ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

———□□□□□□□□———

BÁO CÁO MÔN HỌC
CÁC CÔNG NGHỆ LẬP TRÌNH HIỆN ĐẠI

ĐỀ TÀI: TÌM HIỂU VỀ KIẾN TRÚC TRANSFORMER VÀ ỨNG DỤNG NLP

Sinh viên thực hiện: Nhóm 21

STT	Họ và tên	MSSV
1	Vũ Quốc Vương	3120410629
2	Nguyễn Phong Vũ	3121410579
3	Trần Ngọc Vũ	3121410581

Giảng viên hướng dẫn: ThS. Phạm Thi Vương

Thành phố Hồ Chí Minh, tháng 10 năm 2025

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TP. Hồ Chí Minh, tháng 10 năm 2025

Giảng viên hướng dẫn

Phạm Thi Vương

LỜI CẢM ƠN

Trước hết, nhóm chúng em xin gửi lời cảm ơn chân thành đến thầy **Phạm Thi Vương** – giảng viên môn *Các Công Nghệ Lập Trình Hiện Đại*. Thầy đã tận tình giảng dạy, định hướng và hỗ trợ chúng em trong suốt quá trình học tập cũng như khi thực hiện báo cáo này.

Trong quá trình tìm hiểu và hoàn thiện bài báo cáo, do kiến thức và kinh nghiệm thực tiễn của nhóm còn hạn chế nên khó tránh khỏi những thiếu sót. Nhóm chúng em rất mong nhận được những góp ý quý báu từ thầy để có thể rút kinh nghiệm và nâng cao hơn nữa trong học tập cũng như nghiên cứu sau này.

Một lần nữa, chúng em xin chân thành cảm ơn thầy!

CHƯƠNG I: GIỚI THIỆU

1.1. Lịch sử ra đời và cơ chế hoạt động

1.1.1. Xử lý ngôn ngữ tự nhiên

Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) là một lĩnh vực trong khoa học máy tính và trí tuệ nhân tạo tập trung vào việc tương tác giữa máy tính và ngôn ngữ con người. NLP có mục tiêu chính là cho phép máy tính hiểu, phân tích và tạo ra ngôn ngữ tự nhiên theo cách tương tự như con người.

NLP đã trải qua nhiều giai đoạn đổi mới để đạt được những thành tựu nhất định.

Thời kỳ đầu (1950-1970): Giai đoạn này được đánh dấu bởi sự ra đời của các hệ thống dựa trên luật (rule-based systems). Động lực chính thúc đẩy việc phát triển NLP ban đầu là dịch máy (Machine Translation). Các hệ thống này hoạt động như những cuốn từ điển dịch phức tạp, trong đó các nhà ngôn ngữ học tỉ mỉ xây dựng một bộ quy tắc khổng lồ để nắm bắt cấu trúc ngữ pháp và từ vựng của các ngôn ngữ cụ thể.

Tuy nhiên, phương pháp này có nhiều hạn chế:

- Thiếu linh hoạt: Ngôn ngữ chứa đầy những sắc thái và ngoại lệ. Các hệ thống dựa trên luật gặp khó khăn trong việc xử lý thành ngữ, tiếng lóng và biến thể trong cấu trúc câu

- Vấn đề về khả năng mở rộng: Việc tạo và duy trì một cơ sở quy tắt khổng lồ cho mỗi cặp ngôn ngữ là một nhiệm vụ tốn thời gian và công sức

- Phạm vi hạn chế: Các hệ thống này chủ yếu tập trung vào cú pháp và từ vựng, thường không nắm bắt được ý nghĩa sâu hơn và ngữ cảnh của văn bản

Cuộc cách mạng thống kê (1980s-1990s): Giai đoạn này chứng kiến sự chuyển dịch từ các phương pháp dựa trên luật sang các mô hình thống kê. Các kỹ thuật như N-gram models và Hidden Markov Models (HMMs) được phát triển, cho phép các mô hình học từ dữ liệu lớn để hiểu các mẫu và xác suất trong ngôn ngữ.

Kỷ nguyên Deep Learning (2000s-Hiện tại): Đây là giai đoạn quan trọng nhất trong lịch sử NLP. Sự xuất hiện của word embeddings như Word2Vec (Google) và GloVe (Stanford) đã tạo ra bước đột phá. Những

kỹ thuật này cho phép từ được biểu diễn dưới dạng vector dày đặc trong không gian vector liên tục, nắm bắt các mối quan hệ ngữ nghĩa giữa chúng.

Giai đoạn này cũng chứng kiến sự phát triển của Convolutional Neural Networks (CNNs) và Recurrent Neural Networks (RNNs), đặc biệt là Long Short-Term Memory (LSTM) và Gated Recurrent Units (GRU), được sử dụng rộng rãi cho các tác vụ mô hình hóa chuỗi.

1.1.2. Transformer

Kiến trúc Transformer được giới thiệu vào tháng 6 năm 2017 thông qua bài báo mang tính bước ngoặt "Attention Is All You Need" của tám nhà khoa học làm việc tại Google. Đây được coi là một bài báo nền tảng trong trí tuệ nhân tạo hiện đại và là động lực chính cho sự bùng nổ AI.

Bối cảnh lịch sử trước Transformer:

Trong nhiều năm, việc mô hình hóa và tạo chuỗi được thực hiện bằng cách sử dụng các mạng nơ-ron hồi quy (RNNs) đơn giản. Một ví dụ được trích dẫn sớm là mạng Elman (1990). Về mặt lý thuyết, thông tin từ một token có thể lan truyền một cách tùy ý xa trong chuỗi, nhưng trong thực tế, vấn đề gradient biến mất khiến trạng thái của mô hình ở cuối một câu dài không có thông tin chính xác, có thể trích xuất về các token trước đó.

Một bước đột phá quan trọng là LSTM (1995), một RNN sử dụng các cải tiến khác nhau để khắc phục vấn đề gradient biến mất, cho phép học hiệu quả việc mô hình hóa chuỗi dài. LSTM đã trở thành kiến trúc tiêu chuẩn cho việc mô hình hóa chuỗi dài cho đến khi Transformer được công bố năm 2017.

Tên gọi và nguồn gốc:

Tên "Transformer" được chọn vì Jakob Uszkoreit, một trong những tác giả của bài báo, thích âm thanh của từ này. Tiêu đề bài báo "Attention Is All You Need" là một tham chiếu đến bài hát "All You Need Is Love" của The Beatles. Một tài liệu thiết kế sớm có tiêu đề "Transformers: Iterative Self-Attention and Processing for Various Tasks" và bao gồm một minh họa về sáu nhân vật từ nhượng quyền Transformers.

Đóng góp cách mạng:

Transformer đã giới thiệu một kiến trúc mạng mới đơn giản, dựa hoàn toàn trên cơ chế attention, loại bỏ hoàn toàn tính tái phát (recurrence) và tích chập (convolution). Đây là mô hình chuyển đổi chuỗi (sequence transduction) đầu tiên dựa hoàn toàn trên attention.Điểm đột phá chính là self-attention, cho

phép mô hình cân nhắc tầm quan trọng của các từ so với nhau bất kể vị trí của chúng trong văn bản. Điều này cho phép xử lý song song thực sự, làm cho việc huấn luyện nhanh hơn nhiều và hiệu quả hơn.

Tác động và di sản:

Cho đến năm 2025, bài báo đã được trích dẫn hơn 173.000 lần, đặt nó trong số mười bài báo được trích dẫn nhiều nhất của thế kỷ 21. Transformer ban đầu nhanh chóng dẫn đến các mô hình mạnh mẽ như BERT (2018) và GPT (2018), chứng minh khả năng chưa từng có trong việc hiểu và tạo ra ngôn ngữ con người.

Kiến trúc Transformer không chỉ cách mạng hóa NLP mà còn mở rộng ra ngoài văn bản để xử lý hình ảnh, âm thanh và thậm chí dữ liệu sinh học, củng cố vị trí của nó như một trong những bước đột phá AI quan trọng nhất của thập kỷ.

a. **Cơ chế attention**

b. **Self-attention**

1.2. Thực hành

Ứng dụng nhanh với Transformer: Sentiment Analysis (Hugging Face Transformers)

```
!pip -q install transformers torch --upgrade

# ================================
# Demo ứng dụng: Sentiment Analysis (Hugging Face Transformers)
# ================================

from transformers import pipeline

# Pipeline cảm xúc với DistilBERT đã fine-tuned trên SST-2 (Positive/Negative)
clf = pipeline(
    "sentiment-analysis",
    model="distilbert-base-uncased-finetuned-sst-2-english"
)

# Một vài câu
examples = [
```

```
    "I am happy today!",
    "This movie is terrible...",
    "The product is okay, not great but not bad either.",
    "I absolutely love this! Highly recommended."
]

results = clf(examples)
for s, r in zip(examples, results):
    print(f"{s:60s} -> {r}")
```

Kết quả:

```
I am happy today!                                            -> {'label': 'POSITIVE', 'score': 0.999874472618103}
This movie is terrible...                                    -> {'label': 'NEGATIVE', 'score': 0.9997668862342834}
The product is okay, not great but not bad either.           -> {'label': 'POSITIVE', 'score': 0.9964504241943359}
I absolutely love this! Highly recommended.                  -> {'label': 'POSITIVE', 'score': 0.9998767375946045}
```

Mô hình GPT-2 để tạo sinh văn bản

```
# !pip install transformers --quiet

# from transformers import pipeline
# ==============================
# 2) Sinh văn bản (GPT-2)
# ==============================
generator = pipeline("text-generation", model="gpt2")
# Sử dụng pipeline để tạo văn bản.
# - "Good morning, ":  Prompt - Đoạn văn bản ban đầu mà mô hình sẽ tiếp tục.
# - max_length=30:  Độ dài tối đa của văn bản được tạo ra (tính cả prompt).
# - num_return_sequences=1:  Số lượng văn bản khác nhau mà mô hình sẽ tạo ra.
# - truncation=True:  Cắt bớt prompt nếu nó vượt quá độ dài tối đa mà mô hình
có thể xử lý. Điều này ngăn ngừa lỗi khi sử dụng các prompt dài.
generated_text = generator(
    "Good morning, ",
    max_length=30,
    num_return_sequences=1,
    truncation=True
)

print("\nKết quả sinh văn bản:")
for g in generated_text:
    print(g["generated_text"])
```

Kết quả:

Good morning, "

"Ah, then we'll let you have tea. I'm happy to be home."

"I am, then. I'll be having tea with your master and his wife tonight."

"I'll be fine."

"Ah! I know, my lady. I'm glad to see you."

"I'm glad to see you too, my lady. I like you too, and I'm glad to see you too as well. Ah, my lady, I'm glad to see you too as well. Ah, my lady, I wonder what the hell you think of me, my lady. Ah, my lady, I wonder what the hell you think of me, my lady. Ah, my lady, I wonder what the hell you think of me, my lady. Ah, my lady, I wonder what the hell you think of me, my lady. Ah, my lady, I wonder what the hell you think of me, my lady. Ah, my lady, I wonder what the hell you think of me, my lady. Ah, my lady, I wonder what the hell you think of me, my lady. Ah, my lady, I wonder what the hell you think of me, my lady

1.3. Kết luận

CHƯƠNG II: ATTENTION

2.1. Giới thiệu

Attention là cơ chế giúp mô hình học sâu phân bổ "nguồn chú ý" đến những phần thông tin quan trọng nhất của đầu vào, nhờ đó mô hình có thể tổng hợp ngữ cảnh một cách linh hoạt thay vì bị khóa vào một vector ngữ cảnh cố định như encoder–decoder truyền thống. Lịch sử phát triển của attention khởi đầu từ ý tưởng "alignment" trong dịch máy (đo ánh xạ giữa các từ nguồn–đích), sau đó được hình thức hóa trong công trình Bahdanau (2014) để giải bài toán "bottleneck" của kiến trúc RNN encoder–decoder, và được tổng quát hóa thành cơ chế self-attention trong Transformer (2017), mở đường cho LLM hiện đại. Về mặt nhận thức, attention mô phỏng khả năng chọn lọc thông tin của con người: cùng lúc nhìn toàn cục nhưng chỉ tập trung mạnh vào phần có liên quan nhất cho tác vụ hiện tại.

Cơ chế Attention hoạt động thông qua ba bước chính:

- ➢ Tính điểm số (Scoring): Tính toán độ liên quan giữa các phần tử trong chuỗi đầu vào
- ➢ Chuẩn hóa trọng số (Weighting): Sử dụng hàm softmax để chuyển đổi điểm số thành trọng số xác suất
- ➢ Tổng hợp (Aggregation): Tính tổng có trọng số của các vector giá trị

2.2 Scaled Dot-Product Attention

Scaled Dot-Product Attention là một thành phần cốt lõi trong các mô hình Transformer, giúp mô hình tập trung vào những phần quan trọng nhất của dữ liệu đầu vào bằng cách tính toán mức độ liên quan giữa các phần tử. Đây là "trái tim" của cơ chế self-attention, được dùng trong việc xử lý ngôn ngữ tự nhiên (NLP), dịch máy, phân tích văn bản,…

Nguyên lý hoạt động

- ➢ Cơ chế attention hoạt động thông qua ba bước chính:
- ➢ Tính điểm số (Scoring): Tính toán độ liên quan giữa các phần tử trong chuỗi đầu vào
- ➢ Chuẩn hóa trọng số (Weighting): Sử dụng hàm softmax để chuyển đổi điểm số thành trọng số xác suất
- ➢ Tổng hợp (Aggregation): Tính tổng có trọng số của các vector giá trị

Scaled Dot-Product Attention nhận ba thành phần đầu vào:

- ➢ Truy vấn (Query - Q): Đại diện cho thông tin cần tìm kiếm trong dữ liệu.
- ➢ Khóa (Key - K): Đại diện cho các thông tin tham chiếu để so sánh với truy vấn.
- ➢ Giá trị (Value - V): Chứa thông tin tương ứng với từng khóa, được sử dụng để tổng hợp kết quả đầu ra.

Công thức tổng quát để tính toán được biểu diễn như sau:

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

Trong đó:

- Q: Ma trận truy vấn
- K: Ma trận khóa
- V : Ma trận giá trị.

- d_k: Kích thước của vector khóa, dùng để chia tỷ lệ.
- QK^T: Tích vô hướng giữa truy vấn và khóa để xác định độ tương quan.
- $\sqrt{d_k}$: Hệ số chia tỷ lệ để tránh giá trị lớn gây bất ổn khi tính softmax.
- softmax : Hàm chuẩn hóa, biến các giá trị tương đồng thành xác suất.

Áp dụng vào bài toán:

Để đơn giản hóa và dễ hình dung hơn, áp dụng vào một câu ngắn gồm 4 token: "Cả nhóm làm bài". Mỗi token được biểu diễn thành một vector 2 chiều

$$Q = \begin{bmatrix} 0.2 & 0.4 & 0.1 & 0.5 \\ 0.6 & 0.3 & 0.7 & 0.2 \\ 0.5 & 0.8 & 0.2 & 0.6 \\ 0.9 & 0.1 & 0.4 & 0.3 \end{bmatrix}$$

CHƯƠNG 3: TRANSFORMER

1. Tổng quan:

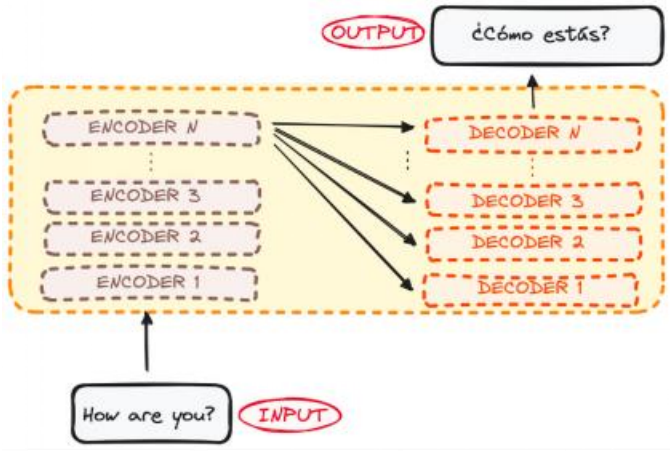
Transformer là một mô hình sequence transduction, ban đầu được thiết kế cho neural machine translation (dịch máy bằng mạng nơ-ron). Điểm đặc biệt của Transformer là nó không sử dụng Recurrent Neural Networks (RNNs) hay Convolutional Neural Networks (CNNs) như các mô hình trước đây, mà hoàn toàn dựa vào Self-Attention – một cơ chế giúp mô hình hiểu toàn bộ ngữ cảnh của câu mà không cần xử lý tuần tự.

Mô hình transformer gồm 2 thành phần chính:

- Bộ mã hóa (Encoder): Xử lý các chuỗi đầu vào và tạo ra biểu diễn liên tục của nó và ghi lại các thông tin đầu vào
- Bộ giãi mã (Decoder): Nhận đầu ra của bộ mã hóa và tạo ra chuỗi đầu ra cuối cùng bằng cách dự đoán từng token một, sử dụng các biểu diễn đã mã hóa và các token đã tạo trước đó.

Trong mô hình dịch ngôn ngữ từ tiếng Anh sang tiếng Tây Ban Nha ta có thể ví Transformer như một hộp đen và có thể hình dung như sau. Đầu vào là một câu tiếng anh ví dụ "*How are you?*" thì đầu ra là "*¿Cómo estás?*".

Encoder và Decoder không chỉ có một lớp, mà là một stack gồm nhiều layer chồng lên nhau.

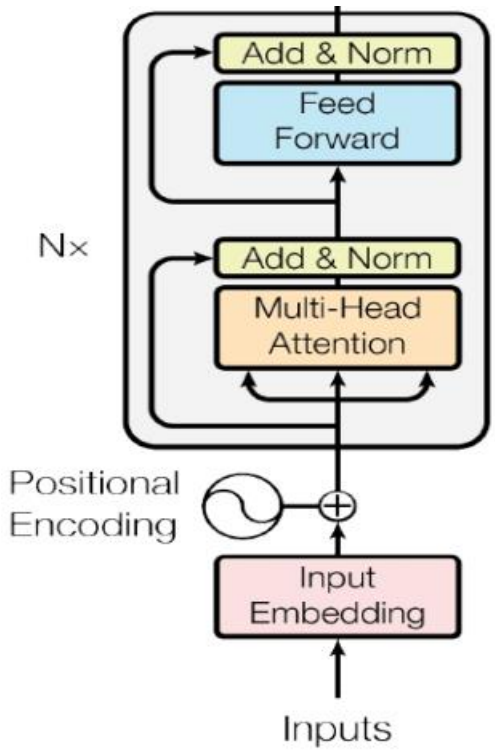


Tất cả các Encoder có cấu trúc giống nhau, nghĩa là dữ liệu được truyền từ layer này sang layer khác. Tất cả các Decoder cũng có cấu trúc giống nhau, nghĩa là nó nhận thông tin từ Encoder cuối cùng và từ chính nó ở bước trước đó để tạo câu dịch.

2. Quy trình hoạt động của Encoder:

Encoder có chức năng chính là biến đổi các token đầu vào thành các biểu diễn có ngữ cảnh (contextualized representations). Khác với các mô hình trước đây xử lý từng token một cách độc lập, Encoder trong Transformer có thể nắm bắt được ngữ cảnh của từng token trong toàn bộ chuỗi.

Cấu trúc của một Encoder trong Transformer và cách nó hoạt động:



Bước 1: Nhúng đầu vào (Input Embbedings)

Câu đầu vào được chia thành các đơn vị nhỏ hơn gọi là token, tùy thuộc vào chiến lược token hóa ta sẽ có nhiều cách biểu diễn khác nhau ví dụ câu "How are you", các token là: ["How, "are", "you"].

Mỗi token được gán một định danh duy nhất (index) dựa trên vị trí của nó trong từ vựng. Từ vựng là danh sách tất cả các token mà mô hình có thể nhận diện. Giả sử từ vựng như sau:

"How" → 0 "are" → 1 "you"→ 2

Các token được biểu diễn thành: Token IDs: [0, 1, 2]. Một ma trận embedding được tạo ra, trong đó mỗi hàng tương ứng với một token trong từ vựng, và mỗi cột biểu thị một chiều trong không gian embedding. Số chiều thường nhỏ hơn nhiều so với kích thước từ vựng.

