Bài 36 - BERT model

23 May 2020 - phamdinhkhanh

Menu

1. Giới thiệu chung

1.1. Một số khái niệm

Trước khi đi vào bài này, chúng ta cần hiểu rõ một số khái niệm:

- **Nhiệm vụ phía sau (Downstream task)**: Là những tác vụ supervised-learning được cải thiện dựa trên những pretrained model. VD: Chúng ta sử dụng lại các biểu diễn từ học được từ những pretrained model trên bộ văn bản lớn vào một tác vụ phân tích cảm xúc huấn luyện trên bộ văn bản có **kích thước nhỏ hơn**. Áp dụng pretrain-embedding đã giúp cải thiện mô hình. Như vậy tác vụ sử dụng pretrain-embedding được gọi là downstream task.
- **Điểm khái quát đánh giá mức độ hiểu ngôn ngữ (GLUE score benchmark)**: GLUE score benchmark (https://gluebenchmark.com/) là một tập hợp các chỉ số được xây dựng để đánh giá khái quát mức độ hiểu ngôn ngữ của các model NLP. Các đánh giá được thực hiện trên các bộ dữ liệu tiêu chuẩn được qui định tại các convention về phát triển và thúc đẩy NLP. Mỗi bộ dữ liệu tương ứng với một loại tác NLP vụ như: Phân tích cảm xúc (Sentiment Analysis), hỏi đáp (Question and Answering), dự báo câu tiếp theo (NSP - Next Sentence Prediction), nhận diện thực thể trong câu (NER - Name Entity Recognition), suy luận ngôn ngữ tự nhiên (NLI - Natural Languague Inference). Nếu bạn muốn tìm hiểu thêm về cách tính GLUE score và các bộ dữ liệu trong GLUE có thể đọc thêm tensorflow - glue (https://www.tensorflow.org/datasets/catalog/glue).
- **Quan hệ văn bản (Textual Entailment)**: Là tác vụ đánh giá mối quan hệ định hướng giữa 2 văn bản? Nhãn output của các cặp câu được chia thành đối lập (contradiction), trung lập (neutral) hay có quan hệ đi kèm (textual entailment). Cụ thể hơn, chúng ta có các câu:

A: Hôm nay trời mưa.

B: Tôi mang ô tới trường.

C: Hôm nay trời không mưa.

D: Hôm nay là thứ 3.

Khi đó (A, B) có mối quan hệ đi kèm. Các cặp câu (A, C) có mối quan hệ đối lập và (A, D) là trung lập.

Top

- **Suy luận ngôn ngữ (Natural Language Inference)**: Là các tác vụ suy luận ngôn ngữ đánh giá mối quan hệ giữa các cặp câu, cũng tương tự như Textual Entailment.
- **Phân tích cảm xúc (Sentiment Analysis)**: Phân loại cảm xúc văn bản thành 2 nhãn tích cực (positive) và tiêu cực (negative). Thường được sử dụng trong các hệ thống đánh giá bình luận của người dùng.
- **Hỏi đáp (Question and Answering)**: Là thuật toán hỏi và đáp. Đầu vào là một cặp câu (pair sequence) bao gồm: câu hỏi (question) có chức năng hỏi và đoạn văn bản (paragraph) chứa thông tin trả lời cho câu hỏi. Một bộ dữ liệu chuẩn nằm trong GLUE dataset được sử dụng để đánh giá tác vụ hỏi và đáp là SQuAD - Stanford Question Answering Dataset (https://rajpurkar.github.io/SQuAD-explorer/). Đây là một bài toán khá thú vị, các bạn có thể xem thêm ứng dụng Question and Answering - BERT model (https://www.facebook.com/TowardDataScience/videos/201232064499053/) mà mình đã sharing.
- **Ngữ cảnh (Contextual)**: Là ngữ cảnh của từ. Một từ được định nghĩa bởi một cách phát âm nhưng khi được đặt trong những câu khác nhau thì có thể mang ngữ nghĩa khác nhau. ngữ cảnh có thể coi là môi trường xung quanh từ để góp phần định nghĩa từ. VD:

A: Tôi `đồng` ý với ý kiến của anh.

B: Lão Hạc phải kiếm từng `đồng` để nuôi cậu Vàng.

Thì từ `đồng` trong câu A và B có ý nghĩa khác nhau. Chúng ta biết điều này vì dựa vào ngữ cảnh của từ.

- **Hiện đại nhất (SOTA)**: state-of-art là những phương pháp, kỹ thuật tốt nhất mang lại hiệu quả cao nhất từ trước đến nay.
- **Mô hình biểu diễn mã hóa 2 chiều dựa trên biến đổi (BERT-Bidirectional Encoder Representation from Transformer)**: Mô hình BERT. Đây là lớp mô hình SOTA trong nhiều tác vụ của `GLUE score benchmark`.
- **LTR model**: là mô hình học bối cảnh theo một chiều duy nhất từ trái sang phải. Chẳng hạn như lớp các model RNN.
- **MLM (Masked Language Model)**: Là mô hình mà bối cảnh của từ được học từ cả 2 phía bên trái và bên phải cùng một lúc từ những bộ dữ liệu unsupervised text. Dữ liệu input sẽ được masked (tức thay bằng một token MASK) một cách ngẫu nhiên với tỷ lệ thấp. Huấn luyện mô hình dự báo từ được masked dựa trên bối cảnh xung quanh là những từ không được masked nhằm tìm ra biểu diễn của từ.

