TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

TRẦN QUỐC VINH - 52000823
TRẦN MINH TRÍ - 52000815
LÊ VĂN VIỆT - 52000822

BÁO CÁO CUỐI KỲ MÔN HỌC SÂU

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

TRẦN QUỐC VINH - 52000823
TRẦN MINH TRÍ - 52000815
LÊ VĂN VIỆT - 52000822

BÁO CÁO CUỐI KỲ MÔN HỌC SÂU

Người hướng dẫn
PGS. TS. Lê Anh Cường

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Em xin bày tỏ lòng biết ơn sâu sắc đến Thầy Lê Anh Cường - người đã hướng dẫn kiến thức và tâm huyết trong suốt quá trình học tập và thực hiện đề tài. Em rất biết ơn về những lời khuyên quý báu và sự dẫn dắt tận tâm của Thầy đã giúp em vượt qua những khó khăn và phát triển trong quá trình học tập và làm đề tài này.

Sự hỗ trợ và sự tận tâm của Thầy đã là nguồn động viên lớn giúp em không chỉ hiểu rõ hơn về các kiến thức và đưa ra lựa chọn đúng đắn giúp em phát triển kỹ năng và tư duy nghiên cứu để thực hiện đề tài trên. Em đánh giá cao những góp ý chi tiết và những lời khuyên rất giá trị từ Thầy để giúp em hoàn thiện tốt hơn đề tài của mình.

Những buổi học trên giảng đường và những trao đổi ý kiến cùng Thầy đã làm cho quá trình học tập trở nên thú vị và hấp dẫn. Em tự hào và biết ơn vì đã có cơ hội được học hỏi dưới sự dạy dỗ của một người thầy xuất sắc như Thầy.

Cuối cùng, em xin chân thành cảm ơn sự tận tâm và nhiệt huyết của Thầy trong thời gian vừa qua. Chúc thầy sẽ có thật nhiều sức khỏe.

Trân trọng cảm ơn thầy!

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

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

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

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

TP. Hồ Chí Minh, ngày 30 tháng 4 năm 2024

Tác giả

(Ký tên và ghi rõ họ tên)

Trần Quốc Vinh

Trần Minh Trí

Lê Văn Việt

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

CHƯƠNG 1. TÌM HIỂU CÁC MÔ HÌNH TRANSFORMER BASED ENCODER VÀ GPT

1.1 Kiến thức nền tảng

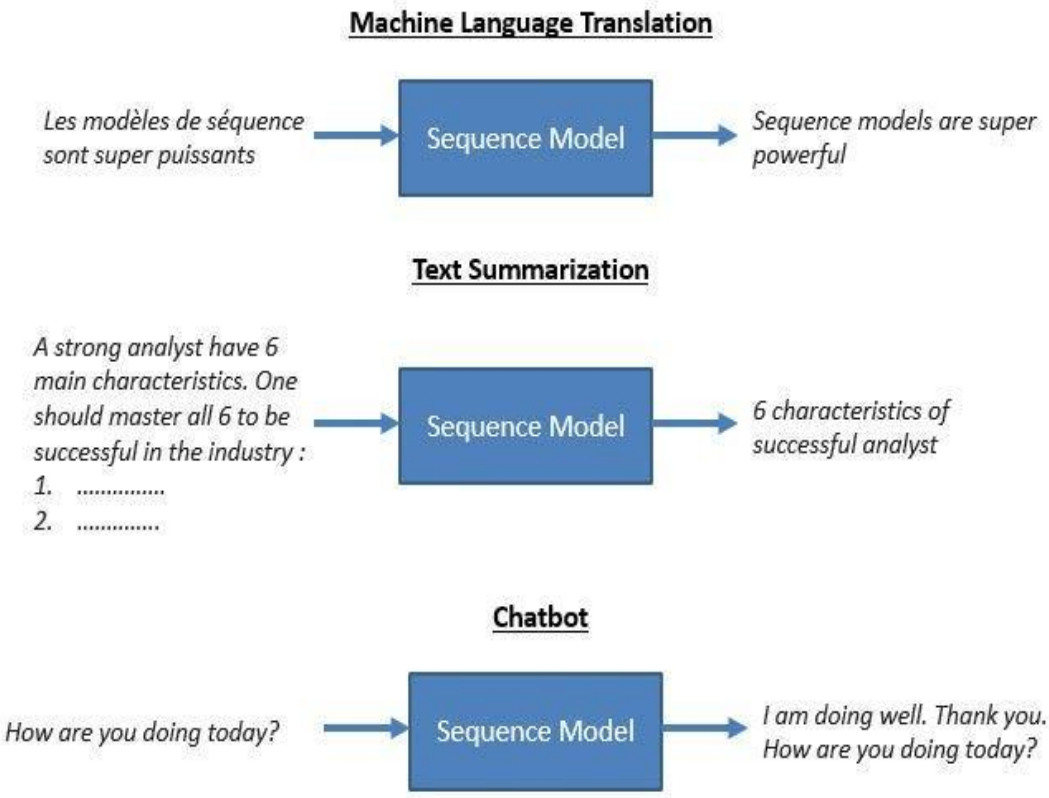
Để hiểu được hoàn toàn kiến trúc của BERT, chúng ta **cần hiểu qua một số khái niệm quan trọng** và có liên quan như sau:

- Seq2Seq model: Decoder and encoder
- Attention mechanism
- Transformer => Chúng ta sẽ đi qua từng phần bên dưới trước khi đi vào kiến trúc chính của 2 model: BERT và GPT-3

1.1.1 Seq2Seq model

Seq2Seq (gọi đầy đủ là Sequence to Sequence) là 1 model thường được sử dụng trong các task liên quan tới NLP được công bố lần đầu vào năm 2014, Seq2Seq có tên như trên là vì input và output đều sẽ là một sequence (tạm dịch: chuỗi ký tự) => Sequence to Sequence

Ví dụ trong bài toán Machine translation, input của ta sẽ là 1 câu tiếng anh và output sẽ là 1 câu tiếng việt. Hoặc trong bài toán Text summarization thì input là một chuỗi ký tự dài và output là chuỗi ký tự ngắn hơn đã được tóm tắt lại