Ví dụ: Giả sử ma trận embedding với các giá trị ngẫu nhiên như sau:

Token ID	Embedding Vector
0	[0.1, 0. 2]
1	[0.3, 0.4]
2	[0.5, 0.6]

Các token ID của câu đầu vào được sử dụng để tra cứu vector embedding tương ứng từ ma trận embedding, tạo ra một ma trận embedding cho toàn bộ câu.

Bước 2: Mã hóa vị trí (Positional Encoding)

Trong khi embedding ghi lại ý nghĩa ngữ nghĩa của các token, chúng không tự cung cấp thông tin về thứ tự của các token trong chuỗi. Để khắc phục hạn chế này, Transformer sử dụng mã hóa vị trí (positional encoding).

Mã hóa vị trí là một kỹ thuật được sử dụng để đưa thông tin về vị trí của mỗi token trong chuỗi đầu vào vào các embedding. Điều này giúp mô hình Transformer duy trì thứ tự của các token, một yếu tố cần thiết để hiểu ngữ cảnh và ý nghĩa của câu.

Mã hóa vị trí được cộng trực tiếp vào các embedding đầu vào để cung cấp thông tin về vị trí của mỗi token trong chuỗi. Thông thường, mã hóa vị trí được tạo ra bằng cách sử dụng hàm sin và cos với các tần số khác nhau, giúp mã hóa cả thông tin vị trí tuyệt đối và tương đối.

Công thức để tính mã hóa vị trí cho mỗi vị trí pos và chiều i trong embedding được xác định như sau:

Cho các chiều chẵn (i là chẵn):

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

Cho các chiều lẻ (i là lẻ):

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

Trong đó:

+ pos: Vị trí của token trong chuỗi (bắt đầu từ 0)
+ i: Chỉ số chiều trong vector embedding.
+ d_model: Tổng số chiều của embedding (ví dụ: 512 hoặc 768).
+ Hàm sin và cos tạo ra các giá trị dao động với tần số khác nhau cho từng vị trí và chiều, đảm bảo mỗi vị trí có một mã hóa duy nhất.
+ Hằng số 10000 được chọn để tạo ra các tần số giảm dần, giúp mã hóa các vị trí khác nhau một cách hiệu quả.
+ Các giá trị mã hóa vị trí có cùng kích thước với embedding (d_model), nên có thể cộng trực tiếp vào embedding mà không cần thay đổi kích thước.

Ví dụ minh họa:

Câu đầu vào: “How are you”

Kích thước embedding: d_model = 2.

Các vị trí pos = 0, 1, 2.

Tính mã hóa vị trí pos = 0:

Chiều 0: PE(0, 0) = sin (0) = 0

Chiều 1: PE(0, 1) = cos(0) = 1

Tương tự, tính cho các vị trí khác (pos = 1, 2), mỗi vị trí sẽ có một vector mã hóa vị trí duy nhất.

Sau đó, vector mã hóa vị trí được cộng vào vector embedding của token tương ứng:

Embedding sau khi thêm mã hóa vị trí = Embedding gốc + Positional Encoding.

BƯỚC 3: Ngăn xếp các lớp bộ mã hóa (Stack of Encoder Layers)

Encoder nhận đầu vào là một ma trận embedding của chuỗi (đã được cộng với mã hóa vị trí để phản ánh thứ tự token) và xử lý qua nhiều Encoder Layer liên tiếp.

Mỗi Encoder Layer có hai thành phần chính:

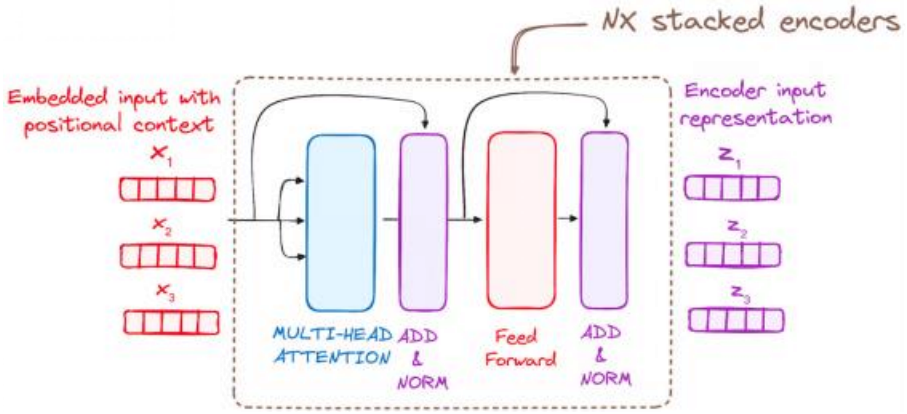
+ Cơ chế tự chú ý đa đầu (Multi-Head Self-Attention Mechanism): Cho phép mô hình tập trung vào các phần khác nhau của chuỗi đầu vào, nắm bắt các mối quan hệ phức tạp giữa các token.

+ Mạng nơ-ron truyền thẳng (Feed-Forward Neural Network):

Các lớp này có cơ chế kết nối dư (Residual Connection) và chuẩn hóa lớp (Layer Normalization) để giúp quá trình học hiệu quả hơn.

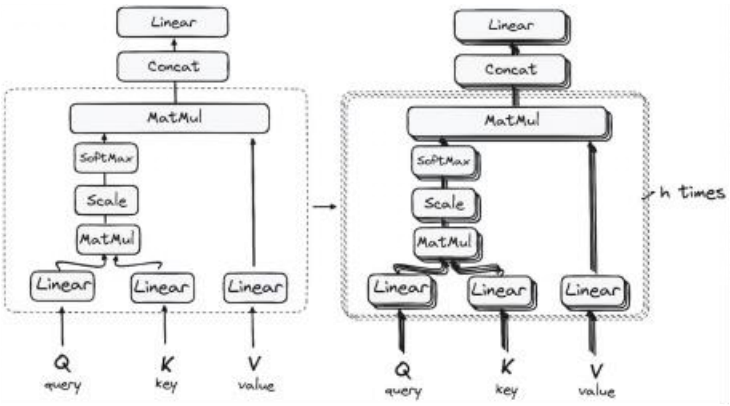
+ Kết nối dư (Residual Connections): Giúp duy trì thông tin từ đầu vào, hỗ trợ gradient lan truyền ngược hiệu quả.

+ Chuẩn hóa lớp (Layer Normalization): Ổn định đầu ra, đảm bảo quá trình huấn luyện mượt mà.



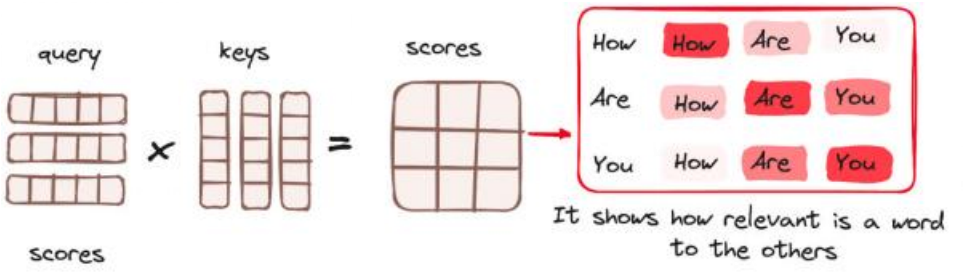
Bước 3.1: Cơ chế Tự chú ý Đa đầu (Multi-Head Self-Attention Mechanism)

Cơ chế tự chú ý đa đầu là nền tảng của bộ mã hóa, cho phép mô hình phân tích toàn bộ chuỗi đầu vào đồng thời, thay vì xử lý tuần tự như các mô hình RNN. Tự chú ý đa đầu mở rộng cơ chế tự chú ý đơn giản bằng cách sử dụng nhiều đầu chú ý (attention heads), mỗi đầu tập trung vào một khía cạnh khác nhau của mối quan hệ giữa các token, từ cú pháp đến ngữ nghĩa.

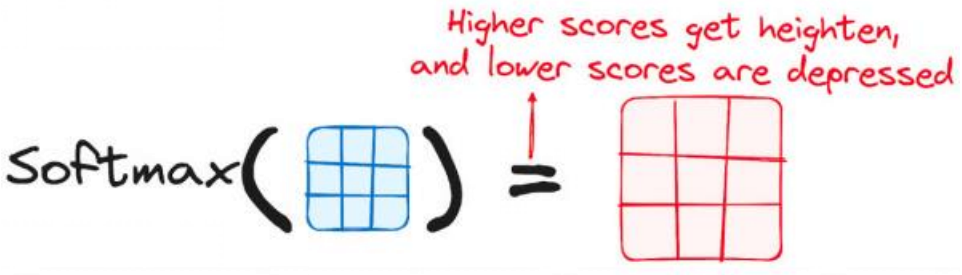


Thực hiện chi tiết như sau:

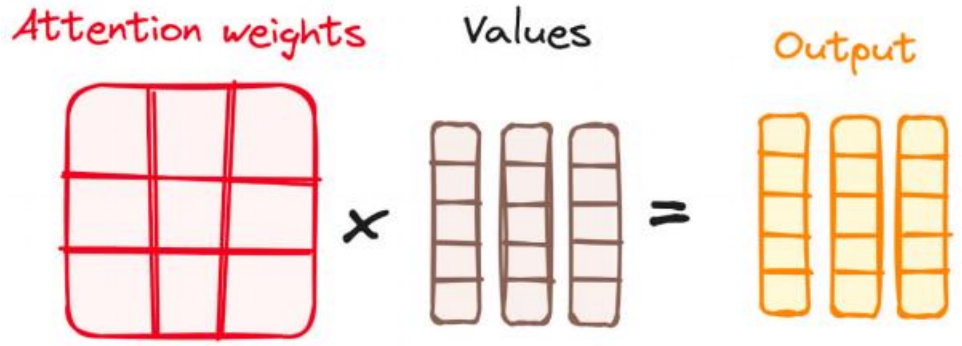
- Nhân Ma Trận (Matrix Multiplication - MatMul) giữa Query và Key: Khi các vector Query, Key và Value đi qua một lớp tuyến tính, phép nhân ma trận (dot product) được thực hiện giữa Query và Key để tạo ra ma trận điểm số (score matrix). Ma trận điểm số xác định mức độ quan trọng của từng từ so với các từ khác. Mỗi từ được gán một điểm số tương ứng với các từ khác. Điểm số cao hơn có nghĩa là mô hình tập trung hơn vào từ đó.



- Giảm biên độ biên chú ý: Các điểm số sau đó được chuẩn hóa bằng cách chia cho căn bậc hai của kích thước vector Query và Key.Mục đích của bước này là duy trì độ ổn định của gradient, tránh trường hợp giá trị quá lớn gây ảnh hưởng xấu đến quá trình huấn luyện.
- Áp dụng Softmax để điều chỉnh điểm số: Sau khi chuẩn hóa, một hàm softmax được áp dụng lên ma trận điểm số để chuyển đổi điểm số thành xác suất từ 0 đến 1. Hàm Softmax giúp tăng cường sự chú ý vào các từ có điểm số cao và giảm ảnh hưởng của các từ có điểm số thấp.



- Kết hợp kết quả Softmax với vector Value: Các trọng số từ Softmax sẽ được nhân với vector giá trị, tạo ra một vector đầu ra. Quá trình này giúp giữ lại những từ có trọng số cao trong mô hình. Vector đầu ra sau đó được đưa qua một lớp tuyến tính để tiếp tục xử lý.



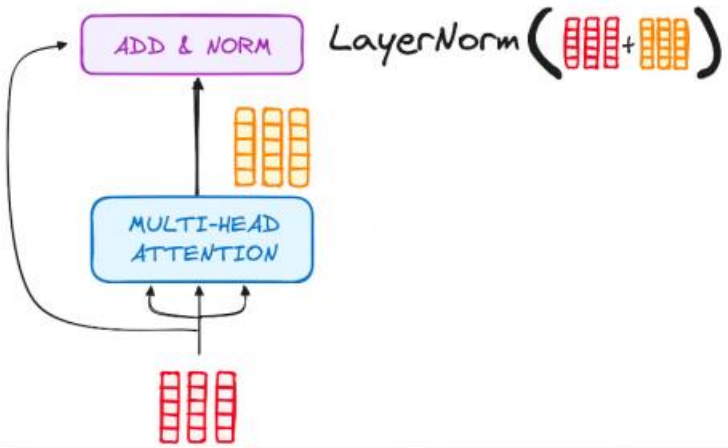
Bước 3.2: Add & Norm

Add & Norm là một bước quan trọng trong mỗi tầng (layer) của kiến trúc bộ mã hóa Transformer. Bước này bao gồm hai thao tác chính: Kết nối dư (Residual Connection - Add) và Chuẩn hóa tầng (Layer Normalization - Norm). Các thao tác này được thực hiện hai lần trong mỗi tầng mã hóa: sau cơ chế tự chú ý đa đầu (multi-head self-attention) và sau mạng nơ-ron tiến (feed-forward neural network). Mục tiêu chính là tăng cường sự ổn định trong huấn luyện, cải thiện dòng gradient và giúp mô hình học hiệu quả hơn trong các mạng sâu.

- Kết nối dư (Residual Connection - Add): Kết nối dư là quá trình cộng đầu vào của một tầng với đầu ra của tầng đó. Nếu đầu vào của tầng là x và phép biến đổi của tầng là F(x), thì đầu ra của kết nối dư được tính như sau: $y = x + F(x)$
- Chuẩn hóa tầng (Layer Normalization - Norm): Layer Normalization chuẩn hóa các đặc trưng đầu ra của một tầng, trên từng đầu vào riêng lẻ. Thay vì chuẩn hóa theo batch như BatchNorm, nó chuẩn hóa trên trục đặc trưng (features) của mỗi sample đầu vào.

Cách thức hoạt động:

- Tính trung bình và phương sai của các giá trị trong vector đầu vào (tức là trên tất cả đặc trưng)
- Chuẩn hóa các đặc trưng sao cho chúng có giá trị trung bình gần 0 và độ lệch chuẩn gần 1.
- Điều chỉnh lại bằng 2 tham số có thể học được:γ (gamma): kéo giãn (scale) ;
β (beta): tịnh tiến (shift).



Bước 3.3: Mạng Nơ-ron Truyền thẳng (Feed-Forward Neural Network - FFNN)

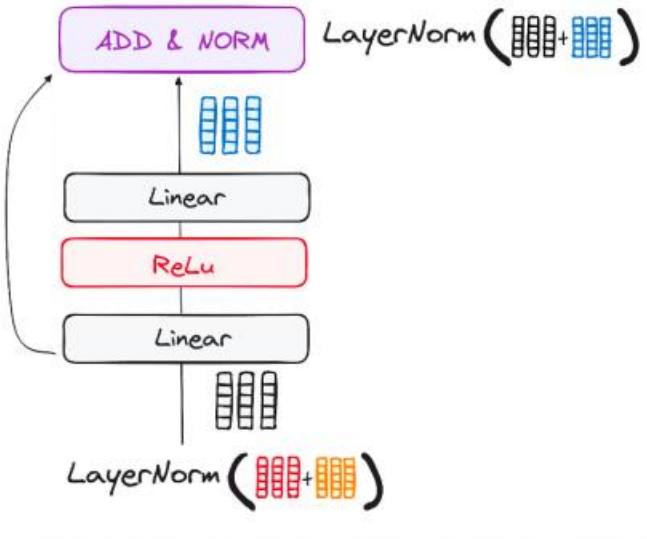
Sau khi đi qua quá trình chuẩn hóa và kết nối dư (Residual Connection), đầu ra tiếp tục được đưa vào mạng nơron truyền thẳng (FFNN). Đây là một bước quan trọng giúp mô hình xử lý dữ liệu sâu hơn và học được những đặc trưng phức tạp hơn.

Hãy tưởng tượng mạng này như một hệ thống gồm hai lớp tuyến tính (Linear Layers), với một hàm kích hoạt ReLU nằm ở giữa đóng vai trò như một cây cầu kết nối hai lớp lại với nhau.

- Lớp tuyến tính đầu tiên: Biến đổi dữ liệu theo một cách nhất định.
- Hàm kích hoạt ReLU: Giữ lại các giá trị dương và loại bỏ giá trị âm để mô hình học tốt hơn.
- Lớp tuyến tính thứ hai: Tiếp tục xử lý dữ liệu để chuẩn bị cho bước tiếp theo.

Sau khi được xử lý qua mạng nơ-ron truyền thẳng, đầu ra không bị mất đi mà quay trở lại và kết hợp với đầu vào ban đầu của chính mạng này. Điều này giúp thông tin gốc không bị lãng quên, đồng thời thêm những đặc trưng mới vào dữ liệu.

Cuối cùng, dữ liệu được chuẩn hóa lại một lần nữa để đảm bảo không có giá trị nào quá lớn hoặc quá nhỏ, giúp mô hình hoạt động ổn định hơn.



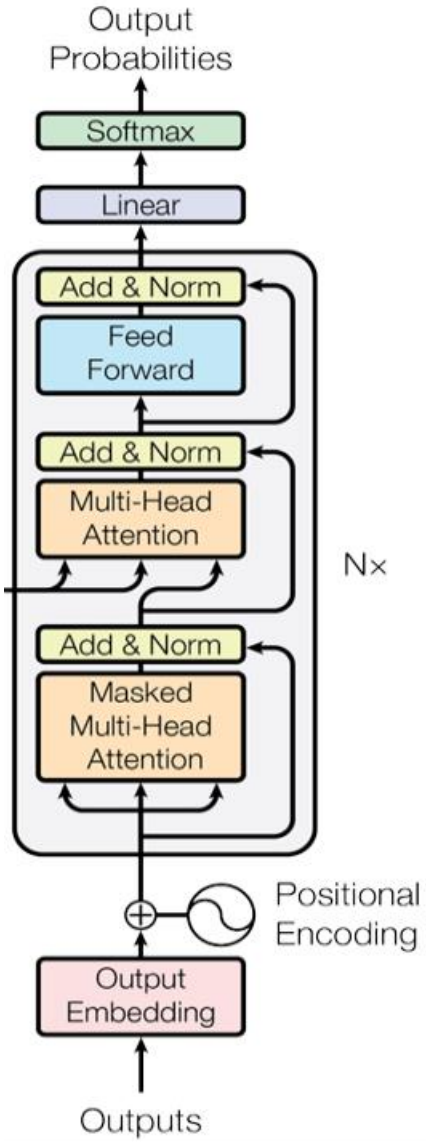
Bước 4: Đầu ra của Bộ mã hóa (Output of the Encoder): Khi dữ liệu đầu vào đi qua tất cả các lớp của bộ mã hóa (encoder layers), đầu ra thu được sẽ là một tập hợp các vector đặc trưng. Mỗi vector này đại diện cho từng từ trong câu đầu vào nhưng đã được bổ sung thêm thông tin ngữ cảnh quan trọng. Đầu ra này sẽ được đưa vào bộ giải mã (decoder) để tiếp tục xử lý và tạo ra câu đầu ra.

3. Quy trình hoạt động của Decoder:

Decoder hoạt động khá giống Encoder nhưng có một số khác biệt quan trọng.

Cấu trúc của bộ giải mã gồm:

- Một lớp self-attention nhiều đầu (multi-heading attention layers): giúp bộ giải mã tập trung vào các phần quan trọng của dữ liệu đầu vào và các từ đã được tạo trước đó.
- Một lớp attention kết hợp với đầu ra của encoder (encoder-decoder attention): cho phép bộ giải mã tham chiếu đến thông tin ngữ cảnh từ đầu vào ban đầu.
- Một lớp mạng nơ-ron truyền thẳng (pointwise feed-forward layer): xử lý và biến đổi dữ liệu đầu ra từ các lớp attention.
- Cơ chế kết nối dư (residual connections) và chuẩn hóa lớp (layer normalization): giúp cải thiện quá trình học và tránh mất mát thông tin.



