Transformer

1 Introduction and Foundations:

1.1 Background Motivation:

1.1.1 Bối cảnh ra đời Transformer:

Trước năm 2017, các hệ thống dịch máy và NLP hiện đại chủ yếu dựa trên kiến trúc seq2seq dùng RNN/LSTM/GRU kết hợp attention, trong khi CNN cũng được khai thác cho mã hóa chuỗi nhưng phải dùng nhiều tầng để bao phủ ngữ cảnh xa. Các pipeline này đạt performance đáng kể nhưng lại gặp phải vấn để về tốc độ huấn luyện, khả năng song song hóa và mô hình hóa phụ thuộc dài hạn trên chuỗi dài.

1.1.2 Hạn chế của RNN:

RNN xử lý tuần tự theo thời gian nên khó song song hóa theo chiều dài chuỗi trên GPU/TPU. Dù các biến thể LSTM/GRU và attention làm giảm gradient vanishing, nhưng việc truyền thông tin qua nhiều bước vẫn khiến phụ thuộc xa bị suy yếu.

1.1.3 Hạn chế của CNN:

CNN nắm bắt tốt đặc trưng cục bộ, nhưng để biểu diễn quan hệ xa cần tăng độ sâu hoặc dùng dilated convolution, làm chi phí và độ phức tạp tăng. Thêm vào đó, thứ tự trong chuỗi phải được mã hóa gián tiếp, khiến việc căn chỉnh linh hoạt giữa nguồn–đích trong dịch máy kém tự nhiên hơn so với cơ chế chú ý toàn cục.

1.1.4 Động lực thiết kế Transformer:

Mục tiêu là loại bỏ ràng buộc tuần tự của RNN và giới hạn cục bộ của CNN, cho phép mô hình hóa phụ thuộc dài hạn trực tiếp giữa mọi cặp token và khai thác tối đa tính song song của phần cứng. Điều này kỳ vọng cải thiện cả chất lượng và tốc độ, đặc biệt trong các bài toán sequence-to-sequence như dịch máy.

Transformer thay thế đệ quy và tích chập bằng self-attention, nơi mỗi token có thể "chú ý" tới toàn bộ chuỗi để gom ngữ cảnh dài hạn. Multi-head attention học song song nhiều kiểu quan hệ, positional encoding đưa thông tin thứ tự vào

mô hình, còn residual connection và layer normalization giúp tối ưu ổn định ở độ sâu lớn.

1.2 Attention Is All You Need

1.2.1 Luận điểm chính

- Transformer loại bỏ hoàn toàn tính **tuần tự** của RNN (phải xử lý từng bước) và tính **cục bộ** của CNN (chỉ nhìn vùng lân cận). Thay vào đó, mô hình sử dụng **self-attention toàn cục**, giúp:
  - Rút ngắn "đường đi" giữa hai token bất kỳ trong câu (từ $O(n)$ còn $O(1)$).
  - Cho phép **song song hóa mạnh** trên GPU, vì không còn phụ thuộc theo chuỗi như RNN.
- Trọng tâm là hai khái niệm:
  - **Scaled Dot-Product Attention**: cơ chế tính toán mức độ liên hệ giữa các token bằng tích vô hướng, được chuẩn hóa để ổn định huấn luyện.
  - **Multi-Head Attention**: thay vì một "đầu chú ý" duy nhất, mô hình sử dụng nhiều đầu chú ý độc lập. Mỗi đầu học một không gian biểu diễn khác nhau, từ đó:
    * Giúp mô hình đồng thời tập trung vào nhiều khía cạnh ngữ nghĩa.
    * Cải thiện "độ phân giải" so với một đầu chú ý đơn lẻ.

1.3 Positional Encoding:

1.3.1 Lợi ích của Positional Encoding:

- Transformer chỉ dựa trên **self-attention**, vốn không có thông tin về **thứ tự tuần tự** giữa các token.
- Nếu không bổ sung thông tin vị trí, mô hình sẽ coi một câu như một **tập hợp không có thứ tự** các từ.
- Do đó, cần một cách để **mã hóa thông tin vị trí** của mỗi token trước khi đưa vào mô hình.