1.2. Lý do tại sao mình viết về BERT?

Tại thời điểm mình viết về model BERT thì BERT đã được ra đời khá lâu. BERT là model biểu diễn ngôn ngữ được google giới thiệu vào năm 2018. Tại thời điểm công bố, BERT đã tạo ra một sự rung động trong cộng đồng NLP bởi những cải tiến chưa từng có ở những model trước đó. Trong bài báo BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (https://arxiv.org/abs/1810.04805) các tác giả đã nêu ra những cải tiến của model BERT trong các tác vụ:

- Tăng GLUE score (General Language Understanding Evaluation score), một chỉ số tổng quát đánh giá mức độ hiểu ngôn ngữ lên `80.5%`.
- Tăng accuracy trên bộ dữ liệu MultiNLI (https://cims.nyu.edu/~sbowman/multinli/) đánh giá tác vụ quan hệ văn bản (text entailment) lên 86.7%.
- Tăng accuracy F1 score trên bộ dữ liệu SQuAD v1.1 (https://rajpurkar.github.io/SQuAD-explorer/) đánh giá tác vụ question and answering lên 93.2%.

Ở thời điểm hiện tại, BERT đã được ứng dụng cho Tiếng Việt. Bạn đọc có thể tham khảo dự án PhoBERT (https://github.com/VinAIResearch/PhoBERT) của VinAI về huấn luyện trước biểu diễn từ (pre-train word embedding) sử dụng model BERT. Một số bạn ứng dụng model PhoBERT vào các tác vụ như sentiment analysis và đạt được kết quả cao như phân loại cảm xúc bình luận - Khôi Nguyễn (https://github.com/suicao/PhoBert-Sentiment-Classification/).

BERT đã được ra đời lâu như vậy và cũng đã được ứng dụng rộng rãi thì tại sao mình lại viết về model này? Đó là vì BERT và các biến thể mô hình của nó đang là hot trend và sẽ định hướng các thuật toán NLP trong tương lai.

1.3. Ngữ cảnh (Contextual) và vai trò trong NLP Top

Trước khi tìm hiểu các kỹ thuật đã tạo ra ưu thế vượt trội cho mô hình BERT. Chúng ta hãy khám phá vai trò của ngữ cảnh trong NLP.

Bản chất của ngôn ngữ là âm thanh được phát ra để diễn giải dòng suy nghĩ của con người. Trong giao tiếp, các từ thường không đứng độc lập mà chúng sẽ đi kèm với các từ khác để liên kết mạch lạc thành một câu. Hiệu quả biểu thị nội dung và truyền đạt ý nghĩa sẽ lớn hơn so với từng từ đứng độc lập.

Ngữ cảnh trong câu có một sự ảnh hưởng rất lớn trong việc giải thích ý nghĩa của từ. Hiểu được vai trò mấu chốt đó, các thuật toán NLP SOTA đều cố gắng đưa ngữ cảnh vào mô hình nhằm tạo ra sự đột phá và cải tiến và mô hình BERT cũng như vậy.

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. VD:

Câu A: `Đơn vị tiền tệ của Việt Nam là [đồng]`

Câu B: `Vợ [đồng] ý với ý kiến của chồng là tăng thêm mỗi tháng 500k tiền tiêu vặt`

Thì từ đồng sẽ mang 2 ý nghĩa khác nhau nên phải có hai biểu diễn từ riêng biệt. Các thuật toán non-context đã không đáp ứng được sự đa dạng về ngữ nghĩa của từ trong NLP.

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 trái. VD:

Câu C: Hôm nay tôi mang 200 tỷ [gửi] ở ngân hàng.

Câu D: Hôm nay tôi mang 200 tỷ [gửi] ….

Như vậy véc tơ biểu diễn của từ `gửi` được xác định thông qua các từ liền trước với nó. Nếu chỉ dựa vào các từ liền trước `Hôm nay tôi mang 200 tỷ` thì ta có thể nghĩ từ phù hợp ở vị trí hiện tại là `cho vay, mua, thanh toán,....`

Ví dụ đơn giản trên đã cho thấy các thuật toán biểu diễn từ có bối cảnh tuân theo theo một chiều sẽ gặp hạn chế lớn trong biểu diễn từ hơn so với biểu diễn 2 chiều.

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.

Thuật toán ELMo đã cải tiến hơn so với word2vec và fasttext đó là tạo ra nghĩa của từ theo bối cảnh. Trong ví dụ về từ `đồng` thì ở mỗi câu A và B chúng ta sẽ có một biểu diễn từ khác biệt.

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 (https://arxiv.org/abs/1810.04805), ULMFit (https://arxiv.org/abs/1801.06146), OpenAI GPT (https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf) đã đạt được những kết quả SOTA trên hầu hết các tác vụ của `GLUE benchmark`.

1.4. Tiếp cận nông và học sâu trong ứng dụng pre-training NLP

1.4.1. Tiếp cận nông (shallow approach)

Imagenet trong Computer Vision

Trong xử lý ảnh chúng ta đều biết tới những pretrained models nổi tiếng trên bộ dữ liệu Imagenet với 1000 classes. Nhờ số lượng classes lớn nên hầu hết các nhãn trong phân loại ảnh thông thường đều xuất hiện trong Imagenet và chúng ta có thể học chuyển giao lại các tác vụ xử lý ảnh rất nhanh và tiện lợi. Chúng ta cũng kỳ vọng NLP có một tợp hợp các pretrained models như vậy, tri thức từ model được huấn luyện trên các nguồn tài nguyên văn bản không nhãn (unlabeled text) rất dồi dào và sẵn có.

Top

Khó khăn học chuyển giao trong NLP

Tuy nhiên trong NLP việc học chuyển giao là không hề đơn giản như Computer Vision. Tại sao vậy?

Các kiến trúc mạng deep CNN của Computer Vision cho phép học chuyển giao trên đồng thời cả low-level và high-level features thông qua việc tận dụng lại các tham số từ những layers của mô hình pretrained.

Nhưng trong NLP, các thuật toán cũ hơn như `GLoVe, word2vec, fasttext` chỉ cho phép sử dụng các biểu diễn véc tơ nhúng của từ là các low-level features như là đầu vào cho layer đầu tiên của mô hình. Các layers còn lại giúp tạo ra high-level features thì dường như được huấn luyện lại từ đầu.

