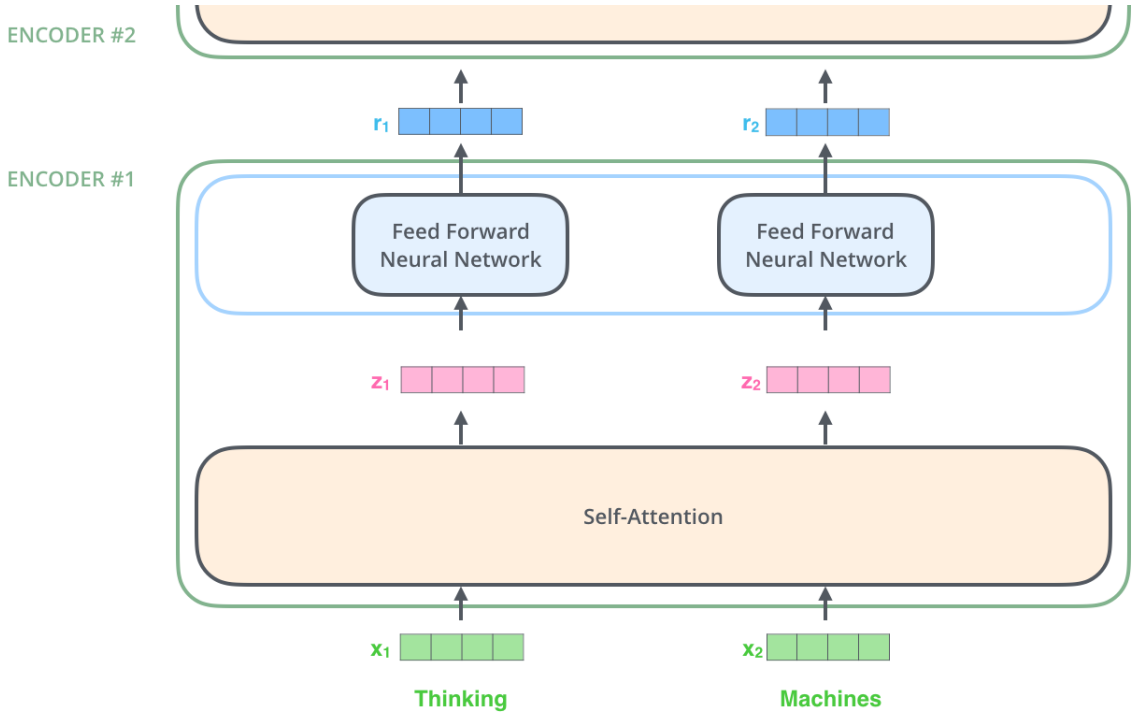
Các lớp của bộ mã hóa là độc lập nhau và có cấu trúc tương tự nhau. Đầu vào của lớp đầu tiên sẽ là các word embeddings đã được xử lý qua lớp positional encoding, các lớp phía trên sẽ có đầu vào là các vector output từ lớp ngay dưới. Các vector đầu vào của các lớp này được gọi là context vector.

Với mỗi lớp mã hóa sẽ bao gồm 2 lớp con:

- Self-attetion: Với mỗi từ trong đoạn văn bản, xem xét độ liên quan của nó với các từ khác trong câu đầu vào. Đưa ra phân bố xác xuất

đối với từng từ một.

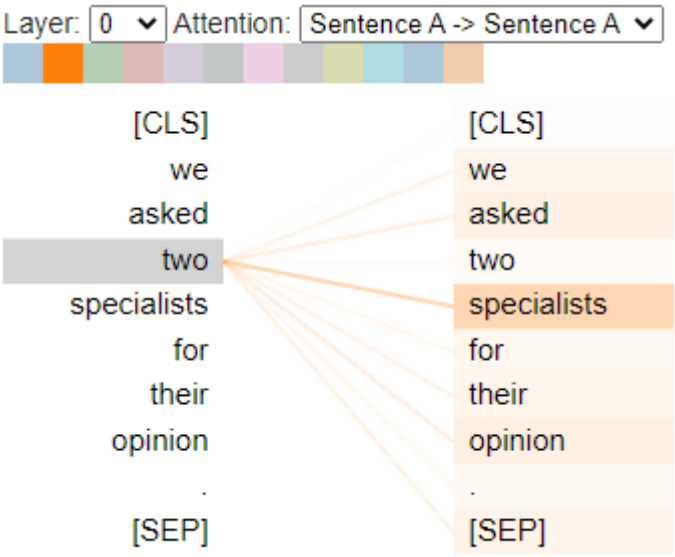
- Feed forwarding: Mã hóa phân bố xác xuất tính toán được thành các context vector để đưa vào các lớp mã hóa phía trên.



Hình 2.3: Encoder

Cơ chế Self-attention

Self-attention là một cơ chế mới được giới thiệu trong "attention is all need". Cơ chế này khác với cơ chế attention trong các mạng Seq2Seq trước đó. Self-attention cho phép ta biểu diễn lại mối quan hệ giữa các từ trong một đoạn văn bản.



Hình 2.4: self-attention

Hình trên cho ta một minh họa về cơ chế self-attention. Xét từ "two", theo tư duy của con người, ta sẽ phân tích xem từ "two" trong câu đang có quan hệ gì với những từ khác. Ngoài ra, ta còn xem xét tầm quan trọng của các từ khác tác động lên ý nghĩa của câu. Từ đó mà ta có thể có được một cái nhìn rõ ràng hơn về vai trò của từ này trong câu.