1.3.2 Cơ chế hoạt động:

- Với mỗi token, ta cộng thêm một vector vị trí (positional vector) có cùng kích thước với embedding của token.

- Vector này mang thông tin về vị trí trong chuỗi, giúp mô hình phân biệt được các từ đứng trước hay sau.
- Cách kết hợp:

$$\mathbf{z}_i = \mathbf{x}_i + \mathbf{p}_i$$

trong đó $\mathbf{x}_i$ là embedding của token thứ i, và $\mathbf{p}_i$ là vector mã hóa vị trí.

1.3.3 Các phương pháp mã hóa vị trí phổ biến

1. **Sinusoidal Positional Encoding (cố định):**
   - Mỗi vị trí *pos* và chiều i được mã hóa bằng:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

   - Ưu điểm:
     - Không cần huấn luyện thêm tham số.
     - Có tính **ngoại suy** cho các chuỗi dài hơn độ dài huấn luyện.
   - Nhược điểm:
     - Biểu diễn vị trí **cứng**, thiếu khả năng thích nghi với từng tác vụ.
2. **Learned Positional Embedding (học được):**
   - Với mỗi vị trí *pos*, mô hình học một embedding $\mathbf{p}_{pos}$ giống như học từ vựng.
   - Ưu điểm:
     - Linh hoạt, mô hình tự học cách biểu diễn vị trí phù hợp với dữ liệu.
     - Thường cho kết quả tốt hơn trên tập huấn luyện cố định.
   - Nhược điểm:
     - Thiếu khả năng ngoại suy cho các chuỗi dài hơn độ dài tối đa đã huấn luyện.
     - Tốn thêm tham số và chi phí huấn luyện.

1.3.4 So sánh hiệu suất và nhược điểm

- **Sin/Cos (cố định):** tốt cho mô hình cần xử lý chuỗi dài, ít tham số, nhưng biểu diễn vị trí kém linh hoạt.
- **Learned Positional Embedding:** hiệu quả hơn khi độ dài chuỗi huấn luyện cố định, nhưng **khó mở rộng** sang chuỗi dài hơn và tốn tài nguyên.

2 Core Components

2.1 Kiến trúc Tổng quan

- Transformer bao gồm hai phần chính: **Encoder** và **Decoder**.
- **Encoder:**
  - Nhận vào một chuỗi token (câu nguồn).
  - Mỗi token được ánh xạ thành embedding + mã hóa vị trí (Positional Encoding).
  - Qua nhiều lớp encoder giống nhau, mỗi lớp gồm:
    - Multi-Head Self-Attention.
    - Feed-Forward Network (FFN).
  - Output là một tập các vector ẩn có thông tin ngữ nghĩa.
- **Decoder:**
  - Nhận vào chuỗi đã dịch một phần (câu đích dịch đến thời điểm hiện tại).
  - Sử dụng **Masked Self-Attention** để tránh nhìn vào token tương lai.
  - Kết hợp với output của Encoder thông qua **Encoder-Decoder Attention**.
  - Cuối cùng, qua FFN và softmax để dự đoán token tiếp theo.
- Trong dịch máy (**Machine Translation**):
  - Encoder trích xuất ý nghĩa của câu nguồn.
  - Decoder từng bước sinh ra câu đích, vừa dựa vào ngữ cảnh đã dịch vừa tham chiếu thông tin từ Encoder.

2.2 Cơ chế Attention

2.2.1 Định nghĩa Attention

- Attention cho phép mô hình tập trung vào các phần quan trọng trong chuỗi khi xử lý một token.
- Công thức **Scaled Dot-Product Attention**:

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

  - Q (Query): biểu diễn "câu hỏi" tại vị trí hiện tại.
  - K (Key): biểu diễn "chìa khóa" của các token khác.
  - V (Value): chứa thông tin cần lấy từ token.
  - $\sqrt{d_k}$: hệ số chuẩn hóa để ổn định gradient.

2.2.2 (Nâng Cao - Optional) Biến đổi tuyến tính để tạo Q, K, V

- Cho đầu vào $X \in \mathbb{R}^{n \times d_{model}}$, mô hình học ba ma trận trọng số:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

  với $W_Q, W_K, W_V \in \mathbb{R}^{d_{model} \times d_k}$.
- Các phép chiếu tuyến tính này cho phép mô hình học các cách "nhìn" khác nhau về cùng một token.