Bộ giải mã (decoder) hoạt động tương tự bộ mã hóa (encoder) nhưng có các lớp attention với nhiệm vụ riêng biệt .

Giai đoạn cuối của quá trình giải mã bao gồm một lớp tuyến tính (linear layer) đóng vai trò như một bộ phân loại (classifier), sau đó là một hàm softmax (softmax function) để tính toán xác suất của các từ khác nhau.

Điều quan trọng cần lưu ý là bộ giải mã hoạt động theo cách tự hồi quy (autoregressive manner), bắt đầu quá trình với một token bắt đầu (start token). Nó sử dụng danh sách các đầu ra đã được tạo trước đó (previously generated outputs) làm đầu vào (inputs), đồng thời kết hợp với đầu ra từ bộ mã hóa (encoder outputs) – vốn chứa nhiều thông tin attention từ đầu vào ban đầu.

Quá trình giải mã tuần tự (sequential decoding) này tiếp tục cho đến khi bộ giải mã đạt đến một khoảnh khắc quan trọng: tạo ra một token báo hiệu kết thúc (end token), chấm dứt quá trình sinh đầu ra.

Bước 1: Đầu vào của Bộ giải mã (Chuỗi mục tiêu - Target Sequence)

Chuỗi mục tiêu là chuỗi các token mà bộ giải mã đã tạo ra cho đến thời điểm hiện tại (trong quá trình suy luận) hoặc chuỗi đầu ra mong đợi (trong quá trình huấn luyện). Chuỗi bắt đầu bằng token đặc biệt [START], bao gồm các token của chuỗi đầu ra, và kết thúc bằng token [END]

Với câu "The cat sat on mat" (Con mèo ngồi trên thảm), chuỗi mục tiêu là:

["[START]", "The", "cat", "sat", "on", "mat", "[END]"]

Bước 2: Embedding đầu ra (Output Embeddings)

Mỗi token trong chuỗi mục tiêu được chuyển thành một vector dày đặc (embedding) bằng một ma trận nhúng đầu ra được học trong quá trình huấn luyện. Ma trận này ánh xạ mỗi token thành một vector trong không gian liên tục, thể hiện ý nghĩa ngữ nghĩa, quá trình này tương tự như bộ mã hóa (encoder).

Bước 3: Mã hóa vị trí (Positional Encoding)

Sau khi được nhúng (embedding), tương tự như trong bộ mã hóa, đầu vào tiếp tục đi qua lớp mã hóa vị trí (positional encoding layer).

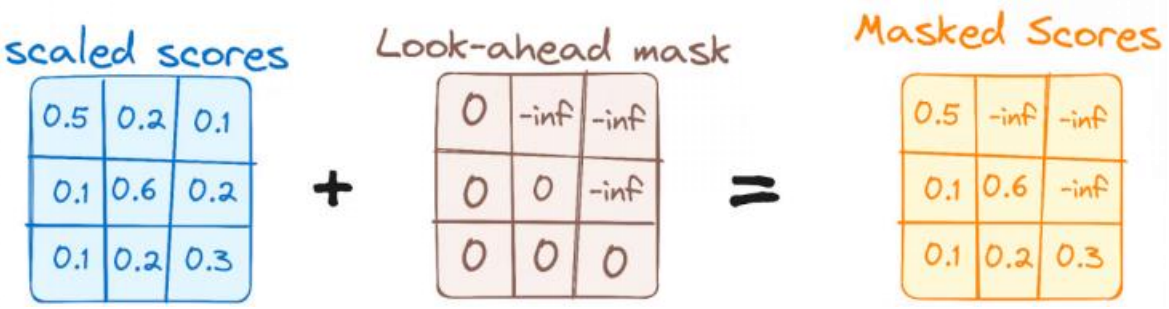
Bước 4: Ngăn xếp các lớp giải mã (Stack of Decoder Layers)

Bộ giải mã bao gồm một chồng các lớp giống hệt nhau (stack of identical layers), với 6 lớp trong mô hình Transformer gốc.

Bước 4.1: Cơ chế tự chú ý có mặt nạ (Masked Self-Attention Mechanism)

Ở bước này, mô hình sử dụng cơ chế masked attention – một biến thể đặc biệt của multi-head attention được dùng riêng trong phần decoder. Khác với multi-head attention trong encoder (nơi mỗi vị trí có thể “nhìn” toàn bộ chuỗi), masked attention chỉ cho phép mỗi vị trí nhìn thấy các token đứng trước hoặc tại chính nó.

Điều này đảm bảo rằng khi mô hình tạo ra từng từ trong chuỗi, nó không bị ảnh hưởng bởi các token chưa được sinh ra, giữ cho quá trình dự đoán diễn ra theo đúng thứ tự từ trái sang phải. Đây là một yếu tố then chốt giúp quá trình sinh ngôn ngữ diễn ra tự nhiên và hợp lý.



Cộng và Chuẩn hóa (Add & Norm):

- Kết nối dư (Residual Connection): Đầu ra của chú ý đa đầu được cộng lại với đầu vào (nhúng có mã hóa vị trí).
- Chuẩn hóa tầng (Layer Normalization): Kết quả được chuẩn hóa để ổn định huấn luyện.

Bước 4.2: Multi-Head Attention Bộ mã hóa - Bộ giải mã (Encoder-Decoder) hoặc Cross Attention

Trong lớp multi-head attention thứ hai của decoder (gọi là encoder-decoder attention), có một sự tương tác đặc biệt giữa các thành phần của encoder và decoder. Cụ thể, các truy vấn (queries) đến từ đầu ra của lớp decoder trước đó (tức là sau lớp self-attention đầu tiên), trong khi các khóa (keys) và giá trị (values) đến từ đầu ra của encoder.

Cấu trúc này cho phép mỗi vị trí trong chuỗi đầu ra của decoder có thể truy cập và tham chiếu đến toàn bộ đầu ra của encoder, giúp decoder học được cách tập trung vào các phần thông tin quan trọng trong chuỗi đầu vào, phù hợp với ngữ cảnh hiện tại đang sinh.

Sự tương tác này đóng vai trò như một cầu nối giữa thông tin đã được mã hóa từ chuỗi đầu vào (encoder) và quá trình sinh ngôn ngữ ở chuỗi đầu ra (decoder). Nhờ đó, decoder có thể căn chỉnh (align) thông tin một cách hiệu quả để đưa ra dự đoán chính xác hơn tại mỗi bước.

Sau khi tính toán attention, đầu ra sẽ trải qua các bước sau:

- Residual connection (Add): kết quả của attention được cộng với đầu vào ban đầu của khối attention này (tức là đầu ra từ lớp self-attention trước đó trong decoder).
- Normalization (Norm): kết quả sau khi cộng được đưa qua một lớp layer normalization, giúp ổn định và tăng hiệu quả huấn luyện.

Bước 4.3: Mạng Nơ-ron Truyền Thẳng (Feed-Forward Neural Network)

Tương tự như trong encoder, mỗi lớp của decoder cũng bao gồm một mạng nơ-ron truyền thẳng (feed-forward neural network) được áp dụng riêng biệt cho từng vị trí trong chuỗi đầu ra. Mạng này là

một mạng hoàn toàn kết nối (fully connected) gồm hai lớp tuyến tính với một hàm kích hoạt phi tuyến nằm giữa.

Sau feed-forward network, kết quả cũng đi qua một bước Add & Norm nữa:

+ Residual connection: đầu vào ban đầu được cộng lại với đầu ra của mạng FFN.

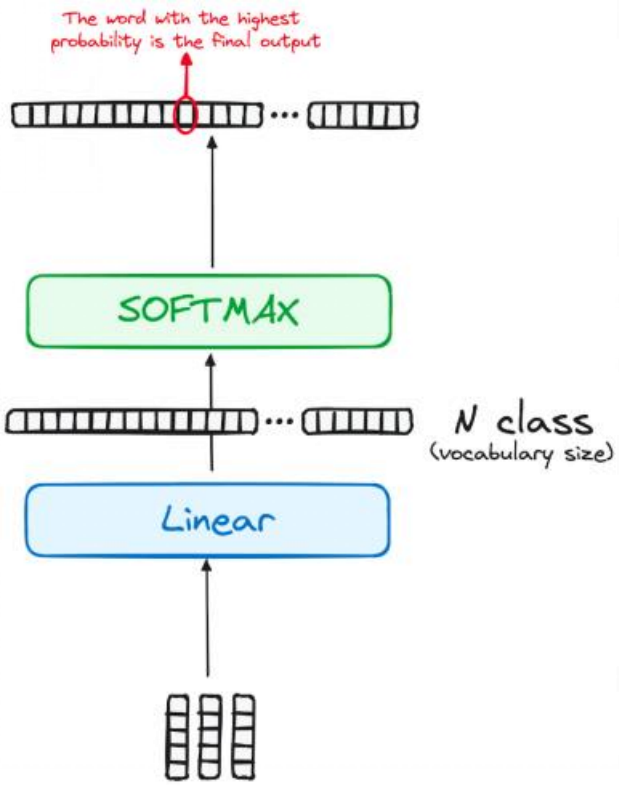
+ Layer normalization: giúp ổn định gradient và tăng hiệu quả huấn luyện.

Bước 5: Bộ phân loại tuyến tính và Softmax để tạo xác suất đầu ra (Linear Classifier and Softmax for Generating Output Probabilities)

Hành trình của dữ liệu qua mô hình Transformer kết thúc bằng việc đi qua một lớp tuyến tính cuối cùng, hoạt động như một bộ phân loại.

Kích thước của bộ phân loại này tương ứng với tổng số lớp đầu ra (chính là số lượng từ trong từ vựng). Ví dụ, nếu có 1000 lớp tương ứng với 1000 từ khác nhau, thì đầu ra của bộ phân loại sẽ là một mảng có 1000 phần tử.

Sau đó, đầu ra này được đưa vào một lớp softmax, chuyển đổi các giá trị thành xác suất nằm trong khoảng từ 0 đến 1. Giá trị xác suất cao nhất sẽ quyết định từ tiếp theo trong chuỗi dự đoán của mô hình.



Đầu ra của Bộ giãi mã (Output of Decoder)

Lớp cuối cùng của bộ giải mã tạo ra một chuỗi dự đoán bằng cách sử dụng một lớp tuyến tính, sau đó áp dụng softmax để tạo xác suất trên toàn bộ từ vựng. Trong quá trình hoạt động, bộ giải mã sẽ thêm đầu ra vừa tạo vào danh sách đầu vào của nó, rồi tiếp tục giải mã.Quá trình này lặp lại cho đến khi mô hình dự đoán ra một token đặc biệt, báo hiệu quá trình hoàn tất. Token có xác suất cao nhất sẽ được chọn làm từ tiếp theo, và thường token kết thúc (end token) sẽ đánh dấu điểm dừng.

Cấu trúc nhiều lớp của decoder:

- Decoder không giới hạn chỉ trong một lớp mà có thể có N lớp.
- Mỗi lớp mới sẽ tiếp tục xử lý thông tin từ encoder và từ các lớp trước đó.
- Cấu trúc nhiều lớp này giúp mô hình có thể tập trung vào nhiều thông tin quan trọng khác nhau trong dữ liệu.
- Cách tiếp cận này cải thiện khả năng dự đoán của mô hình bằng cách tạo ra sự hiểu biết sâu sắc hơn về các mẫu chú ý khác nhau.

4. Thực hành: Xây dựng transformer với Pytorch:

CHƯƠNG 4: BERT – CÁCH TRANSFORMER HIỂU NGỮ CẢNH

Trong xử lý ngôn ngữ tự nhiên (NLP), việc **hiểu ngữ cảnh** là một trong những thách thức lớn nhất. Các mô hình truyền thống như RNN hay LSTM thường gặp khó khăn khi phải nắm bắt mối quan hệ xa trong câu, còn GPT đời đầu lại chỉ xử lý theo một chiều (từ trái sang phải). Chính vì vậy, Google đã giới thiệu **BERT (Bidirectional Encoder Representations from Transformers)** – một bước ngoặt trong NLP.

Điểm đặc biệt của BERT nằm ở 2 cơ chế huấn luyện chính:

- **Masked Language Model (MLM)**: mô hình học cách đoán từ bị che giấu dựa trên cả ngữ cảnh hai bên.
- **Next Sentence Prediction (NSP)**: mô hình học mối quan hệ giữa hai câu liền kề, giúp xử lý tốt hơn các tác vụ liên quan đến văn bản dài.

4.1. Masked Language Model (MLM)

4.1.1. Khái niệm và ý nghĩa

Khái niệm:

Mô hình Ngôn ngữ Che giấu (*Masked Language Model* – **MLM**) là một phương pháp huấn luyện trong học sâu, được thiết kế nhằm giúp mô hình dự đoán các từ bị ẩn trong câu. Trong quá trình huấn luyện, một tỷ lệ nhất định các từ trong câu đầu vào được thay thế bằng token đặc biệt **[MASK]**, và nhiệm vụ của mô hình là **dự đoán lại những từ đã bị che giấu dựa trên ngữ cảnh hai chiều (trước và sau từ đó)**.

Cách tiếp cận này giúp mô hình học được mối quan hệ ngữ nghĩa giữa các từ và hiểu sâu hơn về cấu trúc ngôn ngữ tự nhiên.

Ý nghĩa:

Trước khi BERT ra đời, hầu hết các mô hình ngôn ngữ như GPT hoặc các RNN-based model (ví dụ LSTM, GRU) chỉ học ngữ cảnh **theo một chiều** — từ trái sang phải hoặc từ phải sang trái. Điều này khiến chúng khó có thể hiểu đầy đủ ý nghĩa của câu trong toàn bộ ngữ cảnh.

BERT khắc phục hạn chế này bằng cách sử dụng **Masked Language Model (MLM)**, cho phép mô hình khai thác **ngữ cảnh hai chiều** (bidirectional context). Nhờ đó, mô hình có khả năng "hiểu" câu theo cách tương tự con người, nắm bắt được mối quan hệ giữa các từ trong toàn bộ chuỗi, thay vì chỉ dựa vào phần trước hoặc sau.

Ví dụ minh họa:

Giả sử ta có câu gốc:

"Con mèo đang ngủ trên ghế."

Trong quá trình huấn luyện theo cơ chế **Masked Language Model**, một số từ trong câu được chọn ngẫu nhiên để che giấu. Khi đó, câu trên có thể được chuyển thành:

"Con [MASK] đang ngủ trên ghế."

Mục tiêu của mô hình BERT là **dự đoán từ bị che**, tức là tìm lại từ gốc ban đầu — trong trường hợp này, mô hình dự đoán đúng từ **"mèo"**.

Thông qua quá trình dự đoán này, BERT học được **các mối quan hệ ngữ nghĩa và cú pháp giữa các từ trong câu**, từ đó hiểu rõ hơn ngữ cảnh hai chiều, thay vì chỉ hiểu theo một hướng như các mô hình ngôn ngữ truyền thống.

4.1.2 Cơ chế của mô hình MLM

Sự khác biệt giữa MLM và Mô hình Ngôn ngữ Truyền thống

Đặc điểm	Mô hình truyền thống (GPT)	MLM(BERT)
Hướng đọc	Một chiều (trái → phải)	Hai chiều (trái ↔ phải)
Ngữ cảnh sử dụng	Chỉ ngữ cảnh trước	Ngữ cảnh hai phía
Biểu diễn từ	Không phụ thuộc vào ngữ cảnh	Phụ thuộc vào ngữ cảnh

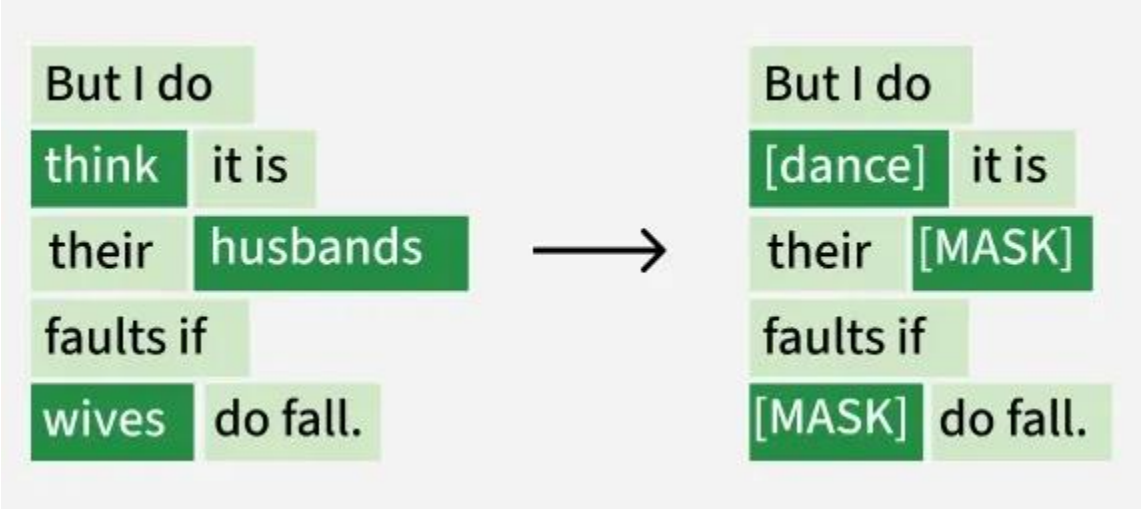
Về mặt trực quan, có thể hình dung rằng khi con người đọc một câu, việc hiểu nghĩa của một từ không chỉ phụ thuộc vào những từ đứng trước mà còn cả những từ theo sau. MLM cũng hoạt động theo cơ chế tương tự – tận dụng ngữ cảnh hai chiều để dự đoán chính xác hơn.

Cơ chế cốt lõi của MLM

Quá trình đào tạo mô hình ngôn ngữ có mặt nạ bao gồm hai bước chính:

- Bước 1: Che từ

Trong quá trình đào tạo, mô hình được trình bày bằng các câu trong đó một số từ được thay thế ngẫu nhiên bằng một mã thông báo đặc biệt, chẳng hạn như "[MASK]". Trong ví dụ dưới đây, hai từ đã được thay thế bằng mã thông báo mặt nạ trong khi một từ khác được thay thế bằng mã thông báo từ khác.



- Bước 2: Dự đoán các từ còn thiếu

Sau đó, mô hình được giao nhiệm vụ dự đoán từ gốc đã bị che khuất. Nó thực hiện việc này bằng cách phân tích các từ xung quanh trong câu. Sử dụng ví dụ trên, mô hình sẽ dự đoán "books" dựa trên ngữ cảnh được cung cấp bởi "reads" và "every evening".

Quá trình này được lặp lại hàng triệu lần trên một lượng lớn dữ liệu văn bản và cho phép mô hình học các mẫu, ngữ pháp và mối quan hệ ngữ nghĩa trong ngôn ngữ.

Nhờ cơ chế này, mô hình có thể biểu diễn mỗi từ theo ngữ cảnh xung quanh, thay vì giữ nghĩa cố định. Đây là nền tảng giúp các mô hình như BERT "hiểu" ngôn ngữ tự nhiên tốt hơn nhiều so với các mô hình cũ.

4.1.3 Kiến trúc Transformer và vai trò trong MLM

Để cơ chế MLM hoạt động hiệu quả, mô hình cần một kiến trúc có khả năng quan sát toàn bộ câu cùng lúc, xác định mối quan hệ giữa mọi cặp từ, và xử lý song song trên quy mô lớn. **Kiến trúc Transformer** ra đời chính là để đáp ứng những yêu cầu này.

Trong BERT, Transformer đảm nhận vai trò **nền tảng** giúp mô hình thực hiện Masked Language Modeling hiệu quả. Nhờ cơ chế **Self-Attention**, mỗi từ trong câu có thể "nhìn thấy" và liên kết với tất cả các từ còn lại, kể cả ở phía trước hay phía sau. Điều này giúp mô hình nắm bắt được ngữ cảnh hai chiều — yếu tố cốt lõi trong MLM.