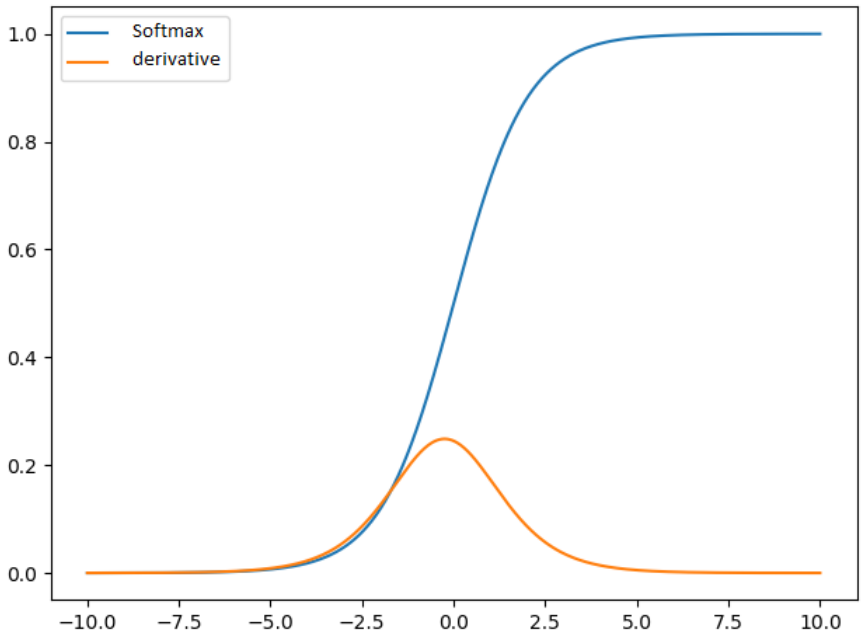
Cụ thể hơn, khi mô hình xử lý từ "two", self-attention cho ta thấy được mối liên kết của nó với từ "specialists". Thể hiện vai trò của nó là dùng để chỉ số lượng của các chuyên gia là hai.

Việc tính toán ở mỗi lớp self attention được tính toán dựa trên công thức sau:

$$Attention = Softmax(\frac{QK^T}{\sqrt{d}})V$$

Trong đó, Q, K, V lần lượt là các ma trận Query, Key, Value. Hai ma trận query và key được dùng để tính toán mối quan hệ giữa các từ trong câu. Còn mỗi dòng thứ của ma trận V đại diện cho từ thứ i trong câu. Từ phân bố softmax, ta tính được context vector, tổng hợp được các thông tin của từ hiện tại và các từ liên quan đến nó.

Trong công thức, ta thấy được tích vô hướng của ma trận Q và K được chuẩn hóa bởi hệ số $\sqrt{d}$ (d là số chiều của vector embedding). Lý do cho việc chuẩn hóa này là do khi d có giá trị lớn, tích vô hướng của Q và K sẽ có giá trị lớn theo do đó, nếu không chuẩn hóa về giá trị nhỏ hơn thì hàm softmax sẽ có độ hội tụ khá chậm



Hình 2.5: Đồ thị hàm softmax và đạo hàm của một thành phần trong tập phân loại

Algorithm 1 Self_attention$(context, w_K, w_Q, w_V)$

Result: Z
$K \leftarrow context \times w_K$
$Q \leftarrow context \times w_Q$
$V \leftarrow context \times w_V$
$Score \leftarrow QK^T$
$Z \leftarrow \frac{softmax(Score)}{\sqrt{d}}V$

Từ công thức tính của self-attention, ta có thể thấy rõ sự khác biệt giữa bộ mã hóa của transformer và bộ mã hóa của các mô hình Seq2Seq

sử dụng RNN. Đối với RNN, dữ liệu đầu vào phải được mã hóa một cách tuần tự. Trong khi đối với self-attention, các từ trong văn bản có thể được mã hóa một cái song song và không bị phụ thuộc vào nhau. Nhờ đó mà có thể tăng được hiệu quả tính toán.

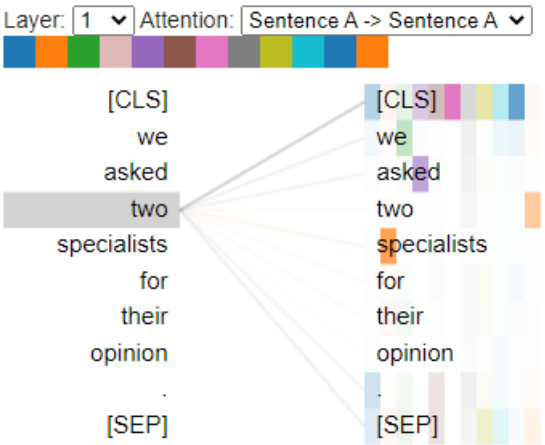
Cơ chế Multihead-attention

Đầu ra của self-attention cho ta biết được mối quan hệ của các từ trong câu dựa trên một góc nhìn nào đó. Bằng các xếp chồng nhiều lớp self-attention lại với nhau. Ta có thể biết được sự liên quan của các từ ngữ trong câu với nhiều gốc nhìn khác nhau. Từ đó mà có được thông tin đầy đủ hơn về câu cần dịch.

Cơ chế xếp chồng nhiều lớp self-attention lại với nhau được gọi là Multihead-attention.

