ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

23TNT1

Đồ án cuối kỳ

Hướng 2: BERT

Môn học: Phương pháp toán cho Trí tuệ nhân tạo

Nhóm 13:

Lê Thái Ngọc - 23122012

Nguyễn Ngọc Khoa - 23122036

Nguyễn Lê Phúc Thắng - 22120332

Giảng viên hướng dẫn:

Nguyễn Ngọc Toàn

TS. Cấn Trần Thành Trung

Ngày 5 tháng 6 năm 2025

Mục lục

Danh sách bảng

Danh sách hình vẽ

Danh Mục Thuật Ngữ Viết Tắt

- **BERT**: Bidirectional Encoder Representations from Transformers
- **NLP**: Natural Language Processing (Xử lý Ngôn ngữ Tự nhiên)
- **LM**: Language Model (Mô hình Ngôn ngữ)
- **MLM**: Masked Language Model (Mô hình Ngôn ngữ Che)
- **NSP**: Next Sentence Prediction (Dự đoán Câu Tiếp theo)
- **SOTA**: State-of-the-art (Hiện đại nhất, Tiên tiến nhất)
- **GLUE**: General Language Understanding Evaluation
- **SQuAD**: Stanford Question Answering Dataset
- **NER**: Named Entity Recognition (Nhận dạng Thực thể Tên)
- **ELMo**: Embeddings from Language Models
- **GPT**: Generative Pre-trained Transformer
- **FFN**: Feed-Forward Network (Mạng Truyền thẳng)
- **ReLU**: Rectified Linear Unit
- **GELU**: Gaussian Error Linear Unit
- **BiLSTM**: Bidirectional Long Short-Term Memory
- **OOV**: Out-of-Vocabulary (Từ Ngoài Từ điển)
- **MNLI**: Multi-Genre Natural Language Inference
- **QQP**: Quora Question Pairs
- **QNLI**: Question Natural Language Inference
- **SST-2**: Stanford Sentiment Treebank

- **CoLA**: Corpus of Linguistic Acceptability
- **STS-B**: Semantic Textual Similarity Benchmark
- **MRPC**: Microsoft Research Paraphrase Corpus
- **RTE**: Recognizing Textual Entailment
- **SWAG**: Situations With Adversarial Generations
- **TPU**: Tensor Processing Unit
- **GPU**: Graphics Processing Unit
- **TF-IDF**: Term Frequency-Inverse Document Frequency
- **CSV**: Comma-Separated Values
- **BPE**: Byte-Pair Encoding

Thông tin nhóm

Nhóm chúng em là **Nhóm 13** của môn *Phương pháp toán cho Trí tuệ nhân tạo*, lớp *23TNT1*

1. Các thành viên

Họ và tên	**MSSV**	**Email**
Lê Thái Ngọc	23122012	23122012@student.hcmus.edu.vn
Nguyễn Ngọc Khoa	23122036	23122036@student.hcmus.edu.vn
Nguyễn Lê Phúc Thắng	22120332	22120332@student.hcmus.edu.vn

2. Cụ thể nội dung thực hiện & Đánh giá mức độ hoàn thành

Công việc	**Thực hiện**	**Mức độ hoàn thành**
Thảo luận: tìm bài báo	Khoa, Ngọc	50%
Report: giới thiệu, tóm tắt bài báo	Khoa	70%
Report: lí do chọn bài báo	Khoa	80%
Report: kế hoạch thực hiện dự kiến	Khoa	70%

1 Giới Thiệu Chung về BERT

1.1 Bối cảnh và Tầm quan trọng của Mô hình Biểu diễn Ngôn ngữ

Trong lĩnh vực Xử lý Ngôn ngữ Tự nhiên (NLP), việc xây dựng các biểu diễn ngôn ngữ (language representations) hiệu quả luôn là một mục tiêu trung tâm. Các biểu diễn này, thường ở dạng vector, cố gắng nắm bắt các đặc tính ngữ nghĩa và cú pháp của từ, cụm từ hoặc câu, làm nền tảng cho nhiều tác vụ NLP phức tạp hơn.

Trước khi BERT ra đời, các phương pháp biểu diễn ngôn ngữ phổ biến bao gồm các word embeddings tĩnh như Word2Vec [14] và GloVe [16]. Để hiểu rõ hạn chế của các phương pháp này, hãy xét một ví dụ cụ thể: từ "bank" trong câu "I went to the bank to deposit money" (ngân hàng) và "The river bank was flooded" (bờ sông) sẽ có cùng một vector biểu diễn, mặc dù ý nghĩa hoàn toàn khác nhau. Đây chính là hạn chế cố hữu của biểu diễn tĩnh - không thể nắm bắt được ngữ cảnh.

Để khắc phục nhược điểm này, các mô hình biểu diễn ngôn ngữ theo ngữ cảnh (contextual language representation models) đã được phát triển. Các mô hình dựa trên kiến trúc tuần tự như Mạng Nơ-ron Hồi quy (RNN) và Bộ nhớ Dài-Ngắn Hạn (LSTM) bắt đầu cho thấy khả năng nắm bắt thông tin ngữ cảnh. Tuy nhiên, các mô hình này gặp phải vấn đề "information bottleneck" - thông tin phải được nén qua một vector trạng thái ẩn có kích thước cố định, dẫn đến mất mát thông tin khi xử lý chuỗi dài.

ELMo (Embeddings from Language Models) [17] đã cố gắng giải quyết vấn đề này bằng cách kết hợp biểu diễn từ cả hai chiều. Cụ thể, ELMo huấn luyện hai LSTM độc lập: một đọc văn bản từ trái sang phải, một đọc từ phải sang trái, sau đó nối (concatenate) các biểu diễn. Tuy nhiên, đây chỉ là kết hợp "nông" (shallow) - hai LSTM không thực sự tương tác với nhau trong quá trình học.

Sự ra đời của kiến trúc Transformer [23] với cơ chế tự chú ý (self-attention) đã mang lại một cuộc cách mạng. Khác với RNN phải xử lý tuần tự từng từ, Transformer có thể xử lý toàn bộ chuỗi song song, cho phép mô hình hóa các phụ thuộc tầm xa một cách trực tiếp. OpenAI GPT [19] đã tận dụng Transformer nhưng vẫn giữ hướng huấn luyện từ trái sang phải, giới hạn khả năng hiểu ngữ cảnh toàn diện.

BERT (Bidirectional Encoder Representations from Transformers) [5], được Google giới thiệu năm 2018, đã giải quyết triệt để vấn đề này. BERT sử dụng một chiến lược đột phá: che ngẫu nhiên một số từ trong câu và yêu cầu mô hình dự đoán chúng dựa trên ngữ cảnh hai chiều. Điều này cho phép

BERT học được biểu diễn sâu sắc hai chiều (deeply bidirectional) - mỗi từ được hiểu dựa trên toàn bộ ngữ cảnh xung quanh, không chỉ một phía.

1.2 Mục tiêu và Đóng góp chính của BERT

Mục tiêu chính của bài báo BERT [5] là giới thiệu một phương pháp tiền huấn luyện (pre-training) các biểu diễn ngôn ngữ sâu sắc hai chiều, có khả năng nắm bắt thông tin ngữ nghĩa và cú pháp phong phú từ dữ liệu văn bản không gán nhãn. Đây là một bước đột phá quan trọng vì dữ liệu không gán nhãn rất dồi dào và dễ thu thập, trong khi dữ liệu có nhãn cho các tác vụ cụ thể thường khan hiếm và tốn kém.
Những đóng góp chính của BERT bao gồm:

- **Kiến trúc hai chiều sâu sắc (Deeply Bidirectional):** Đây là điểm khác biệt quan trọng nhất của BERT. Trong khi các mô hình trước đó như GPT chỉ nhìn ngữ cảnh một chiều, hoặc như ELMo kết hợp hai chiều một cách độc lập, BERT cho phép mỗi lớp của mô hình xem xét ngữ cảnh hai chiều đồng thời. Điều này có nghĩa là khi xử lý từ "bank" trong câu, BERT có thể đồng thời xem xét cả "went to the" (bên trái) và "to deposit money" (bên phải) để hiểu đây là "ngân hàng" chứ không phải "bờ sông".