Transformer gồm hai thành phần chính: **Encoder** và **Decoder**, nhưng BERT chỉ sử dụng phần **Encoder**. Mỗi lớp Encoder bao gồm hai khối:

- **Multi-Head Self-Attention:** cho phép mô hình học nhiều mối quan hệ ngữ nghĩa khác nhau giữa các từ.
- **Feed-Forward Network:** giúp tinh chỉnh và tổng hợp thông tin sau khi đã được chú ý.

Nhờ sự kết hợp giữa MLM và kiến trúc Transformer, BERT có thể học biểu diễn ngữ nghĩa giàu thông tin, phản ánh ngữ cảnh toàn câu. Đây chính là nền tảng giúp BERT trở thành một trong những mô hình hiểu ngôn ngữ mạnh mẽ nhất hiện nay.

4.1.4. Quá trình huấn luyện MLM

Mục tiêu huấn luyện

MLM được huấn luyện để **giảm thiểu sai số** giữa từ dự đoán và từ thực tế bị che, thường sử dụng hàm mất mát **cross-entropy**.

Ví dụ:

Câu gốc: *"The quick brown fox jumps over the lazy dog."*
Sau khi che: *"The ___ brown fox jumps over the lazy dog."*

- Nếu mô hình dự đoán "fast" thay vì "quick", nghĩa gần đúng nhưng không khớp, mô hình sẽ bị phạt bằng một mức loss cao.
- Ngược lại, nếu dự đoán chính xác "quick", loss sẽ nhỏ và mô hình biết rằng nó đang học đúng hướng.

Quá trình này lặp đi lặp lại trên hàng triệu câu văn khác nhau. Nhờ đó, mô hình dần học được quy tắc ngữ pháp, ngữ nghĩa và cả những mối liên hệ phức tạp trong ngôn ngữ.

Tiền xử lý dữ liệu

Trước khi đưa văn bản vào huấn luyện, dữ liệu cần qua các bước xử lý:

- **Tokenization (tách từ):** Văn bản được tách thành các đơn vị nhỏ gọi là token. Với BERT, thường sử dụng kỹ thuật **WordPiece**, cho phép chia từ thành subword, giúp xử lý từ mới hoặc từ hiếm.
- **Masking Strategy (chiến lược che giấu):** Khoảng 15% token sẽ bị thay bằng `[MASK]`. Nhiệm vụ của mô hình là dự đoán lại từ gốc dựa trên ngữ cảnh.

Ví dụ:

- Câu gốc: *"Con mèo ngồi trên tấm thảm."*
- Sau khi che: *"Con mèo ngồi [MASK] tấm thảm."*
- Mục tiêu: dự đoán đúng từ *"trên"*.

Thách thức trong huấn luyện

- **Dữ liệu thưa thớt:** Vì chỉ che 15% token, mô hình chỉ học được một phần nhỏ dữ liệu ở mỗi lượt huấn luyện.
- **Chi phí tính toán cao:** Huấn luyện BERT hoặc RoBERTa cần GPU/TPU mạnh, thời gian kéo dài từ nhiều ngày đến vài tuần.
- **Nguy cơ quá khớp (overfitting):** Mô hình có thể học thuộc dữ liệu thay vì học khái quát. Để tránh điều này, người ta thường dùng dropout, regularization, hoặc early stopping.

Tối ưu hóa và tinh chỉnh

- Phần lớn các mô hình MLM dùng **AdamW optimizer**, giúp kiểm soát sự thay đổi trọng số và tăng khả năng khái quát.
- Sau tiền huấn luyện, **fine-tuning** trên dữ liệu gắn nhãn sẽ giúp mô hình "học nghề". Ví dụ, để phân loại đánh giá phim, ta chỉ cần tinh chỉnh BERT trên tập dữ liệu review phim có gắn nhãn tích cực/tiêu cực.

4.1.5. Các mô hình ngôn ngữ che giấu phổ biến

- **BERT:**
  Trước BERT, các mô hình như GPT chỉ xử lý văn bản một chiều, nên chỉ nắm được một nửa ngữ cảnh. BERT thay đổi điều này bằng cách khai thác ngữ cảnh hai chiều, quan sát đồng thời cả từ trước và sau từ bị che. Nhờ vậy, nó học được ngữ nghĩa sâu hơn và cải thiện rõ rệt trên hầu hết các tác vụ NLP.
- **RoBERTa:**
  Được phát triển dựa trên BERT nhưng tối ưu hơn:
  - Bỏ nhiệm vụ NSP (Next Sentence Prediction) vốn không mang lại nhiều lợi ích.
  - Tăng quy mô dữ liệu và chiều dài câu huấn luyện.
  - Thay đổi chiến lược che giấu để mô hình học tốt hơn.
    Kết quả, RoBERTa đạt hiệu năng vượt trội hơn BERT trong nhiều benchmark. Nếu ví BERT như một vận động viên tài năng, thì RoBERTa chính là phiên bản "được huấn luyện bài bản và chuyên nghiệp hơn".
- **ALBERT và DistilBERT:**
  - **ALBERT (A Lite BERT):** Giảm số lượng tham số bằng cách chia sẻ trọng số giữa các lớp và nén ma trận embedding, giúp mô hình nhẹ nhưng vẫn mạnh mẽ.
  - **DistilBERT:** Là phiên bản rút gọn của BERT, nhanh hơn và tiết kiệm tài nguyên, phù hợp để triển khai trên thiết bị di động hoặc hệ thống có hạn chế về phần cứng.

4.1.6. Ứng dụng của Mô hình Ngôn ngữ Bị Che giấu (MLM)

Các mô hình ngôn ngữ bị che giấu, tiêu biểu là **BERT** và các biến thể của nó, đã trở thành nền tảng trong nhiều ứng dụng xử lý ngôn ngữ tự nhiên hiện đại. Sau khi được tiền huấn luyện bằng cơ chế **Masked Language Modeling (MLM)**, chúng có khả năng nắm bắt ngữ cảnh và biểu diễn ngôn ngữ ở mức độ sâu sắc, từ đó dễ dàng tinh chỉnh để phục vụ cho nhiều tác vụ khác nhau.

Phân loại văn bản (Text Classification)

MLM được ứng dụng mạnh mẽ trong các bài toán phân loại như:

- **Phân tích cảm xúc (Sentiment Analysis):** Xác định một bài viết, đánh giá phim hay bình luận trên mạng xã hội mang cảm xúc tích cực hay tiêu cực. Ví dụ, Google và nhiều công ty thương mại điện tử đã dùng BERT để phân loại đánh giá sản phẩm, giúp lọc phản hồi tiêu cực nhanh hơn.
- **Lọc thư rác (Spam Detection):** Các email hoặc tin nhắn được gán nhãn spam/không spam. Nhờ biểu diễn ngữ cảnh của BERT, độ chính xác của hệ thống cao hơn so với mô hình cũ.

Trả lời câu hỏi (Question Answering – QA)

Một trong những ứng dụng nổi bật nhất của MLM là hệ thống hỏi – đáp. Với cơ chế hiểu ngữ cảnh hai chiều, BERT có thể xác định vị trí câu trả lời trong một đoạn văn bản.

- Ví dụ: Trong tập dữ liệu **SQuAD (Stanford Question Answering Dataset)**, BERT đã đạt kết quả vượt trội, thậm chí vượt qua cả mức trung bình của con người trên một số chỉ số.
- Ứng dụng thực tế: trợ lý ảo (Google Assistant, Siri) hay chatbot dịch vụ khách hàng.

Nhận diện thực thể có tên (Named Entity Recognition – NER)

Trong NER, hệ thống cần phát hiện và phân loại các thực thể trong văn bản như tên người, địa điểm, tổ chức, thời gian…

- Ví dụ: trong câu *"Bill Gates thành lập Microsoft vào năm 1975"*, mô hình cần nhận ra *"Bill Gates"* là người, *"Microsoft"* là tổ chức và *"1975"* là mốc thời gian.
- BERT và các biến thể giúp cải thiện độ chính xác nhờ khả năng hiểu ngữ cảnh sâu rộng.

Dịch máy và tóm tắt văn bản

- **Dịch máy (Machine Translation):** Mặc dù BERT không được thiết kế trực tiếp cho dịch, nhưng nhờ khả năng học biểu diễn ngôn ngữ hai chiều, nó trở thành nền tảng quan trọng trong các hệ thống dịch đa ngữ (ví dụ: mBERT – Multilingual BERT).
- **Tóm tắt văn bản (Text Summarization):** Các mô hình dựa trên BERT được dùng để tạo bản tóm tắt tự động, giữ lại ý chính mà vẫn đảm bảo độ mạch lạc.

Tìm kiếm thông tin và hệ thống gợi ý

- **Tìm kiếm thông tin (Information Retrieval):** Google đã áp dụng BERT để cải thiện công cụ tìm kiếm, giúp hiểu rõ hơn ý định của người dùng, đặc biệt trong các truy vấn ngôn ngữ tự nhiên.
- **Hệ thống gợi ý:** Trong thương mại điện tử hoặc nền tảng học tập, mô hình dự đoán và đề xuất sản phẩm/dịch vụ dựa trên ngữ cảnh trong truy vấn của người dùng.

Ứng dụng chuyên ngành

- **Y tế:** Mô hình MLM hỗ trợ phân tích hồ sơ bệnh án, trích xuất thông tin quan trọng từ tài liệu y khoa (BioBERT, ClinicalBERT).
- **Pháp lý:** Dùng để phân tích hợp đồng, phát hiện điều khoản quan trọng hoặc bất thường.
- **Tài chính:** Xử lý báo cáo tài chính, dự đoán xu hướng thị trường dựa trên tin tức.

4.2. Next Sentence Prediction (NSP)

4.2.1 Khái niệm và ý nghĩa

Khái niệm:
Next Sentence Prediction (NSP) là một trong hai tác vụ huấn luyện trước (pre-training tasks) của mô hình BERT, bên cạnh Masked Language Modeling (MLM). Mục tiêu của NSP là giúp mô hình hiểu **mối quan hệ giữa hai câu liên tiếp** – liệu câu thứ hai có thực sự là **tiếp theo của câu thứ nhất trong văn bản gốc** hay không.

Ví dụ minh họa:

Câu A: "Tôi đến thư viện để học."
Câu B: "Tôi đọc sách suốt buổi chiều." → ***Cặp đúng (IsNext)***
Câu B': "Trời hôm nay thật đẹp." → ***Cặp sai (NotNext)***

Ý nghĩa:

Tác vụ **Next Sentence Prediction (NSP)** trong mô hình BERT có ý nghĩa quan trọng trong việc giúp mô hình hiểu được **mối quan hệ ngữ nghĩa giữa các câu trong văn bản**. Trong khi Masked Language Modeling (MLM) giúp BERT học ngữ cảnh ở mức từ, thì NSP mở rộng khả năng này lên mức câu, cho phép mô hình nhận biết xem **một câu có thực sự nối tiếp và liên quan đến câu trước đó hay không**. Nhờ đó, BERT không chỉ hiểu nghĩa của từng từ riêng lẻ mà còn nắm bắt được **logic và sự liên kết giữa các câu**, giúp cải thiện hiệu suất ở các bài toán yêu cầu xử lý ngữ cảnh đa câu như **trả lời câu hỏi (Question Answering)**, **suy luận ngữ nghĩa (Natural Language Inference)** hay **so khớp cặp câu (Sentence Pair Classification)**. Nói cách khác, NSP đóng vai trò giúp BERT phát triển khả năng "hiểu" văn bản một cách tự nhiên hơn, tương tự như cách con người liên kết ý nghĩa giữa các câu khi đọc. Tuy nhiên, trong các mô hình sau này như RoBERTa, tác vụ NSP được loại bỏ do không mang lại cải thiện rõ rệt trong nhiều tác vụ thực tế, song về mặt khái niệm, NSP vẫn được xem là **một bước quan trọng trong quá trình hình thành khả năng hiểu ngữ cảnh sâu của BERT**.

4.2.2 Cơ chế hoạt động

Cơ chế hoạt động của NSP gồm 5 bước:

Bước 1: Tạo cặp câu huấn luyện (Sentence A – Sentence B)

Trong quá trình huấn luyện, dữ liệu đầu vào của mô hình gồm **hai câu được ghép thành một cặp**:

- **50% trường hợp**, câu thứ hai (**Sentence B**) là **câu thật sự nối tiếp** với câu đầu (**Sentence A**) trong văn bản gốc.
  → Cặp câu này được gán nhãn là **"IsNext"**.
- **50% còn lại**, câu thứ hai (**Sentence B**) được chọn **ngẫu nhiên từ một nơi khác** trong tập dữ liệu, không liên quan gì đến câu A.
  → Cặp này được gán nhãn là **"NotNext"**.

Mục đích:

Mô hình học cách **phân biệt khi nào hai câu có quan hệ ngữ cảnh thực sự**, và khi nào thì chúng hoàn toàn độc lập.

Sentence A	Sentence B	Nhãn

Tôi đi học vào buổi sáng.	Trường của tôi nằm ở trung tâm thành phố.	IsNext
Tôi đi học vào buổi sáng.	Con mèo đang ngủ trên ghế sofa.	NotNext

Như vậy, dữ liệu huấn luyện của NSP được chia đôi giữa hai loại cặp câu này để mô hình không bị lệch về một hướng nhất định.

Bước 2: Chuẩn bị đầu vào cho mô hình

Sau khi chọn được hai câu, BERT sẽ **nối hai câu lại thành một chuỗi duy nhất** để đưa vào mô hình.

Chuỗi này được đánh dấu bằng **các token đặc biệt**:

- `[CLS]`: Đánh dấu **đầu chuỗi**, dùng để đại diện cho toàn bộ cặp câu.
- `[SEP]`: Dùng để **phân tách** giữa hai câu.

Ví dụ về chuỗi đầu vào:

```
[CLS] Tôi đi học vào buổi sáng. [SEP] Trường của tôi nằm ở trung tâm thành phố. [SEP]
```

Ngoài ra, BERT còn có thêm **segment embedding** để mô hình biết từ nào thuộc câu A, từ nào thuộc câu B:

- Các token thuộc **Sentence A** được gán nhãn là **Segment A (0)**.
- Các token thuộc **Sentence B** được gán nhãn là **Segment B (1)**.

Nhờ đó, mô hình hiểu rõ ranh giới giữa hai câu và học được **mối quan hệ ngữ nghĩa giữa chúng**.

Bước 3: Mã hóa bằng Transformer Encoder

Khi chuỗi đã được mã hóa với token và embedding, BERT sẽ đưa toàn bộ dữ liệu qua **một chuỗi các lớp Transformer Encoder**.

Trong quá trình này:

- Mỗi token "nhìn thấy" toàn bộ các token khác nhờ **cơ chế Self-Attention**.
- Mô hình học được **ngữ cảnh toàn cục**: tức là không chỉ hiểu nội dung từng câu, mà còn học **mối quan hệ giữa câu trước và câu sau**.

Đặc biệt, token `[CLS]` đóng vai trò rất quan trọng — nó được xem như **đại diện cho toàn bộ cặp câu**.

Sau khi qua toàn bộ các lớp Transformer, vector đầu ra của `[CLS]` chính là **tóm tắt thông tin của cả Sentence A và Sentence B**.

Bước 4: Dự đoán mối quan hệ giữa hai câu

Cuối cùng, vector của `[CLS]` sẽ được đưa qua một **mạng phân loại (classification layer)**.

Mạng này sẽ dự đoán xem:

- Hai câu **có mối quan hệ liên tiếp hợp lý → IsNext**
- Hay là **không liên quan đến nhau → NotNext**

Ví dụ:

- "Tôi ăn sáng xong thì đi làm." → "Tôi đến công ty lúc 8 giờ."
  → Dự đoán: **IsNext**
- "Tôi ăn sáng xong thì đi làm." → "Con mèo của tôi rất dễ thương."
  → Dự đoán: **NotNext**

Bước 5: Quá trình học lặp lại

Quá trình này được **lặp lại hàng triệu lần** với vô số cặp câu khác nhau trong kho dữ liệu lớn (Wikipedia, BooksCorpus,…).

Mỗi lần như vậy, mô hình sẽ:

- Cập nhật trọng số (weights) để cải thiện khả năng nhận diện quan hệ giữa hai câu.
- Dần dần "học" được cách **nhận biết khi nào các câu có sự liên kết tự nhiên trong văn bản**.

4.2.3 Vai trò của NSP trong hiệu suất mô hình

Cơ chế **Next Sentence Prediction (NSP)** đóng vai trò quan trọng trong việc nâng cao hiệu suất của mô hình BERT, đặc biệt là khả năng hiểu ngữ cảnh giữa các câu trong một đoạn văn. Trước khi có NSP, các mô hình ngôn ngữ chủ yếu chỉ học được mối quan hệ giữa các từ trong cùng một câu, dẫn đến hạn chế trong việc nắm bắt ngữ nghĩa toàn đoạn. NSP được thiết kế để giải quyết vấn đề này bằng cách huấn luyện mô hình dự đoán xem một câu có thực sự theo sau câu trước đó trong văn bản gốc hay không.

Nhờ cơ chế này, BERT học được cách phân biệt giữa các cặp câu có liên kết logic và những cặp câu không liên quan, giúp mô hình hiểu rõ hơn các mối quan hệ ngữ nghĩa, nhân quả hay đối lập giữa các câu. Điều này đặc biệt hữu ích trong các tác vụ như **hỏi – đáp (Question Answering)** hay **suy luận ngôn ngữ tự nhiên (Natural Language Inference)**, nơi việc xác định mối liên kết giữa hai câu hoặc hai đoạn văn là yếu tố quyết định độ chính xác của mô hình. Ngoài ra, NSP còn giúp cải thiện chất lượng biểu diễn ngữ nghĩa ở cấp độ câu (sentence-level embedding), hỗ trợ mô hình ổn định và hội tụ nhanh hơn trong giai đoạn **fine-tuning**, nhờ đã được huấn luyện trước để hiểu mối quan hệ giữa các đoạn văn bản.

Tuy nhiên, một số nghiên cứu sau này, như **RoBERTa (Facebook AI, 2019)**, cho thấy khi quy mô dữ liệu huấn luyện đủ lớn, việc loại bỏ NSP không làm giảm mà thậm chí còn cải thiện hiệu suất mô hình, do tránh được sự phân tán trong quá trình học. Mặc dù vậy, NSP vẫn được xem là một thành phần có giá trị trong việc hình thành nền tảng giúp BERT hiểu ngữ cảnh sâu và học được mối quan hệ giữa các câu, đóng góp quan trọng vào thành công của BERT trong nhiều tác vụ xử lý ngôn ngữ tự nhiên.

4.2.4 Giới hạn của NSP

Mặc dù nhiệm vụ Next Sentence Prediction (NSP) mang lại nhiều lợi ích trong việc giúp mô hình BERT hiểu được mối quan hệ giữa các câu, nhưng các nghiên cứu sau này cho thấy cơ chế này vẫn tồn tại một số hạn chế đáng kể.

Thứ nhất, NSP có thể cung cấp tín hiệu học khá “nông” (shallow semantic signal), vì việc phân biệt hai câu có liên kết hay không đôi khi không đòi hỏi mô hình phải thật sự hiểu ngữ nghĩa sâu của chúng.

Điều này khiến mô hình dễ học theo hướng ghi nhớ thống kê bề mặt hơn là nắm bắt mối quan hệ ngữ cảnh thực sự.

Thứ hai, nhiệm vụ NSP có thể làm chậm quá trình huấn luyện vì thêm một mục tiêu phụ không quá mạnh, dẫn đến việc phân tán tài nguyên tính toán mà không mang lại cải thiện đáng kể về hiệu suất. Theo báo cáo của Facebook AI trong mô hình **RoBERTa (2019)**, việc loại bỏ hoàn toàn NSP không những không làm giảm hiệu quả mà còn giúp mô hình đạt kết quả tốt hơn trong nhiều bài toán như *Question Answering* hay *Natural Language Inference*.

Cuối cùng, NSP không phản ánh chính xác sự đa dạng của các quan hệ ngữ nghĩa giữa các câu trong ngữ cảnh thực tế (chẳng hạn như nguyên nhân – kết quả, so sánh, hoặc điều kiện), vì nó chỉ đơn giản hóa bài toán thành một lựa chọn nhị phân "tiếp theo hoặc không".