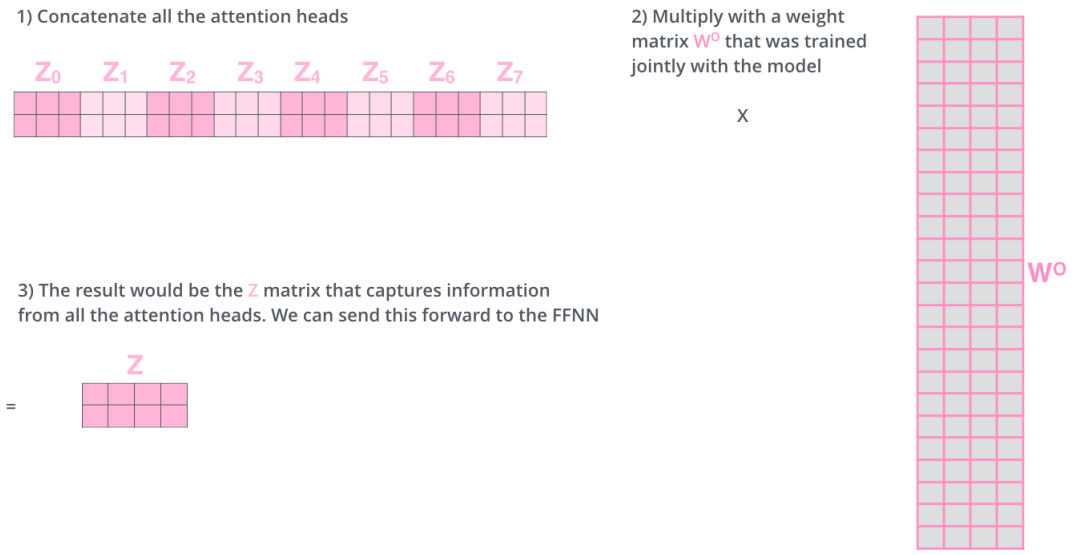
Ngoài ra, sử dụng cơ chế self-attention còn giúp ta tránh được trường hợp một từ phụ thuộc hoàn toàn vào chính nó. Ta mong muốn một phân bố xác suất quan hệ giữa một từ với các từ có ảnh hưởng đến nó.

Với việc sử dụng N lớp self-attention chạy song song với các bộ trọng số khác nhau, ta có được N ma trận context khác nhau. Lúc này, ta cần có một phương phát khác để tổng hợp thông tin từ các lớp self-attention này lại để đưa vào lớp feed-forward.



Hình 2.6: Multihead-context-vector

Để làm được việc đó, transformer ghép theo chiều ngang các ma trận context lại rồi nhân với một bộ trọng số để đưa ra một kết quả duy nhất là một ma trận với các dùng là các context vector cũng đồng thời là đầu vào cho bộ giải mã ở phía trên.

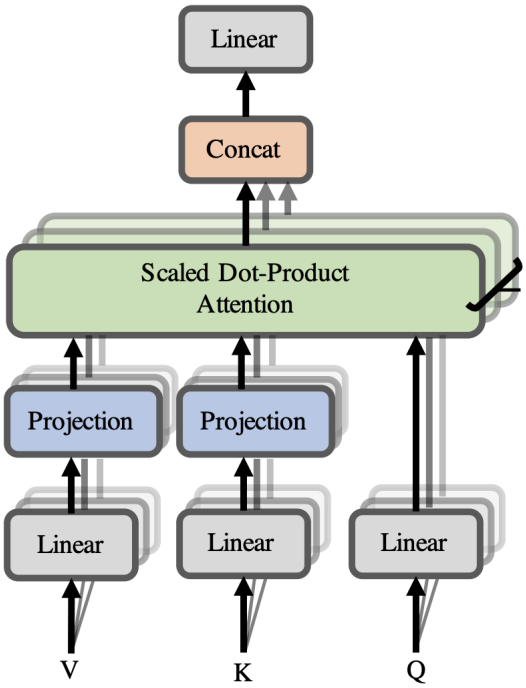


Hình 2.7: Multihead-context-vector

Hình dưới minh họa cho việc sử dụng nhiều lớp self-attention trên cùng một câu với mỗi một màu tương ứng với kết quả của một lớp self-attention khác nhau.

Algorithm 2 Multihead attention(*context*)

1: **Result:** $Z \times w_O$
2: **for** $1...number\_of\_head$ **do**
3: $\quad Z_i \leftarrow self\_attention(context, wK_i, wQ_i, wV_i)$
4: $Z \leftarrow [Z_1, Z_2, ..., Z_{number\_of\_head}]$



Hình 2.8: Multihead attention

Positional encoding

Cách tính toán song song của cơ chế self-attention dẫn đến một vấn đề đối với các từ trong đầu vào. Embeedings biểu diễn các từ này chưa biểu diễn được thông tin về thứ tự của các từ trong câu. Trong khi thông tin này là một thông tin quan trọng vì thay đổi vị trí của các từ có thể dẫn đến một câu hoàn toàn khác ý nghĩa.

Một ví dụ về việc đổi chỗ một từ trong câu khiến cho ý nghĩa của câu trở nên trái ngược hoàn toàn:

- "Please don't go! I love you!"
- "Please go! I don't love you!"

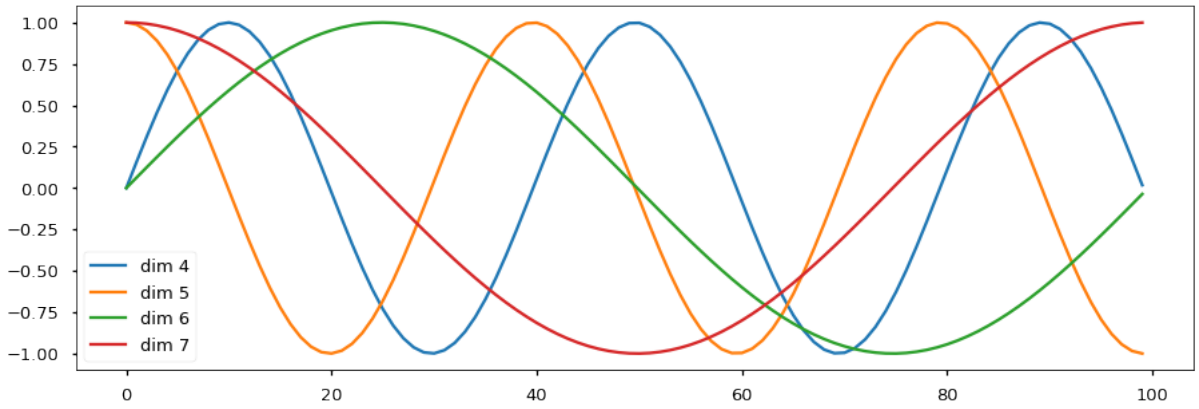
Transformer sử dụng cơ chế positional encoding. Cơ chế này giúp đưa thông tin về vị trí của các từ vào trong các embeddings. Cụ thể hơn, trước khi embeddings được đưa vào trong mô hình, nó được cộng với một vector

tương ứng với vị trí trong câu. Những vector này tuân theo một một quy định nhất định. Chúng phải thể hiện được sự khác biệt về vị trí của các từ trong câu và cả khoảng cách của các từ trong câu.