- **Hai tác vụ tiền huấn luyện sáng tạo:**

  - **Masked Language Model (MLM):** BERT che ngẫu nhiên 15% các từ trong câu và yêu cầu mô hình dự đoán chúng. Ví dụ, từ câu "The cat sat on the mat", BERT có thể che từ "sat" thành "[MASK]" và học cách dự đoán từ đó dựa trên ngữ cảnh. Điều quan trọng là BERT sử dụng chiến lược 80-10-10: 80% thời gian thay bằng [MASK], 10% thay bằng từ ngẫu nhiên, 10% giữ nguyên. Chiến lược này giúp giảm sự khác biệt giữa giai đoạn huấn luyện (có [MASK]) và sử dụng thực tế (không có [MASK]).

  - **Next Sentence Prediction (NSP):** BERT học dự đoán xem hai câu có liên tiếp trong văn bản gốc hay không. Ví dụ, "The weather is nice today. Let's go for a walk" (IsNext) so với "The weather is nice today. Penguins live in Antarctica" (NotNext). Tác vụ này giúp BERT hiểu mối quan hệ giữa các câu, rất quan trọng cho các ứng dụng như hỏi đáp hay suy luận ngôn ngữ.

- **Hiệu suất vượt trội:** BERT đã thiết lập kỷ lục mới trên 11 tác vụ NLP khác nhau. Trên GLUE benchmark, $\text{BERT}_{\text{LARGE}}$ đạt điểm trung bình 80.5, vượt xa mô hình tốt nhất trước đó (OpenAI GPT với 72.8). Trên SQuAD v1.1, BERT thậm chí vượt qua hiệu suất con người (F1: 93.2 so với 91.2).
- **Tính linh hoạt và dễ áp dụng:** BERT được thiết kế với triết lý "pre-train once, fine-tune for everything". Sau khi tiền huấn luyện trên dữ liệu lớn (BookCorpus và Wikipedia), BERT có thể được tinh chỉnh cho bất kỳ tác vụ NLP nào chỉ bằng cách thêm một lớp đầu ra đơn giản. Quá trình tinh chỉnh thường chỉ mất vài giờ trên GPU, so với hàng tuần cho việc huấn luyện từ đầu.

Sự thành công của BERT không chỉ nằm ở hiệu suất vượt trội mà còn ở việc nó mở ra một hướng nghiên cứu mới - sử dụng mô hình ngôn ngữ lớn được tiền huấn luyện làm nền tảng cho mọi tác vụ NLP. Điều này đã dẫn đến sự phát triển của cả một thế hệ mô hình mới như RoBERTa, ALBERT, và xa hơn là GPT-3, ChatGPT - định hình lại toàn bộ lĩnh vực AI ngôn ngữ.

2 Kiến trúc BERT và các Thành phần Cốt lõi

Kiến trúc của BERT dựa trên kiến trúc Transformer [23], cụ thể là phần bộ mã hóa (encoder). Để hiểu sâu về BERT, chúng ta cần phân tích từng thành phần cốt lõi và cách chúng kết hợp để tạo nên sức mạnh của mô hình.

2.1 Tổng quan về Kiến trúc Transformer

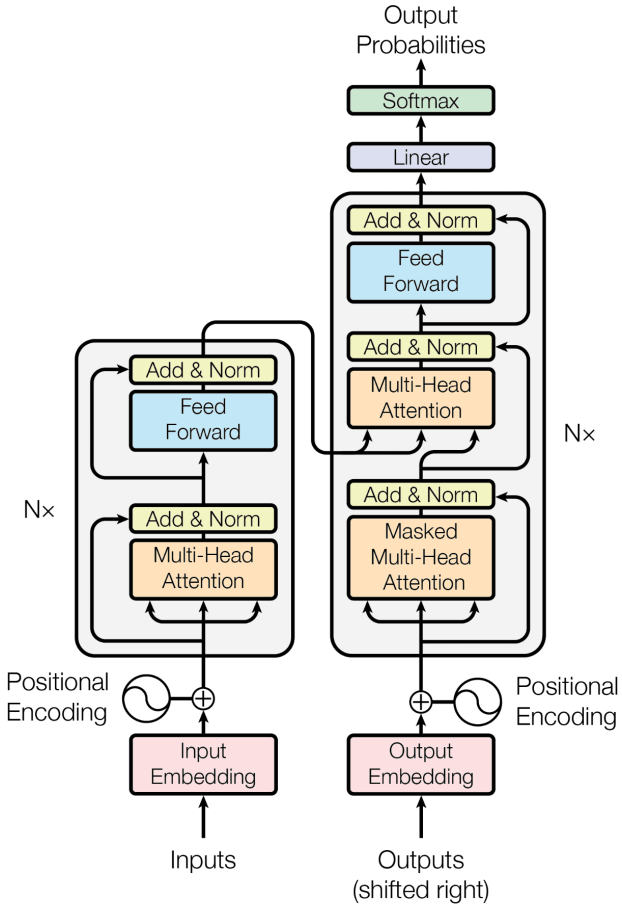
Transformer là một kiến trúc mạng nơ-ron được giới thiệu với mục tiêu ban đầu là cải thiện các mô hình dịch máy. Điểm đột phá của Transformer là loại bỏ hoàn toàn các cấu trúc tuần tự (RNN, LSTM) và thay thế bằng cơ chế attention, cho phép xử lý song song toàn bộ chuỗi.
Kiến trúc Transformer gồm hai phần chính:

- **Encoder (Bộ mã hóa):** Xử lý chuỗi đầu vào và tạo ra biểu diễn ngữ cảnh
- **Decoder (Bộ giải mã):** Sinh ra chuỗi đầu ra dựa trên biểu diễn từ encoder

BERT chỉ sử dụng phần Encoder vì mục tiêu là tạo biểu diễn ngôn ngữ, không phải sinh văn bản. Mỗi khối encoder trong BERT bao gồm:

- **Lớp Multi-Head Self-Attention:** Cho phép mô hình "chú ý" đến các vị trí khác nhau trong chuỗi khi xử lý mỗi từ. Ví dụ, khi xử lý từ "it" trong câu "The animal didn't cross the street because it was too tired", lớp attention giúp mô hình liên kết "it" với "animal" chứ không phải "street".
- **Lớp Feed-Forward Network:** Xử lý độc lập từng vị trí với cùng một mạng nơ-ron. Điều này giúp biến đổi phi tuyến các biểu diễn sau khi đã tổng hợp thông tin ngữ cảnh từ attention.

Mỗi lớp con đều có kết nối phần dư (residual connection) và chuẩn hóa lớp (layer normalization), giúp huấn luyện ổn định các mạng sâu.



Hình 1: Kiến trúc tổng quan của mô hình Transformer. BERT chỉ sử dụng phần Encoder (bên trái). Mỗi khối encoder chứa lớp Multi-Head Attention và Feed-Forward, được kết nối bằng residual connections và layer normalization.

2.2 Cơ chế Tự chú ý (Self-Attention) - Trái tim của BERT

Self-attention là cơ chế cho phép mỗi từ trong chuỗi "nhìn" và tương tác với mọi từ khác để xây dựng biểu diễn của mình. Điều này khác biệt hoàn toàn với RNN chỉ có thể xem thông tin tuần tự.

2.2.1 Trực quan về Attention

Hãy tưởng tượng bạn đang đọc câu và cần hiểu từ "bank". Não bộ của bạn tự động "chú ý" đến các từ xung quanh như "river", "money", "deposit" để xác định nghĩa. Self-attention mô phỏng chính quá trình này.