Như vậy chúng ta chỉ chuyển giao được các đặc trưng ở mức độ rất nông nên phương pháp này còn được gọi là tiếp cận nông (shallow approach). Việc tiếp cận với các layers sâu hơn là không thể. Điều này tạo ra một hạn chế rất lớn đối với NLP so với Computer Vision trong việc học chuyển giao. Cách tiếp cận nông trong học chuyển giao còn được xem như là **feature-based**.

Khi áp dụng feature-based, chúng ta sẽ tận dụng lại các biểu diễn từ được huấn luyện trước trên những kiến trúc mô hình cố định và những bộ văn bản có kích thước **rất lớn** để nâng cao khả năng biểu diễn từ trong không gian đa chiều. Một số pretrained feature-based bạn có thể áp dụng trong tiếng anh đã được huấn luyện sẵn đó là GloVe, word2vec (https://wikipedia2vec.github.io/wikipedia2vec/pretrained/), fasttext (https://fasttext.cc/docs/en/english-vectors.html), ELMo (https://arxiv.org/abs/1802.05365).

1.4.2. Học sâu (deep-learning)

Các mô hình NLP đột phá trong hai năm trở lại đây như BERT (https://arxiv.org/abs/1810.04805), ELMo (https://arxiv.org/pdf/1802.05365), ULMFit (https://arxiv.org/abs/1801.06146), OpenAI GPT (https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf) đã cho phép việc chuyển giao layers trong NLP khả thi hơn.

Chúng ta không chỉ học chuyển giao được các đặc trưng mà còn chuyển giao được kiến trúc của mô hình nhờ số lượng layers nhiều hơn, chiều sâu của mô hình sâu hơn trước đó.

Các kiến trúc mới phân cấp theo level có khả năng chuyển giao được những cấp độ khác nhau của đặc trưng từ low-level tới high-level. Trong khi học nông chỉ chuyển giao được low-level tại layer đầu tiên. Tất nhiên low-level cũng đóng vai trò quan trọng trong các tác vụ NLP. Nhưng high-level là những đặc trưng có ý nghĩa hơn vì đó là những đặc trưng đã được tinh luyện.

Người ta kỳ vọng rằng `ULMFit, OpenAI GPT, BERT` sẽ là những mô hình pretrained giúp tiến gần hơn tới việc xây dựng một lớp các pretrained models `ImageNet for NLP` . Các bạn có thể xem thêm ý tưởng về xây dựng Imagenet for NLP (https://ruder.io/nlp-imagenet/).

Khi học chuyển giao theo phương pháp học sâu chúng ta sẽ tận dụng lại kiến trúc từ mô hình pretrained và bổ sung một số layers phía sau để phù hợp với nhiệm vụ huấn luyện. Các tham số của các layers gốc sẽ được **fine-tunning** lại. Chỉ một số ít các tham số ở layers bổ sung được huấn luyện lại từ đầu. Bạn đọc có thể tìm hiểu thêm về fine-tuning tại Bài 33 - Phương pháp Transfer Learning (https://phamdinhkhanh.github.io/2020/04/15/TransferLearning.html).

1.2. Phương pháp transformer

1.2.1. Encoder và decoder trong BERT

Trước khi hiểu về BERT chúng ta cùng ôn lại về kỹ thuật transformer. Mình đã diễn giải kỹ thuật này tại Bài 4 - Attention is all you need (https://phamdinhkhanh.github.io/2019/06/18/AttentionLayer.html). Đây làm một lớp mô hình seq2seq gồm 2 phrase encoder và decoder. Mô hình hoàn toàn không sử dụng các kiến trúc Recurrent Neural Network của RNN mà chỉ sử dụng các layers attention để embedding các từ trong câu. Kiến trúc cụ thể của mô hình như sau:

Top

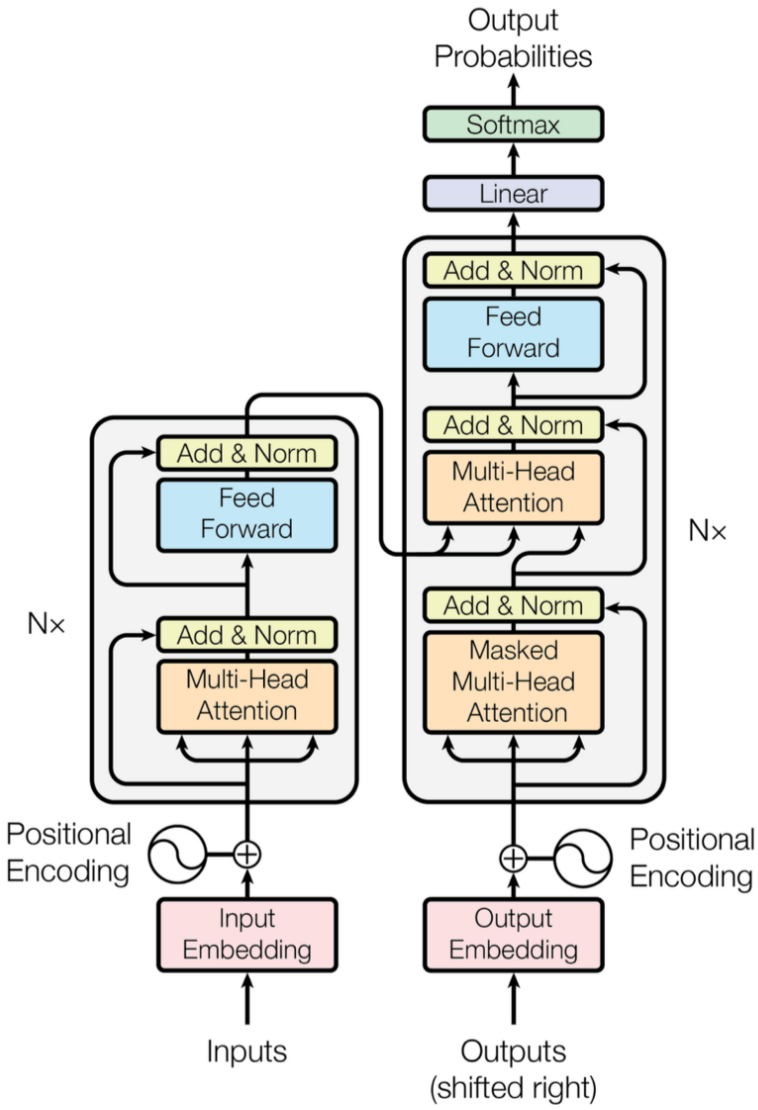


Figure 1: The Transformer - model architecture.