Công thức tính toán các vector này như sau:

$$PE_{(pos,2i)} = \sin \frac{pos}{10000^{\frac{2i}{d_model}}}$$

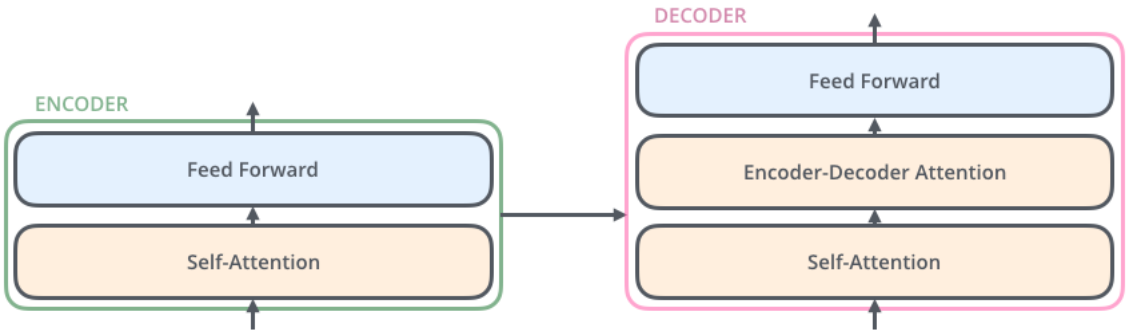
$$PE_{(pos,2i+1)} = \cos \frac{pos}{10000^{\frac{2i}{d_model}}}$$



Hình 2.9: Đồ thị positional embedding

2.1.3 Decoder

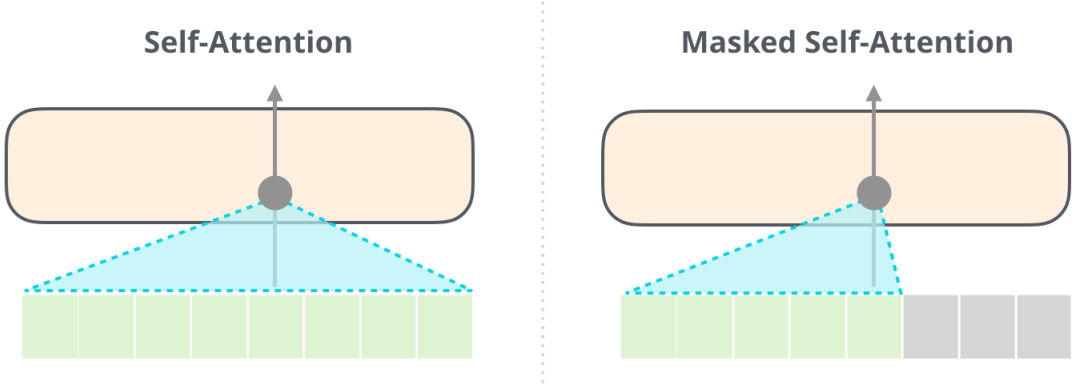
Các lớp của bộ giải mã cũng có 2 lớp con là Self-attention và Feed-forwarding giống với bộ mã hóa. Tuy nhiên giữa 2 lớp con này có một lớp trung gian là Encoder-Decoder attention. Lớp này có cơ chế giống với cơ chế attention trong mô hình Seq2Seq với thông tin của các hidden layer chính là đầu ra của bộ mã hóa.



Hình 2.10: Transformer decoder

Masked self attention

Khác với lớp self-attention ở bộ mã hóa, lớp masked self attention ở bộ giải mã chỉ tìm kiếm các mối quan hệ của từ hiện tại với các từ đã được giải mã trước đó. Do đó, trước khi thực hiện tính toán trên hàm softmax, giá trị score của các từ chưa được giải mã sẽ được gắn nhãn là $-\infty$. Sau đó, các bước tính toán sẽ tương tự như trong bộ mã hóa.



Hình 2.11: So sánh self attention và maked self attention

Ta có thể điều chỉnh lại hàm self attention như sau:

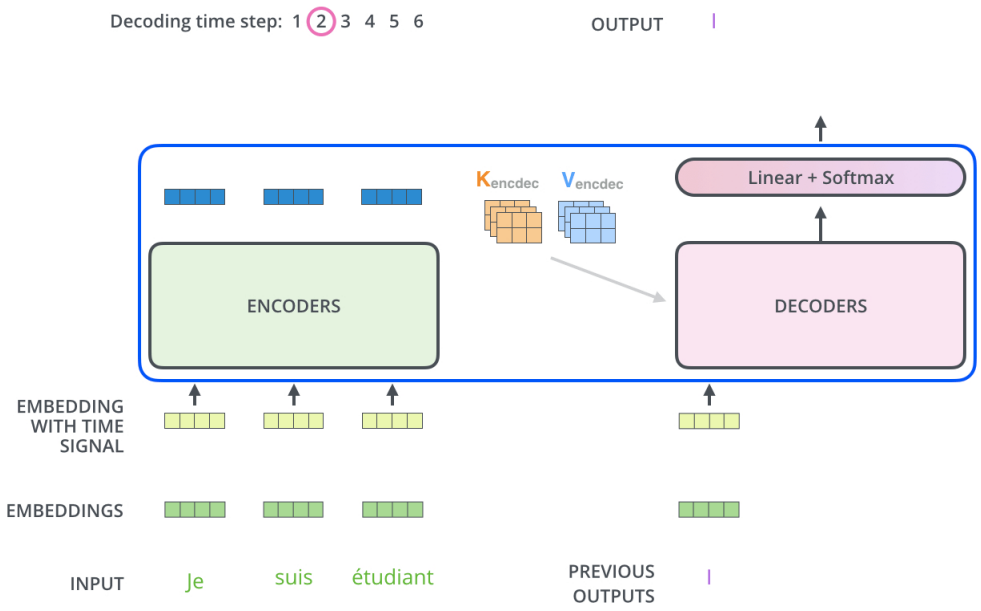
Algorithm 3 Self_attention($context, w_K, w_Q, w_V, masked$)