Chính vì vậy, các mô hình cải tiến sau này như **ALBERT** hay **ELECTRA** đã thay thế NSP bằng nhiệm vụ khác như **Sentence Order Prediction (SOP)**, giúp mô hình học được thứ tự tự nhiên giữa các câu hiệu quả hơn.

4.2.5 Tổng kết

Cơ chế **Next Sentence Prediction (NSP)** là một trong hai thành phần huấn luyện quan trọng của BERT, cùng với **Masked Language Modeling (MLM)**. Trong khi MLM giúp mô hình học được mối quan hệ ngữ nghĩa giữa các từ trong câu, thì NSP lại cho phép BERT hiểu sâu hơn mối liên kết giữa các câu trong cùng một đoạn văn. Cơ chế này giúp mô hình không chỉ dừng lại ở việc dự đoán từ còn thiếu, mà còn có khả năng nhận biết xem hai câu có mối quan hệ liền mạch về mặt ngữ nghĩa hay không.

Mặc dù các nghiên cứu sau này chỉ ra rằng NSP có thể chưa tối ưu trong việc mô hình hóa quan hệ câu, nhưng trong giai đoạn phát triển của BERT, NSP vẫn đóng vai trò quan trọng trong việc hình thành khả năng hiểu ngữ cảnh ở cấp độ cao hơn. Chính nhờ sự kết hợp giữa MLM và NSP, BERT đã đạt được bước đột phá trong việc nắm bắt ngữ nghĩa ngôn ngữ tự nhiên, trở thành nền tảng cho hàng loạt mô hình xử lý ngôn ngữ tiên tiến ra đời sau này.

Sau khi hoàn thành giai đoạn tiền huấn luyện (pre-training) với hai cơ chế chính là **Masked Language Modeling (MLM)** và **Next Sentence Prediction (NSP)**, mô hình BERT đã học được kiến thức ngữ pháp, ngữ nghĩa và mối quan hệ giữa các câu trong ngôn ngữ tự nhiên. Tuy nhiên, để BERT có thể áp

dụng hiệu quả vào các tác vụ cụ thể như **phân loại văn bản, trả lời câu hỏi, hay phân tích cảm xúc**, mô hình cần trải qua giai đoạn **fine-tuning**. Đây là bước tinh chỉnh lại các trọng số đã học từ giai đoạn pre-training, giúp BERT thích nghi và đạt hiệu suất cao nhất đối với từng bài toán cụ thể.

4.3. Fine-tuning

4.3.1 Khái niệm và ý nghĩa

Khái niệm:
Fine-tuning là giai đoạn điều chỉnh mô hình sau khi đã được huấn luyện trước (pre-training) trên tập dữ liệu ngôn ngữ lớn. Trong mô hình BERT, quá trình pre-training giúp mô hình học được các đặc trưng ngữ nghĩa và ngữ pháp chung của ngôn ngữ, nhưng những kiến thức này chỉ mang tính tổng quát. Do đó, để áp dụng vào các tác vụ cụ thể như **phân loại cảm xúc (Sentiment Analysis)**, **trả lời câu hỏi (Question Answering)** hay **suy luận ngữ nghĩa (Natural Language Inference)**, mô hình cần được tinh chỉnh lại. Fine-tuning chính là quá trình tiếp tục huấn luyện mô hình BERT trên **tập dữ liệu nhỏ hơn nhưng có gán nhãn**, nhằm giúp mô hình thích ứng với đặc điểm và yêu cầu của bài toán cụ thể.

Ý nghĩa:
Fine-tuning đóng vai trò then chốt trong việc biến BERT từ một **mô hình ngôn ngữ tổng quát** thành một **công cụ mạnh mẽ cho các tác vụ NLP cụ thể**. Nhờ quá trình này, BERT có thể nhanh chóng thích nghi và đạt hiệu suất cao trong nhiều ứng dụng khác nhau chỉ với một lượng dữ liệu huấn luyện nhỏ. Fine-tuning giúp người dùng tận dụng tri thức ngôn ngữ sâu rộng đã được học trong giai đoạn pre-training, từ đó **tối ưu hóa hiệu quả học máy**, **giảm chi phí**, và **tăng độ chính xác** trong các tác vụ thực tế như phân tích cảm xúc, tóm tắt văn bản, hay nhận diện thực thể. Có thể xem fine-tuning là **cầu nối giữa nghiên cứu và ứng dụng**, giúp BERT không chỉ hiểu ngôn ngữ ở mức lý thuyết mà còn có khả năng áp dụng thực tế linh hoạt và hiệu quả.

4.3.2. Cơ chế hoạt động của Fine-tuning

Trong giai đoạn **fine-tuning**, mô hình BERT đã được tiền huấn luyện (pre-trained) sẽ được **tinh chỉnh lại toàn bộ các tham số (parameters)** dựa trên tập dữ liệu nhỏ hơn nhưng có gán nhãn phù hợp với tác vụ cụ thể.

Khác với pre-training, giai đoạn này **không huấn luyện từ đầu**, mà chỉ **cập nhật lại trọng số** của mô hình đã học sẵn để nó thích ứng với đặc trưng của bài toán.

Quy trình fine-tuning trong BERT thường bao gồm các bước chính:

1. **Chuẩn bị dữ liệu có gán nhãn:**
    a. Dữ liệu được chia thành các cặp *input – output* tương ứng với tác vụ.
    b. Ví dụ: trong bài toán phân loại cảm xúc, đầu vào là câu văn, đầu ra là nhãn "tích cực" hoặc "tiêu cực".
2. **Thêm lớp đầu ra (output layer):**
    a. BERT gốc được giữ nguyên, chỉ thêm một **lớp phân loại (classification layer)** phía trên — thường là một lớp Dense (Fully Connected) với số nút đầu ra tương ứng số nhãn cần dự đoán.
    b. Đầu ra đặc trưng `[CLS]` từ BERT được dùng làm **đại diện cho toàn bộ câu** và truyền vào lớp này.
3. **Huấn luyện toàn mô hình:**
    a. Cả mô hình (BERT + lớp phân loại) đều được tinh chỉnh đồng thời, với tốc độ học nhỏ hơn nhiều so với pre-training (ví dụ: `learning rate = 2e-5`).
    b. Mô hình học cách liên kết kiến thức ngôn ngữ tổng quát với đặc trưng của bài toán cụ thể.
4. **Đánh giá và tối ưu:**
    a. Sau khi huấn luyện, mô hình được kiểm thử trên tập dữ liệu chưa thấy (test set).
    b. Các tham số như *batch size*, *learning rate* hay *epoch* được điều chỉnh để đạt kết quả tối ưu.

Nhờ quy trình này, BERT có thể nhanh chóng thích nghi với nhiều tác vụ khác nhau mà không cần huấn luyện lại từ đầu.

4.3.3. Ứng dụng của Fine-tuning trong các tác vụ NLP

Fine-tuning giúp BERT được áp dụng linh hoạt vào nhiều bài toán xử lý ngôn ngữ tự nhiên (NLP), tiêu biểu như:

- **Phân loại văn bản (Text Classification):**
  BERT được fine-tuning để xác định chủ đề, cảm xúc hoặc loại văn bản (ví dụ: tích cực / tiêu cực trong đánh giá sản phẩm).
- **Trả lời câu hỏi (Question Answering):**
  Mô hình được tinh chỉnh để xác định vị trí đoạn văn chứa câu trả lời trong một đoạn văn bản dài, như trong bộ dữ liệu SQuAD.
- **Suy luận ngôn ngữ tự nhiên (Natural Language Inference – NLI):**
  Fine-tuning giúp BERT xác định mối quan hệ giữa hai câu — ví dụ: câu B có mâu thuẫn, suy ra, hay trung lập với câu A.
- **Nhận dạng thực thể có tên (Named Entity Recognition – NER):**
  Mô hình được huấn luyện để xác định các thực thể như tên người, địa điểm, tổ chức trong câu.
- **Tóm tắt văn bản, dịch máy, và sinh ngôn ngữ tự nhiên (Summarization, Translation, Generation):**
  Dù BERT không phải mô hình sinh (generative model), fine-tuning kết hợp với các kiến trúc khác giúp mô hình tham gia các tác vụ này.

4.3.4. Tổng kết quá trình Fine-tuning

Tổng kết lại, **fine-tuning** là giai đoạn mang tính quyết định trong toàn bộ quy trình huấn luyện của mô hình BERT. Nếu như **pre-training** giúp mô hình học được những tri thức ngôn ngữ tổng quát — bao gồm ngữ pháp, ngữ nghĩa, và mối quan hệ giữa các từ, câu trong ngôn ngữ — thì **fine-tuning** chính là bước biến những tri thức đó thành khả năng ứng dụng thực tế. Thông qua việc tinh chỉnh các tham số đã được học từ trước trên tập dữ liệu có gán nhãn phù hợp với từng tác vụ cụ thể, BERT có thể thích nghi nhanh chóng và đạt hiệu suất vượt trội mà không cần huấn luyện lại từ đầu.

Điểm mạnh nổi bật của fine-tuning nằm ở khả năng **tái sử dụng tri thức ngôn ngữ tổng quát** từ giai đoạn pre-training. Điều này giúp tiết kiệm đáng kể **thời gian, tài nguyên tính toán và dữ liệu huấn luyện**, đồng thời vẫn đảm bảo độ chính xác cao trong các tác vụ như phân tích cảm xúc, trả lời câu hỏi, suy luận ngữ nghĩa, hay nhận dạng thực thể. Ngoài ra, fine-tuning còn mang lại **tính linh hoạt cao**, cho

phép cùng một mô hình BERT có thể được điều chỉnh và tái sử dụng cho nhiều mục tiêu khác nhau chỉ bằng cách thay đổi lớp đầu ra.

Từ góc độ tổng thể, fine-tuning không chỉ giúp BERT phát huy toàn bộ tiềm năng của quá trình tiền huấn luyện mà còn mở ra **một hướng tiếp cận mới trong lĩnh vực học chuyển giao (Transfer Learning)** cho xử lý ngôn ngữ tự nhiên. Nhờ cơ chế này, BERT đã trở thành nền tảng cho hàng loạt mô hình kế thừa và cải tiến sau này như **RoBERTa, ALBERT, DistilBERT**, góp phần định hình nên kỷ nguyên mới của các mô hình ngôn ngữ thông minh. Có thể khẳng định rằng, chính giai đoạn fine-tuning đã biến BERT từ một mô hình hiểu ngôn ngữ thành một công cụ mạnh mẽ có khả năng **ứng dụng thực tiễn sâu rộng và linh hoạt trong nhiều lĩnh vực NLP hiện đại**.

$$K = \begin{bmatrix} 0.3 & 0.7 & 0.2 & 0.6 \\ 0.5 & 0.2 & 0.8 & 0.4 \\ 0.6 & 0.3 & 0.5 & 0.7 \\ 0.4 & 0.9 & 0.1 & 0.2 \end{bmatrix}$$

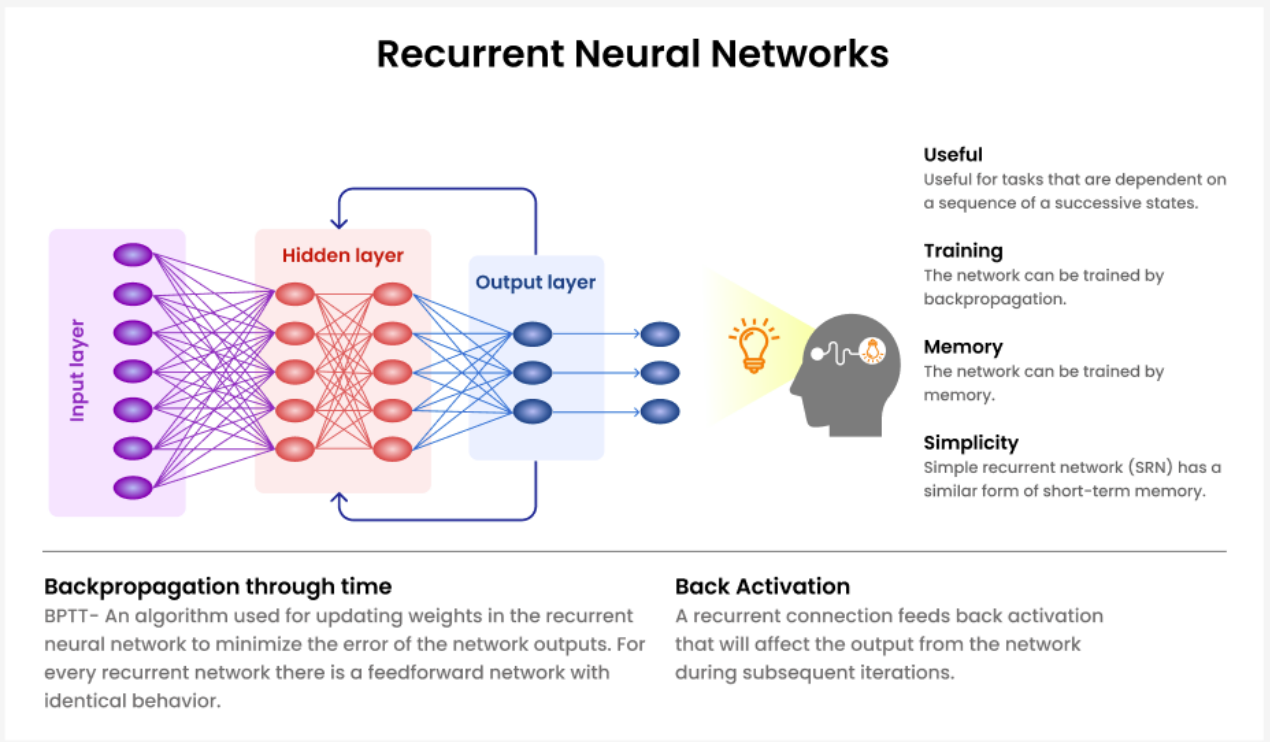
$$V = \begin{bmatrix} 0.7 & 0.2 & 0.5 & 0.1 \\ 0.3 & 0.6 & 0.4 & 0.8 \\ 0.5 & 0.3 & 0.9 & 0.7 \\ 0.2 & 0.8 & 0.6 & 0.4 \end{bmatrix}$$

CHƯƠNG 5: GPT – MÔ HÌNH SINH VĂN BẢN TỰ HỒI QUY

5.1. Nguyên lý tự hồi quy trong mô hình ngôn ngữ

5.1.1. Định nghĩa và vai trò của tự hồi quy

Tự hồi quy (Autoregressive) là một khái niệm thống kê chỉ mô hình trong đó giá trị hiện tại của một chuỗi thời gian được mô hình hóa như một hàm tuyến tính hoặc phi tuyến của các giá trị quá khứ. Trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), mô hình tự hồi quy được áp dụng để mô hình hóa xác suất của một từ trong văn bản dựa trên các từ đã xuất hiện trước đó trong câu.



Mô hình tự hồi quy trong NLP hoạt động dựa trên nguyên lý cốt lõi: xác suất của một chuỗi từ được tính bằng tích của các xác suất có điều kiện. Cụ thể, cho một chuỗi từ $W = (w_1, w_2, ..., w_n)$, xác suất của chuỗi này được biểu diễn:

$$P(W) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_1,w_2) \times ... \times P(w_n|w_1,w_2,...,w_{n-1})$$

Định nghĩa này phản ánh cách con người tiếp nhận và sinh ngôn ngữ tự nhiên - chúng ta thường hiểu và dự đoán từ tiếp theo dựa trên ngữ cảnh đã biết từ các từ trước đó. Mô hình tự hồi quy tận dụng đặc tính này để học cách sinh văn bản một cách tự nhiên và mạch lạc.

Vai trò của tự hồi quy trong mô hình ngôn ngữ vô cùng quan trọng:

- Sinh văn bản tự nhiên: Cho phép mô hình tạo ra văn bản mới bằng cách dự đoán từng từ một cách tuần tự, tạo thành các câu và đoạn văn có ý nghĩa.
- Dự đoán ngữ cảnh: Khả năng hiểu và dự đoán ngữ cảnh dựa trên các từ đã xuất hiện, giúp mô hình đưa ra dự đoán chính xác về từ tiếp theo.
- Học không giám sát hiệu quả: Tự hồi quy cho phép mô hình học từ dữ liệu văn bản thô mà không cần nhãn, vì mỗi từ trong câu đều có thể được sử dụng làm "mục tiêu" để dự đoán.

Nhiều bài toán NLP khác nhau - từ phân loại văn bản, trả lời câu hỏi đến dịch máy - đều có thể được biểu diễn dưới dạng dự đoán chuỗi hoặc hưởng lợi từ một mô hình ngôn ngữ tự hồi quy mạnh đã được tiền huấn luyện. Đặc biệt, sê-ri mô hình GPT của OpenAI là những ví dụ tiêu biểu cho mô hình tự hồi quy, đã thể hiện hiệu quả vượt trội trong nhiều ứng dụng sinh ngôn ngữ tự nhiên

5.1.2. Cơ chế dự đoán từ tiếp theo trong autoregressive

Cơ chế dự đoán từ tiếp theo trong mô hình autoregressive dựa trên việc mã hóa ngữ cảnh hiện tại và sử dụng nó để tính toán xác suất của mỗi từ trong từ vựng. Quá trình này diễn ra theo các bước sau:

Bước 1: Mã hóa ngữ cảnh

Mô hình nhận đầu vào là chuỗi từ đã xuất hiện (w_1, w_2, ..., w_t) và chuyển đổi chúng thành các vector biểu diễn thông qua một lớp embedding. Các vector này sau đó được xử lý qua nhiều lớp transformer để tạo ra biểu diễn ngữ cảnh có chiều sâu.

Bước 2: Tính toán xác suất

Từ biểu diễn ngữ cảnh hiện tại h_t, mô hình tính toán xác suất có điều kiện cho mỗi từ trong từ vựng V:

$P(w_{t+1} = v_i \mid h_t) = \exp(f(h_t, v_i)) / \Sigma_j \exp(f(h_t, v_j))$

Trong đó $f(h_t, v_i)$ là điểm số được tính bởi mạng neural cho từ v_i dựa trên ngữ cảnh h_t.

Bước 3: Lấy mẫu hoặc chọn từ

Có hai cách chọn từ tiếp theo:

- Greedy decoding: Chọn từ có xác suất cao nhất

- Stochastic sampling: Lấy mẫu từ phân phối xác suất để tạo ra văn bản đa dạng hơn

  Ví dụ minh họa:

  Giả sử mô hình đã nhận được câu "Hôm nay thời tiết" và cần dự đoán từ tiếp theo. Mô hình sẽ:

a) Mã hóa ngữ cảnh "Hôm nay thời tiết" thành vector biểu diễn
b) Tính xác suất cho các từ khả dĩ: "đẹp" (0.6), "xấu" (0.2), "mưa" (0.15), "nắng" (0.05)
c) Chọn từ dựa trên chiến lược decoding (greedy sẽ chọn "đẹp")

5.1.3. Ưu điểm và hạn chế của mô hình autoregressive

Ưu điểm:

- Sinh văn bản chất lượng cao: Mô hình autoregressive có khả năng tạo ra văn bản mạch lạc, tự nhiên và có ngữ cảnh cao, vì nó được huấn luyện để tối ưu hóa xác suất của toàn bộ chuỗi.
- Không cần dữ liệu có nhãn: Có thể huấn luyện không giám sát từ dữ liệu văn bản thô, giảm chi phí chuẩn bị dữ liệu.
- Linh hoạt trong sinh văn bản: Có thể điều khiển quá trình sinh thông qua các tham số như temperature và top-k sampling để tạo ra văn bản đa dạng hoặc tập trung.
- Hiểu ngữ cảnh dài hạn: Các mô hình transformer-based autoregressive như GPT có khả năng mô hình hóa các phụ thuộc dài hạn trong văn bản.