Hình 1: Sơ đồ kiến trúc transformer kết hợp với attention. Nguồn attention is all you need (https://arxiv.org/abs/1706.03762).

Mô hình sẽ bao gồm 2 phase.

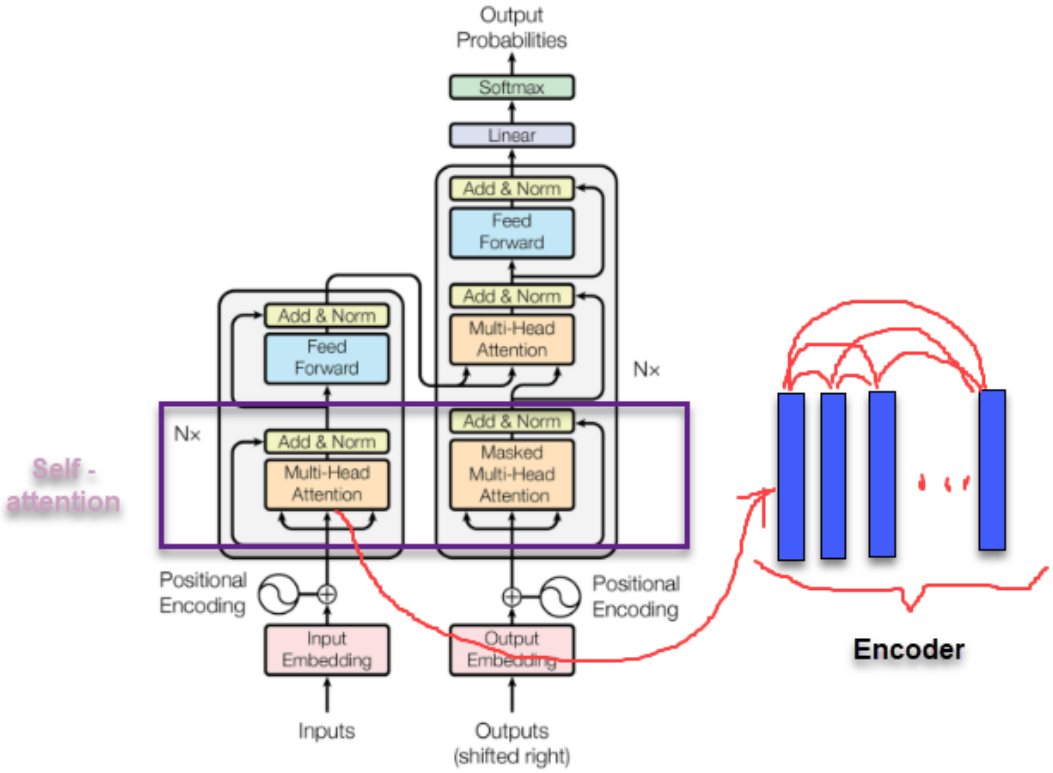
- **Encoder**: Bao gồm 6 layers liên tiếp nhau. Mỗi một layer sẽ bao gồm một sub-layer là Multi-Head Attention kết hợp với fully-connected layer như mô tả ở nhánh encoder bên trái của hình vẽ. Kết thúc quá trình encoder ta thu được một vector embedding output cho mỗi từ.
- **Decoder**: Kiến trúc cũng bao gồm các layers liên tiếp nhau. Mỗi một layer của Decoder cũng có các sub-layers gần tương tự như layer của Encoder nhưng bổ sung thêm sub-layer đầu tiên là `Masked Multi-Head Attention` có tác dụng loại bỏ các từ trong tương lai khỏi quá trình attention.

1.2.2. Các tiến trình self-attention và encoder-decoder attention

Trong kiến trúc transformer chúng ta áp dụng 2 dạng attention khác nhau tại từng bước huấn luyện.

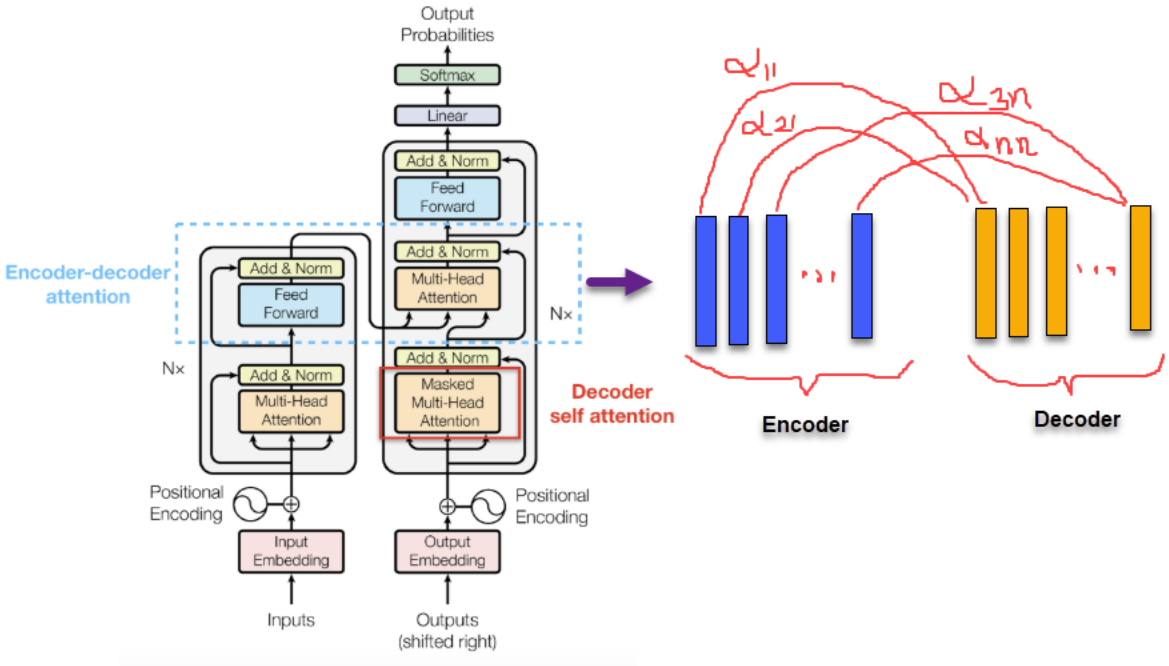
self-attention: Được sử dụng trong cùng một câu input, tại encoder hoặc tại decoder. Đây chính là attention được áp dụng tại các Multi-Head Attention ở đầu vào của cả 2 phase encoder và decoder.