2.2.2 Công thức toán học

Với đầu vào là các vector biểu diễn của từ, self-attention tính toán ba ma trận:

- **Query (Q):** "Tôi đang tìm kiếm thông tin gì?"
- **Key (K):** "Tôi có thông tin gì để cung cấp?"
- **Value (V):** "Thông tin thực sự tôi sẽ truyền đi là gì?"

Công thức attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Trong đó:

- QK^T: Tính điểm tương đồng giữa queries và keys (ma trận attention scores)
- $\sqrt{d_k}$: Hệ số chuẩn hóa để tránh gradient vanishing khi d_k lớn
- softmax: Chuyển điểm số thành xác suất (tổng = 1)
- Nhân với V: Tổng hợp thông tin dựa trên trọng số attention

2.2.3 Ví dụ minh họa

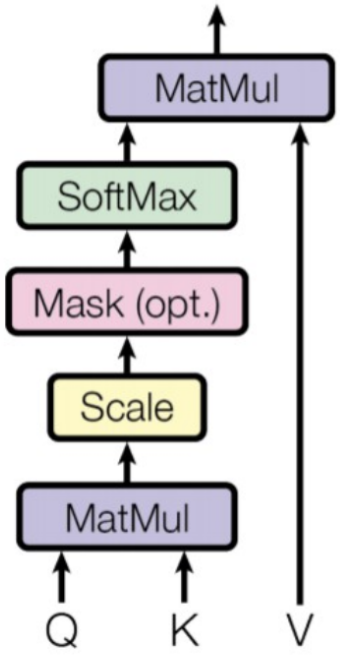
Xét câu: "The cat sat on the mat" Khi tính attention cho từ "sat":

1. Query của "sat" được so sánh với Key của mọi từ

2. Điểm số cao với "cat" (chủ ngữ) và "mat" (địa điểm)

3. Sau softmax, ta có trọng số attention (ví dụ: cat=0.3, sat=0.1, on=0.1, the=0.15, mat=0.35)

4. Biểu diễn mới của "sat" = tổng có trọng số của các Value



Hình 2: Cơ chế Scaled Dot-Product Attention. MatMul thực hiện phép nhân ma trận QK^T, Scale chia cho $\sqrt{d_k}$, Mask (tùy chọn) che các vị trí không hợp lệ, Softmax chuẩn hóa thành xác suất, và MatMul cuối cùng tính tổng có trọng số của Values.

2.3 Chú ý Đa đầu (Multi-Head Attention) - Sức mạnh từ sự đa dạng

Một cơ chế attention duy nhất chỉ có thể nắm bắt một loại mối quan hệ. Multi-Head Attention giải quyết hạn chế này bằng cách chạy nhiều attention song song, mỗi cái tập trung vào khía cạnh khác nhau.

2.3.1 Ý tưởng chính

Thay vì có một attention lớn, ta chia thành h attention nhỏ hơn (heads). Mỗi head học một "góc nhìn" khác:

- Head 1 có thể học quan hệ cú pháp (chủ ngữ - vị ngữ)

- Head 2 có thể học quan hệ ngữ nghĩa (từ đồng nghĩa)
- Head 3 có thể học khoảng cách (từ gần nhau)
- ...

2.3.2 Công thức toán học

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$$

$$\text{với head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Trong đó:

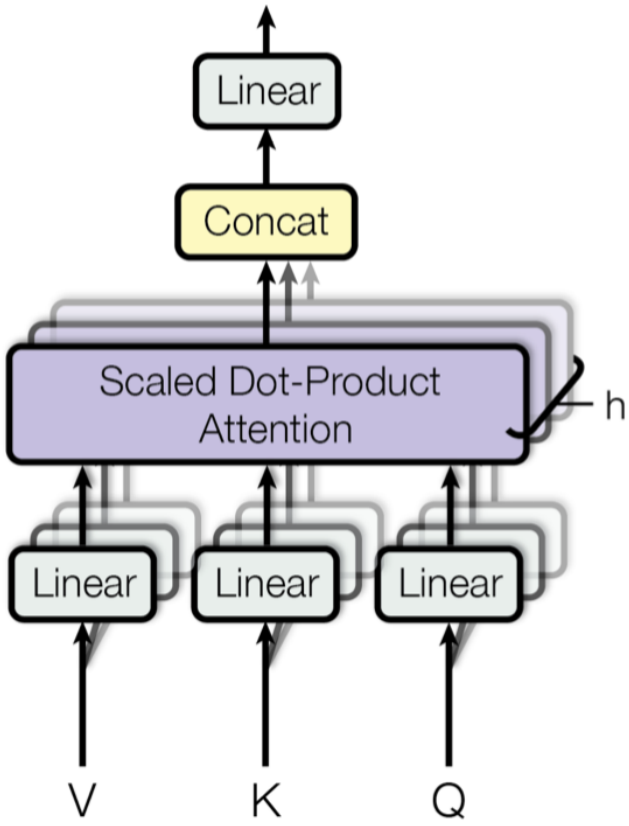
- $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d_{model} \times d_k}$: Ma trận chiếu cho head thứ i
- $W^O \in \mathbb{R}^{hd_v \times d_{model}}$: Ma trận chiếu đầu ra
- $d_k = d_v = d_{model}/h$: Kích thước mỗi head

$\text{BERT}_{\text{BASE}}$ sử dụng $h = 12$ heads với $d_{model} = 768$, nên mỗi head có kích thước $d_k = 64$.

2.3.3 Ví dụ thực tế

Trong câu "The bank will not approve the loan", các head khác nhau có thể học:

- Head 1: Liên kết "bank" với "loan" (quan hệ ngữ nghĩa tài chính)
- Head 2: Liên kết "will not" với "approve" (phủ định)
- Head 3: Liên kết "the" với danh từ theo sau (quan hệ ngữ pháp)



Hình 3: Cơ chế Multi-Head Attention. Đầu vào được chiếu thành h bộ Q, K, V độc lập. Mỗi head thực hiện attention riêng, sau đó các kết quả được nối lại và chiếu qua W^O để tạo đầu ra cuối cùng.

2.4 Position-wise Feed-Forward Networks - Biến đổi phi tuyến

Sau khi tổng hợp thông tin qua attention, mỗi vị trí được xử lý qua một mạng FFN giống nhau:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Đặc điểm quan trọng:

- Cùng một FFN được áp dụng cho mọi vị trí (parameter sharing)
- Lớp ẩn có kích thước lớn hơn ($d_{ff} = 3072$ so với $d_{model} = 768$)
- ReLU activation tạo tính phi tuyến

FFN đóng vai trò như bộ xử lý đặc trưng cục bộ sau khi đã có ngữ cảnh toàn cục từ attention.