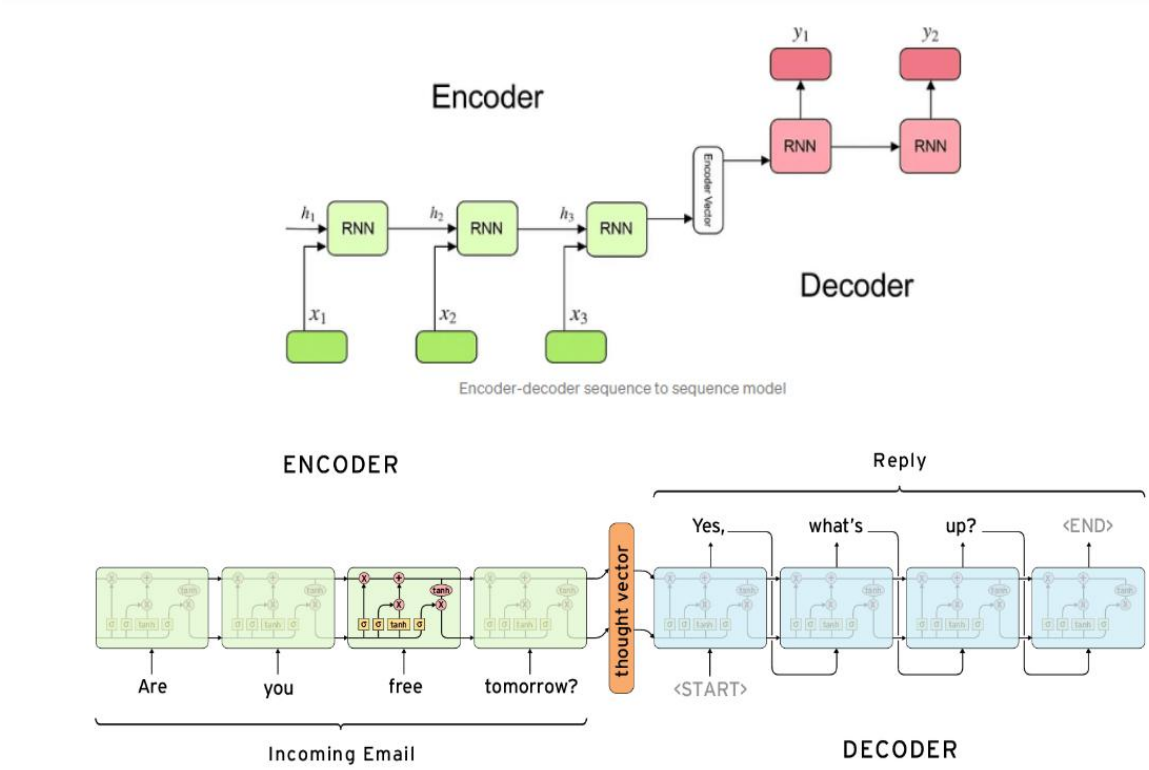


Hình 1. 1: Ví dụ bài toán Machine Translation

Kiến trúc của Seq2Seq model bao **gồm 2 bộ phận chính: Encoder và Decoder:**

- Encoder thường là lớp RNN, nó đọc vào một sequence và trích xuất thông tin của sequence đó, thông tin này sẽ được lưu trữ trong những vector gọi là context vectors hoặc internal state vector

- **Decoder** cũng là lớp RNN, input của nó chính là các context vectors từ Encoder. Decoder có nhiệm vụ dự đoán từng từ sau đó cập nhật lại hidden state, và sử dụng hidden state đó để dự đoán từ tiếp theo



Hình 1. 2: Kiến trúc Encoder-Decoder

Vì kiến trúc encoder-decoder chuyển đổi toàn bộ chuỗi đầu vào thành một vector có độ dài cố định và sau đó dự đoán chuỗi đầu ra nên **ta gặp các vấn đề như sau:**

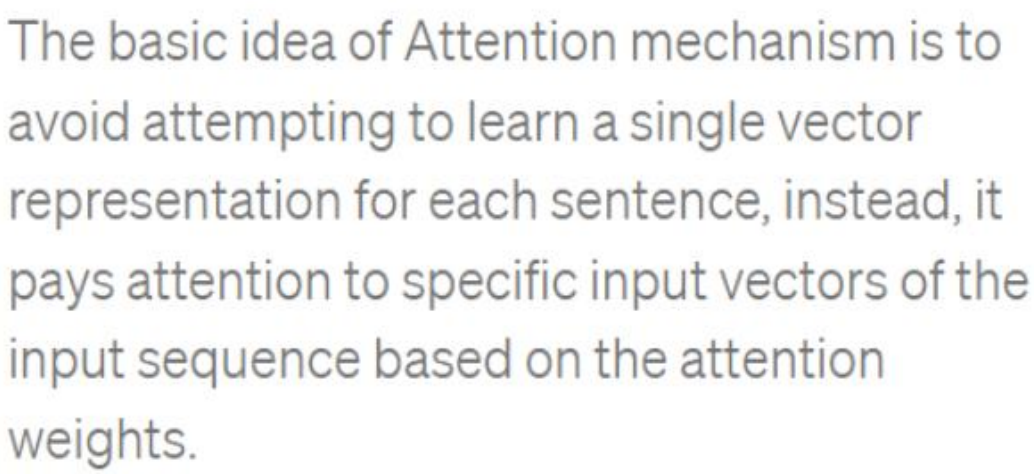
- Kiến trúc này chỉ hoạt động tốt với các chuỗi ngắn. Khi các chuỗi dài hơn thì ta gặp tình trạng mất mát thông tin (loss of information) khi chuyển đổi thành context vector => vấn đề này được biết tới như là vanishing gradient
- Khó để encoder ghi nhớ các chuỗi dài thành một vector có độ dài cố định
- Việc encode tất cả các thông tin trong 1 sequence dài có thể chiếm rất nhiều tài nguyên, liệu model chỉ có thể tập trung vào 1 số keyword trong sequence thay vì cả sequence hay không => **Cơ chế Attention (Attention mechanism) được tạo ra để giải quyết các vấn đề trên**

1.1.2 Attention mechanism

1.1.2.1 Giới thiệu

Cho 1 ví dụ như sau: "Hôm này trời mưa nhưng anh ấy quên mang theo ... ". Với ô trống phía trên thì chúng ta có thể dự đoán dễ dàng từ còn thiếu là "dù" do có keyword là "trời mưa". Vậy làm sao chúng ta có thể cho keyword "trời mưa" trọng số lớn để model biết là từ này quan trọng và mang nhiều ý nghĩa để dự đoán cho từ còn thiếu trong câu => cơ chế Attention

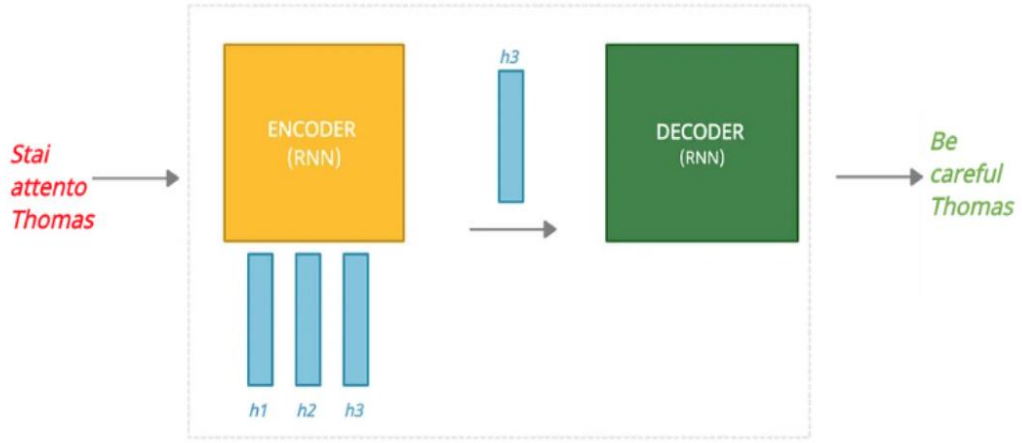
Ý tưởng cơ bản của cơ chế Attention là thay vì model học tất cả các vectors cho từng sentence (câu) thì model chỉ chú ý vào một số input vector của một số sentence nhất định dựa trên **attention weight**



Hình 1. 3: Ý tưởng cơ bản của cơ chế Attention

1.1.2.2 Cách hoạt động của cơ chế attention

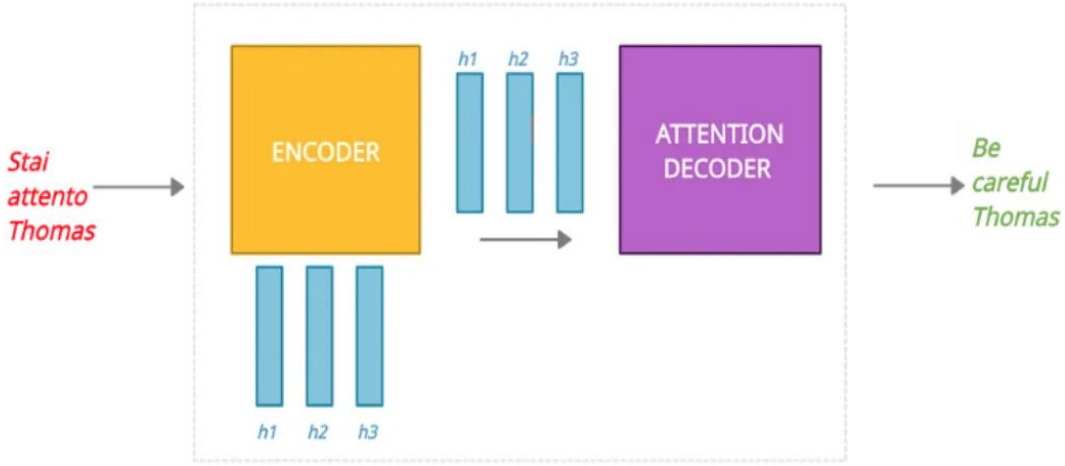
Đối với kiến trúc encoder-decoder bình thường mà chưa có attention layer thì context vector sẽ là hidden state cuối cùng của encoder. **Tất cả thông tin của sequence đều được chứa trong 1 vector duy nhất sẽ dẫn tới việc mất mát thông tin, từ đó decoder cho ra kết quả không chính xác**