Top



Hình 2: Sơ đồ vị trí áp dụng self-attention trong kiến trúc transformer. Các véc tơ embedding của cùng một chuỗi encoder hoặc decoder tự liên kết với nhau để tính toán attention như hình bên phải.

Encoder-decoder attention:



Hình 3: Bên trái là vị trí áp dụng encoder-decoder attention. Bên phải là cách tính trọng số attention khi kết hợp mỗi véc tơ embedding ở decoder với toàn bộ các véc tơ embedding ở encoder.

Sở dĩ được gọi là encoder-decoder attention vì đây là kiến trúc attention tương tác giữa các véc tơ embedding của encoder và decoder. véc tơ context được tính toán trên encoder đã được tính tương quan với véc tơ decoder nên sẽ có ý nghĩa giải thích bối cảnh của từ tại vị trí time step decoder tương ứng. Sau khi kết hợp giữa véc tơ context và véc tơ decoder ta sẽ project tiếp qua một fully connected layer để tính phân phối xác suất cho output.

Mặc dù có kiến trúc chỉ gồm các biến đổi attention nhưng Transformer lại có kết quả rất tốt trong các tác vụ NLP như sentiment analysis và dịch máy.

2. Giới thiệu về BERT

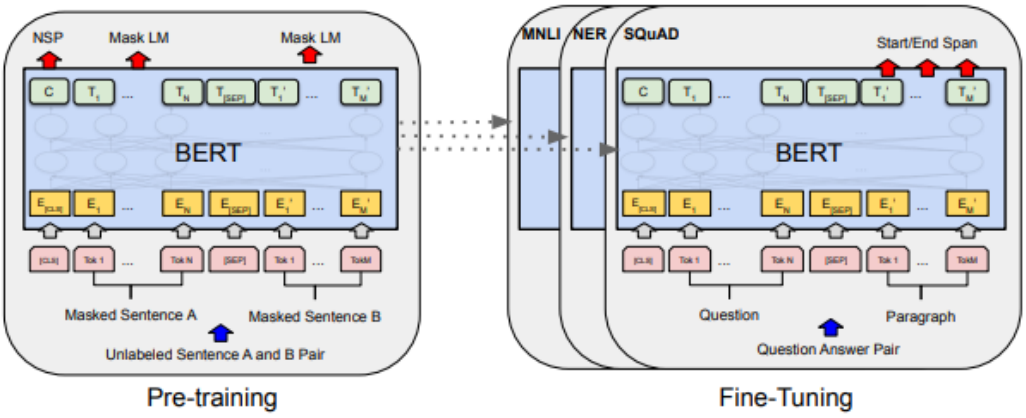
Top

BERT (https://arxiv.org/pdf/1810.04805.pdf) là viết tắt của cụm từ `Bidirectional Encoder Representation from Transformer` có nghĩa là mô hình biểu diễn từ theo 2 chiều ứng dụng kỹ thuật `Transformer` . BERT được thiết kế để huấn luyện trước các biểu diễn từ (pre-train word embedding). Điểm đặc biệt ở BERT đó là nó có thể điều hòa cân bằng bối cảnh theo cả 2 chiều trái và phải.

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

2.1. Fine-tuning model BERT

Một điểm đặc biệt ở BERT mà các model embedding trước đây chưa từng có đó là kết quả huấn luyện có thể fine-tuning được. Chúng ta sẽ thêm vào kiến trúc model một output layer để tùy biến theo tác vụ huấn luyện.



Hình 4: Toàn bộ tiến trình pre-training và fine-tuning của BERT. Một kiến trúc tương tự được sử dụng cho cả pretrain-model và fine-tuning model. Chúng ta sử dụng cùng một tham số pretrain để khởi tạo mô hình cho các tác vụ down stream khác nhau. Trong suốt quá trình fine-tuning thì toàn bộ các tham số của layers học chuyển giao sẽ được fine-tune. Đối với các tác vụ sử dụng input là một cặp sequence (pair-sequence) ví dụ như `question and answering` thì ta sẽ thêm token khởi tạo là `[CLS]` ở đầu câu, token `[SEP]` ở giữa để ngăn cách 2 câu.