2.5 Residual Connections và Layer Normalization

Mỗi sublayer (attention hoặc FFN) được bọc bởi:

$$\text{LayerNorm}(x + \text{Sublayer}(x))$$

Lợi ích:

- **Residual connection ($x+$):** Cho phép gradient flow trực tiếp, giúp huấn luyện mạng sâu
- **Layer normalization:** Chuẩn hóa theo chiều features, ổn định quá trình học

2.6 Biểu diễn Đầu vào của BERT - Kết hợp thông tin đa chiều

BERT xây dựng biểu diễn đầu vào bằng cách cộng ba loại embedding:

$$\text{Input}_i = \text{TokenEmb}_i + \text{SegmentEmb}_i + \text{PositionEmb}_i$$

2.6.1 Token Embeddings

- Sử dụng WordPiece tokenization với vocabulary $\approx$ 30,000 tokens
- Ví dụ: "playing" $\rightarrow$ ["play", "##ing"]
- Giúp xử lý từ hiếm và từ mới (out-of-vocabulary)

2.6.2 Segment Embeddings

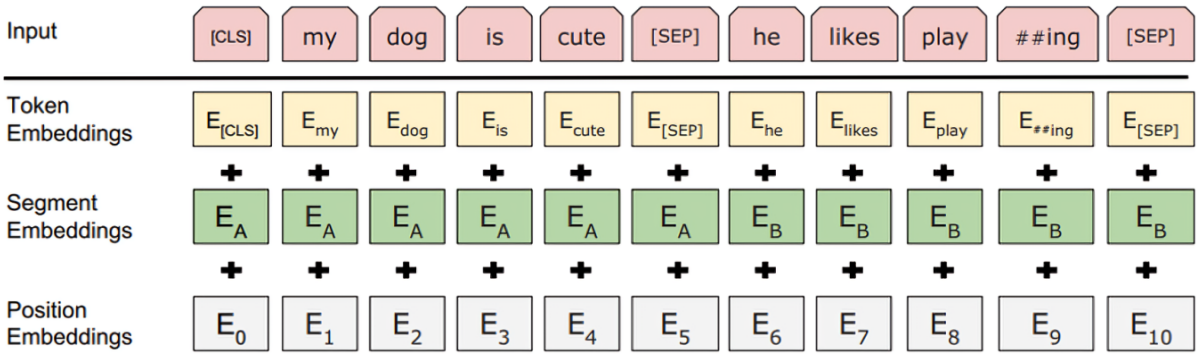
- Phân biệt câu A và câu B trong các tác vụ cặp câu
- Chỉ có 2 giá trị: E_A cho câu đầu, E_B cho câu thứ hai
- Ví dụ: "[CLS] How are you ? [SEP] I am fine [SEP]"
  - Tokens của "How are you ?" nhận E_A
  - Tokens của "I am fine" nhận E_B

2.6.3 Position Embeddings

- Transformer không có cấu trúc tuần tự nên cần thông tin vị trí
- BERT học position embeddings (khác với Transformer gốc dùng sinusoidal)
- Giới hạn 512 positions (có thể mở rộng khi cần)

2.6.4 Special Tokens

- **[CLS]:** Token đầu tiên, biểu diễn cuối cùng dùng cho classification
- **[SEP]:** Phân tách câu và đánh dấu kết thúc
- **[MASK]:** Dùng trong pre-training MLM
- **[PAD]:** Padding để các chuỗi có cùng độ dài



Hình 4: Biểu diễn đầu vào của BERT. Ví dụ cho cặp câu "He likes playing" và "My dog is cute". Ba loại embeddings được cộng theo từng vị trí để tạo input embedding cuối cùng. Token [CLS] ở đầu sẽ được dùng cho các tác vụ phân loại.

2.7 Các phiên bản BERT

Google phát hành hai phiên bản chính:

Bảng 1: So sánh các phiên bản BERT

Phiên bản	Layers (L)	Hidden (H)	Heads (A)	Parameters	Training Time
$\text{BERT}_{\text{BASE}}$	12	768	12	110M	4 days on 16 TPUs
$\text{BERT}_{\text{LARGE}}$	24	1024	16	340M	4 days on 64 TPUs

Sự khác biệt không chỉ về kích thước mà còn về khả năng học:

- $\text{BERT}_{\text{LARGE}}$ có capacity lớn hơn, phù hợp cho tác vụ phức tạp
- $\text{BERT}_{\text{BASE}}$ hiệu quả hơn về tài nguyên, phù hợp cho deployment
- Cả hai đều vượt trội so với các mô hình trước đó

3 Các Tác vụ Tiền huấn luyện của BERT

BERT được tiền huấn luyện đồng thời trên hai tác vụ tự giám sát (self-supervised): Masked Language Model (MLM) và Next Sentence Prediction (NSP). Các tác vụ này cho phép BERT học từ dữ liệu văn bản không cần gán nhãn thủ công.

3.1 Masked Language Model (MLM) - Đột phá cho học hai chiều

3.1.1 Vấn đề cốt lõi

Các mô hình ngôn ngữ truyền thống chỉ có thể học một chiều vì lý do đơn giản: nếu cho phép mô hình "nhìn" cả hai chiều khi dự đoán từ tiếp theo, nó sẽ "gian lận" bằng cách nhìn trực tiếp vào từ cần dự đoán. BERT giải quyết vấn đề này một cách khéo léo: che ngẫu nhiên một số từ và yêu cầu mô hình dự đoán chúng từ ngữ cảnh.

3.1.2 Chiến lược Masking 80-10-10

BERT che 15% tokens trong mỗi chuỗi, nhưng không phải lúc nào cũng thay bằng [MASK]:

- **80% trường hợp → [MASK]:**
  - Input: "The cat sat on the mat"
  - Masked: "The cat [MASK] on the mat"
  - Target: Dự đoán "sat"
- **10% trường hợp → Random token:**
  - Input: "The cat sat on the mat"

  - Masked: "The cat apple on the mat"
  - Target: Dự đoán "sat" (không phải "apple")

- **10% trường hợp → Giữ nguyên:**
  - Input: "The cat sat on the mat"
  - Masked: "The cat sat on the mat"
  - Target: Vẫn dự đoán "sat"

3.1.3 Tại sao cần chiến lược phức tạp này?

1. **Vấn đề [MASK] token:** Trong quá trình fine-tuning và sử dụng thực tế, không có token [MASK]. Nếu chỉ train với [MASK], mô hình sẽ học cách xử lý [MASK] tốt nhưng kém với các từ thực.
2. **Random replacement (10%):** Buộc mô hình phải học biểu diễn tốt cho MỌI token, không chỉ [MASK]. Mô hình phải "nghi ngờ" mọi từ có thể bị thay thế.
3. **Keep unchanged (10%):** Đảm bảo mô hình cũng học cách biểu diễn các từ không bị thay đổi. Tạo cân bằng trong học tập.

3.1.4 Ví dụ huấn luyện MLM

Câu gốc: "BERT revolutionized natural language processing in 2018"
Bước 1: Chọn ngẫu nhiên 15% tokens (giả sử chọn "revolutionized" và "2018")
Bước 2: Áp dụng chiến lược:

- "revolutionized" → [MASK] (80% chance)
- "2018" → "cat" (10% chance - random)

Input cho BERT: "BERT [MASK] natural language processing in cat"
Mục tiêu:

- Tại vị trí [MASK]: dự đoán "revolutionized"
- Tại vị trí "cat": dự đoán "2018"
- Các vị trí khác: không tính loss

3.1.5 Hàm loss cho MLM

Loss chỉ được tính cho các vị trí bị mask:

$$\mathcal{L}_{MLM} = -\sum_{i \in \text{masked}} \log P(w_i|\text{context})$$

Trong đó $P(w_i|\text{context})$ là xác suất dự đoán đúng từ gốc tại vị trí i.

3.2 Next Sentence Prediction (NSP) - Học quan hệ giữa câu

3.2.1 Động lực

Nhiều tác vụ NLP quan trọng yêu cầu hiểu mối quan hệ giữa các câu:

- **Question Answering:** Câu hỏi và đoạn văn có liên quan?
- **Natural Language Inference:** Câu A kéo theo câu B?
- **Paraphrasing:** Hai câu có cùng ý nghĩa?

MLM chỉ học quan hệ trong câu, NSP bổ sung khả năng học quan hệ giữa câu.