Hình 1. 4: Context vector là hidden state cuối cùng của encoder

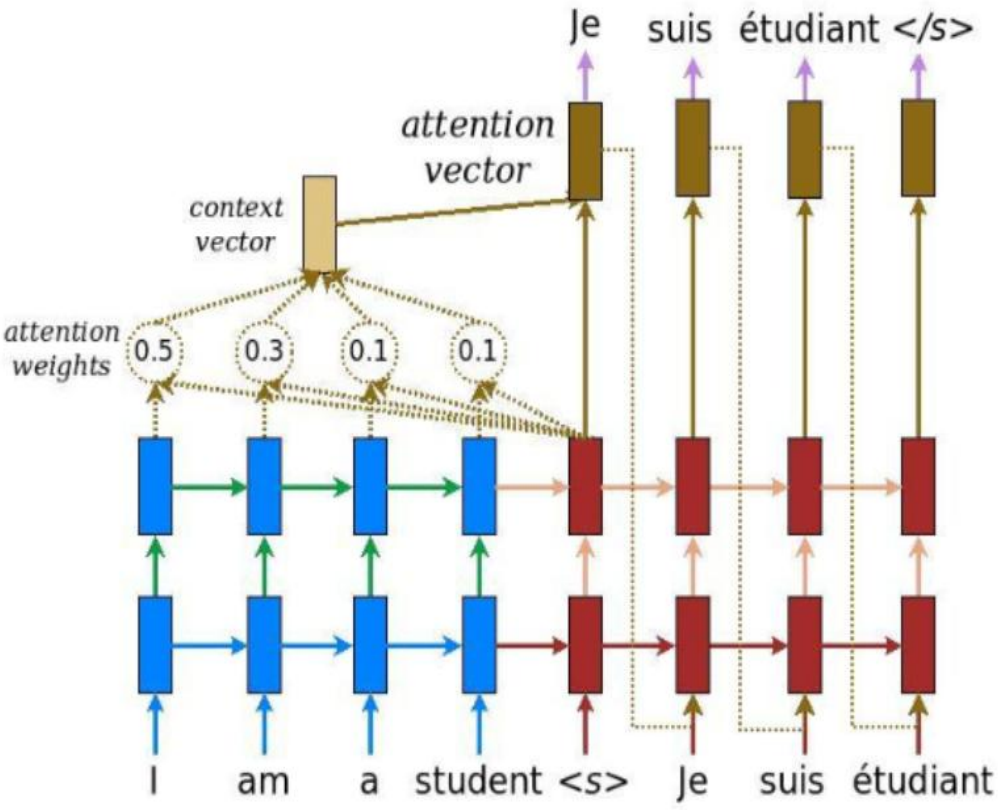
Trong ví dụ bên trên, chỉ có h3 (hidden state cuối cùng) là context vector và làm input cho decoder

Attention khác phục vấn đề này bằng cách không chỉ riêng hidden state cuối cùng được vào input mà tất cả các hidden state khác đều được



Hình 1. 5: Tất cả các hidden state được vào input

Mỗi hidden state được gán 1 score, score này sẽ được dùng làm trọng số để tạo thành context vector cuối cùng attention mechanism



Hình 1. 6: Kiến trúc Attention mechanism

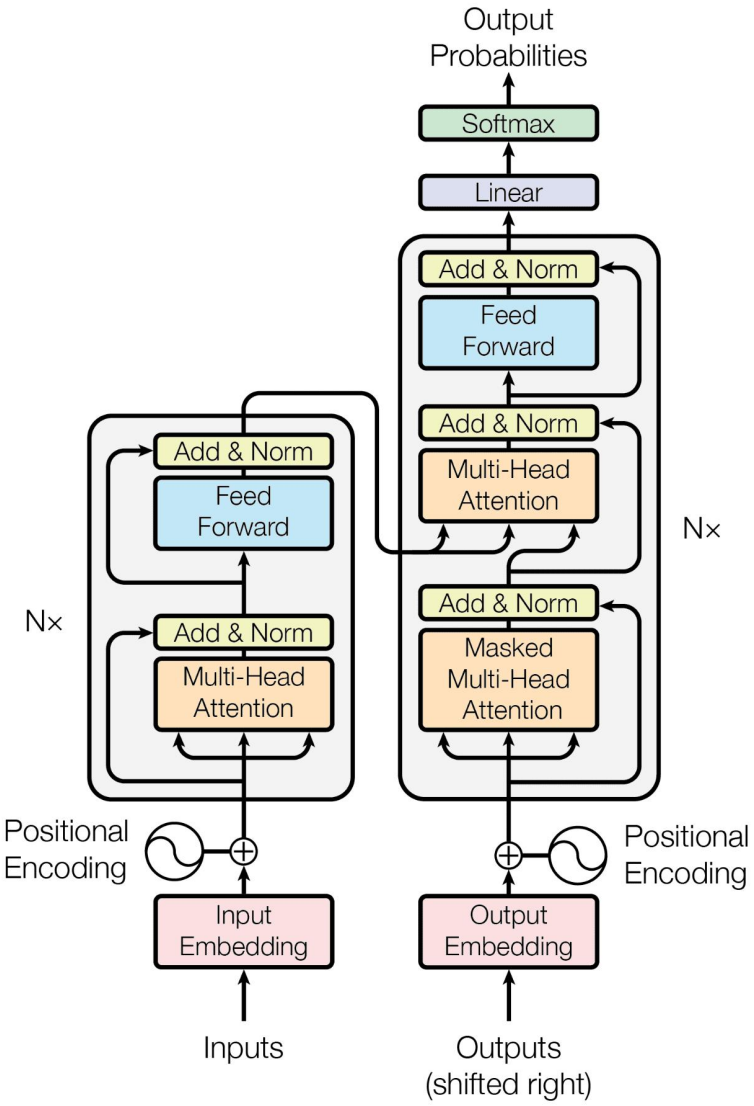
Để hiểu thêm về Attention mechanism, bạn có thể tham khảo bài viết dưới đây: Attention: Sequence 2 Sequence model with Attention Mechanism | by Renu Khandelwal | Towards Data Science

1.2 Transformer

Tổng quát: Model Tranformer chính là nền tảng của rất nhiều mô hình khác mà nổi tiếng nhất là BERT (Bidirectional Encoder Representations from Transformers) một mô hình dùng để học biểu diễn của các từ tốt nhất hiện tại và đã tạo ra một bước ngoặt lớn cho động đồng NLP trong năm 2019. Và chính Google cũng đã áp dụng BERT trong cỗ máy tìm kiếm của họ

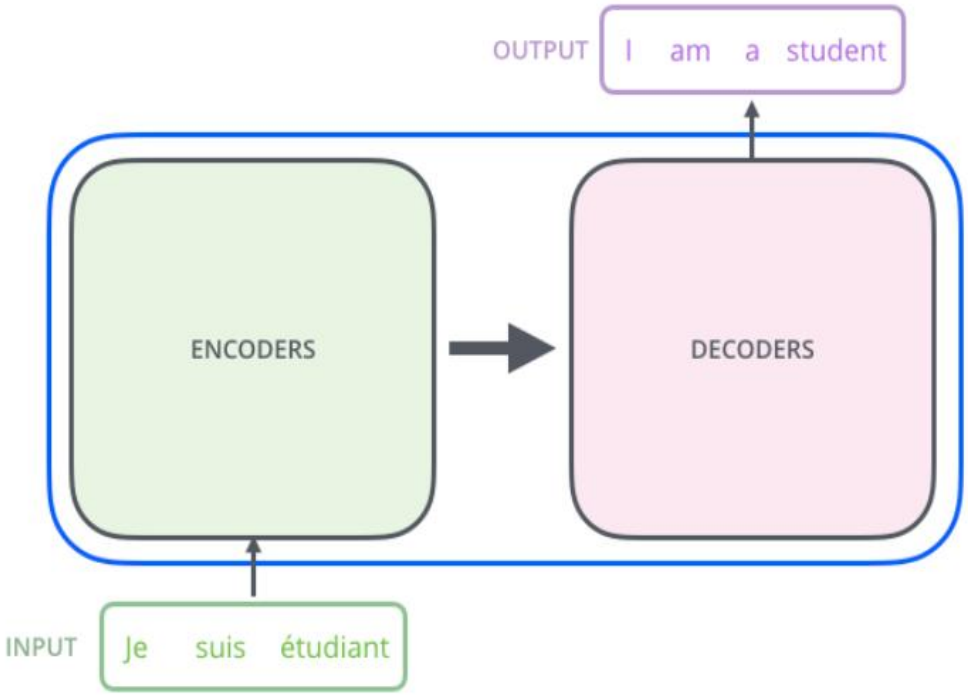
1.2.1 Kiến trúc của Transformer

Kiến trúc của transformer sẽ được mình thể hiện thông qua hình ảnh dưới đây:



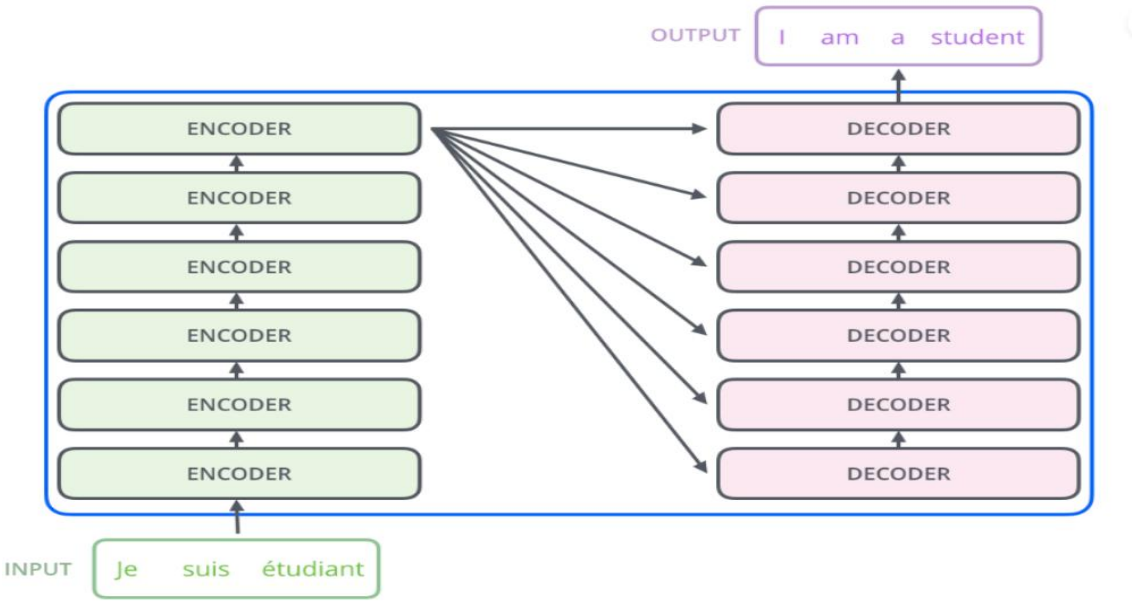
Hình 1. 7: Kiến trúc chi tiết của transformer

Có thể nói model bao gồm 2 component chính là: Encoder stack và Decoder stack



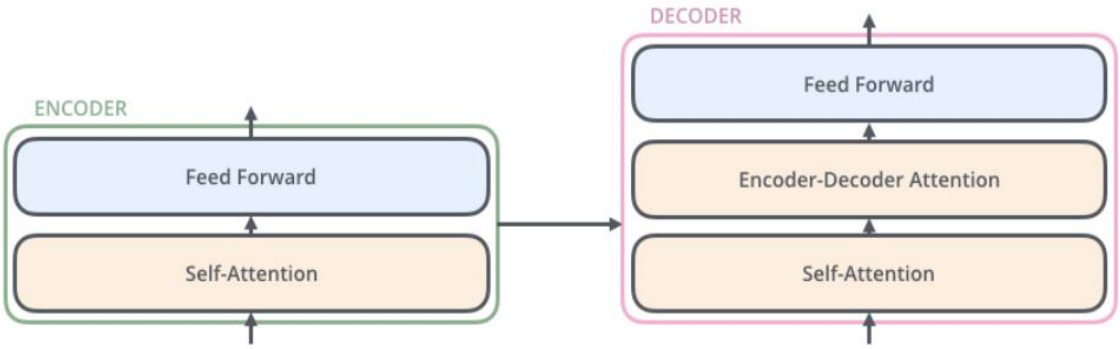
Hình 1. 8: Encoder stack và Decoder stack

Mỗi 1 stack như vậy sẽ bao gồm 6 layer (6 layers trong encoder và 6 layers trong decoder)



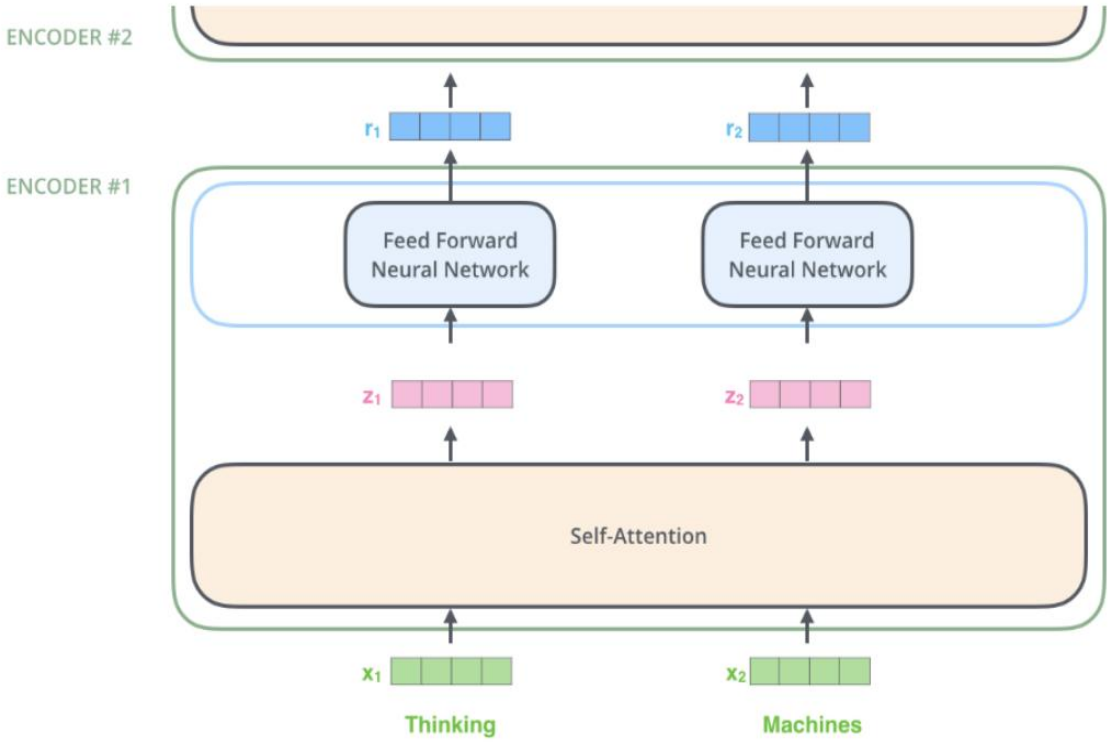
Hình 1. 9: Mỗi một stack bao gồm 6 layer

Mỗi encoder layer sẽ bao gồm 2 sublayers đó là: Self-attention và Feed Forward Neural Network (NN) layer. Decoder layer cũng bao gồm 2 sublayers như trên, nhưng giữa chúng có thêm 1 attention layer nữa



Hình 1. 10: Mỗi Encoder layer bao gồm 2 sublayers

Input của encoder sẽ là một list các embedding vectors. Các vector này thường có độ dài là 512, tuy nhiên đây là tham số và ta có thể tùy chỉnh được. Các vectors đi qua self-attention layer trước và sau đó là tới Feed Forward Neural Network