Tiến trình áp dụng fine-tuning sẽ như sau:

- **Bước 1**: Embedding toàn bộ các token của cặp câu bằng các véc tơ nhúng từ pretrain model. Các token embedding bao gồm cả 2 token là `[CLS]` và `[SEP]` để đánh dấu vị trí bắt đầu của câu hỏi và vị trí ngăn cách giữa 2 câu. 2 token này sẽ được dự báo ở output để xác định các phần `Start/End Spand` của câu output.
- **Bước 2**: Các embedding véc tơ sau đó sẽ được truyền vào kiến trúc multi-head attention với nhiều block code (thường là 6, 12 hoặc 24 blocks tùy theo kiến trúc BERT). Ta thu được một véc tơ output ở encoder.
- **Bước 3**: Để dự báo phân phối xác suất cho từng vị trí từ ở decoder, ở mỗi time step chúng ta sẽ truyền vào decoder véc tơ output của encoder và véc tơ embedding input của decoder để tính encoder-decoder attention (cụ thể về encoder-decoder attention là gì các bạn xem lại mục 2.1.1). Sau đó projection qua liner layer và softmax để thu được phân phối xác suất cho output tương ứng ở time step t.
- **Bước 4**: Trong kết quả trả ra ở output của transformer ta sẽ cố định kết quả của câu Question sao cho trùng với câu Question ở input. Các vị trí còn lại sẽ là thành phần mở rộng `Start/End Span` tương ứng với câu trả lời tìm được từ câu input.

Lưu ý quá trình huấn luyện chúng ta sẽ fine-tune lại toàn bộ các tham số của model BERT đã cut off top linear layer và huấn luyện lại từ đầu các tham số của linear layer mà chúng ta thêm vào kiến trúc model BERT để customize lại phù hợp với bài toán.

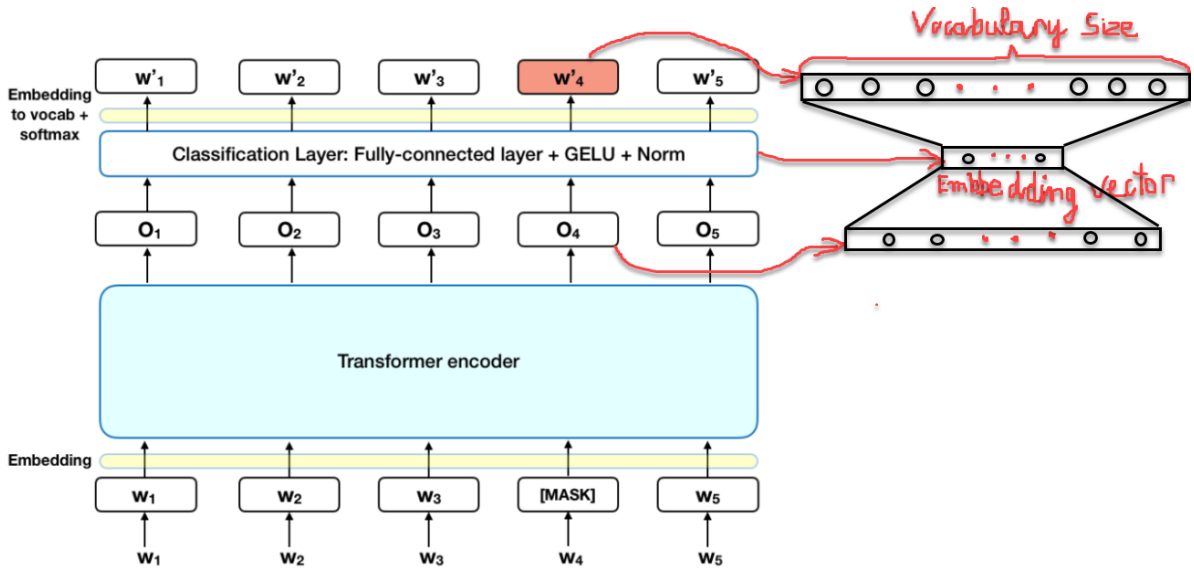
Như vậy các bạn đã hình dung được model BERT được fine-tuning trong một tác vụ như thế nào rồi chứ? Tôi cá rằng qua quá trình thực hành ở bài sau các bạn sẽ nắm vững hơn cách thức fine-tune BERT model.

2.3. Masked ML (MLM)

Top

Masked ML là một tác vụ cho phép chúng ta fine-tuning lại các biểu diễn từ trên các bộ dữ liệu unsupervised-text bất kỳ. Chúng ta có thể áp dụng Masked ML cho những ngôn ngữ khác nhau để tạo ra biểu diễn embedding cho chúng. Các bộ dữ liệu của tiếng anh có kích thước lên tới vài vài trăm tới vài nghìn GB được huấn luyện trên BERT đã tạo ra những kết quả khá ấn tượng.

Bên dưới là sơ đồ huấn luyện BERT theo tác vụ Masked ML



Hình 5: Sơ đồ kiến trúc BERT cho tá vụ Masked ML.

Theo đó:

- 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 là $O_1, O_2, \ldots, O_5$
- 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.
- Cuối cùng ta thu được véc tơ nhúng của mỗi một từ tại vị trí MASK sẽ là embedding véc tơ giảm chiều của véc tơ O_i sau khi đi qua fully connected layer như mô tả trên hình vẽ bên phải.

Hàm loss function của BERT sẽ bỏ qua mất mát từ những từ không bị che dấu và chỉ đưa vào mất mát của những từ bị che dấu. Do đó mô hình sẽ hội tụ lâu hơn nhưng đây là đặc tính bù trừ cho sự gia tăng ý thức về bối cảnh. Việc lựa chọn ngẫu nhiên 15% số lượng các từ bị che dấu cũng tạo ra vô số các kịch bản input cho mô hình huấn luyện nên mô hình sẽ cần phải huấn luyện rất lâu mới học được toàn diện các khả năng.

2.4. 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.