3.2.2 Cách thức hoạt động

1. **Tạo dữ liệu huấn luyện:**
   - 50% mẫu: Câu B thực sự theo sau câu A trong văn bản (label: IsNext)
   - 50% mẫu: Câu B là câu ngẫu nhiên từ corpus (label: NotNext)
2. **Input format:**

   ```
   [CLS] Sentence A [SEP] Sentence B [SEP]
   ```

3. **Prediction:** Sử dụng biểu diễn cuối cùng của [CLS] token qua một classifier:

$$P(\text{IsNext}) = \text{softmax}(W_{NSP} \cdot h_{[CLS]} + b_{NSP})$$

3.2.3 Ví dụ cụ thể

Positive example (IsNext):

- A: "The weather forecast predicts heavy rain tomorrow."
- B: "I should bring an umbrella to work."
- Logic: B là hệ quả logic của A

Negative example (NotNext):

- A: "The weather forecast predicts heavy rain tomorrow."
- B: "Python is a popular programming language."
- Logic: Không có quan hệ ngữ nghĩa

3.2.4 Vai trò của [CLS] token

[CLS] token đóng vai trò như "summary vector" của cả cặp câu:

- Vị trí đầu tiên, có thể attend đến mọi token khác
- Qua các layer, tích lũy thông tin từ cả hai câu
- Biểu diễn cuối cùng chứa thông tin về mối quan hệ câu
- Được sử dụng cho mọi tác vụ classification sau này

3.2.5 Combined Loss

BERT được train với combined loss:

$$\mathcal{L}_{total} = \mathcal{L}_{MLM} + \mathcal{L}_{NSP}$$

Cả hai loss được tối ưu đồng thời, giúp BERT học cả biểu diễn từ (MLM) và quan hệ câu (NSP).

3.2.6 Tranh cãi về NSP

Các nghiên cứu sau (RoBERTa, ALBERT) cho thấy:

- NSP có thể không cần thiết hoặc thậm chí có hại cho một số tác vụ
- Vấn đề: Negative samples (random sentences) quá dễ phân biệt
- Cải tiến: Sentence Order Prediction (SOP) trong ALBERT - dự đoán thứ tự câu

Tuy nhiên, trong thiết kế gốc, NSP đóng góp vào thành công của BERT trên nhiều benchmark.

4 Tinh chỉnh (Fine-tuning) BERT cho các Tác vụ Cụ thể

Sau khi được tiền huấn luyện trên một lượng lớn dữ liệu, BERT có thể được tinh chỉnh cho các tác vụ NLP cụ thể (downstream tasks). Quá trình tinh chỉnh tương đối đơn giản và không đòi hỏi thay đổi nhiều về kiến trúc.

4.1 Quy trình Tinh chỉnh Chung

Đối với mỗi tác vụ, các chuỗi đầu vào (một câu hoặc cặp câu) được đưa vào BERT theo cách tương tự như quá trình tiền huấn luyện. Sau đó, một hoặc nhiều lớp đầu ra đơn giản được thêm vào phía trên các biểu diễn cuối cùng của BERT. Tất cả các tham số của BERT và các lớp mới thêm vào này đều được huấn luyện (tinh chỉnh) trên dữ liệu gán nhãn của tác vụ đó.
So với tiền huấn luyện, quá trình tinh chỉnh thường nhanh hơn nhiều và đòi hỏi ít dữ liệu hơn. Các siêu tham số (hyperparameters) như tốc độ học (learning rate), kích thước batch (batch size), số epochs thường nhỏ hơn so với tiền huấn luyện.

4.2 Ví dụ về Tinh chỉnh cho các Tác vụ Phổ biến

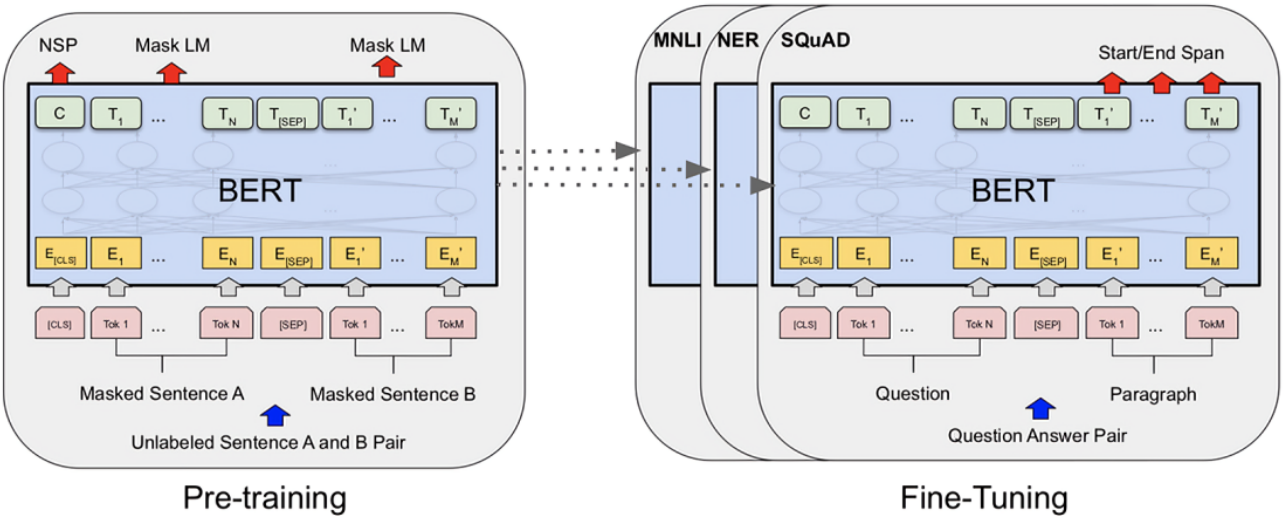
BERT có thể được áp dụng cho nhiều loại tác vụ NLP khác nhau:

- **Phân loại cặp câu (Sentence Pair Classification Tasks):** Ví dụ: Natural Language Inference (NLI), Semantic Textual Similarity (STS). Đầu vào là một cặp câu (A, B). Biểu diễn cuối cùng của token '[CLS]' được đưa qua một lớp softmax để phân loại.

- **Phân loại một câu (Single Sentence Classification Tasks):** Ví dụ: Phân loại cảm xúc (Sentiment Analysis), Phân loại chủ đề văn bản. Tương tự như phân loại cặp câu, nhưng đầu vào chỉ có một câu. Biểu diễn của '[CLS]' được sử dụng.

- **Trả lời câu hỏi (Question Answering Tasks):** Ví dụ: SQuAD. Đầu vào là một cặp (câu hỏi, đoạn văn chứa câu trả lời). BERT cần dự đoán vị trí bắt đầu (start span) và kết thúc (end span) của câu trả lời trong đoạn văn. Điều này được thực hiện bằng cách học hai vector mới S và E. Xác suất token i là điểm bắt đầu được tính bằng $softmax(S \cdot T_i)$, và là điểm kết thúc bằng $softmax(E \cdot T_i)$, trong đó T_i là biểu diễn cuối cùng của token thứ i trong đoạn văn.

- **Gán nhãn chuỗi (Single Sentence Tagging Tasks):** Ví dụ: Named Entity Recognition (NER). Đối với mỗi token trong câu đầu vào, BERT cần dự đoán một nhãn (ví dụ: PER, ORG, LOC, O). Biểu diễn cuối cùng của mỗi token T_i được đưa qua một lớp softmax để phân loại.



Hình 5: Cách tinh chỉnh BERT cho các tác vụ NLP khác nhau (Nguồn: [5]). (a) Phân loại cặp câu, (b) Phân loại một câu, (c) Trả lời câu hỏi, (d) Gán nhãn chuỗi.

5 Thực nghiệm với BERT: Phân loại Cảm xúc Văn bản

Để minh họa khả năng ứng dụng của BERT, chương này trình bày quy trình thực nghiệm áp dụng BERT cho bài toán phân loại cảm xúc văn bản (Sentiment Analysis). Đây là một tác vụ NLP phổ biến, nhằm xác định thái độ hoặc cảm xúc (ví dụ: tích cực, tiêu cực, trung tính) được thể hiện trong một đoạn văn bản.

5.1 Lựa chọn Bài toán và Bộ dữ liệu

5.1.1 Bài toán Phân loại Cảm xúc

Bài toán được chọn là phân loại cảm xúc nhị phân (binary sentiment classification), tức là phân loại một đoạn văn bản (ví dụ: một bình luận về sản phẩm, một đánh giá phim) thành hai lớp: "Tích cực" (Positive) hoặc "Tiêu cực" (Negative).