Hạn chế:

- Tốc độ sinh chậm: Do phải sinh từng từ một cách tuần tự, không thể parallel hóa hoàn toàn quá trình sinh, dẫn đến tốc độ chậm hơn so với các mô hình song song.
- Lỗi khuếch đại: Lỗi trong các bước đầu có thể được khuếch đại và ảnh hưởng đến toàn bộ chuỗi sinh ra, dẫn đến văn bản không mạch lạc hoặc sai lệch.
- Tiêu tốn tài nguyên tính toán: Cần lượng lớn tham số và bộ nhớ để lưu trữ trạng thái ngữ cảnh, đặc biệt với các mô hình lớn như GPT-3.
- Khó kiểm soát nội dung: Việc sinh từng từ một cách tự động có thể khiến mô hình tạo ra nội dung không mong muốn hoặc không phù hợp với yêu cầu cụ thể.

5.2. Kiến trúc tổng quan của GPT và Transformer

5.2.1. Mối quan hệ giữa GPT và Transformer Decoder

Mô hình GPT (Generative Pre-trained Transformer) được xây dựng dựa trên kiến trúc Transformer, cụ thể hơn là phần decoder của Transformer. Trong kiến trúc Transformer gốc【258†】 (Vaswani et al., 2017), mô hình bao gồm hai khối chính: bộ mã hóa (encoder) và bộ giải mã (decoder). GPT loại bỏ hoàn toàn phần encoder, chỉ sử dụng các lớp decoder theo cơ chế tự hồi quy để sinh văn bản. Mỗi lớp decoder trong GPT đảm nhận việc tiếp nhận ngữ cảnh (các token trước đó) và dự đoán token kế tiếp, giống như decoder trong mô hình dịch máy nhưng không có encoder tương ứng. Nói cách khác, GPT chính là một mô hình Transformer "decoder-only", tức chỉ gồm chuỗi các khối giải mã Transformer được xếp chồng lên nhau, hoạt động theo cơ chế dự đoán tuần tự một chiều (từ trái sang phải)

Vì sử dụng cấu trúc decoder, GPT thừa hưởng đầy đủ những thành phần kiến trúc tinh túy của Transformer decoder: bao gồm cơ chế tự chú ý đa đầu có mặt nạ, các mạng Feed-Forward ở mỗi lớp, cùng với kết nối tắt (residual) và chuẩn hóa lớp. Tuy nhiên, do không có encoder độc lập, phần cross-attention (chú ý đến trạng thái encoder) trong các lớp decoder GPT được loại bỏ – mỗi lớp chỉ còn lại hai phần là self-attention và feed-forward. Cách thiết kế này giúp mô hình tập trung hoàn toàn vào việc mô hình hóa phân phối ngôn ngữ một chiều (causal language modeling). Thực tiễn cho thấy kiến trúc decoder-only như GPT rất hiệu quả cho các nhiệm vụ sinh văn bản, khi mô hình chỉ cần nhìn lịch sử (ngữ cảnh trước) để tạo ra nội dung kế tiếp. So sánh với các biến thể Transformer khác: BERT là encoder-only (chỉ mã hóa bidirectional, không phù hợp sinh văn bản tự do), trong khi GPT là decoder-only (phù hợp sinh văn bản nhưng không thu nhận ngữ cảnh hai chiều). Tóm lại, GPT chính là một trường hợp áp dụng đặc thù của Transformer, trong đó toàn bộ mô hình chính là một Transformer decoder nhiều tầng, được huấn luyện theo mục tiêu tự hồi quy để sinh ngôn ngữ.

5.2.2. Token-by-token Decoding trong mô hình sinh

Quá trình giải mã từng token một (token-by-token decoding) là đặc trưng của mô hình GPT khi sinh văn bản. Do GPT là mô hình autoregressive, nó tạo đầu ra theo trình tự từng bước: tại mỗi bước, mô hình xem xét chuỗi token đầu vào (bao gồm cả prompt ban đầu và các token đã sinh ra trước đó), rồi dự đoán token kế tiếp. Cơ chế này nghĩa là mỗi lần suy luận mô hình GPT chỉ sinh được một token, sau đó token này được gắn vào cuối chuỗi đầu vào để mô hình tiếp tục dự đoán

token tiếp theo ở bước kế tiếp. Việc giải mã tuần tự này tiếp tục đến khi đạt độ dài yêu cầu hoặc gặp ký hiệu kết thúc.

Khi áp dụng vào thực tế, quá trình giải mã token-by-token cho phép kiểm soát sát sao việc sinh nội dung. Chẳng hạn, ta có thể áp dụng các thuật toán tìm kiếm/tạo mẫu mỗi khi chọn token tiếp theo: có thể chọn tham lam (greedy) token xác suất cao nhất, hoặc lấy mẫu ngẫu nhiên dựa trên phân phối xác suất (với tham số nhiệt độ, top-k, top-p) để nội dung đa dạng hơn. Mỗi quyết định sinh một token sẽ ảnh hưởng đến toàn bộ ngữ cảnh cho các bước sau, do đó mô hình luôn điều chỉnh linh hoạt theo những gì nó đã sinh ra. Ưu điểm của decoding tuần tự là giúp mô hình GPT theo sát ngữ cảnh và duy trì logic xuyên suốt đoạn văn – vì tại bước nào mô hình cũng chỉ căn cứ vào quá khứ (đã cố định) chứ không bị ảnh hưởng bởi thông tin "nhìn trước" nào khác.

Tuy nhiên, decoding token-by-token cũng đồng nghĩa với chi phí tính toán tăng tuyến tính theo độ dài văn bản đầu ra. Mặc dù mỗi bước dự đoán của GPT có thể song song hóa tính toán bên trong (nhờ kiến trúc Transformer), ta vẫn phải thực hiện lần lượt từng bước cho mỗi token, do đó thời gian suy luận tăng theo chiều dài đầu ra. Dẫu vậy, chính nhờ phương thức tạo chuỗi từng token mà GPT và các mô hình tương tự có khả năng sinh văn bản dài với nội dung phong phú, vì chúng có thể tự điều chỉnh theo ngữ cảnh sinh ra một cách tự nhiên, tương tự như con người viết từng chữ một. Đây là điểm khác biệt cơ bản so với các mô hình sinh khác (ví dụ RNN cũ vốn sinh tuần tự nhưng khó parallel, hay mô hình encoder-decoder có thể tạo đồng loạt nhưng chỉ áp dụng khi chiều dài đầu ra cố định như trong dịch máy). Nói chung, token-by-token decoding là cơ chế tất yếu để GPT thực hiện mục tiêu mô hình hóa ngôn ngữ tự nhiên – tạo văn bản tự do và linh hoạt theo ngữ cảnh.

5.2.3. Masked Multi-Head Self-Attention, Layer Norm & Residual Connection

Trong mỗi lớp decoder của GPT: đầu vào đi qua masked multi-head self-attention (với LayerNorm và residual đi kèm), sau đó qua mạng FFN (cũng kèm LayerNorm và residual). Nhờ kiến trúc này, mô hình có thể học hiệu quả quan hệ giữa các token trong ngữ cảnh (qua attention) đồng thời duy trì tính ổn định và biểu diễn đa dạng (qua residual connections và LayerNorm). Đây chính là bí quyết kiến trúc giúp GPT trở thành mô hình sinh ngôn ngữ mạnh mẽ, học được phụ thuộc dài và sinh văn bản lưu loát.

5.3. Các phương pháp huấn luyện mô hình ngôn ngữ

5.3.1. Pre-training không giám sát (Unsupervised Pre-training)

Tiền huấn luyện không giám sát là giai đoạn đầu tiên và cốt lõi trong việc huấn luyện các mô hình ngôn ngữ hiện đại như GPT. Ở bước này, mô hình được huấn luyện trên một tập dữ liệu văn bản rất lớn và không có nhãn, mục tiêu là để mô hình học được biểu diễn ngôn ngữ tổng quát thông qua nhiệm vụ dự đoán chính các dữ liệu đó. Cụ thể, với GPT, nhiệm vụ tiền huấn luyện là mô hình hóa ngôn ngữ tự nhiên: mô hình được cung cấp các đoạn văn bản từ tập corpus lớn và học cách dự đoán token kế tiếp dựa trên các token trước đó (objective của mô hình tự hồi quy). Quá trình này hoàn toàn không giám sát vì dữ liệu không cần nhãn mục tiêu bên ngoài – bản thân mỗi từ trong chuỗi đóng vai trò như một "nhãn" cho ngữ cảnh phía trước nó. Radford và cộng sự (2018) đã tiến hành tiền huấn luyện GPT-1 trên một tập dữ liệu đa dạng (BooksCorpus ~800 triệu từ) bằng cách tối ưu mục tiêu ngôn ngữ tự hồi quy nói trên. Kết quả, sau quá trình tiền huấn luyện, mô hình đã học được các tham số ban đầu thể hiện kiến thức ngôn ngữ phong phú, sẵn sàng để áp dụng cho các nhiệm vụ cụ thể khác.

Mục đích chính của unsupervised pre-training là giúp mô hình tích lũy "kiến thức nền" về ngôn ngữ: từ ý nghĩa từ vựng, cú pháp, đến các mẫu ngữ cảnh, thông tin thực tế,... Quá trình này giống như việc mô hình đọc rất nhiều sách và tự hình thành hiểu biết trước khi được giao làm bài tập cụ thể. Ưu điểm lớn là tận dụng được khối lượng đồ sộ văn bản không gán nhãn có sẵn (trên internet, Wikipedia, sách báo, ...) để huấn luyện, thay vì đòi hỏi tập dữ liệu có nhãn nhỏ và đắt đỏ. Các nghiên cứu đã chỉ ra tiền huấn luyện trên dữ liệu lớn mang lại cải thiện vượt trội: mô hình GPT-2 (2019) được tiền huấn luyện trên 40GB văn bản WebText đã đạt trạng thái-of-the-art trên nhiều benchmark mô hình ngôn ngữ mà không cần dữ liệu gán nhãn thêm. Như vậy, pre-training không giám sát thiết lập nền tảng cho mô hình ngôn ngữ, cung cấp bộ tham số khởi tạo rất tốt (thay vì khởi tạo ngẫu nhiên) để từ đó tinh chỉnh cho các nhiệm vụ cụ thể sau này.

Một điểm kỹ thuật quan trọng trong giai đoạn tiền huấn luyện là lựa chọn mục tiêu tự giám sát phù hợp. Đa số các mô hình GPT sử dụng mục tiêu dự đoán token kế tiếp (next-token prediction) – đây là một dạng self-supervised learning phổ biến, vì mô hình tự tạo ra nhãn từ chính dữ liệu. Ngoài ra, có thể kết hợp các mục tiêu khác (như mô hình hóa câu tiếp theo, hay sắp xếp lại đoạn văn) nhưng phương pháp đơn giản dự đoán từ tiếp theo tỏ ra hiệu quả và đủ sức giúp mô hình học được biểu diễn mạnh mẽ. Tóm lại, unsupervised pre-training là bước "học hỏi" chính của mô hình

ngôn ngữ: huấn luyện dài hơi trên dữ liệu thô để hình thành hiểu biết ngôn ngữ nền tảng, tạo tiền đề cho các bước huấn luyện có giám sát ở giai đoạn sau.

5.3.2. Fine-tuning có giám sát (Supervised Fine-tuning)

Sau khi hoàn thành tiền huấn luyện, mô hình GPT sẽ được đưa vào giai đoạn fine-tuning có giám sát nhằm điều chỉnh nó cho từng tác vụ ngôn ngữ cụ thể. Ở bước này, mô hình được huấn luyện tiếp trên một tập dữ liệu có gán nhãn (supervised) tương ứng với nhiệm vụ đích – ví dụ: dữ liệu phân loại cảm xúc (có nhãn tích cực/tiêu cực), dữ liệu hỏi đáp (có câu hỏi và câu trả lời đúng), v.v. Mục tiêu của fine-tuning là tinh chỉnh các tham số của mô hình đã tiền huấn luyện sao cho nó tối ưu với nhiệm vụ cụ thể, trong khi vẫn tận dụng kiến thức ngôn ngữ tổng quát đã học trước đó. Quá trình fine-tuning thường sử dụng một hàm mất mát có giám sát (ví dụ cross-entropy trên nhãn phân loại, hoặc độ đo độ chính xác trả lời) và cập nhật trọng số toàn bộ mô hình (hoặc đôi khi chỉ một phần, tùy kỹ thuật) để mô hình học lên chi tiết phù hợp với task.

Chẳng hạn, với GPT-1, sau khi tiền huấn luyện trên BooksCorpus, nhóm nghiên cứu đã fine-tune mô hình trên từng task của bộ GLUE (như MRPC – phân loại cặp câu, RACE – trả lời đọc hiểu, v.v.) bằng cách thêm một lớp đầu ra phù hợp (ví dụ lớp softmax 2 nhãn cho phân loại) và huấn luyện trên dữ liệu có nhãn của task đó. Kết quả cho thấy mô hình chuyển đổi rất tốt: chỉ cần một lượng nhỏ dữ liệu giám sát, GPT-1 fine-tuned đã vượt qua nhiều mô hình đặc thù trước đó trên 9/12 tác vụ NLU. Điều này chứng tỏ kiến thức mà mô hình tích lũy được qua pre-training thực sự giúp ích lớn cho việc học tác vụ mới (chỉ cần ít dữ liệu giám sát hơn mà vẫn đạt hiệu quả cao).

Về mặt kỹ thuật, fine-tuning đòi hỏi một vài điều chỉnh: (1) Thêm tầng đầu ra phù hợp cho tác vụ (chẳng hạn thêm một feed-forward + softmax cho phân loại, hay điều chỉnh cách mô hình sinh text để phù hợp với định dạng đầu ra mong muốn). (2) Thiết kế đầu vào: có thể cần định dạng lại input để cung cấp ngữ cảnh cho mô hình hiểu tác vụ (ví dụ chèn câu hỏi và câu trả lời mẫu vào prompt để GPT hiểu cần làm gì). (3) Điều chỉnh siêu tham số: thường fine-tune với learning rate nhỏ hơn, có thể dùng early stopping vì dữ liệu ít hơn pre-training nhiều. Radford (2018) đã đề xuất kỹ thuật "task-specific input transformations", tức là chuẩn bị đầu vào dưới dạng một chuỗi văn bản đặc biệt để mô hình GPT có thể tiếp nhận nhiệm vụ mới mà không cần thay đổi kiến trúc. Nhờ đó, quá trình fine-tuning hầu như chỉ yêu cầu thay thế và huấn luyện lại lớp đầu ra cuối cùng (đôi khi gọi là "head" của mô hình) và tinh chỉnh nhẹ toàn bộ tham số, thay vì phải thiết kế lại kiến trúc mô hình cho từng tác vụ.

Nhìn chung, supervised fine-tuning là bước chuyên môn hóa mô hình: từ một mô hình ngôn ngữ tổng quát, GPT được "hướng dẫn" bằng dữ liệu gán nhãn để thực hiện tốt một nhiệm vụ cụ thể. Sự kết hợp pre-training + fine-tuning này tạo nên phương pháp học bán giám sát rất hiệu quả: pre-training lo học "đề cương", fine-tuning lo "ôn thi" cho từng môn. Kết quả đã cách mạng hóa NLP: với cùng một kiến trúc GPT tiền huấn luyện, chỉ cần fine-tune là có thể đạt hiệu năng hàng đầu trên nhiều tác vụ NLP khác nhau.

5.3.3. Học bán giám sát (Semi-Supervised Learning)

Phương pháp huấn luyện mô hình GPT có thể được xem là một chiến lược học bán giám sát độc đáo, kết hợp cả dữ liệu không gán nhãn và có gán nhãn trong hai giai đoạn liên tiếp. Thuật ngữ "bán giám sát" ở đây xuất phát từ việc mô hình trước tiên được học một cách không giám sát trên dữ liệu lớn (pre-training), sau đó mới học có giám sát trên dữ liệu ít hơn của từng tác vụ (fine-tuning). Cách tiếp cận này được Radford và cộng sự mô tả trong GPT-1 (2018) như một phương pháp bán giám sát cho hiểu ngôn ngữ: mục tiêu là học được một biểu diễn ngôn ngữ phổ quát từ dữ liệu thô, để có thể sử dụng cho nhiều nhiệm vụ khác nhau mà chỉ cần tinh chỉnh nhẹ.

Quy trình bán giám sát gồm hai bước chính đã trình bày: (1) *Unsupervised pre-training*: mô hình học từ dữ liệu không nhãn, (2) *Supervised fine-tuning*: mô hình được dạy cụ thể cho task có nhãn. Sự kết hợp này mang lại lợi ích đôi bên: pre-training cung cấp cho mô hình kiến thức nền rộng, còn fine-tuning đảm bảo hiệu suất cao cho tác vụ hẹp. So với mô hình học hoàn toàn có giám sát (chỉ dùng dữ liệu nhãn), phương pháp bán giám sát này giúp mô hình tận dụng tối đa dữ liệu sẵn có, đặc biệt có ý nghĩa khi dữ liệu nhãn khan hiếm. Đồng thời, so với mô hình hoàn toàn không giám sát (chỉ học xong rồi dùng trực tiếp), việc có bước fine-tuning giúp điều chỉnh mô hình cho sát mục tiêu hơn, tránh việc mô hình quá "tổng quát" mà không đạt độ chính xác cần thiết cho tác vụ cụ thể.

Trong bối cảnh GPT, phương pháp bán giám sát đã chứng minh hiệu quả rõ rệt. GPT-1 được tiền huấn luyện trên dữ liệu sách và sau đó fine-tune đã đạt kết quả SOTA trên nhiều tập benchmark mà trước đó yêu cầu kiến trúc mô hình riêng. Đến GPT-2, nhóm nghiên cứu thậm chí còn loại bỏ hoàn toàn bước fine-tuning cho nhiều nhiệm vụ: họ chỉ tiền huấn luyện mô hình lớn hơn trên dữ liệu đa dạng và cho mô hình tự thực hiện nhiệm vụ ở chế độ zero-shot (không huấn luyện thêm), nhưng vẫn thu được kết quả đáng kinh ngạc. Điều này cho thấy với dung lượng đủ lớn, mô hình

ngôn ngữ tự hồi quy có thể tự học để giải quyết nhiều nhiệm vụ mà không cần học có giám sát từng tác vụ như trước – một dạng mở rộng của học bán giám sát thành học tự giám sát đa nhiệm. Tóm lại, chiến lược huấn luyện GPT là sự kết hợp linh hoạt giữa không giám sát và có giám sát, tận dụng ưu điểm của cả hai: không giám sát để học đặc trưng tiềm ẩn phong phú của ngôn ngữ, và có giám sát để điều chỉnh mô hình tinh tế theo yêu cầu từng bài toán. Cách tiếp cận bán giám sát này đã mở ra một hướng mới trong NLP, cho phép xây dựng những mô hình ngôn ngữ nền tảng (foundation models) có thể thích nghi với nhiều nhiệm vụ khác nhau chỉ với ít dữ liệu bổ sung.

5.4. Sự phát triển qua các thế hệ GPT

5.4.1. Sơ lược về từng mô hình: GPT (2018), GPT-2 (2019), GPT-3 (2020), GPT-Neo, GPT-J (mở nguồn)

GPT (2018) – còn được gọi là GPT-1 – là mô hình GPT đầu tiên do OpenAI giới thiệu vào tháng 6/2018. GPT-1 có kiến trúc Transformer decoder gồm 12 lớp, ~117 triệu tham số, được tiền huấn luyện trên tập dữ liệu BookCorpus (800 triệu từ). Điểm đột phá của GPT-1 là chứng minh được sức mạnh của tiền huấn luyện không giám sát kết hợp với fine-tuning: mô hình đạt kết quả tốt trên 9/12 bài toán hiểu ngôn ngữ tự nhiên (NLU) mà không cần kiến trúc đặc thù cho mỗi bài toán. GPT-1 mở đầu cho trào lưu sử dụng mô hình ngôn ngữ tiền huấn luyện lớn như một nền tảng chung cho nhiều tác vụ NLP, tức mô hình "một dùng cho tất cả" thay vì mỗi tác vụ một mô hình riêng.