Top

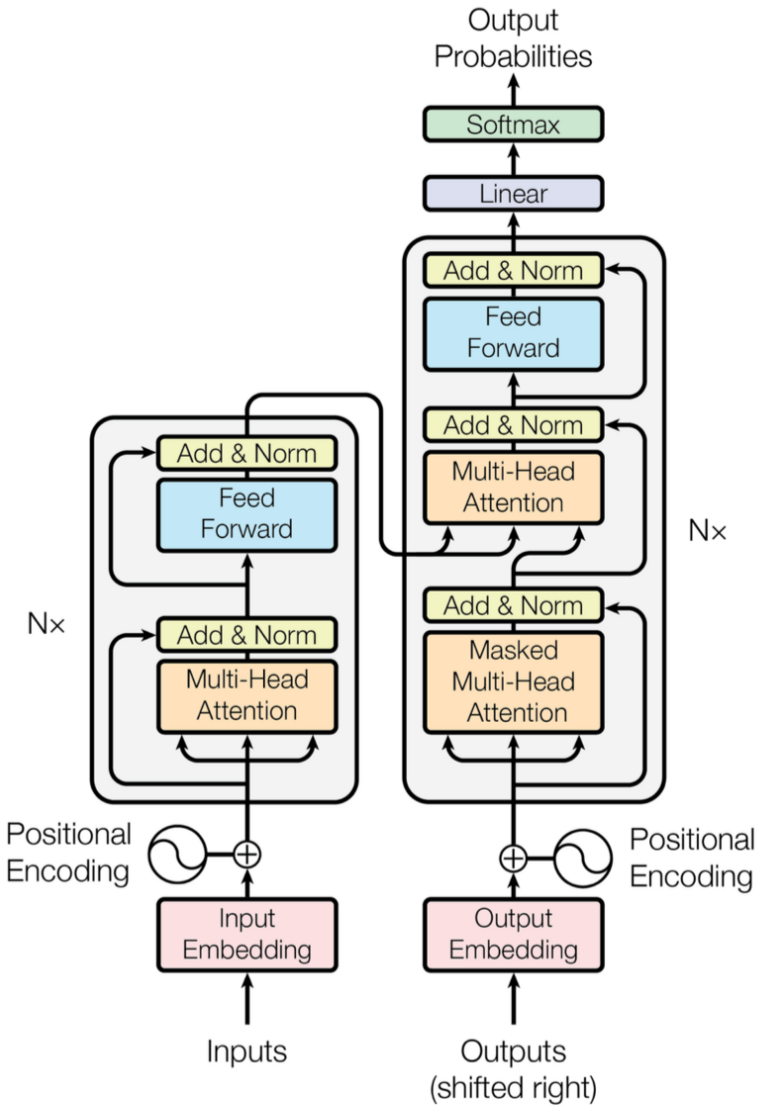


Figure 1: The Transformer - model architecture.

Hình 6: Sơ đồ kiến trúc model BERT cho tác vụ NSP.

Thông tin input được preprocessing trước khi đưa vào mô hình huấn luyện bao gồm:

- Ngữ nghĩa của từ (token embeddings): Thông qua các embedding véc tơ cho từng từ. Các véc tơ được khởi tạo từ pretrain model.

Ngoài embedding biểu diễn từ của các từ trong câu, mô hình còn embedding thêm một số thông tin:

- Loại câu (segment embeddings): Gồm hai véc tơ là E_A nếu từ thuộc câu thứ nhất và E_B nếu từ thuộc câu thứ hai.
- Vị trí của từ trong câu (position embedding): là các véc tơ $E_0, \ldots, E_{10}$. Tương tự như positional embedding trong transformer.

Véc tơ input sẽ bằng tổng của cả ba thành phần embedding theo `từ, câu và vị trí.`

3. Các kiến trúc model BERT

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:

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

Top

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.

4. Thực hành model BERT

4.1. Giới thiệu về bài toán

Chúng ta sẽ cùng xây dựng một ứng dụng Question and Answering có chức năng hỏi đáp.

Dữ liệu bao gồm:

Input: Một cặp câu <Question, Paragraph>, Question là câu hỏi và Paragraph là đoạn văn bản chứa câu trả lời cho câu hỏi.

Output: Câu trả lời được trích suất từ Paragraph.

Đây là một ứng dụng khá thú vụ mà tôi đã compile thành công trên thiết bị android. Các bạn có thể download ứng dụng về và chạy thử nghiệm BERT - Tensorflow Lite - Khanh Blog (https://www.facebook.com/TowardDataScience/videos/201232064499053/).

Để thực hiện tác vụ này tôi sẽ sử dụng pretrain model từ package transformer. Chúng ta có thể cài thông qua câu lệnh bên dưới.

```
1     !pip install transformers
```

4.2. Xây dựng một ứng dụng Question and Answering

Các bước dữ liệu:

- **Tokenize**: Tạo chuỗi token là concatenate của cặp câu `<Question, Paragraph>` , thêm các token `[CLS]` đánh dấu vị trí bắt đầu câu `Question` và `[SEP]` đánh dấu vị trí kết thúc câu. Sau đó Tokenize toàn bộ cặp câu `<Question, Paragraph>` thành chuỗi index từ từ điển.
- **Set Segment IDs**: Tạo véc tơ segment cho cặp câu `Question` và `Paragraph` . Trong đó index 0 đánh dấu các vị trí thuộc câu A và index 1 đánh dấu các vị trí thuộc câu B.
- **Evaluate**: Khởi tạo model từ pretrain model `bert-large-uncased-whole-word-masking-finetuned-squad` . Và dự báo các vị trí `start` và `end` nằm trong chuỗi token.
- **Reconstruct Answer**:Trích suất thông tin câu trả lời.