5.1.2 Bộ dữ liệu

Một bộ dữ liệu phổ biến thường được sử dụng cho bài toán này là **IMDB Movie Reviews dataset** [12].

- **Mô tả:** Bộ dữ liệu này chứa 50,000 bình luận phim được lấy từ trang web Internet Movie Database (IMDB).
- **Phân chia:** Được chia thành 25,000 bình luận cho tập huấn luyện (training set) và 25,000 bình luận cho tập kiểm thử (test set). Mỗi tập này lại được chia đều thành 12,500 bình luận tích cực và 12,500 bình luận tiêu cực.
- **Đặc điểm:** Các bình luận có độ dài đa dạng và chứa ngôn ngữ tự nhiên, đời thường.

5.2 Quy trình Thực nghiệm

5.2.1 Tiền xử lý Dữ liệu

1. **Làm sạch văn bản (Text Cleaning):** Loại bỏ các thẻ HTML (nếu có trong bình luận IMDB), chuyển văn bản về chữ thường (lowercase). Có thể cân nhắc loại bỏ các ký tự đặc biệt không cần thiết.

2. **Tokenization:** Sử dụng tokenizer của BERT (ví dụ: `BertTokenizer` từ thư viện Hugging Face Transformers). Đối với IMDB, nếu dùng `bert-base-uncased`, tokenizer sẽ sử dụng WordPiece.

   ```
   Ví dụ: "This movie is great!"
   -> ["[CLS]", "this", "movie", "is", "great", "!", "[SEP]"]
   ```

3. **Chuyển đổi Tokens thành IDs:** Ánh xạ mỗi token sang ID tương ứng trong bộ từ vựng của BERT.

4. **Padding và Truncation:** Đảm bảo tất cả các chuỗi đầu vào có cùng một độ dài cố định (ví dụ: `max_length = 256 tokens`). Các chuỗi ngắn hơn `max_length` sẽ được đệm (padding) bằng token `[PAD]`. Các chuỗi dài hơn sẽ bị cắt bớt (truncation).

5. **Tạo Attention Masks:** Tạo một attention mask để chỉ cho mô hình biết token nào là token thực sự và token nào là token đệm (để không tính toán attention trên các token đệm).

5.2.2 Mô hình

1. **Sử dụng BERT đã tiền huấn luyện:** Tải một mô hình BERT đã tiền huấn luyện, ví dụ `bert-base-uncased` từ Hugging Face Transformers.

2. **Thêm lớp Phân loại (Classification Head):** Phía trên output của token `[CLS]` từ BERT (là một vector có kích thước $H = 768$ đối với `bert-base-uncased`), thêm:

   - Một lớp `Dropout` để chống quá khớp (overfitting), ví dụ dropout rate là 0.1.
   - Một lớp `Linear` (fully connected layer) với H đầu vào và K đầu ra, trong đó K là số lớp cảm xúc (ở đây $K = 2$: Positive, Negative).
   - Không cần hàm `softmax` ở lớp cuối này nếu sử dụng `CrossEntropyLoss`, vì nó đã bao gồm `LogSoftmax` và `NLLLoss`.

   Mô hình hoàn chỉnh sẽ là `BertForSequenceClassification` trong thư viện Hugging Face.

5.2.3 Huấn luyện (Fine-tuning)

1. **Hàm mất mát (Loss Function):** Sử dụng `CrossEntropyLoss`, phù hợp cho bài toán phân loại đa lớp (hoặc nhị phân).

2. **Trình tối ưu (Optimizer):** Sử dụng `AdamW` (Adam with weight decay), một biến thể của Adam thường được dùng cho Transformer. Tốc độ học (learning rate) ban đầu có thể chọn nhỏ, ví dụ $2e-5$ hoặc $3e-5$.

3. **Lịch trình Tốc độ học (Learning Rate Scheduler):** Thường sử dụng một lịch trình tuyến tính với một giai đoạn khởi động (warmup), ví dụ `get_linear_schedule_with_warmup`.

4. **Chia dữ liệu:** Chia tập huấn luyện IMDB thành tập huấn luyện nhỏ hơn và một tập validation (ví dụ: 80% train, 20% validation) để theo dõi hiệu suất trong quá trình huấn luyện và tránh quá khớp. Tập kiểm thử IMDB gốc được giữ riêng để đánh giá cuối cùng.

5. **Thông số huấn luyện:**
   - Kích thước batch (Batch size): Ví dụ 16 hoặc 32 (tùy thuộc vào bộ nhớ GPU).
   - Số epochs: Ví dụ 2-4 epochs. BERT thường hội tụ nhanh trong quá trình tinh chỉnh.

6. **Quá trình huấn luyện:** Lặp qua dữ liệu huấn luyện theo từng batch, tính toán loss, thực hiện backpropagation và cập nhật trọng số mô hình. Sau mỗi epoch (hoặc sau một số bước nhất định), đánh giá mô hình trên tập validation.

5.3 Đánh giá Kết quả

Sau khi huấn luyện xong, mô hình tốt nhất (thường là mô hình có hiệu suất cao nhất trên tập validation) được đánh giá trên tập kiểm thử IMDB. Các độ đo thường được sử dụng bao gồm:

- **Accuracy (Độ chính xác):** Tỷ lệ số mẫu được phân loại đúng trên tổng số mẫu.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- **Precision (Độ chuẩn):** Trong số các mẫu được dự đoán là Positive, có bao nhiêu mẫu thực

sự là Positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall (Độ phủ, hay Độ nhạy):** Trong số các mẫu thực sự là Positive, có bao nhiêu mẫu được dự đoán là Positive.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **F1-score:** Trung bình điều hòa của Precision và Recall.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

(TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative)

- **Confusion Matrix (Ma trận nhầm lẫn):** Một bảng cho thấy số lượng dự đoán đúng và sai cho từng lớp, giúp hiểu rõ hơn về các loại lỗi mà mô hình mắc phải.

Bảng 2: Kết quả ví dụ của BERT trên tập kiểm thử IMDB (giả định).

Độ đo	**Giá trị**
Accuracy	0.935
Precision (Positive)	0.93
Recall (Positive)	0.94
F1-score (Positive)	0.935
Precision (Negative)	0.94
Recall (Negative)	0.93
F1-score (Negative)	0.935

5.4 Thảo luận và Hướng phát triển

5.4.1 Phân tích Kết quả

(Phần này nhóm cần tự điền dựa trên kết quả thực nghiệm nếu có)

- Mô hình BERT có đạt được hiệu suất tốt trên bộ dữ liệu IMDB không? So sánh với các baseline (ví dụ: mô hình Naive Bayes, LSTM) hoặc các kết quả đã công bố khác.

- Phân tích các trường hợp lỗi điển hình từ confusion matrix. Mô hình thường nhầm lẫn ở những loại bình luận nào? (Ví dụ: bình luận có chứa sự mỉa mai, câu phức tạp, từ ngữ đa

nghĩa).

5.4.2 Những Thách thức Gặp phải

(Phần này nhóm cần tự điền dựa trên kinh nghiệm thực tế)

- **Giới hạn tài nguyên tính toán:** Huấn luyện BERT, ngay cả khi chỉ fine-tuning, cũng đòi hỏi GPU có bộ nhớ tương đối lớn. Thời gian huấn luyện có thể kéo dài nếu không có GPU mạnh.
- **Lựa chọn siêu tham số:** Việc tìm ra bộ siêu tham số tối ưu (learning rate, batch size, số epochs) có thể cần nhiều thử nghiệm.
- **Xử lý dữ liệu tiếng Việt (nếu có):** Nếu làm với dữ liệu tiếng Việt, các thách thức có thể bao gồm việc tìm tokenizer phù hợp, xử lý các vấn đề đặc thù của tiếng Việt (dấu câu, thanh điệu, từ ghép).