2.3 Vai trò cốt lõi của Attention

- Giúp mô hình nắm bắt quan hệ **dài hạn** giữa các token.
- Loại bỏ tính tuần tự bắt buộc của RNN, cho phép **xử lý song song** trên GPU.
- Là nền tảng giúp Transformer đạt hiệu suất vượt trội trong các tác vụ NLP như dịch máy, tóm tắt văn bản, sinh ngôn ngữ.

2.4 Multi-Head Attention

- Thay vì chỉ một đầu Attention, Transformer sử dụng Multi-Head Attention độc lập.
- Mỗi head có bộ W_Q, W_K, W_V riêng, học cách tập trung vào những quan hệ khác nhau.
- Công thức:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \ldots, head_h)W_O$$

  trong đó:

$$head_i = \text{Attention}(QW_Q^{(i)}, KW_K^{(i)}, VW_V^{(i)})$$

- **Lợi ích:**
  - Cho phép mô hình **nhìn từ nhiều góc độ** (cú pháp, ngữ nghĩa, quan hệ từ xa).
  - Tăng khả năng biểu diễn và khái quát so với chỉ một head duy nhất.

3 Masked Attention và Các Khối Phụ trong Transformer

3.1 Masked Self-Attention trong Decoder

- **Vấn đề:** Khi dịch máy (Machine Translation), ở bước t, mô hình chỉ được phép dự đoán token tiếp theo dựa trên các token đã sinh ra trước đó, không được nhìn "tương lai".
- **Giải pháp: Masked Self-Attention**
  - Trong khi tính toán Attention:

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

  - M là một ma trận mask với giá trị $-\infty$ tại các vị trí không được phép chú ý (các token phía sau).
  - Kết quả: softmax loại bỏ các vị trí bị che (mask), đảm bảo tại thời điểm t chỉ dùng thông tin từ các token $1 \dots t$.
- **So sánh với Self-Attention trong Encoder:**
  - Encoder: không cần mask $\rightarrow$ mỗi token nhìn toàn bộ chuỗi đầu vào.
  - Decoder: cần mask $\rightarrow$ đảm bảo mô hình sinh tuần tự, không "gian lận" bằng cách nhìn token tương lai.

3.2 Các Khối Phụ trong Encoder/Decoder

3.2.1 Feed-Forward Network (FFN)

- Mỗi layer trong Encoder và Decoder đều có một mạng con FFN:

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

- Cấu trúc: hai tầng tuyến tính (Linear) xen giữa là hàm kích hoạt ReLU.
- Chức năng:
  - Biến đổi biểu diễn từ không gian attention sang không gian đặc trưng phi tuyến tính hơn.
  - Tăng khả năng biểu diễn và trích xuất ngữ nghĩa phức tạp.

3.2.2 Add & Norm: Residual Connections và Layer Normalization

- Sau mỗi khối Attention và FFN, Transformer áp dụng:

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

  trong đó x là đầu vào của sublayer, và Sublayer có thể là Attention hoặc FFN.
- **Residual Connections:**
  - Cho phép tín hiệu gốc x đi thẳng qua, giảm nguy cơ biến mất gradient.
  - Giúp mô hình huấn luyện ổn định hơn ngay cả với kiến trúc sâu hàng trăm lớp.
- **Layer Normalization vs. Batch Normalization:**
  - **Batch Normalization (BN):** chuẩn hóa theo batch và từng feature. Phù hợp với CNN nhưng phụ thuộc kích thước batch.
  - **Layer Normalization (LN):** chuẩn hóa theo toàn bộ chiều ẩn (hidden dimension) của một token, độc lập giữa các token.
  - Trong Transformer:
    - LN ổn định hơn vì chuỗi có độ dài thay đổi, batch size có thể nhỏ.
    - BN không hiệu quả trong NLP do phụ thuộc vào thống kê toàn batch, không phù hợp với dữ liệu chuỗi.
- **Tại sao LayerNorm phù hợp hơn cho NLP/chuỗi?**
  - Dữ liệu ngôn ngữ có độ dài chuỗi khác nhau, batch nhỏ → LN không phụ thuộc vào batch size.
  - LN giữ tính ổn định của phân phối ẩn trong mỗi token, quan trọng cho việc học quan hệ ngữ cảnh.
- **Vai trò của Residual Connections:**
  - Giữ lại thông tin gốc, tránh mất mát do biến đổi phi tuyến.
  - Truyền gradient trực tiếp ngược qua nhiều lớp, giảm hiện tượng **vanishing gradients**.
  - Là chìa khóa giúp Transformer có thể huấn luyện sâu và hiệu quả.