1: **Result:** Z
2: $K \leftarrow context \times w_K$
3: $Q \leftarrow context \times w_Q$
4: $V \leftarrow context \times w_V$
5: $Score \leftarrow QK^T$
6: **for** $i = masked + 1...len(context)$ **do**
7: Cột thứ i của $Score = -\infty$
8: $Z \leftarrow \frac{softmax(Score)}{\sqrt{d}}V$

Tham số masked thể hiện kể từ vị trí này, các phần tử phía sau của score sẽ được gán giá trị $-\infty$. Hàm self_attention được gọi bởi hàm multihead_attention. Do đó, ta cần truyền thêm tham số masked vào hàm này. Ta mặc định masked = len(context), nhờ vậy mà mà cần truyền thêm tham số này khi gọi trong bộ mã hóa.

encoder-decoder attention

Một điểm khác biệt dễ thấy giữa bộ mã hóa và bộ giải mà là lớp encoder-decoder attention được đặt giữa 2 lớp self attetion lvâf feed-forward. Công dụng của lớp này giúp cho decoder có thể giải mã ra được các từ của văn bản đích dựa trên độ liên quan của mỗi từ với các từ trong văn bản nguồn.



Hình 2.12: encoder-decoder-atention

Đầu ra lớp trên cùng của mã hóa sẽ được biến đổi thành 2 vector K và V. Quá trình giải mã được thực hiện như sau:

Algorithm 4 decoder(context, pos)

1: **Result:** $feed\_forward(context)$
2: $context \leftarrow multihead\_attention(context, masked = pos)$
3: $context \leftarrow normalize(context)$
4: $context \leftarrow encoder\_decoder\_attention(K_{enc}, V_{env}, context)$

Để biểu thị cho mở đầu của đoạn văn bản và kết thúc của đoạn văn bản đến đánh dấu việc bắt đầu giải mã và kết thúc giải mã. Transformer sử dụng 2 token đặc biệt nằm tách biệt với bộ từ vựng: "<bos>" (begin of sentence) và "<eos>" (end of sentence).

Quá trình giải mã sẽ được bắt đầu với token "<bos>" và thực hiện một cách tuần từ cho đến khi giải mã ra token "<eos>". Từ được giải mã ở bước trước sẽ được sử dụng làm đầu vào cho bước giải mã tiếp theo. Cụ thể thuật toán của toàn bộ quá trình giải mã như sau:

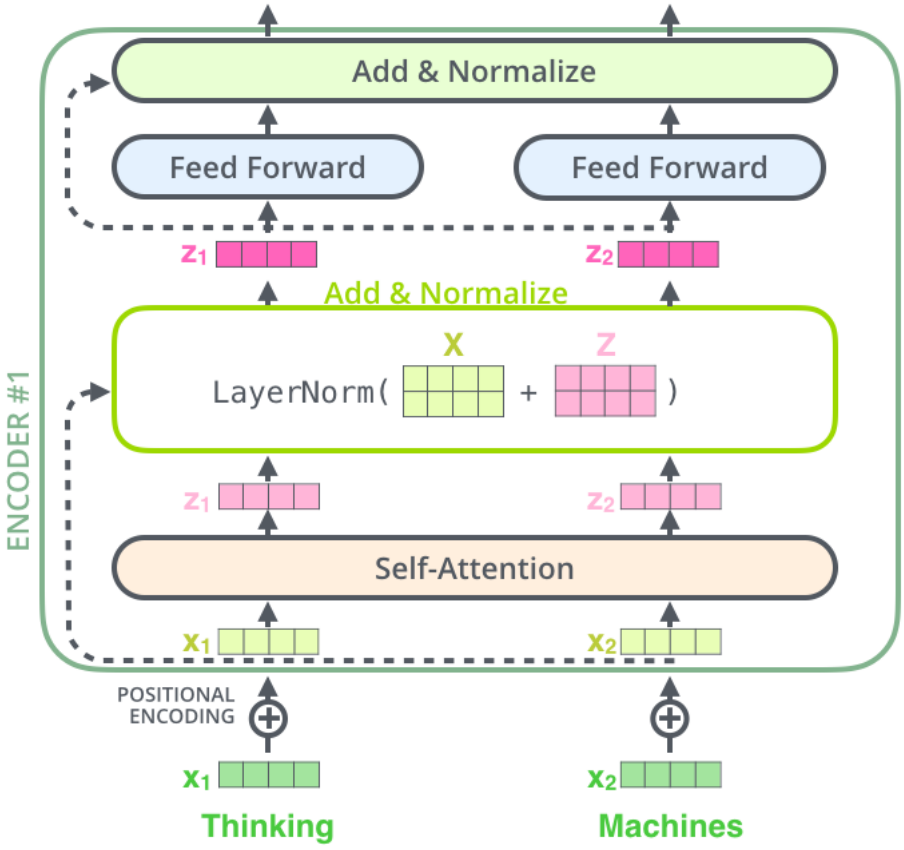
Algorithm 5 Quá trình giải mã

Khởi tạo $output_i \leftarrow$ " $< bos >$ "
Khởi tạo $i \leftarrow 1$
do
    $embed_i \leftarrow position\_encoding(output_{i-1})$
    $context \leftarrow embed_i$
    for do$j = 1...number\_decoder\_layer$
        $context \leftarrow decoder(context, masked = i)$
    $output_i \leftarrow softmax(context)$
while $output_i \neq$ " $< eos >$ "