5.4.3 Hướng Cải thiện Tiềm năng

- **Thử nghiệm với các mô hình BERT lớn hơn hoặc biến thể khác:** Ví dụ, 'bert-large-uncased' hoặc các mô hình mới hơn như RoBERTa, ALBERT, ELECTRA (nếu phù hợp với phạm vi đồ án).
- **Tinh chỉnh siêu tham số (Hyperparameter Tuning):** Sử dụng các kỹ thuật như Grid Search, Random Search, hoặc các công cụ tự động như Optuna, Ray Tune.
- **Kỹ thuật Augmentation Dữ liệu (Data Augmentation):** Nếu dữ liệu huấn luyện hạn chế, có thể áp dụng các kỹ thuật như back-translation, synonym replacement để tăng cường dữ liệu.
- **Ensemble Models:** Kết hợp dự đoán từ nhiều mô hình BERT khác nhau hoặc kết hợp BERT với các loại mô hình khác.
- **Phân tích lỗi chi tiết hơn:** Sử dụng các công cụ diễn giải mô hình (model interpretability tools) để hiểu rõ hơn tại sao BERT đưa ra dự đoán cụ thể.

Chương thực nghiệm này cho thấy BERT, với khả năng học các biểu diễn ngữ nghĩa mạnh mẽ, có thể được tinh chỉnh hiệu quả cho các tác vụ NLP cụ thể như phân loại cảm xúc, đạt được kết quả ấn tượng.

6 Kết quả và So sánh

Bài báo gốc của BERT [5] đã trình bày kết quả vượt trội trên nhiều bộ dữ liệu benchmark tiêu chuẩn.

6.1 Kết quả trên GLUE Benchmark

GLUE (General Language Understanding Evaluation) [24] là một tập hợp các tác vụ đa dạng nhằm đánh giá khả năng hiểu ngôn ngữ tự nhiên của mô hình. BERT đã đạt được những cải thiện đáng kể so với các mô hình trước đó trên tất cả các tác vụ của GLUE.

Bảng 3: Kết quả của BERT so với các mô hình khác trên GLUE test set. Các kết quả được lấy từ [5]. $\text{BERT}_{\text{LARGE}}$ đạt điểm trung bình GLUE là 80.5.

Model	**MNLI-(m/mm)** (Acc)	**QQP** (Acc/F1)	**QNLI** (Acc)	**SST-2** (Acc)	**CoLA** (Matthews)	**STS-B** (Pearson/Spearman)	**MRPC** (Acc/F1)	**RTE** (Acc)	**Avg.**
OpenAI GPT	82.1/81.4	70.3/88.3	88.1	91.3	35.0	80.0/78.8	82.3/87.0	56.0	72.8
ELMo+Attn	76.4/76.1	64.8/85.0	81.0	90.4	36.0	73.7/72.4	84.9/89.1	56.8	70.0
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2/89.2	90.5	93.5	52.1	85.8/84.9	88.9/91.8	66.4	78.3
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1/89.3**	**92.7**	**94.9**	**60.5**	**86.5/85.9**	**89.3/92.3**	**70.1**	**80.5**

6.2 Kết quả trên SQuAD (Stanford Question Answering Dataset)

SQuAD là một bộ dữ liệu phổ biến cho tác vụ trả lời câu hỏi dựa trên việc đọc hiểu đoạn văn.

- **SQuAD v1.1:** $\text{BERT}_{\text{LARGE}}$ đạt F1 score là 93.2 trên test set, vượt qua con người (F1=91.2).
- **SQuAD v2.0:** Tác vụ này khó hơn vì một số câu hỏi không có câu trả lời trong đoạn văn. $\text{BERT}_{\text{LARGE}}$ đạt F1 score là 83.1.

Bảng 4: Kết quả của BERT trên SQuAD v1.1 và v2.0 test sets (Nguồn: [5]).

Model	**SQuAD v1.1**		**SQuAD v2.0**	
	EM	F1	EM	F1
Human Performance	82.3	91.2	86.8	89.5
OpenAI GPT (fine-tuned)	-	-	71.0	75.8
ELMo + BiDAF++ (ensemble)	81.0	88.5	-	-
$\text{BERT}_{\text{BASE}}$	80.8	88.5	73.7	76.3
$\text{BERT}_{\text{LARGE}}$	**84.1**	**90.9**	**79.0**	**81.8**
$\text{BERT}_{\text{LARGE}}$ (w/ TriviaQA)	**87.4**	**93.2**	**83.1**	**86.4**

6.3 Phân tích Ablation Study

Bài báo BERT cũng thực hiện các nghiên cứu loại bỏ (ablation studies) để đánh giá tầm quan trọng của các thành phần khác nhau trong mô hình:

- **Tầm quan trọng của MLM và NSP:** Kết quả cho thấy cả hai tác vụ tiền huấn luyện đều đóng góp vào hiệu suất của BERT, đặc biệt NSP quan trọng cho các tác vụ hiểu mối quan hệ giữa các câu. Tuy nhiên, các nghiên cứu sau này đã có những phát hiện khác nhau về vai trò của NSP.
- **Tầm quan trọng của tính hai chiều (Bidirectionality):** So sánh với các mô hình chỉ sử dụng ngữ cảnh từ trái sang phải (LTR) hoặc kết hợp nông LTR và RTL, BERT (sử dụng MLM hai chiều sâu sắc) cho thấy hiệu suất vượt trội, khẳng định lợi ích của việc học biểu diễn hai chiều thực sự.
- **Kích thước mô hình:** $\text{BERT}_{\text{LARGE}}$ luôn cho kết quả tốt hơn $\text{BERT}_{\text{BASE}}$, cho thấy rằng việc tăng kích thước mô hình (số lớp, kích thước ẩn, số đầu chú ý) có thể cải thiện đáng kể khả năng học biểu diễn ngôn ngữ.

7 Hạn chế và Hướng Phát triển

7.1 Hạn chế của BERT

Mặc dù BERT đã đạt được những thành công to lớn, mô hình này vẫn có một số hạn chế:

- **Chi phí tính toán cao:** Việc tiền huấn luyện BERT đòi hỏi một lượng lớn dữ liệu và tài nguyên tính toán (GPU/TPU) đáng kể. Ngay cả việc tinh chỉnh các mô hình lớn cũng có thể tốn kém.
- **Sự khác biệt giữa tiền huấn luyện và tinh chỉnh (Pre-train/Fine-tune Discrepancy):** Token '[MASK]' được sử dụng trong tác vụ MLM ở giai đoạn tiền huấn luyện không xuất hiện trong dữ liệu ở giai đoạn tinh chỉnh. Mặc dù các chiến lược che từ (80-10-10) đã cố gắng giảm thiểu vấn đề này, sự khác biệt vẫn tồn tại.
- **Độ dài chuỗi đầu vào cố định:** BERT thường xử lý các chuỗi có độ dài cố định (ví dụ: 512 token). Điều này có thể là một hạn chế khi xử lý các văn bản rất dài. Các mô hình sau này như Longformer [2] đã cố gắng giải quyết vấn đề này.
- **Tác vụ NSP chưa tối ưu:** Như đã đề cập, hiệu quả của tác vụ NSP đã bị đặt câu hỏi trong các nghiên cứu sau này. Một số mô hình như RoBERTa [11] đã loại bỏ NSP và vẫn đạt được hiệu suất tốt hoặc thậm chí tốt hơn.
- **Không phù hợp cho các tác vụ sinh văn bản (Generative Tasks):** Do bản chất là một mô hình encoder, BERT không được thiết kế trực tiếp cho các tác vụ sinh văn bản tự hồi quy như dịch máy hoặc tóm tắt văn bản (mặc dù có thể được điều chỉnh, ví dụ BART [10], T5 [20] sử dụng kiến trúc encoder-decoder).
- **Khả năng diễn giải hạn chế (Limited Interpretability):** Giống như nhiều mô hình học sâu khác, việc hiểu rõ tại sao BERT đưa ra một dự đoán cụ thể vẫn là một thách thức.