Source code của mô hình được tham khảo tại Question answering with fine tuned BERT (https://mccormickml.com/2020/03/10/question-answering-with-a-fine-tuned-BERT)

Top

```
from transformers import BertTokenizer
from transformers import BertForQuestionAnswering
import torch
# Initialize tokenizer for corpus of bert-large-uncased
tokenizer = BertTokenizer.from_pretrained('bert-large-uncased-whole-word-masking-f

# Initialize model BertForQuestionAnswering for bert-large-uncased
model = BertForQuestionAnswering.from_pretrained('bert-large-uncased-whole-word-ma

def answer_question(question, answer_text):
    '''
    Lấy input là chuỗi string của câu question và answer_text chứa nội dung câu tr
    Xác định từ trong answer_text là câu trả lời và in ra.
    '''
    # ======== Tokenize ========
    # Áp dụng tokenizer cho cặp câu <question, answer_text>. input_ids là concaten
    input_ids = tokenizer.encode(question, answer_text)

    # ======== Set Segment IDs ========
    # Xác định vị trí đầu tiên chứa token [SEP] trong câu.
    sep_index = input_ids.index(tokenizer.sep_token_id)

    # Tạo segment index đánh dấu các vị trí từ thuộc question (giá trị 0) và answe
    num_seg_a = sep_index + 1
    num_seg_b = len(input_ids) - num_seg_a
    segment_ids = [0]*num_seg_a + [1]*num_seg_b

    # Kiểm tra độ dài segment_ids phải bằng input_ids
    assert len(segment_ids) == len(input_ids)

    # ======== Evaluate ========
    # Dự báo phân phối xác suất của vị trí của từ start và từ end trong chuỗi conc
    start_scores, end_scores = model(torch.tensor([input_ids]), # chuỗi index biểu
                                     token_type_ids=torch.tensor([segment_ids])) #

    # ======== Reconstruct Answer ========
    # Tìm ra vị trí start, end với score là cao nhất
    answer_start = torch.argmax(start_scores)
    answer_end = torch.argmax(end_scores)

    # Chuyển ngược từ input_ids sang list tokens
    tokens = tokenizer.convert_ids_to_tokens(input_ids)

    # Token đầu tiên của câu trả lời
    answer = tokens[answer_start]

    # Lựa chọn các thành phần còn lại của câu trả lời và join chúng với whitespace
    for i in range(answer_start + 1, answer_end + 1):

        # Nếu token là một subword token (có dấu ## ở đầu) thì combine vào answer
        if tokens[i][0:2] == '##':
            answer += tokens[i][2:]

        # Nếu trái lại thì combine trực tiếp vào answer.
        else:
            answer += ' ' + tokens[i]
    print('Question: "' + question + '"')
    print('Answer: "' + answer + '"')
```

Thử nghiệm kết quả của mô hình trên một vài cặp câu `<Question, Paragraph>`.

Top

```
question = "what is my dog name?"
paragraph = "I have a dog. It's name is Ricky. I get it at my 15th birthday, when i
answer_question(question, paragraph)
```

```
Question: "what is my dog name?"
Answer: "ricky"
```

Thử nghiệm một văn bản khác dài hơn. Tôi sẽ lấy một đoạn văn mô tả tiểu sử của ông vua toán học `Euler` và hỏi thuật toán ngày sinh của ông ấy. Các bạn hãy xem kết quả nhé.

```
question = "when Leonhard Euler was born?"
paragraph = "Leonhard Euler: 15 April 1707 – 18 September 1783 was a Swiss mathemat
physicist, astronomer, geographer, logician and engineer who made important and inf
such as infinitesimal calculus and graph theory, \
while also making pioneering contributions to several branches such as topology and
He also introduced much of the modern mathematical terminology and notation, \
particularly for mathematical analysis, such as the notion of a mathematical functi
answer_question(question, paragraph)
```

```
Question: "when Leonhard Euler was born?"
Answer: "15 april 1707"
```

Ta có thể thấy kết quả là chính xác.

Việc áp dụng pretrain model sẵn có trên package transformer cho tác vụ `Question and Answering` là khá dễ dàng. Chúng ta cũng có thể fine-tuning lại các kiến trúc model question and answering cho dữ liệu Tiếng Việt để tạo ra các ứng dụng hỏi đáp cho riêng mình. Để thực hiện được điều đó đòi hỏi phải nắm vững kiến trúc của model BERT được trình bày trong bài viết này. Có lẽ ở một bài sau tôi sẽ hướng dẫn các bạn thực hành điều này.

5. Tổng kết

Như vậy qua bài này tôi đã hướng dẫn các bạn kiến trúc tổng quát của model BERT và cách thức áp dụng model BERT vào trong các tác vụ down stream task trong NLP như Masked ML, Next Sentence Prediction và thực hành xây dựng một ứng dụng Question and Answering ngay trên pretrain model của transformer package.

Các kiến trúc biến thể mới của BERT hiện tại vẫn đang được nghiên cứu và tiếp tục phát triển như ROBERTA (https://huggingface.co/transformers/model_doc/roberta.html), ALBERT (https://huggingface.co/transformers/model_doc/albert.html), CAMEBERT (https://huggingface.co/transformers/model_doc/camembert.html), XLMROBERTA (https://huggingface.co/transformers/model_doc/xlmroberta.html), …

Ngày càng có nhiều các pretrain model trên BERT áp dụng cho nhiều ngôn ngữ khác nhau trên toàn thế giới và tạo ra một sự đột phá trong NLP. Ngôn ngữ Tiếng Việt của chúng ta cũng đã được VinAI nghiên cứu và huấn luyện pretrain model thành công. Bạn đọc muốn sử dụng pretrain model này trong các tác vụ NLP có thể tham khảo thêm tại PhoBERT (https://github.com/VinAIResearch/PhoBERT).

6. Tài liệu

1. From word embeddings to Pretrained Language models (https://towardsdatascience.com/from-word-embeddings-to-pretrained-language-models-a-new-age-in-nlp-part-2-e9af9a0bdcd9)
2. BERT explained state of the art language model for NLP (https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270)
3. huggingface - transformer github package (https://github.com/huggingface/transformers/)
4. question answering with a fine tuned BERT (https://mccormickml.com/2020/03/10/question-answering-with-a-fine-tuned-BERT)

Top

5. BERT fine-tuning with cloud (https://colab.research.google.com/github/tensorflow/tpu/blob/master/tools/colab/bert_finetuning_with_cloud_tpus.ipynb
6. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (https://arxiv.org/abs/1810.04805)
7. OpenAI GPT - paper (https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf)
8. ULMFit paper (https://arxiv.org/abs/1801.06146)

Top