4 Key Variants and Developments

4.1 Sự chia tách kiến trúc

- Ban đầu, Transformer được thiết kế với kiến trúc **Encoder–Decoder đầy đủ**.

- Tuy nhiên, các nghiên cứu sau này cho thấy có thể sử dụng chỉ phần **Encoder** hoặc chỉ phần **Decoder** cho những mục tiêu cụ thể:
  - **Encoder-only:** tập trung vào việc hiểu ngữ cảnh hai chiều (ví dụ: BERT).
  - **Decoder-only:** tập trung vào việc sinh văn bản tuần tự (ví dụ: GPT).

4.2 Mô hình chỉ dùng Encoder: BERT

- **BERT** (Bidirectional Encoder Representations from Transformers) chỉ sử dụng phần **Encoder** của Transformer.
- **Ý tưởng chính:** học biểu diễn ngôn ngữ **hai chiều** (bidirectional), cho phép mô hình hiểu ngữ cảnh trước và sau cùng lúc.
- **Tác vụ tiền huấn luyện (pre-training tasks):**
  1. **Masked Language Modeling (MLM):**
     - Một số token trong câu được thay bằng token đặc biệt [MASK].
     - Mô hình phải dự đoán token gốc dựa trên ngữ cảnh hai chiều.
  2. **Next Sentence Prediction (NSP):**
     - Mô hình được cho hai câu và phải dự đoán xem câu thứ hai có thực sự là câu tiếp theo của câu thứ nhất hay không.
- **Ứng dụng phổ biến:**
  - Hiểu ngôn ngữ tự nhiên (Natural Language Understanding).
  - Các tác vụ downstream như: phân loại văn bản, trích xuất thực thể (NER), trả lời câu hỏi, phân tích cảm xúc.

4.3 Mô hình chỉ dùng Decoder: GPT

- **GPT** (Generative Pre-trained Transformer) chỉ sử dụng phần **Decoder** của Transformer.
- **Ý tưởng chính:** huấn luyện theo cơ chế **Auto-regressive (Causal Language Modeling)**:

$$P(x) = \prod_{t=1}^{T} P(x_t \mid x_{<t})$$

  tức là mô hình dự đoán token tiếp theo dựa vào các token trước đó.
- **Đặc điểm:**
  - Chỉ nhìn ngữ cảnh phía trước (causal mask).
  - Được huấn luyện trên lượng dữ liệu khổng lồ để sinh văn bản mượt mà.

- **Vai trò:** Đây là nền tảng cho các **Mô hình ngôn ngữ lớn (LLMs)** tạo sinh, ví dụ: ChatGPT, GPT-4, LLaMA, Mistral,...

4.4 Các biến thể khác (Nâng cao - Optional)

- **RoBERTa:** cải tiến từ BERT, huấn luyện lâu hơn, dữ liệu lớn hơn, bỏ nhiệm vụ NSP để tăng hiệu quả.
- **XLNet:** khắc phục hạn chế của BERT bằng **Permutation Language Modeling**, cho phép tận dụng tính tự hồi quy và ngữ cảnh hai chiều.
- **T5 (Text-to-Text Transfer Transformer):** coi mọi tác vụ NLP (dịch, tóm tắt, QA, phân loại) đều là **bài toán sinh văn bản**.
- **Longformer, Reformer:** cải tiến cơ chế Attention để xử lý chuỗi rất dài với chi phí thấp hơn (từ $O(n^2)$ xuống gần $O(n)$).