7.2 Các Hướng Phát triển Kế thừa BERT

BERT đã đặt nền móng cho một loạt các mô hình ngôn ngữ lớn và các hướng nghiên cứu mới trong NLP:

- **Cải tiến quy trình tiền huấn luyện:**
  - **RoBERTa (Robustly Optimized BERT Pretraining Approach) [11]:** Tối ưu hóa các siêu tham số tiền huấn luyện của BERT, loại bỏ NSP, sử dụng dynamic masking, và huấn luyện trên nhiều dữ liệu hơn với batch size lớn hơn.

  - **ALBERT (A Lite BERT for Self-supervised Learning of Language Representations) [8]:** Giảm số lượng tham số của BERT bằng cách chia sẻ tham số giữa các lớp và tách rời ma trận embedding từ vựng, giúp mô hình nhẹ hơn và huấn luyện nhanh hơn.
  - **ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) [3]:** Sử dụng một tác vụ tiền huấn luyện mới gọi là replaced token detection, hiệu quả hơn MLM về mặt tính toán. Mô hình học cách phân biệt token "thật" và token "giả" được tạo ra bởi một generator nhỏ.
  - **SpanBERT [7]:** Che các đoạn (span) từ liên tiếp thay vì các từ đơn lẻ và huấn luyện mô hình dự đoán toàn bộ đoạn bị che từ các token ở biên của đoạn đó.

- **Mở rộng cho các ngôn ngữ khác và đa ngôn ngữ:**
  - **mBERT (Multilingual BERT):** Phiên bản BERT được tiền huấn luyện trên dữ liệu từ hơn 100 ngôn ngữ.
  - **XLM-R (Cross-lingual Language Model - RoBERTa) [4]:** Một mô hình đa ngôn ngữ dựa trên RoBERTa, đạt hiệu suất mạnh mẽ trên các tác vụ cross-lingual.
  - Các mô hình đặc thù cho từng ngôn ngữ, ví dụ PhoBERT cho tiếng Việt [15], CamemBERT cho tiếng Pháp [13].

- **Kết hợp kiến thức bên ngoài (Knowledge-enhanced LMs):** Các mô hình như ERNIE (Enhanced Representation through Knowledge Integration) [22], KnowBERT [18] cố gắng tích hợp kiến thức từ các knowledge graph vào quá trình tiền huấn luyện.

- **Hiệu quả tính toán và mô hình nhỏ hơn:** DistilBERT [21], TinyBERT [6] sử dụng các kỹ thuật chưng cất kiến thức (knowledge distillation) để tạo ra các phiên bản BERT nhỏ hơn, nhanh hơn mà vẫn giữ được phần lớn hiệu suất.

- **Ứng dụng trong các lĩnh vực chuyên biệt:** BioBERT [9] cho lĩnh vực y sinh, SciBERT [1] cho văn bản khoa học.

- **Các mô hình Encoder-Decoder dựa trên Transformer:** BART [10], T5 [20] mở rộng ý tưởng tiền huấn luyện cho các tác vụ sinh văn bản.

Sự phát triển không ngừng của các mô hình dựa trên Transformer, khởi nguồn từ những ý tưởng đột phá của BERT, tiếp tục định hình tương lai của Xử lý Ngôn ngữ Tự nhiên.

8 Kết luận

BERT (Bidirectional Encoder Representations from Transformers) đã tạo ra một cuộc cách mạng trong lĩnh vực Xử lý Ngôn ngữ Tự nhiên. Bằng cách giới thiệu một phương pháp tiền huấn luyện các biểu diễn ngôn ngữ sâu sắc hai chiều dựa trên kiến trúc Transformer encoder và hai tác vụ tiền huấn luyện mới là Masked Language Model (MLM) và Next Sentence Prediction (NSP), BERT đã thành công trong việc nắm bắt các đặc trưng ngữ nghĩa và cú pháp phong phú từ một lượng lớn dữ liệu không gán nhãn.

Kiến trúc hai chiều sâu sắc cho phép BERT hiểu ngữ cảnh của một từ dựa trên tất cả các từ xung quanh nó, cả bên trái và bên phải, trong tất cả các lớp của mô hình. Điều này mang lại lợi thế đáng kể so với các mô hình một chiều hoặc hai chiều nông trước đó. Quy trình tinh chỉnh đơn giản của BERT cho phép nó dễ dàng thích ứng và đạt được hiệu suất vượt trội trên một loạt các tác vụ NLP, từ phân loại văn bản, suy luận ngôn ngữ tự nhiên đến trả lời câu hỏi, mà không cần những thay đổi kiến trúc phức tạp đặc thù cho từng tác vụ.

Thành công vang dội của BERT, được minh chứng qua việc thiết lập hàng loạt kỷ lục mới trên các bộ dữ liệu benchmark tiêu chuẩn như GLUE và SQuAD, đã khẳng định sức mạnh của phương pháp tiền huấn luyện hai chiều sâu. Quan trọng hơn, BERT không chỉ là một mô hình cụ thể mà còn là một "công thức" và một triết lý thiết kế, định hình lại cách tiếp cận các bài toán NLP. Nó đã mở đường cho sự phát triển của một thế hệ các mô hình ngôn ngữ lớn (LLMs) dựa trên Transformer, thúc đẩy mạnh mẽ các ứng dụng AI dựa trên ngôn ngữ và tiếp tục là nguồn cảm hứng cho nhiều nghiên cứu tiên tiến trong lĩnh vực này.

Trong khuôn khổ đồ án này, nhóm đã trình bày chi tiết về bối cảnh ra đời, kiến trúc cốt lõi, các cơ chế hoạt động chính của BERT, bao gồm cơ chế tự chú ý, chú ý đa đầu, các tác vụ tiền huấn luyện MLM và NSP, cũng như quy trình tinh chỉnh cho các tác vụ cụ thể. Một ví dụ thực nghiệm về việc áp dụng BERT cho bài toán phân loại cảm xúc cũng đã được đề xuất và phân tích. Mặc dù có những hạn chế về chi phí tính toán và một số khía cạnh của tác vụ tiền huấn luyện, những đóng góp nền tảng của BERT cho sự phát triển của NLP là không thể phủ nhận và tiếp tục có ảnh hưởng sâu rộng.

9 Tài liệu tham khảo

Tài liệu

[1] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[2] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[3] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[4] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. Version 2, 24 May 2019.

[6] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[7] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[8] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[9] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[10] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[11] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[12] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, 2011.

[13] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. CamemBERT: a tasty french language model. *arXiv preprint arXiv:1911.03894*, 2019.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 3111–3119, 2013.

[15] Dat Quoc Nguyen and Nguyen Anh Tuan. PhoBERT: Pre-trained language models for vietnamese. *arXiv preprint arXiv:2003.00744*, 2020.

[16] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[17] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, 2018.

[18] Matthew E Peters, Mark Neumann, Robert L Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. Knowledge enhanced contextual word representations. *arXiv preprint arXiv:1909.04164*, 2019.

[19] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf, 2018. Accessed: May 25, 2025.

[20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[21] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[22] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. ERNIE: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*, 2019.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 5998–6008, 2017.

[24] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

A Phụ lục

- Template này **không phải** là template chính thức của Khoa Công nghệ thông tin - Trường Đại học Khoa học Tự nhiên.
- Các hình ảnh, bảng biểu, thuật toán trong template chỉ mang tính chất ví dụ.
- Nhóm tác giả phân phối **miễn phí** template này trên GitHub và trên Overleaf với Giấy phép GNU General Public License v3.0. Nhóm tác giả không chịu trách nhiệm với các bản phân phối không nằm trong hai kênh phân phối chính thức nêu trên.