Hình 1. 11: Input của Encoder

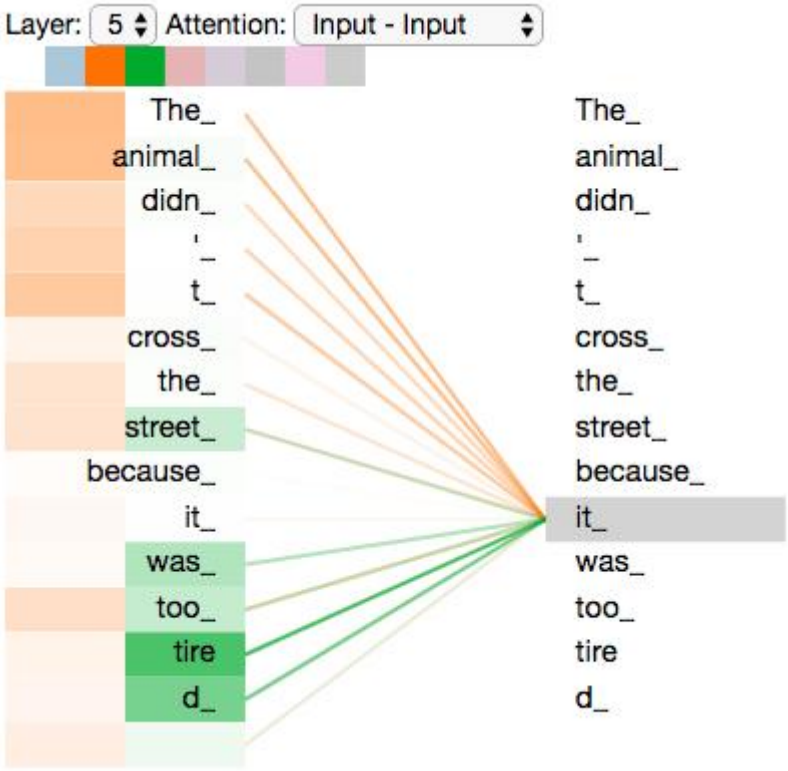
1.2.2 Cơ chế self-attention của model Transformer

Giả sử câu sau là câu đầu vào mà chúng ta cần dịch: **The animal didn't cross the street because it was too tired**

Từ "it" trong câu trên đại diện cho cái gì? "Con vật" (animal) hay "đường phố" (street)? Câu hỏi này đơn giản với con người nhưng không đơn giản với các thuật toán

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

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



Hình 1. 12: Mô hình xử lý từng từ

Chi tiết về cách thực hiện self-attention các bạn có thể tham khao bài viết dưới đây:

The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)

1.2.3 Transformer dựa trên Encoder hoặc Decoder

Đến thời điểm hiện tại thì hầu hết kiến trúc mà **large language model** sử dụng đều là Transformer (ngoại trừ mô hình RWKV sử dụng RNN). Thông thường, người ta chia large language model thành 3 loại dựa trên mục đích sử dụng :

- **Encoder only**: cho phép model đọc hết cả câu đầu vào và encode ra vector context nên có khả năng tổng hợp ngữ nghĩa khá mạnh, dạng model này phù hợp cho những task mang tính đọc hiểu như : classification và sentiment analysis, text summarization, named entity recognition.
- **Decoder only**: nhờ vào khả năng sinh chữ auto-regressive, model dạng decoder có khả năng tạo văn bản mạch lạc và có liên quan theo ngữ cảnh dựa trên prompt hoặc đầu vào nhất định, phù hợp với những task sinh chữ mang tính sáng tạo cao như : text completion, summarization, question-answering, và generating creative text. Đây là hướng phát triển chính của LLM hiện nay bởi nhờ được train unsupervised với lượng dữ liệu unlabel khổng lồ và kiến trúc được tối ưu cũng như kích thước model được scale lên rất lớn mà model dạng decoder đã có thể làm tốt cả các task của encoder only và encoder-decoder
- **Encoder - Decoder**: model sở hữu cả encoder và decoder như transformer, có khả năng đọc hiểu và sinh text. Tuy nhiên, người ta không hay chọn model dạng encoder-decoder để scale kích thước lên thành large language model bởi một số lí do: Training complexity, Inference complexity, Task specificity

1.3 Kiến trúc tổng quát của BERT

Trước khi đi qua kiến trúc chính, ta sẽ nói qua về **1 số điểm khiến cho BERT là 1 bước đột phá trong lĩnh vực NLP thời gian gần đây**

1.3.1 Mức độ phát triển của embedding trong NLP

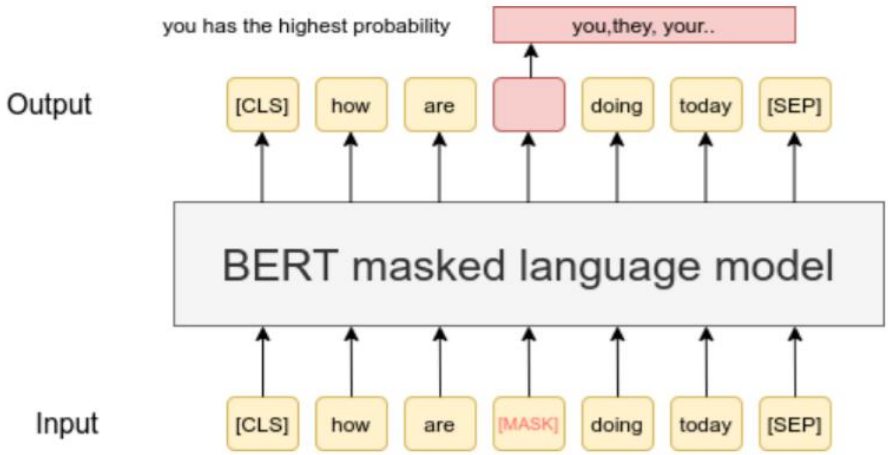
Phân cấp mức độ phát triển của các phương pháp embedding từ trong NLP có thể bao gồm các nhóm:

- **Non-context (không bối cảnh)**: Là các thuật toán không tồn tại bối cảnh trong biểu diễn từ. Đó là các thuật toán NLP đời đầu như word2vec, GLoVe, fasttext. Chúng ta chỉ có duy nhất một biểu diễn véc tơ cho mỗi một từ mà không thay đổi theo bối cảnh
- **Uni-directional (một chiều)**: Là các thuật toán đã bắt đầu xuất hiện bối cảnh của từ. Các phương pháp nhúng từ base trên RNN là những phương pháp nhúng từ một chiều. Các kết quả biểu diễn từ đã có bối cảnh nhưng chỉ được giải thích bởi một chiều từ trái qua phải hoặc từ phải qua trá. ELMo là một ví dụ cho phương pháp một chiều. Mặc dù ELMo có kiến trúc dựa trên một mạng BiLSTM xem xét bối cảnh theo hai chiều từ trái sang phải và từ phải sang trái nhưng những chiều này là độc lập nhau nên ta coi như đó là biểu diễn một chiều
- **Bi-directional (hai chiều)**: Ngữ nghĩa của một từ không chỉ được biểu diễn bởi những từ liền trước mà còn được giải thích bởi toàn bộ các từ xung quanh. Luồng giải thích tuân theo đồng thời từ trái qua phải và từ phải qua trái cùng một lúc. Đại diện cho các phép biểu diễn từ này là những mô hình sử dụng kỹ thuật transformer mà chúng ta sẽ tìm hiểu bên dưới. Gần đây, những thuật toán NLP theo trường phái bidirectional như BERT

Vì thế, **embedding vectors của BERT là Bi-directional, từ đó giúp cho thuật toán hiểu được context trong câu và mang lại độ chính xác cao**

1.3.2 Mask Language Model (MLM)

MLM là một quá trình pre-training của BERT. Trước khi sequence được cho vào BERT, 15% số từ trong mỗi sequence sẽ được thay thế bằng token **[MASK]**. Sau đó model sẽ cố gắng dự đoán 15% số từ này dựa trên ngữ cảnh mà 85% từ còn lại cung cấp(những từ chưa bị **MASK.** Theo đó:

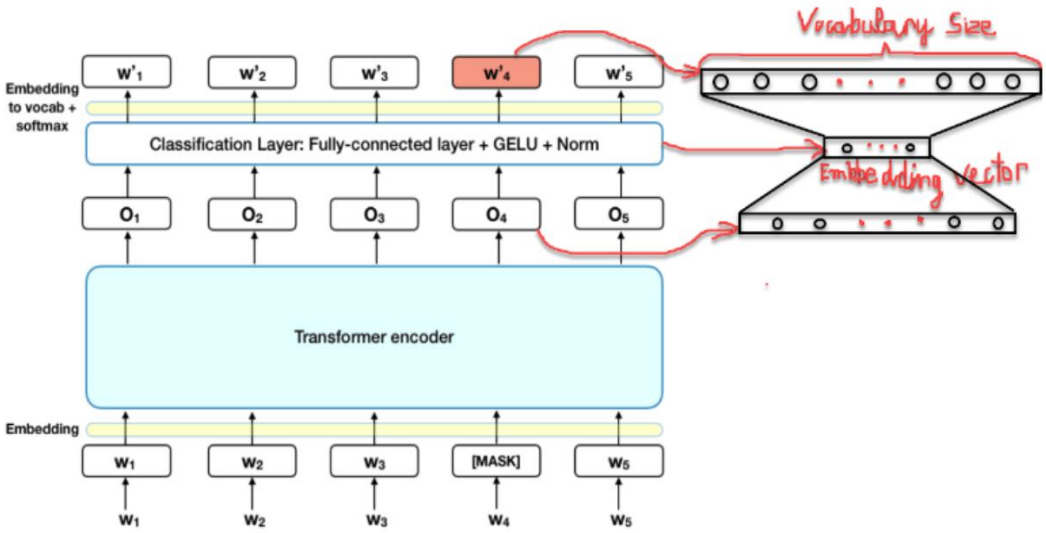


Hình 1. 13: BERT Mask Language Model

- Khoảng 15 % các token của câu input được thay thế bởi [MASK] token trước khi truyền vào model đại diện cho những từ bị che dấu (masked). Mô hình sẽ dựa trên các từ không được che (non-masked) dấu xung quanh [MASK] và đồng thời là bối cảnh của [MASK] để dự báo giá trị gốc của từ được che dấu. Số lượng từ được che dấu được lựa chọn là một số ít (15%) để tỷ lệ bối cảnh chiếm nhiều hơn (85%).

- Bản chất của kiến trúc BERT vẫn là một mô hình seq2seq gồm 2 phase encoder giúp embedding các từ input và decoder giúp tìm ra phân phối xác suất của các từ ở output. Kiến trúc Transfomer encoder được giữ lại trong tác vụ Masked ML. Sau khi thực hiện self-attention và feed forward ta sẽ thu được các véc tơ embedding ở output

- Để tính toán phân phối xác suất cho từ output, chúng ta thêm một Fully connect layer ngay sau Transformer Encoder. Hàm softmax có tác dụng tính toán phân phối xác suất. Số lượng units của fully connected layer phải bằng với kích thước của từ điển



Hình 1. 14: Classification Layer

Vì thế, **MLM là một pre-training strategy rất hiệu quả nhằm giúp BERT học các embedding sentences trước khi thực hiện các Downstream task (những tác vụ supervised-learning được cải thiện dựa trên những pretrained model)**

1.3.3 Next Sentence Prediction (NSP)

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

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

Chi tiết về Positional encoding trong model transformer các bạn có thể xem tại video này:

Visual Guide to Transformer Neural Networks - (Episode 1) Position Embeddings - YouTube

Vì vậy, **Trong quá trình training BERT, MLM và NSP được train cùng nhau với mục tiêu cuối cùng là minimize hàm loss**

1.3.4 Kiến trúc của BERT dựa trên model transformer

Hiện tại có nhiều phiên bản khác nhau của model BERT. Các phiên bản đều dựa trên việc thay đổi kiến trúc của Transformer tập trung ở 3 tham số:

- L: số lượng các block sub-layers trong transformer,

- H: kích thước của embedding véc tơ (hay còn gọi là hidden size),

- A: Số lượng head trong multi-head layer, mỗi một head sẽ thực hiện một self-attention

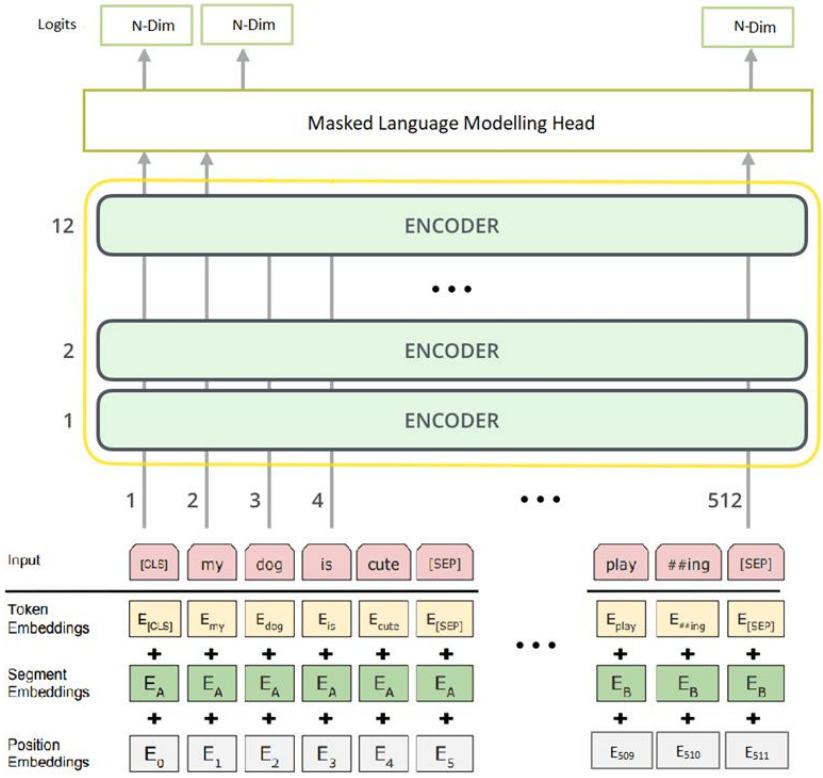
Tên gọi của 2 kiến trúc bao gồm:

- $\text{BERT}_{\text{BASE}}(L = 12, H = 768, A = 12)$: Tổng tham số 110 triệu.
- $\text{BERT}_{\text{LARGE}}(L = 24, H = 1024, A = 16)$: Tổng tham số 340 triệu.

Hình 1. 15: Tên gọi của hai kiến trúc

Như vậy ở kiến trúc BERT Large chúng ta tăng gấp đôi số layer, tăng kích thước hidden size của embedding véc tơ gấp 1.33 lần và tăng số lượng head trong multi-head layer gấp 1.33 lần

Minh họa Input của BERT, input **dựa trên 3 embedding vectors là Token, Segment, Position**