GPT-2 (2019) là thế hệ kế tiếp, được OpenAI công bố tháng 2/2019. Mục tiêu chính của GPT-2 là mở rộng quy mô mô hình và dữ liệu nhằm xem hiệu quả sinh văn bản tăng lên thế nào. GPT-2 thực chất là một phiên bản phóng to của GPT: số tham số tăng hơn 10 lần (lên 1,5 tỷ), dữ liệu huấn luyện tăng ~10 lần (lên 40GB văn bản thu thập từ 8 triệu trang web – tập WebText). Kiến trúc GPT-2 vẫn là decoder Transformer nhưng sâu hơn (48 lớp) và rộng hơn (chiều ẩn ~1600, 16 head). Nhờ quy mô lớn, GPT-2 có thể sinh đoạn văn bản dài rất mạch lạc và trôi chảy, đạt chất lượng gần như con người trong nhiều trường hợp. Sự tiến bộ rõ rệt so với GPT-1 khiến OpenAI lúc đầu lo ngại về khả năng lạm dụng, nên họ đã thực hiện việc phát hành có kiểm soát: ban đầu

chỉ công bố phiên bản nhỏ (117M tham số), dần dần mới công bố mô hình đầy đủ 1.5B tham số vào cuối 2019. GPT-2 được ứng dụng nhiều trong các hệ thống sinh văn bản tự động: từ viết tin tức, truyện ngắn, đến hỗ trợ viết mã giả, viết thơ,.... Sự thành công của GPT-2 khẳng định xu hướng rằng mô hình càng lớn, dữ liệu càng nhiều, thì chất lượng mô hình ngôn ngữ càng cao.

GPT-3 (2020) đưa xu hướng trên lên một tầm cao mới. Ra mắt tháng 5/2020, GPT-3 gây chấn động khi có tới 175 tỷ tham số – lớn hơn GPT-2 khoảng 100 lần. Mô hình được huấn luyện trên một tập dữ liệu khổng lồ ~500 tỷ token (bao gồm nhiều nguồn như Common Crawl, Wikipedia, sách,....). Kiến trúc GPT-3 về cơ bản mở rộng từ GPT-2: 96 lớp Transformer decoder, vector ẩn kích thước 12,288, sử dụng 96 đầu chú ý (attention heads), ngữ cảnh đầu vào tối đa 2048 token. GPT-3 đạt được một năng lực mới nổi bật: học từ ngữ cảnh (in-context learning) hay few-shot learning. Điều này nghĩa là GPT-3 có thể thực hiện tốt nhiều tác vụ NLP khác nhau mà không cần fine-tuning, chỉ bằng cách cung cấp một số ví dụ trong prompt. Ví dụ, nếu muốn GPT-3 dịch một câu, ta chỉ cần cho nó vài câu song ngữ mẫu, mô hình sẽ tiếp tục dịch các câu tiếp theo khá chuẩn dù chưa từng được huấn luyện có giám sát cho tác vụ dịch. GPT-3 đánh dấu bước chuyển quan trọng: mô hình ngôn ngữ lớn giờ đây vừa là bộ sinh văn bản đa dụng, vừa đóng vai trò như một mô hình nền tảng giải quyết nhiều bài toán theo kiểu zero-shot hoặc few-shot. Mặc dù OpenAI không phát hành công khai GPT-3 dưới dạng mô hình có sẵn (họ chỉ cung cấp API thương mại), GPT-3 đã được sử dụng rộng rãi thông qua dịch vụ để xây dựng các ứng dụng như ChatGPT (phiên bản GPT-3.5 fine-tuned cho hội thoại), các hệ thống trả lời tự động, hỗ trợ viết code (GitHub Copilot sử dụng Codex – một biến thể GPT-3 fine-tune trên mã nguồn),....

GPT-Neo & GPT-J (2021): Đây là các mô hình GPT mã nguồn mở do nhóm nghiên cứu độc lập EleutherAI phát triển, nhằm tạo ra các đối trọng mở với GPT-3 của OpenAI. Vào tháng 3/2021, EleutherAI công bố hai mô hình GPT-Neo có 1,3 tỷ và 2,7 tỷ tham số được huấn luyện trên bộ dữ liệu The Pile 825GB. GPT-Neo được xây dựng theo kiến trúc GPT-3 (Transformer decoder) và được cung cấp miễn phí mã nguồn lẫn trọng số. Tiếp đó, tháng 6/2021, nhóm này ra mắt GPT-J – một mô hình 6 tỷ tham số, cũng huấn luyện trên The Pile, với chất lượng tiệm cận GPT-3 kích thước tương đương. Theo đánh giá, GPT-J 6B đạt độ chính xác tương đương mô hình GPT-3 6.7B của OpenAI trên nhiều tác vụ zero-shot. Sự ra đời của GPT-Neo và GPT-J có ý nghĩa quan trọng: chúng cung cấp cho cộng đồng nghiên cứu một nền tảng mô hình lớn mở để thử nghiệm và ứng dụng, mà không bị giới hạn bởi API thương mại. Dù quy mô chưa bằng GPT-3 full (175B), nhưng GPT-Neo 2.7B đã lớn hơn GPT-2 và tiệm cận GPT-3 nhỏ nhất (Ada 2.7B), đủ sức thực hiện nhiều tác vụ sinh văn bản đa dạng. GPT-J 6B thậm chí còn được đánh giá là mô hình Transformer mở tốt nhất tại thời điểm 2021 về hiệu năng zero-shot trên các bài toán chuẩn. Tóm lại, GPT-Neo và

GPT-J đại diện cho nỗ lực của cộng đồng nhằm dân chủ hóa công nghệ mô hình ngôn ngữ lớn, cho phép bất kỳ ai cũng có thể tải về và triển khai mô hình GPT tương tự GPT-3 ở mức độ vừa phải.

5.4.2. Kiến trúc, thông số, ứng dụng tiêu biểu từng thế hệ

5.5. Phân tích chi tiết và so sánh các mô hình

5.5.1. GPT-2 – Kiến trúc, thông số, ứng dụng

GPT-2 là mô hình đã tạo bước ngoặt lớn trong khả năng sinh văn bản tự động nhờ độ lớn và chất lượng vượt trội so với thế hệ trước. Về kiến trúc, GPT-2 vẫn trung thành với thiết kế Transformer decoder-only như GPT-1, nhưng số tầng và tham số tăng lên đáng kể: phiên bản GPT-2 đầy đủ có 48 lớp decoder, mỗi lớp ẩn kích thước ~1600, tổng số tham số ~1,5 tỷ. Mô hình sử dụng 16 đầu attention mỗi lớp và ngữ cảnh tối đa 1024 token. Nhóm OpenAI đã huấn luyện GPT-2 trên tập dữ liệu WebText ~40GB (gồm hàng triệu trang web có độ dài tối thiểu 全文) – lớn hơn hẳn so với BookCorpus dùng cho GPT-1.

Nhờ cấu trúc sâu hơn và dữ liệu huấn luyện lớn hơn, GPT-2 có khả năng mô hình hóa ngôn ngữ rất tinh vi, sinh ra những đoạn văn bản có ngữ pháp, ngữ nghĩa tự nhiên đến mức đôi khi khó phân biệt với con người viết. Ví dụ, GPT-2 có thể viết một đoạn tin tức giả với văn phong thuyết phục dựa trên một tiêu đề cho trước. Một mẫu nổi tiếng do GPT-2 tạo ra bắt đầu bằng prompt: “In a shocking finding, scientists discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains...” (Trong một phát hiện đáng kinh ngạc, các nhà khoa học tìm thấy một đàn kỳ lân sinh sống trong thung lũng hẻo lánh chưa từng được khám phá, thuộc dãy núi Andes...) và mô hình đã tiếp tục viết thành một bản tin hoàn chỉnh về đàn kỳ lân biết nói tiếng Anh hết sức thú vị. Khả năng sinh văn bản dài, mạch lạc về chủ đề bất kỳ của GPT-2 đã vượt xa các mô hình trước đó (vốn thường vấp lỗi lặp hoặc vô nghĩa sau vài câu).

Về ứng dụng, GPT-2 chủ yếu được sử dụng như một mô hình sinh văn bản tự động đa mục đích. Các nhà phát triển có thể fine-tune GPT-2 trên một tập dữ liệu hội thoại để tạo chatbot đơn giản, hoặc fine-tune trên một thể loại văn bản (như thơ, truyện) để mô hình sáng tác theo phong cách

đó. GPT-2 cũng được dùng trong hỗ trợ viết nội dung: gợi ý câu tiếp theo cho người viết, mở rộng ý tưởng dựa trên đoạn văn hiện có. Trong nghiên cứu, GPT-2 trở thành baseline để so sánh khi phát triển các mô hình mới hoặc đánh giá mức độ "hiểu" ngôn ngữ của mô hình ngôn ngữ lớn. Tuy nhiên, GPT-2 cũng gióng lên hồi chuông về mặt đạo đức và an toàn: vì nó có thể tạo tin giả và nội dung có hại một cách khá thuyết phục, OpenAI đã phải cân nhắc kỹ khi công bố mô hình. Việc phát hành GPT-2 theo từng giai đoạn (từ nhỏ đến lớn) cũng nhằm nghiên cứu tác động xã hội và thiết lập tiền lệ cho việc công bố mô hình ngôn ngữ lớn sau này.

Tóm lại, GPT-2 với kiến trúc Transformer 1.5 tỷ tham số là một cột mốc quan trọng, chứng minh rằng tăng quy mô mô hình sẽ đem lại bước nhảy vọt về chất lượng trong sinh ngôn ngữ. Nó đặt nền móng cho sự ra đời của các mô hình lớn hơn (GPT-3) và khẳng định mô hình ngôn ngữ lớn có tiềm năng ứng dụng rộng rãi, đồng thời đặt ra các vấn đề cần giải quyết về kiểm soát nội dung và đạo đức AI.

5.5.2. GPT-3 – Kiến trúc, quy mô, khả năng few-shot learning

GPT-3 là phiên bản mở rộng vượt bậc của GPT-2, được thiết kế với giả thuyết rằng quy mô cực lớn sẽ tạo ra chất lượng vượt trội và khả năng mới. Về kiến trúc, GPT-3 vẫn là Transformer decoder-only, nhưng số lượng tham số tăng lên 175 tỷ, gấp hơn 100 lần GPT-2. Cấu hình cụ thể của GPT-3 lớn nhất (GPT-3 "Davinci"): 96 lớp decoder, kích thước ẩn 12,288, số đầu attention 96, ngữ cảnh tối đa 2048 token. Ngoài ra, OpenAI cũng huấn luyện các phiên bản nhỏ hơn của GPT-3 để nghiên cứu hiệu quả theo quy mô: các model 125M, 350M, 1.3B, 2.7B, 6B, 13B, 175B (7 phiên bản). Tất cả đều dùng cùng kiến trúc, chỉ khác về số tầng và chiều ẩn. Dữ liệu huấn luyện GPT-3 rất đa dạng và khổng lồ, tổng cộng khoảng 500 tỷ token, bao gồm Common Crawl (các trang web), Wikipedia, corpora sách, và các nguồn văn bản khác. Việc huấn luyện một mô hình khổng lồ như vậy đòi hỏi tài nguyên tính toán cực lớn (ước tính hàng chục triệu USD chi phí compute).

Điểm nổi bật nhất của GPT-3 là khả năng few-shot learning – mô hình có thể giải quyết nhiều nhiệm vụ NLP chỉ bằng cách quan sát một vài ví dụ trong phần prompt mà không cần tinh chỉnh tham số. Đây là một dạng năng lực mới xuất hiện khi mô hình đạt đến quy mô đủ lớn. Ví dụ, GPT-3 có thể dịch một câu tiếng Anh sang tiếng Pháp khá tốt nếu người dùng cung cấp trong prompt vài cặp câu Anh-Pháp mẫu làm hướng dẫn. Tương tự, GPT-3 có thể trả lời câu hỏi kiến thức, thực hiện suy luận logic ngắn, thậm chí viết mã theo yêu cầu, chỉ thông qua tương tác bằng ngôn ngữ

tự nhiên trong prompt. Nghiên cứu trong bài báo GPT-3 cho thấy hiệu năng của GPT-3 175B trong thiết lập few-shot và zero-shot trên nhiều benchmark đã tiệm cận hoặc vượt qua các mô hình SOTA đã được fine-tune chuyên biệt. Điều này rất đáng chú ý vì GPT-3 không hề được huấn luyện có giám sát cho các tác vụ đó – kỹ năng giải quyết nhiệm vụ dường như đến từ chính quá trình tiền huấn luyện quy mô lớn (model tự hình thành nội dung kiến thức và phương pháp tổng quát). GPT-3 lần đầu tiên cho thấy viễn cảnh về một mô hình ngôn ngữ tổng quát, có thể coi như một "bộ não" AI thống nhất cho nhiều công việc khác nhau, thay vì phải có mô hình riêng cho từng việc.

Tuy nhiên, GPT-3 cũng bộc lộ những thách thức. Mặc dù rất thông minh trong ngữ cảnh ngắn hạn, GPT-3 đôi khi vẫn mắc những lỗi cơ bản: ví dụ trả lời sai các câu hỏi đơn giản đòi hỏi suy luận dài hơn vài bước, hoặc tạo thông tin không có thực (ảo giác thông tin). Mặt khác, do GPT-3 quá lớn, chi phí sử dụng cũng là vấn đề – không phải ai cũng có điều kiện chạy một model 175B. Chính vì vậy, OpenAI không mở mã hay trọng số GPT-3, mà cung cấp qua API trả phí, dẫn đến một số chỉ trích về tính đóng và thiếu minh bạch. Dẫu sao, về mặt nghiên cứu, GPT-3 đã khẳng định rằng tăng thông số + dữ liệu là hướng đi hiệu quả để nâng cao năng lực mô hình ngôn ngữ, và mở ra kỷ nguyên mô hình học sâu quy mô cực lớn (các công ty khác cũng chạy đua mô hình tỷ tỷ tham số sau GPT-3). Từ GPT-3, nhiều hướng nghiên cứu mới cũng xuất hiện: tối ưu mô hình lớn, nén mô hình (distillation), tìm hiểu tại sao mô hình lớn lại học được kỹ năng few-shot,...

Tổng kết lại, GPT-3 với kiến trúc 175 tỷ tham số là bước nhảy vọt trong mô hình ngôn ngữ, không chỉ về quy mô mà còn về chất lượng và khả năng thích ứng. Khả năng few-shot learning của GPT-3 thể hiện một mức độ "hiểu" ngôn ngữ tự nhiên ở phạm vi rộng hơn hẳn các thế hệ trước. GPT-3 đã đặt nền móng cho hàng loạt ứng dụng AI sáng tạo và thúc đẩy cả cộng đồng tiến gần hơn đến mục tiêu AI ngôn ngữ tổng quát.

5.5.3. So sánh hiệu suất, khả năng sinh ngôn ngữ giữa GPT-2, GPT-3, GPT-Neo

5.6. Thực hành sinh văn bản bằng GPT-2 qua thư viện Hugging Face

5.6.1. Chuẩn bị mô hình và dữ liệu

5.6.2. Tiến hành thực thi, phân tích kết quả

5.7. Kết luận chương 5

CHƯƠNG 6: T5 VÀ BART

1. T5

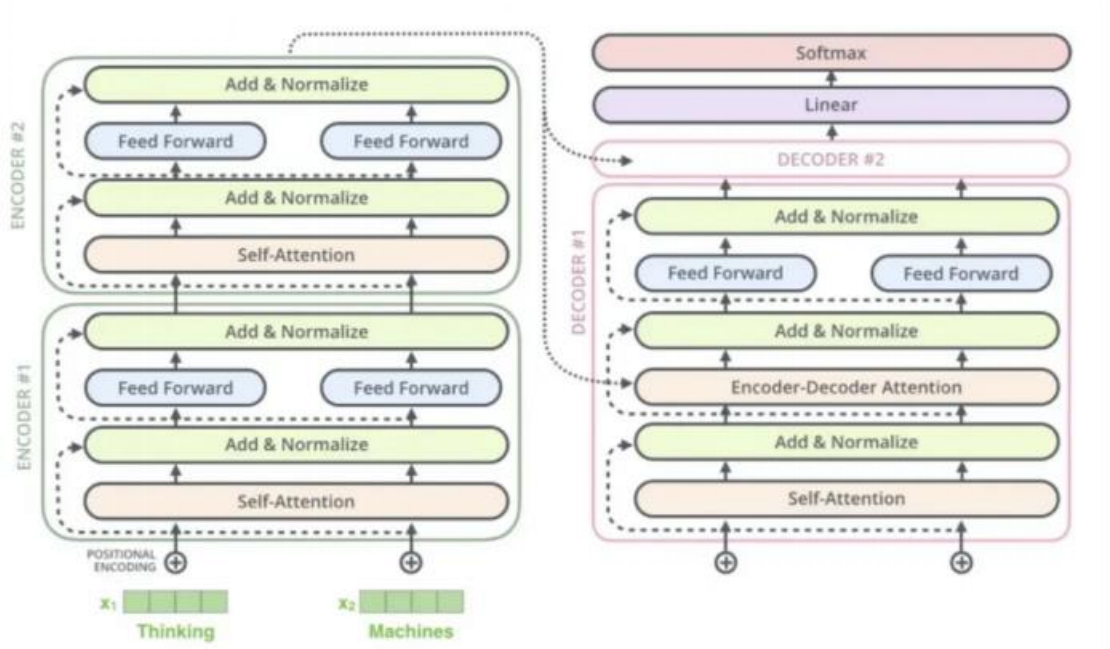
1.1 T5 là gì:

T5 (Text-to-Text Transfer Transformer) do Google phát triển, được giới thiệu trong bài báo "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer" của Colin Raffel và các cộng sự. Mô hình tiếp cận tất cả các tác vụ NLP dưới dạng bài toán "văn bản sang văn bản". Mô hình này được huấn luyện trên bộ dữ liệu C4 (Colossal Clean Crawled Corpus) khổng lồ, sử dụng kiến trúc encoder-decoder và đạt hiệu suất ấn tượng trên nhiều benchmark. Ý tưởng cốt lõi của T5 là chuyển đổi mọi tác vụ NLP thành định dạng văn bản sang văn bản (text-to-text), trong đó cả đầu vào và đầu ra đều là chuỗi văn bản. Cách tiếp cận thống nhất này đơn giản hóa khung công tác (framework) (cấu trúc tổng quát để giải quyết vấn đề) để giải quyết các tác vụ NLP khác nhau như dịch thuật (translation), tóm tắt văn bản (summarization), và trả lời câu hỏi (question answering).

1.2 Đặc điểm của T5:

- T5 chuyển mọi tác vụ xử lý ngôn ngữ tự nhiên (NLP) thành bài toán text-to-text, tức đầu vào và đầu ra đều dưới dạng văn bản. Điều này giúp mô hình có kiến trúc thống nhất, linh hoạt cho đa dạng nhiệm vụ như dịch máy, tóm tắt, trả lời câu hỏi, phân loại,....
- Kiến trúc encoder-decoder theo mô hình Transformer, giúp xử lý ngữ cảnh theo cả hai chiều mã hóa và giải mã, nâng cao hiệu quả học ngữ nghĩa và ngữ cảnh dài.
- T5 sử dụng cơ chế attention đa đầu (multi-head attention) trong cả encoder và decoder để nắm bắt các mối quan hệ ngữ nghĩa phức tạp giữa các từ trong câu.
- Mỗi tác vụ được đặt trước bằng một tiền tố (prefix) chuyên biệt mô tả nhiệm vụ, ví dụ như "translate English to French:", "summarize:", "question:", giúp mô hình hiểu bối cảnh và thực hiện đúng
- Mô hình được tiền huấn luyện (pre-train) trên tập dữ liệu lớn với nhiệm vụ làm sạch và dự đoán phần bị che khuất trong văn bản (span corruption), từ đó tăng cường khả năng hiểu ngôn ngữ tổng quát, sau đó có thể fine-tune cho các nhiệm vụ cụ thể.