2.1.4 residuals

Để tránh các trường hợp vanishing graident, Transformer kết hợp kiến trúc residual vào mạng encoder. Từ đó mà thông tin từ các lớp trước có thể được sử dụng lại trong khi huấn luyện các lớp sau.

với mỗi lớp con trong cả bộ mã hóa lẫn bộ giải mã, ta đặt thêm một lớp normalize vào ngay trên lớp con đó. Lớp normalize này nhận vào input là kết quả của lớp con ngay dưới của nó và cả đầu vào trước khi đi vào lớp con đó. Thực hiện phép cộng 2 ma trận này và chuẩn hóa lại trước khi đưa vào lớp con phía trên.



Hình 2.13: Residual

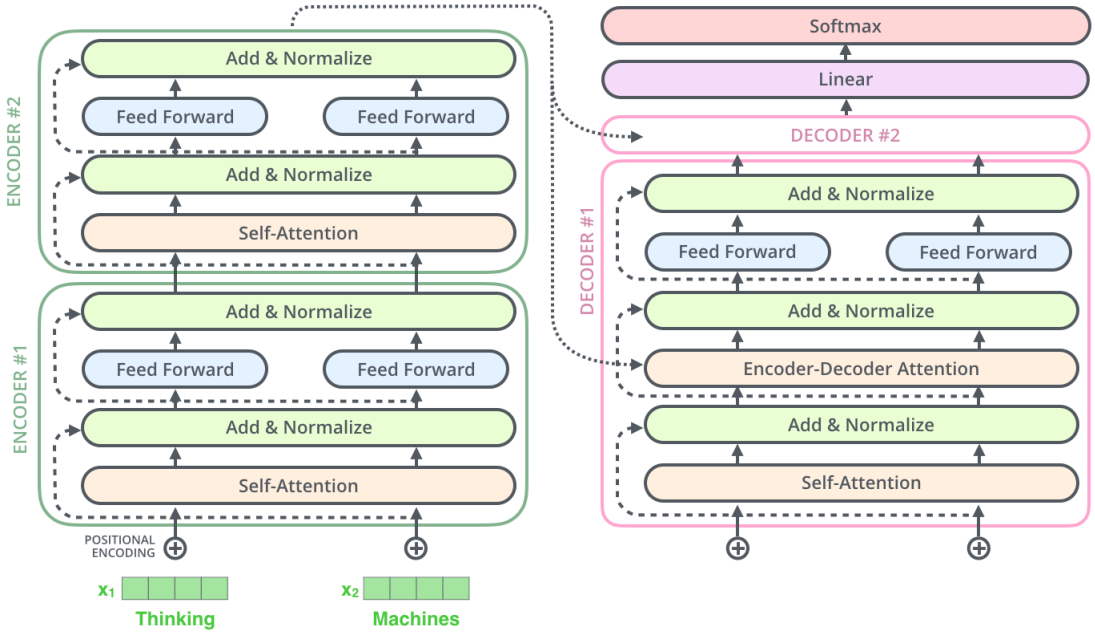
2.1.5 Tổng kết mô hình

Tổng kết lại, kiến trúc của Transformer bao gồm 2 thành phần chính là bộ mã hóa và bộ giải mã. Bộ mã hóa được tạo bởi nhiều lớp con gọi là các lớp mã hóa. Tương tự, bộ giải mã cũng được hình thành từ các lớp giải mã được xếp chồng lên nhau. Số lượng lớp con của bộ mã hóa bằng với số lượng lớp con của bộ giải mã.

Với mỗi lớp con của bộ mã hóa sẽ bao gồm một lớp multihead-attention và một lớp feed-forward. Ngoài ra còn áp dụng thêm kiến trúc residual block với các lớp normalize.

Với mỗi lớp giải mã, chúng được xếp chồng bởi 3 lớp con lần lượt là: masked-multihead-attention, encoder-decoder-attention và feed-forwarding.

Lớp giải mã cũng được áp dụng kiến trúc residual block giúp mô hình tránh được hiện tượng vanishing gradient.



Hình 2.14: Tổng kết mô hình transformer

2.2 Mô hình WordGCN huấn luyện embedding cho bộ ngữ liệu

2.2.1 Lý thuyết đồ thị cơ bản

Định nghĩa đồ thị

Đồ thị được định nghĩa là một cấu trúc rời rạc gồm các đỉnh và các cạnh nối các đỉnh đó. Ký hiệu biểu diễn một đồ thị như sau:

$$G = (V, E)$$

Trong đó, V là tập các đỉnh (Vertices) và E là tập các cạnh (Edges). Các cạnh trong tập E được bởi diễn bởi một cặp (x,y) với $x, y \in V$. Số

lượng đỉnh của đồ thị là $|V| = n$, còn số lượng cạnh của đồ thị là $|E| = m$

Các loại đồ thị

Đơn đồ thị (simple graph) là đồ thị thỏa $\forall x, y \in V|$ không tồn tại quá 1 cạnh nối giữa hai đỉnh này.

Đa đồ thị (multigraph) là đồ thị thỏa $\forall x, y \in V|$ có thể có nhiều hơn 1 cạnh nối giữa 2 đỉnh này

Giả đồ thị (speudo graph) là đồ thị chứa các đỉnh có khả năng nối cạnh với chính nó.