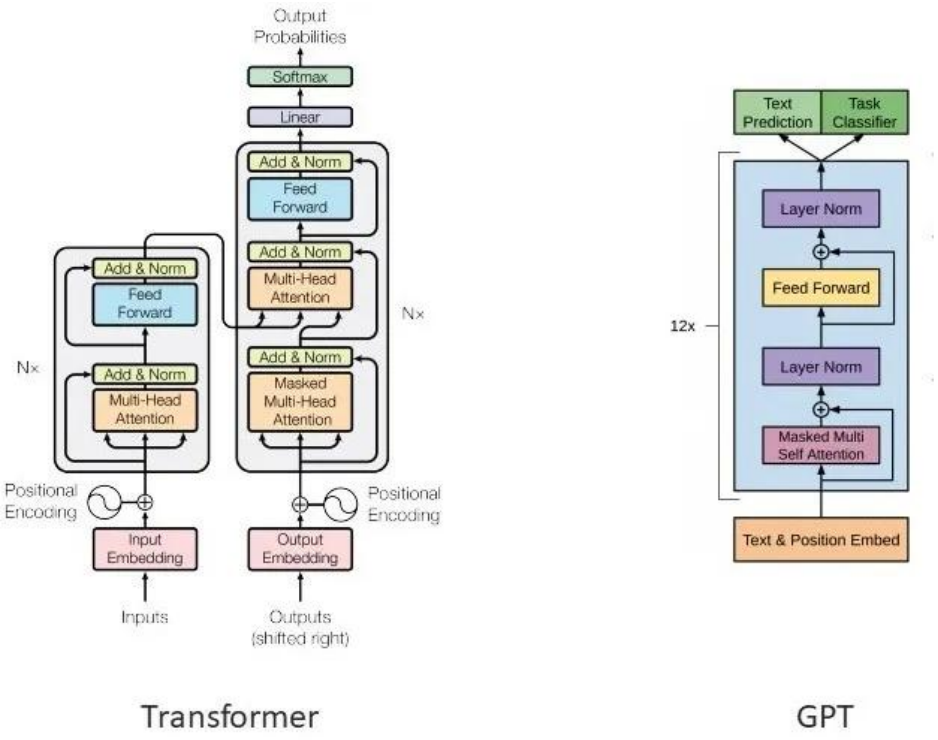


Hình 1. 16: Minh họa Input của BERT

1.4 Kiến trúc tổng quát của GPT-3

Giới thiệu: GPT-3 (Generative Pre-trained Transformer 3) là một mô hình ngôn ngữ được tạo bởi OpenAI. Mô hình học sâu với 175 tỷ tham số này có khả năng tạo ra văn bản giống con người và được huấn luyện trên các bộ dữ liệu văn bản lớn với hàng trăm tỷ từ

GPT (Generative Pre-trained Transformer) sử dụng kiến trúc Transformer gốc, ngoại trừ việc thiếu phần Encoder. Chúng ta có thể thấy điều này trong sơ đồ phía dưới



Hình 1. 17: Kiến trúc của GPT-3 sử dụng kiến trúc Tràormer gốc

GPT-1, GPT-2 và GPT-3 được xây dựng dựa trên Transformer Decoder. Ngược lại, BERT sử dụng các Transformer Encoder. GPT-3 được huấn luyện trên bộ dữ liệu văn bản khổng lồ trên internet - tổng cộng 570GB. Khi được phát hành, nó là model lớn nhất với 175 tỷ tham số (gấp 100 lần GPT-2). GPT-3 có 96 attention blocks, mỗi attention blocks chứa 96 attention head

1.4.1 Cách GPT-3 hoạt động: Pretraining

GPT-3 được huấn luyện bằng cách "dự đoán từ tiếp theo". Trong đó - nó dự đoán từ tiếp theo trong một câu, chuỗi đầu vào thực sự được cố định ở 2048 từ (đối với GPT-3)

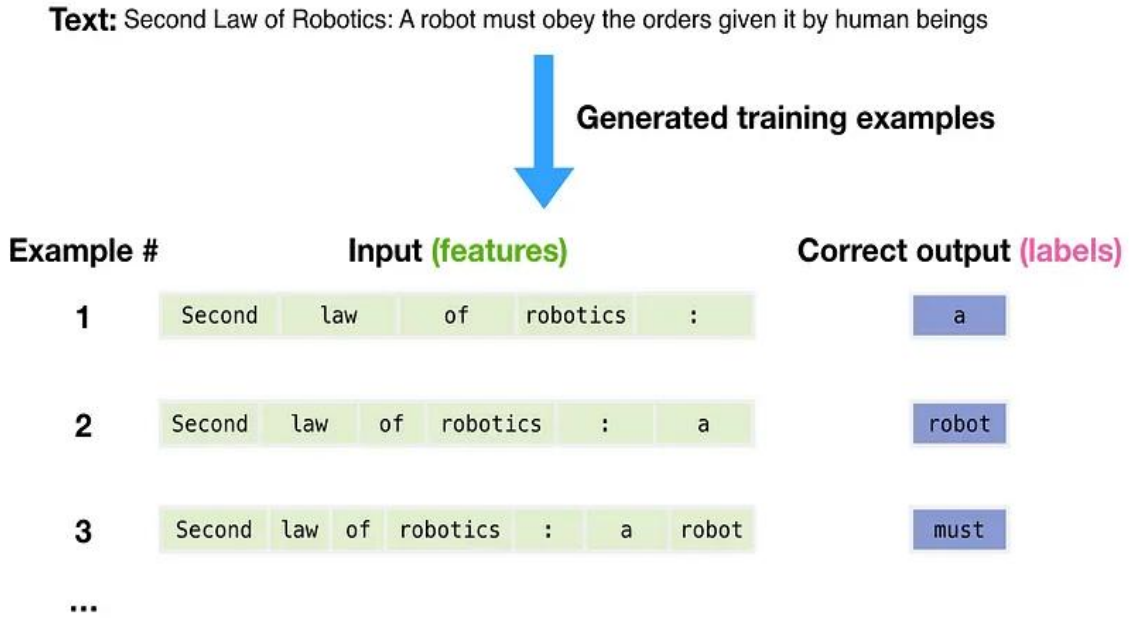
Tuy nhiên, chúng ta vẫn có thể truyền các chuỗi ngắn làm đầu vào: chúng ta chỉ đơn giản điền tất cả các vị trí thừa bằng các giá trị "Padding". Đầu ra của GPT không chỉ là một dự đoán duy nhất, mà là một chuỗi (dài 2048) các dự đoán (xác suất cho mỗi từ có thể xuất hiện)

Các bước pretrain của GPT-3:

- **Encoding**: Bước đầu tiên là duy trì một bộ Key-word cho tất cả vocab, nó cho phép chúng ta gán giá trị cho mỗi từ. GPT sử dụng các thuật toán Sub Word Algorithm để tạo ra bộ từ vựng. Nếu một từ không có trong từ điển hiện tại của mô hình, GPT-3 sử dụng Byte Pair Encoding (BPE) sub word tokenization => GPT-3 tạo ra Positional Encoding từ đó
- **GPT-3 tính toán Key, Query, Value vector** cho mỗi token trong câu
- **Attention Score của mỗi token cuối cùng sẽ được đưa vào hàm Softmax để cho về khoảng 0-1** => Giá trị càng cao thì từ đó càng có khả năng xuất hiện

1.4.2 Cách GPT-3 hoạt động: Training

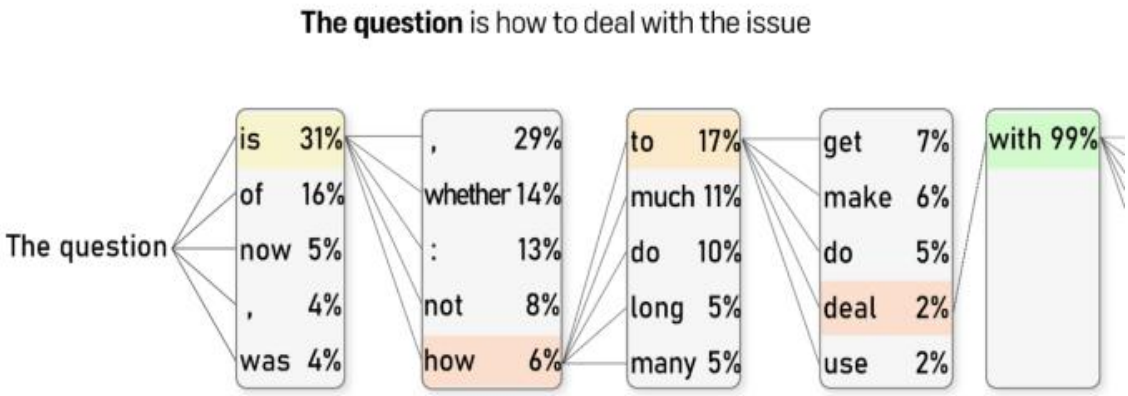
Bộ dữ liệu khổng lồ gồm 300 tỷ token văn bản được sử dụng để tạo các ví dụ huấn luyện cho mô hình. Ví dụ, đây là ba ví dụ huấn luyện được tạo từ một câu



Hình 1. 18: Ba ví dụ huấn luyện được tạo từ một câu

Chúng ta truyền các token vào trong model. GPT-3 sẽ dự đoán từng token một vì nó là auto regressive model và mục tiêu là Next word Prediction. Lần đầu tiên, dự đoán của mô hình có thể sẽ sai. Chúng ta tính toán lỗi bằng cách so sánh với đầu ra chính

xác và cập nhật các tham số của mô hình. Nhờ đó, lần tiếp theo mô hình sẽ đưa ra dự đoán tốt hơn. Quá trình này được lặp đi lặp lại nhiều lần



Hình 1. 19: Truyền các token vào và dự đoán từng token

CHƯƠNG 2. PRETRAINED VÀ FINE-TUNING GPT-2 MODEL XÂY DỰNG CHATBOT

GitHub: https://github.com/VinhQuocTran/finalterm-DeepLearning

2.1 Dữ liệu

Tập dữ liệu có khoảng 3000 dòng text về các cuộc hội thoại từ các nhóm chat.