5 Bài tập:

5.1 Tổng quan

Mô hình Seq2Seq với cơ chế Attention đã được triển khai để tóm tắt văn bản tiếng Việt từ tập dữ liệu *nam194/vietnews*. Mô hình bao gồm các thành phần chính:

- **Encoder:** Mã hóa bài viết đầu vào thành các trạng thái ẩn.
- **Attention:** Cho phép decoder tập trung vào các phần quan trọng của bài viết.
- **Decoder:** Tạo ra bản tóm tắt dựa trên trạng thái ẩn và attention.

5.2 Dữ liệu

- **Tập dữ liệu:** *nam194/vietnews* (sử dụng 50,000 mẫu từ tập huấn luyện)
- **Tiền xử lý:** Loại bỏ URL, ký tự đặc biệt, chuyển đổi thành chữ thường, tách từ bằng *underthesea*.
- **Độ dài tối đa:** Bài viết (150 từ), Bản tóm tắt (40 từ).
- **Kích thước từ vựng:** Bài viết (163601), Bản tóm tắt (56756).

5.3 Mô hình

- **Kiến trúc:** Seq2Seq với Attention.
- **Embedding:** 256 chiều.
- **LSTM:** 512 chiều.

- **Optimizer:** Adam (learning rate 0.001).
- **Loss:** CrossEntropyLoss (bỏ qua padding).

5.4 Huấn luyện

- **Số epochs:** 10
- **Thời gian mỗi epoch:** Khoảng 24-25 phút (trên GPU).
- **Xu hướng Loss:** Giảm dần qua các epochs.

Epoch	Loss
1	6.2659
2	5.5000
3	5.1022
4	4.7551
5	4.4525
6	4.2065
7	3.9979
8	3.8284
9	3.6652
10	3.5453

5.5 Kết quả

- **Bài viết gốc:** ngày 273 cơquan cảnhsát điềutra côngan tp hưngyên tỉnh hưngyên cho biết đơnvị vừa ra quyếtđịnh khởitố vụ án khởitố bịcan đốivới đốitượng maivănthương sn 1989 trú tại đội 11 thôn anchiểu 1 xã liênphương tp hưngyên để điềutra về hànhvi trộmcắp tàisản theo tàiliệu điềutra của cơquan côngan vàokhoảng 7 ...
- **Bản tóm tắt được tạo ra:** sau khi độtnhập tiềnán tiềnsự ổnhóm nghiện matuý <unk> đã độtnhập tiềnán tiềnsự ổnhóm nghiện matuý đã độtnhập vào nhà chú ruột của mình để mua lại nghiện để kiếm tiền tiêuxài lười
- **Bản tóm tắt tham khảo:** <sos> với bảntính ham chơi lười làm có nhiều tiềnán tiềnsự lại nghiện matuý_thương đã độtnhập vào nhà chú ruột để trộm hơn 1 tạ thóc và hơn 8 triệu đồng mang đi tiêuxài <eos>

5.6 Đánh giá

Ưu điểm:

- Mô hình học được cách giảm loss trong quá trình huấn luyện.
- Bản tóm tắt được tạo ra có chứa một số từ khóa liên quan đến bài viết gốc.

Hạn chế:

- Bản tóm tắt còn thiếu mạch lạc và có vẻ lặp lại.
- Vẫn còn sự xuất hiện của token `<unk>` (unknown), cho thấy từ vựng chưa bao phủ hết các từ trong dữ liệu.
- Bản tóm tắt chưa thực sự "tóm tắt" được ý chính của bài viết.

5.7 Kết luận và hướng cải thiện

Mô hình Seq2Seq với Attention đã cho thấy tiềm năng trong việc tóm tắt văn bản tiếng Việt, tuy nhiên, cần cải thiện thêm để tạo ra các bản tóm tắt chất lượng hơn. Một số hướng cải thiện có thể bao gồm:

- Tăng kích thước tập dữ liệu huấn luyện.
- Điều chỉnh các siêu tham số (ví dụ: learning rate, kích thước embedding, số lớp LSTM).
- Sử dụng các kỹ thuật attention phức tạp hơn.
- Sử dụng pre-trained word embeddings (ví dụ: fastText, PhoBERT) để cải thiện khả năng biểu diễn từ.
- Áp dụng các kỹ thuật regularization để tránh overfitting.