Đồ thị vô hướng (undirected graph) là đồ thị chứa các cạnh không định hướng. Nói cách khác, cạnh nối hai đỉnh x và y cũng sẽ được hiểu là cạnh đỉnh y với đỉnh x.

Đồ thị có hướng (directed graph) là đồ thị chứa các cạnh có hướng. nói cách khác, cạnh nói hai đỉnh x và y sẽ phân biệt với cạnh nối từ y đến x.

Đồ thị có trọng số là đồ thị mà các cạnh của đồ thị được biểu diễn bằng một trọng số nào đó. Lúc này mỗi cạnh $e \in E$ sẽ được biểu diễn bởi (x, y, w). Với x và y là hai đỉnh của cạnh và w là trọng số của cạnh đó

Đồ thị không trọng số là đồ thị mà các cạnh của đồ thị không được biểu diễn bởi một trọng số.

(Ảnh minh họa các loại đồ thị)

Ngoài ra, khi kết hợp cách tính chất của các loại đồ thị khác nhau, ta có thể có được các dạng đồ thị khác như: Đơn đồ thị vô hướng, Đa đồ thị có hướng, Giả Đa đồ thị có hướng có trọng số,..

Đồ thị còn có thể phân loại theo đồ thị vô hạn và độ thị hữu hạn, chỉ số lượng cạnh và đỉnh của đồ thị là vô hạn hay có thể đếm được.

vector đặc trưng (feature vectors)

Lý thuyết đồ thị khi được áp dụng vào các bài toán học máy thường sẽ được bổ sung thêm khác khái niệm vector đặc trưng. Vector đặc trưng là những vector chỉ các đặc trưng của một đỉnh hoặc một cạnh của đồ thị.

Các vector đặc trưng biểu diễn cho các đối tượng giống nhau phải có số lượng chiều như nhau.

Các vector này có thể có một cấu trúc đã được định nghĩa trước với tập các trường được khảo sát trên đối tượng tương ứng với đỉnh đang xét. Ngoài ra các vector của có thể được huấn luyện nhờ vào các mô hình học máy. Một bài toán được quan tâm đến khi nhắc đến các vector đặc trưng này là từ các liên kết của đồ thị và các context vector đã được định nghĩa trước, tìm ra các vector đặc trưng biểu diễn tốt cho các đối tượng đỉnh hoặc/và cạnh của đồ thị đó. Bài toán này được gọi là representation learning.

(Ảnh minh họa vector đặc trưng)

Ma trận kề (adjacent matrix)

Ma trận kề là ma trận chứa các thông tin về các cạnh nối giữa các đỉnh trong đồ thị. ma trận kề có kích thước $|V| \times |V|$. Phần tử ở hàng x cột y của ma trận chứa thông tin về kết nối giữa 2 đỉnh u và v và có giá trị như sau:

- Đồ thị không trọng số: 0 hoặc x biểu thị hoặc không có cạnh nối giữa 2 điểm u và v hoặc có x cạnh đang nối giữa 2 đỉnh này.
- Đồ thị có trọng số: 0 hoặc w biểu thị hoặc không có cạnh nối giữa u và v hoặc có cạnh nối giữa u và v và trọng số của cạnh này là w

(Minh họa ma trận kề)

Bậc của đồ thị

Bậc của một đỉnh được định nghĩa là tổng số cạnh nối với đỉnh đó. Đối với đơn đồ thị, số cạnh nối với đỉnh đang xét cũng bằng số đỉnh kề với đỉnh đang xét. Bậc của đỉnh u được ký hiệu là deg(u).

Tính chất:

- tổng số bậc của tất cả các đỉnh là số chẵn và có giá trị là $2m$ với m là số lượng cạnh của đồ thị

$$\sum_{v \in V} deg(v) = 2m$$

- Tổng số đỉnh có bậc lẻ là số chẵn.
- Đối với đồ thị vô hướng, tổng giá trị của hàng x hoặc cột x của ma trận kề biểu diễn đồ thị đó chính bằng deg(x)

Đối với đồ thị có hướng, bậc của một đỉnh còn được phân ra làm 2 thành phần:

- Bán bậc vào là số lượng cạnh nối với đỉnh đang xét và có hường vào đỉnh đó. Ký hiệu bán bậc vào của đỉnh u là $deg^-(u)$
- Bán bậc ra là số lượng cạnh nối với đỉnh đang xét và có hường ra khỏi đỉnh đó. Ký hiệu bán bậc ra của đỉnh u là $deg^+(u)$

$$deg(u) = deg^-(u) + deg^+(u)$$

2.2.2 Mô hình Graph Convolution Network và bài toán representation learning

Để giải bài toán representation learning trên đồ thị, người ta đưa ra giả thuyết rằng, các nút trên đồ thị có khoảng cách càng gần nhau thì sẽ có các đặc tính giống nhau. Từ đó mà vector đặc trưng của những nút gần kế nhau cũng sẽ có khoảng cách gần nhau trong không gian latent.

(Hình minh họa representation learning)

Từ giả thuyết này, người ta đưa ra được ý tưởng chính của mạng GCN là sử dụng các vector đặc trưng của các nút kề với nút đang xét để học được đặc trưng của nút này.

Cho đồ thị $G = (V, E)$, có ma trận kề $A_{n\times n}$ và ma trận đặc trưng $X_{n\times d}$ với $n = |V|$ và d là chiều của ma trận đặc trưng. Mô hình GCN được kiến trúc bởi nhiều lớp ẩn chồng lên nhau. Gọi $H^{(l)}$ là đầu ra của lớp ẩn thứ l. $H^{(l)}$ là một ma trận có kích thước $n \times d_i$ với d_i là số lượng đặc trưng của từng nút tại lớp l. Công thức tính $H^{(l)}$ được biểu diễn một cái đệ quy như sau:

$$H^{(l)} = \begin{cases} f(H^{(l-1)}, A), & l > 0 \\ X, & l = 0 \end{cases}$$

Trong đó, f là hàm tính đặc trưng của các nút dựa trên các lớp kề với nút đó. Do đó, f có biểu diễn như sau:

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)})$$

Với W là ma trận chứa các tham số học có kích thước $d_l \times d_{l+1}$. Và σ là hàm kích hoạt phi tuyến của mô hình.