Cấu trúc dữ liệu:


```
    }
  ],
  "eval_score": 1,
  "profile_match": 1
},
{
  "dialog_id": "0x31b35caa",
  "dialog": [
    { "id": 0, "sender": "participant2", "text": "/start" },
    { "id": 1, "sender": "participant1", "text": "hello how are you" },
    { "id": 2, "sender": "participant2", "text": "/start" },
    { "id": 3, "sender": "participant1", "text": "are you ok?" },
    {
      "id": 4,
      "sender": "participant2",
      "text": "i love animals , i have a dog , and a dog ."
    },
    { "id": 5, "sender": "participant1", "text": "i have a dog too" },
    {
      "id": 6,
      "sender": "participant2",
      "text": "i see , i am a black belt , so i can not say that , but i am not sure ."
    },
    { "id": 7, "sender": "participant1", "text": "i enjoy american sports" },
    {
      "id": 8,
      "sender": "participant2",
      "text": "that is cool , i am a dancer , and i am a teacher , and you ?"
    },
    {
      "id": 9,
      "sender": "participant1",
      "text": "i am working in a company for 15 years"
    },
    {
      "id": 10,
      "sender": "participant2",
      "text": "i am a black belt , so i am a fan of it , but i am not sure ."
    }
  ],
  "eval_score": 2,
  "profile_match": 1
},
{
  "dialog_id": "0x823c044",
  "dialog": [
```


Hình 2. 1: Cấu trúc dữ liệu

Mỗi dialog là một conversation giữa các senders

Ta trích ra các đoạn text (khoảng 3500 dòng) thành tập dữ liệu để train.

2.2 Mô hình

Mô hình: GPT2LMHeadModel

Đây là mô hình pretrained text generation. Ta sẽ sử dụng để train bộ dữ liệu trên

2.2.1 Xử lý dữ liệu đầu vào

Ta sử dụng **GPT2Tokenizer** để tokenize dữ liệu đầu vào với tham số:

- max_length: 40
- truncation: True
- padding: “max_length”
- return_tensors: “pt”

2.2.2 Training

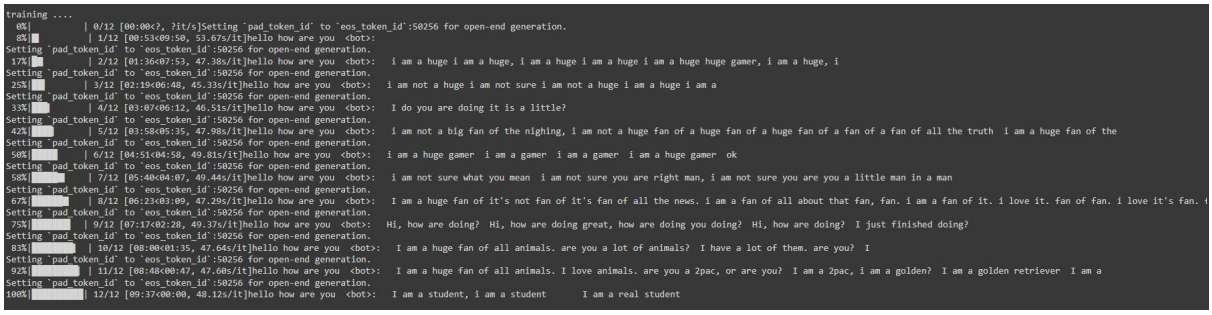
```
def train(chatData, model, optim):
    epochs = 12
    for i in tqdm.tqdm(range(epochs)):
        for X, a in chatData:
            X = X.to(device)
            a = a.to(device)
            optim.zero_grad()
            loss = model(X, attention_mask=a, labels=X).loss
            loss.backward()
            optim.step()
        torch.save(model.state_dict(), "model_state.pt")
```

Hình 2. 2: Hàm train

Ta huấn luyện mô hình với các tham số:

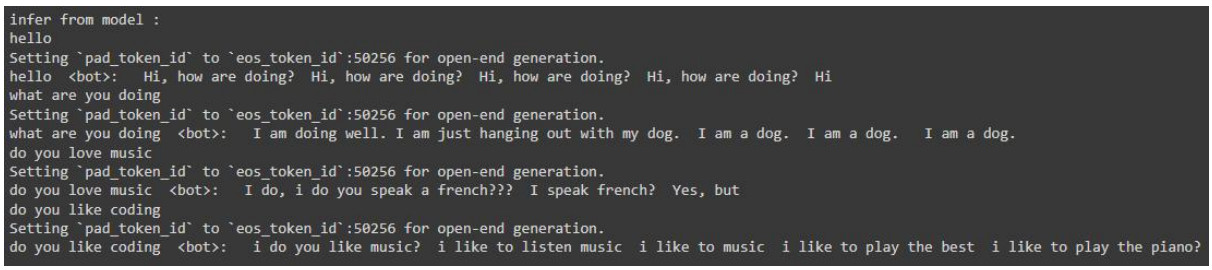
- epochs: 12
- optimize: Adam với learning_rate = 1e - 3

Quá trình huấn luyện:



Hình 2. 3: Quá trình huấn luyện

Kết quả thử nghiệm:



Hình 2. 4: Kết quả thử nghiệm

TÀI LIỆU THAM KHẢO

BERT

1. https://phamdinhkhanh.github.io/2020/05/23/BERTModel.html#12-l%C3%BD-do-t%E1%BA%A1i-sao-m%C3%ACnh-vi%E1%BA%BFt-v%E1%BB%81-bert
2. https://viblo.asia/p/hieu-hon-ve-bert-buoc-nhay-lon-cua-google-eW65GANOZDO
3. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
4. https://towardsdatascience.com/bert-for-next-sentence-prediction-466b67f8226f#:~:text=Next%20sentence%20prediction%20.
5. https://trituenhantao.io/tin-tuc/minh-hoa-transformer/
6. https://viblo.asia/p/hieu-hon-ve-bert-buoc-nhay-lon-cua-google-eW65GANOZDO
7. https://huggingface.co/blog/bert-101?text=Paris+is+the+%5BMASK%5D+of+France.
8. https://www.sbert.net/examples/unsupervised_learning/MLM/README.html
9. https://jalammar.github.io/illustrated-bert/

https://phamdinhkhanh.github.io/2019/06/18/AttentionLayer.html

GPT-3

11. https://viblo.asia/p/tu-transformer-den-language-model-bai-2-kien-truc-va-phuong-phap-generative-pretraining-cua-gpt-model-bXP4Wx1xJ7G
12. https://medium.com/nerd-for-tech/gpt3-and-chat-gpt-detailed-architecture-study-deep-nlp-horse-db3af9de8a5d

13. https://medium.com/@prudhvithtavva/bert-vs-gpt-a-tale-of-two-transformers-that-revolutionized-nlp-11fff8e61984

GPT-2

14. https://huggingface.co/docs/transformers/v4.40.1/en/model_doc/gpt2?fbclid=IwZXh0bgNhZW0CMTAAAR2i_95etyk-QKBsccX32P6NVPXL9gSMo_NZ0Emwj4yhrysmSK5q32k_jNU_aem_AeT3onsQt5kT-60iGecmCH45lpylay9_bP_NZP6Fh7sXLe6XnWtlDCsHbcHgI-yK423xUhBzs6ZNPvUffbVa07RW#transformers.GPT2LMHeadModel