1.3 Kiến trúc của T5:



- Bộ mã hóa (Encoder):

+ Chuỗi đầu vào (dưới dạng token) được ánh xạ thành một chuỗi các embedding và sau đó được đưa vào bộ mã hóa. Bộ mã hóa bao gồm một chồng các "khối" (blocks), mỗi khối chứa hai thành phần con: một lớp tự chú ý (self-attention layer) và một mạng truyền thẳng nhỏ (feed-forward network)

+ Cơ chế chú ý (Self-Attention Mechanism): Bộ mã hóa sử dụng cơ chế chú ý "hoàn toàn có thể nhìn thấy" (fully-visible), cho phép mỗi vị trí trong chuỗi đầu vào có thể chú ý đến tất cả các vị trí khác trong chuỗi đó

+ Mạng Nơ-ron Tiến Thẳng (Feed-Forward Neural Network - FFN): Là một mạng gồm hai lớp kết nối đầy đủ (fully connected layers) với hàm kích hoạt ReLU (giúp đưa tính phi tuyến vào mô hình).Bao gồm dropout để giảm nguy cơ học quá mức (overfitting).

+ Lớp chuẩn hóa (Layer normalization): Được áp dụng cho đầu vào của mỗi thành phần con. T5 sử dụng một phiên bản chuẩn hóa lớp đơn giản hóa, chỉ điều chỉnh lại các giá trị kích hoạt mà không cộng thêm độ lệch (bias).

+ Kết Nối Dư (Residual Connections): Là các kết nối bỏ qua (skip connections), giúp giữ lại thông tin từ các lớp trước đó. Hỗ trợ lan truyền gradient hiệu quả khi huấn luyện các mô hình sâu.

- Bộ giải mã (Decoder):

+ Cơ chế chú ý (Self-Attention Mechanism): Lớp tự chú ý trong bộ giải mã sử dụng một dạng tự chú ý "tự hồi quy" (autoregressive) hay "nhân quả" (causal). Cơ chế này chỉ cho phép mô hình chú ý

đến các đầu ra trong quá khứ (các token đã được tạo ra trước đó), ngăn không cho mô hình "nhìn vào tương lai" khi tạo ra chuỗi đầu ra

+ Cơ Chế Chú Ý Chéo (Cross-Attention Mechanism):Mỗi token trong bộ giải mã chú ý đến toàn bộ đầu ra của bộ mã hóa cho phép mô hình kết hợp thông tin từ đầu vào (encoder) trong quá trình tạo văn bản đầu ra.

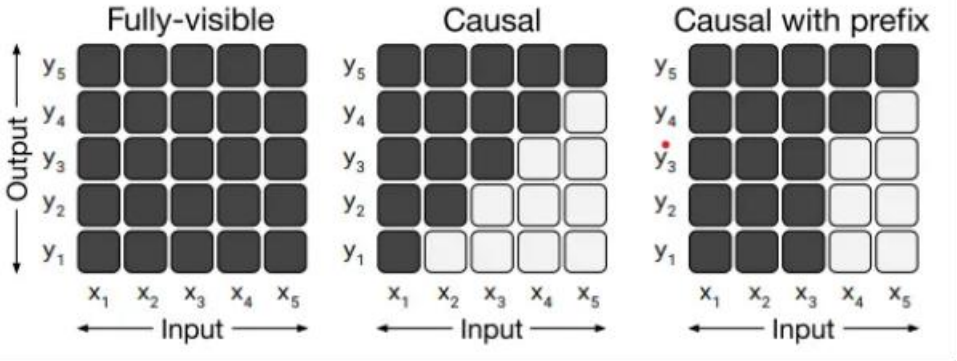
+ Mạng Nơ-ron Tiến Thẳng (Feed-Forward Neural Network - FFN): Cấu trúc giống với bộ mã hóa: gồm hai lớp fully connected và hàm kích hoạt ReLU, có thêm dropout để chống overfitting.

+ Lớp chuẩn hóa (Layer Normalization): Áp dụng trước mỗi khối (pre-norm), giúp huấn luyện ổn định và hiệu quả hơn.

+ Kết Nối Dư (Residual Connections): Dùng để bỏ qua thông tin từ đầu vào đến đầu ra của mỗi khối. Giúp duy trì thông tin gốc và hỗ trợ lan truyền gradient trong mạng sâu.

1.4 Các biến thể kiến trúc:

Một yếu tố phân biệt chính giữa các kiến trúc khác nhau là "mặt nạ" (mask) được sử dụng bởi các cơ chế chú ý khác nhau trong mô hình. Hoạt động tự chú ý trong Transformer lấy một chuỗi làm đầu vào và xuất ra một chuỗi mới có cùng độ dài. Mỗi phần tử của chuỗi đầu ra được tạo ra bằng cách tính toán giá trị trung bình có trọng số của các phần tử của chuỗi đầu vào.



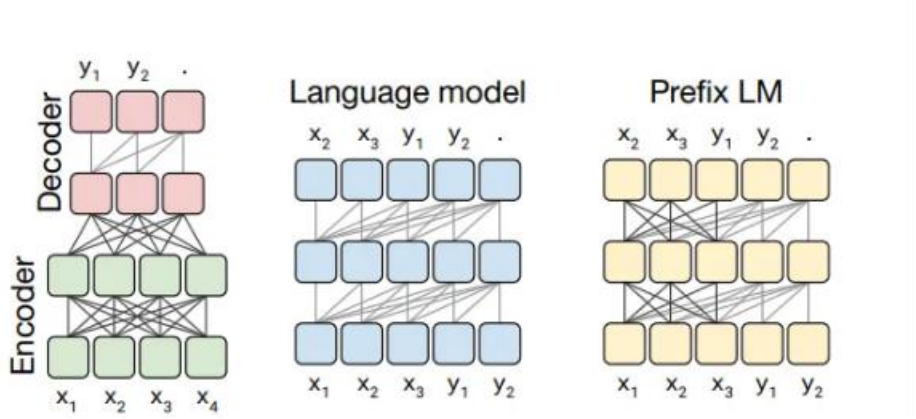
Ma trận minh họa các loại mặt nạ khác nhau được dùng trong cơ chế tự chú ý:

- x là đầu vào, y là đầu ra của self-attention.
- Ô màu đậm tại hàng i, cột j: cho biết self-attention được phép chú ý từ vị trí đầu ra i đến phần tử đầu vào j.
- Ô màu nhạt: cho biết self-attention không được phép chú ý đến cặp vị trí tương ứng.

Các biến thể masking:

- Mặt nạ hoàn toàn hiển thị (fully-visible mask).Mọi vị trí đầu ra (y_1 đến y_5) đều được nhìn toàn bộ đầu vào (x_1 đến x_5). Dùng trong encoder hoặc các tác vụ không cần "che tương lai" (ví dụ: hiểu toàn bộ câu đầu vào).
- Mặt nạ nhân quả (causal mask):Chỉ cho phép nhìn các phần tử trước hoặc chính nó trong chuỗi.Ví dụ: y_3 chỉ được nhìn: x_1, x_2, x_3 và không được nhìn x_4, x_5 (vì "tương lai"). Dùng trong decoder khi mô hình đang "tự viết ra một câu" → không được nhìn trước các từ chưa sinh ra

- Mặt nạ nhân quả với tiền tố (causal masking with a prefix): Một phần chuỗi đầu vào (gọi là tiền tố/prefix) được nhìn thấy hoàn toàn, còn phần còn lại vẫn theo kiểu causal. Ví dụ: Giả sử x_1, x_2 là tiền tố thì mọi y đều được nhìn x_1, x_2 và các phần còn lại thì vẫn phải "che tương lai".Dùng trong mô hình như prefix-tuning hoặc T5: cho mô hình một đoạn gợi ý (prefix), rồi yêu cầu tiếp tục sinh nội dung.



Các biến thể kiến trúc được thể hiện trong sơ đồ dưới với:
- Các khối vuông đại diện cho các phần tử trong một chuỗi (ví dụ: các từ).
- Các đường nối thể hiện khả năng "chú ý" (attention visibility) giữa các phần tử.
- Màu sắc khác nhau của các nhóm khối biểu thị các khối lớp Transformer khác nhau.
- Đường màu xám đậm tượng trưng cho "mặt nạ nhìn toàn bộ" (fully-visible masking)
- Đường màu xám nhạt biểu thị "mặt nạ nhân quả" (causal masking), tức là chỉ được nhìn về phía trước trong chuỗi.
- Ký hiệu "." đại diện cho token kết thúc chuỗi đặc biệt (end-of-sequence), đánh dấu nơi kết thúc của quá trình dự đoán.
- Chuỗi đầu vào và đầu ra được ký hiệu lần lượt là x và y.

Bên trái là kiến trúc Encoder-Decoder tiêu chuẩn: Sử dụng mặt nạ nhìn toàn bộ trong phần mã hóa (encoder) và trong quá trình attention giữa encoder và decoder. Phần giải mã (decoder) sử dụng mặt nạ nhân quả để đảm bảo chỉ nhìn các từ đầu ra trước đó khi dự đoán từ tiếp theo.

Ở giữa là Mô hình ngôn ngữ (Language Model): Gồm một khối Transformer duy nhất. Được cung cấp chuỗi đầu vào và đầu ra đã được nối lại với nhau. Sử dụng mặt nạ nhân quả xuyên suốt – tức là mỗi vị trí chỉ được nhìn thấy những phần tử phía trước nó trong chuỗi. Dùng nhiều trong các tác vụ như viết tiếp câu, sinh đoạn văn.

Bên phải là Prefix Language Model: Là một biến thể của mô hình ngôn ngữ. Cho phép phần đầu vào (prefix) được attention toàn bộ (fully-visible), tức là có thể nhìn qua lại giữa các phần tử đầu vào. Phần đầu ra vẫn tuân theo mặt nạ nhân quả – chỉ nhìn thấy các từ được tạo ra trước đó. Rất thích hợp cho các bài toán trả lời câu hỏi, sinh đoạn văn dựa trên gợi ý.

1.5 Ứng dụng của T5:

- Dịch máy: T5 có thể dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác với chất lượng cao nhờ kiến trúc encoder-decoder và khả năng học ngữ cảnh sâu.
- Tóm tắt văn bản: T5 được sử dụng để rút gọn các bài báo, tài liệu dài thành các bản tóm tắt ngắn gọn mà vẫn giữ được ý chính.
- Trả lời câu hỏi: Mô hình này giúp xây dựng hệ thống hỏi đáp tự động bằng cách nhận câu hỏi và cung cấp câu trả lời dưới dạng văn bản, hiệu quả trong các chatbot, trợ lý ảo.
- Phân loại văn bản: T5 phân loại các đoạn văn bản vào các nhóm định nghĩa trước, ví dụ như phân tích cảm xúc, phân loại chủ đề.
- Phân Tích Cảm Xúc (Sentiment Analysis): T5 xác định cảm xúc (sentiment) của văn bản (tích cực, tiêu cực, hoặc trung lập), hữu ích cho việc phân tích đánh giá khách hàng, bài đăng mạng xã hội, và dữ liệu văn bản khác.

2. BART

2.1 BART là gì:

BART (Bidirectional and Auto-Regressive Transformers) do Facebook AI đề xuất, được giới thiệu trong bài báo BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. Mô hình là sự kết hợp ưu điểm của BERT (mã hóa hai chiều) và GPT (sinh văn bản tự hồi quy). BART được huấn luyện theo cơ chế làm nhiễu đầu vào và học cách tái tạo lại văn bản ban đầu, giúp mô hình thể hiện hiệu quả đặc biệt trong các tác vụ tạo sinh ngôn ngữ như tóm tắt, dịch máy hay trả lời câu hỏi. Mô hình được triển khai dưới dạng

một kiến trúc sequence-to-sequence, với một encoder hai chiều xử lý văn bản bị nhiễu và một decoder tự hồi tiếp theo chiều trái sang phải

2.2 Kiến trúc:

Kiến trúc cốt lõi của BART là một bộ tự mã hóa khử nhiễu (denoising autoencoder) được xây dựng dựa trên kiến trúc Transformer sequence-to-sequence tiêu chuẩn của Vaswani và cộng sự (2017). Mô hình này được thiết kế để nhận một văn bản đã bị làm hỏng (corrupted text) và tái tạo lại văn bản gốc.

Encoder :

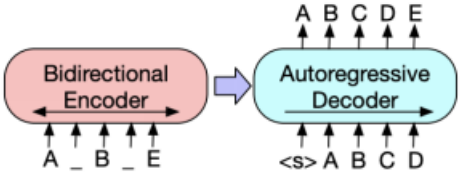
- Bộ mã hóa hai chiều (Bidirectional Encoder): Tương tự như trong BERT, bộ mã hóa của BART xử lý văn bản đầu vào (văn bản đã bị làm hỏng) và có thể xem xét ngữ cảnh từ cả hai phía (trái và phải) của một token. Điều này cho phép nó xây dựng một biểu diễn ngữ cảnh phong phú cho toàn bộ chuỗi đầu vào

Decoder:

- Bộ giải mã tự hồi quy (Autoregressive Decoder): Tương tự như trong GPT, bộ giải mã của BART tạo ra văn bản đầu ra từ trái sang phải, mỗi lần một token. Khi tạo một token mới, nó sẽ chú ý đến các token đã được tạo ra trước đó và đầu ra từ bộ mã hóa.

(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.

(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Figure 1: A schematic comparison of BART with BERT (Devlin et al., 2019) and GPT (Radford et al., 2018).

Các đặc điểm kỹ thuật:

- Hàm kích hoạt (Activation Functions): BART sử dụng hàm kích hoạt GeLUs (Gaussian Error Linear Units) thay vì ReLUs, một sửa đổi được lấy cảm hứng từ GPT.
- Khởi tạo tham số (Parameter Initialization): Các tham số được khởi tạo từ phân phối chuẩn N(0, 0.02).
- Cross-Attention trong bộ giải mã: Mỗi lớp trong bộ giải mã của BART thực hiện thêm cơ chế chú ý chéo (cross-attention) lên lớp ẩn cuối cùng của bộ mã hóa. Đây là một đặc điểm tiêu chuẩn của các mô hình sequence-to-sequence dựa trên Transformer.
- Không có lớp Feed-Forward bổ sung: BERT sử dụng một mạng truyền thẳng (feed-forward network) bổ sung trước lớp dự đoán từ cuối cùng, nhưng BART không có thành phần này.
- Số lượng tham số: Tổng cộng, BART chứa nhiều hơn khoảng 10% tham số so với một mô hình BERT có kích thước tương đương

Sự kết hợp này cho phép BART được xem như một sự tổng quát hóa của các mô hình trước đó: nó có bộ mã hóa hai chiều của BERT và bộ giải mã từ trái sang phải của GPT. Sơ đồ trong tài liệu mô tả rõ điều này: văn bản bị hỏng được đưa vào bộ mã hóa hai chiều, sau đó bộ giải mã tự hồi quy sẽ tính toán xác suất của văn bản gốc.

2.3 Tiền huấn luyện (Pre-Training):

BART được huấn luyện bằng cách làm hỏng tài liệu và sau đó tối ưu hóa độ mất mát tái tạo — entropy chéo giữa đầu ra của bộ giải mã và tài liệu gốc. Không giống như các bộ mã hóa tự động khử nhiễu hiện có, được thiết kế riêng cho các lược đồ nhiễu cụ thể, BART cho phép áp dụng bất kỳ loại làm hỏng tài liệu nào. Các phép biến đổi được tóm tắt bên dưới.

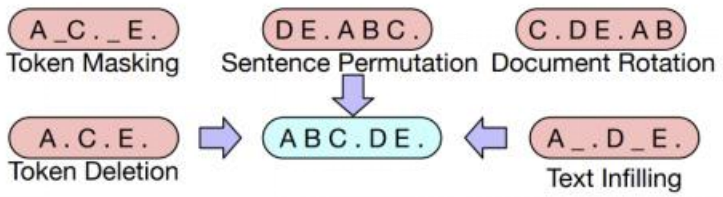
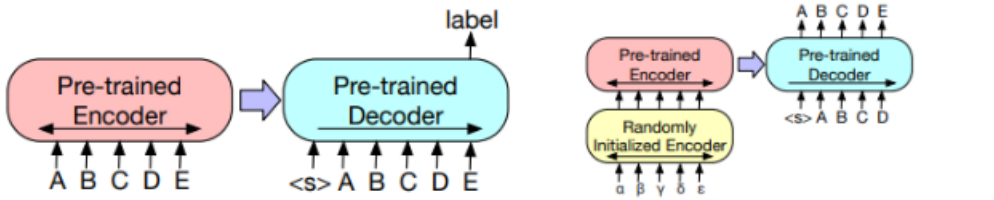


Figure 2: Transformations for noising the input that we experiment with. These transformations can be composed.

- Token Masking: Một số token ngẫu nhiên trong văn bản bị thay bằng token [MASK], giống như BERT.
- Token Deletion: Một số token bị xóa hoàn toàn khỏi văn bản. Mô hình phải tự suy đoán vị trí và nội dung còn thiếu.

- Text Infilling: Một số đoạn văn bản (span) có độ dài ngẫu nhiên (theo phân phối Poisson $\lambda = 3$) được thay bằng một token [MASK]. Kỹ thuật này dạy mô hình đoán số lượng và nội dung token bị thiếu.
- Sentence Permutation: Các câu trong văn bản bị xáo trộn thứ tự.
- Document Rotation: Một token được chọn ngẫu nhiên làm điểm bắt đầu mới cho văn bản, và văn bản bị "xoay vòng" theo điểm này.

2.4 Tinh chỉnh BART (Fine-tuning BART):



(a) To use BART for classification problems, the same input is fed into the encoder and decoder, and the representation from the final output is used.

(b) For machine translation, we learn a small additional encoder that replaces the word embeddings in BART. The new encoder can use a disjoint vocabulary.

Figure 3: Fine tuning BART for classification and translation.

- Sequence Classification (Phân loại chuỗi):Đối với các bài toán phân loại văn bản, cùng một đầu vào được đưa vào cả encoder và decoder. Trạng thái ẩn cuối cùng (final hidden state) của token cuối cùng trong decoder sẽ được sử dụng làm đầu vào cho một bộ phân loại tuyến tính nhiều lớp (multi-class linear classifier). Cách tiếp cận này tương tự như token [CLS] trong BERT, nhưng thay vì đặt đầu chuỗi, BART thêm một token đặc biệt vào cuối, để đảm bảo token này có thể "chú ý" đến toàn bộ trạng thái trong decoder.
- Token Classification (Phân loại theo token): Trong các tác vụ như tìm điểm kết thúc câu trả lời (ví dụ: SQuAD), toàn bộ văn bản được đưa vào cả encoder và decoder. Trạng thái ẩn cao nhất (top hidden state) tại mỗi vị trí trong decoder sẽ đại diện cho từng token, từ đó được sử dụng để phân loại từng token cụ thể.
- Sequence Generation (Sinh chuỗi): Do decoder của BART là autoregressive (sinh tuần tự), mô hình có thể được fine-tune trực tiếp cho các tác vụ sinh văn bản như tóm tắt văn bản (abstractive summarization) hoặc trả lời câu hỏi (abstractive QA). Trong các tác vụ này, thông tin từ đầu vào được sử dụng lại nhưng có thể bị thay đổi để phù hợp với mục tiêu đầu ra: đây chính là điểm liên hệ mật thiết với mục tiêu huấn luyện sơ cấp của BART (denoising).

- Machine Translation (Dịch máy): BART còn được mở rộng để áp dụng vào bài toán dịch ngôn ngữ. Cụ thể, lớp embedding encoder ban đầu của BART sẽ được thay thế bằng một encoder mới, được khởi tạo ngẫu nhiên và học từ dữ liệu song ngữ (bitext). Quá trình huấn luyện gồm 2 bước:

+ Bước 1: Giữ nguyên (freeze) hầu hết các tham số của BART, chỉ cập nhật encoder mới, positional embeddings và ma trận chiếu attention đầu tiên của encoder.

+ Bước 2: Huấn luyện toàn bộ mô hình (bao gồm cả BART và encoder mới) trong một số vòng lặp nhỏ.

Cách làm này cho phép mô hình chuyển đổi từ ngôn ngữ gốc sang tiếng Anh thông qua khả năng "denoise" của BART: một phương pháp sáng tạo trong áp dụng pre-trained models cho dịch máy.