Với công thức của hàm f như trên, ta thấy được 2 phần thiếu sót như sau:

- vector đặc trưng của nút u chỉ được tính bởi đặc trưng của các nút kề nó mà chưa được xét bởi các đặc trưng nội tại của nút u. Ta có thể giải quyết vấn đề này bằng các thêm các liên kết vòng (self connected edge) từ một nút đến chính nó. Ta định nghĩa ma trận $\tilde{A}$ là ma trận kề của G đã bổ sung các liên kết vòng:

$$\tilde{A} = A + I$$

- Các đỉnh có bậc cao sẽ gây ảnh hưởng đến nhiều nút hơn các đỉnh khác. Và nó cũng sẽ được ảnh hưởng bởi nhiều đỉnh hơn các đỉnh khác. Theo như công thức f ở trên, các đỉnh có bậc cao sẽ ma giá trị rất lớn hoặc rất nhỏ dẫn đến việc cập nhập gradient chậm lại trong quá trình back-propagation. Để giải quyết vấn đề trên, GCN sử dụng phép chuẩn hóa Symetric normalized Laplacian để tránh sự thiên vị

trong quá trình học. Từ đó mà ta có công thức đầy đủ của hàm f như sau:

$$H^{(l+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{1/2}H^{(l)}W^{(l)})$$

Trong đó, $\tilde{D}$ là ma trận bậc của đồ thị G phép chuẩn hóa đối xứng laplacian có thể được hiểu là với mỗi $\tilde{A}_{u,v}$ sẽ được nhân với một lượng $\frac{1}{\sqrt{deg(u)}\sqrt{deg(v)}}$

2.2.3 Mô hình WordGCN

WordGCN là mô hình được ghép bởi 2 mô hình con là SynGCN và SemGCN. Cả hai mô hình trên đều áp dụng kiến trúc GCN để trích xuất các embeddings của các từ trong bộ ngữ liệu. Trong khi SynGCN trích xuất thông tin cú pháp (syntactic) của các từ thông qua các liên kết của chúng trong các đoạn văn bản. Thì SemGCN rút trích các đặc trưng về ngữ nghĩa (semantic) của các từ thông qua các mối quan hệ như: đồng nghĩa, trái nghĩa, từ viết tắt,...

Mô hình SynGCN

SynGCN quan tâm đến đặc trưng của một từ thông qua ví trị và vai trò của từ đó trong câu. theo universal dependency Mỗi từ trong câu sẽ có một mối quan hệ với một từ khác trong câu hoặc là chủ thể của cả câu (không có liên kết đến từ khác). Các mối quan hệ đó được gọi là các universal dependency. Từ đây, ta có thể xem mỗi câu $s = \{w_1, w_2, ..., w_n\}$ trong bộ ngữ liệu là một đồ thị con dạng cây có hướng $G_s = (V_s, E_s)$ với các nút là các từ $V_s = \{w_1, w_2, w_3..., w_n\}$ và các dependency là các cạnh của đồ thị có dạng (w_i, w_j, d_{ij}) với d_{ij} là mối quan hệ giữa w_i và w_j. Lúc này, ta có thể áp dụng Thuật toán GCN lên đồ thị này.

(Hình minh họa GCN)

Nhận xét thấy GCN áp dụng vào trường hợp này khá tương tự với mô hình CBOW. Mô hình này sử dụng tập các từ xung quanh w_i để học các

đặc trưng của w_i. Tập các từ như vậy được gọi là tập Context của w_i.

$$C_{w_i} = \{w_{i+j} : -c \leq j \leq c, j \neq 0\}$$

Trong đó, c là kích thước của cửa sổ trượt.

Đối với SynGCN, tập Context của w_i được định nghĩa là tập các các từ tương ứng với nút kề với w_i:

$$C_{w_i} = N(w_i) = \{w_j : (w_i, w_j, l_{i,j}) \in E_s\}$$

Mô hình SemGCN

Các embedding sau khi được huấn luyện qua mạng SynGCN sẽ được đưa vào mạng SemGCN để học các đặc trưng về ngữ nghĩa (semantic context). Mô hình cho phép học các mối quan hệ có tính đối xứng lẫn các mối quan hệ không đối xứng. Đối với các mô hình trước đây, hoặc là chúng chỉ có thể giải quyết các mối quan hệ đối xứng hoặc chưa xử lý tốt vối các môi không đối xứng. Gọi G =(V,E) là đồ thị được tạo thành từ các ngữ nghĩa trong bộ dữ liệu. Ta sẽ có các đỉnh là các từ trong bộ nghữ liệu còn các cạnh là các mối quan hệ giữa chúng. Các cạnh trong G có định hướng. Đối với 2 từ x và y có liên kết đối xứng thì giữa 2 đỉnh tương ứng với x và y sẽ có 2 cạnh ngược hướng.

(Hình minh họa SemGCN)

Sau khi có được đồ thì biểu diễn, ta có thể sử dụng thuật toán GCN để tối ưu các embeddings đã được huấn luyện từ mô hình SynGCN.

Chương 3

Phương pháp

3.0.1 Tạo và chuyển dữ liệu cho chủ dữ liệu

3.0.2 Chia sẻ dữ liệu ngang hàng

Chương 4

Ứng dụng và cài đặt

4.0.1 Web Application (cho nhà cung cấp nội dung)

4.0.2 Mobile Application (cho người dùng)

Tài liệu tham khảo