

TRƯỜNG ĐẠI HỌC MỎ ĐỊA CHẤT
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN HỆ THỐNG THÔNG TIN VÀ TRI THỨC

BÁO CÁO SEMINAR
KIẾN TRÚC TRANSFORMER VÀ ỨNG DỤNG

Người trình bày: ThS. Bùi Thị Vân Anh

Hà Nội 6/2026

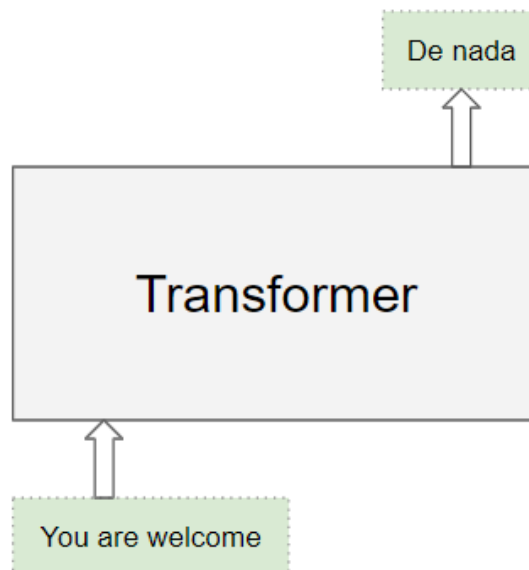
1. Mở đầu

Trong thế giới của trí tuệ nhân tạo và học máy, transformer đã trở thành một khái niệm không thể thiếu trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Kể từ khi được giới thiệu vào năm 2017 thông qua bài báo [Attention is All You Need](#) của các nhà khoa học ở Google, mô hình transformer đã cách mạng hóa cách chúng ta xây dựng các hệ thống hiểu và sinh ngôn ngữ tự nhiên. Bài viết lần này sẽ giải thích transformer là gì, cơ chế hoạt động và tầm quan trọng của nó.

2. Transformer là gì?

Transformer là một kiến trúc mô hình học sâu dựa trên cơ chế self-attention, cho phép mô hình này hiểu được mối quan hệ giữa các từ trong một câu mà không cần đến kiến trúc tuần tự truyền thống như RNN (Recurrent Neural Networks) hay LSTM (Long Short-Term Memory). Transformer có khả năng xử lý toàn bộ câu cùng một lúc, điều này giúp tăng tốc độ huấn luyện và cải thiện hiệu quả xử lý.

Kiến trúc Transformer vượt trội trong việc xử lý dữ liệu văn bản có tính chất tuần tự. Chúng nhận một chuỗi văn bản làm đầu vào và tạo ra một chuỗi văn bản khác làm đầu ra. Ví dụ: để dịch một câu tiếng Anh đầu vào sang tiếng Tây Ban Nha.



3. Các công nghệ xử lý ngôn ngữ tự nhiên trước Transformer?

Trước khi mô hình Transformer được phát triển và trở nên phổ biến trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), các kiến trúc chủ yếu được sử dụng là các

mạng nơ-ron hồi quy (RNNs), bao gồm cả Long Short-Term Memory (LSTM) và Gated Recurrent Units (GRUs). Mặc dù những công nghệ này đã đạt được nhiều thành công nhất định, chúng vẫn tồn tại một số hạn chế nghiêm trọng đã thúc đẩy sự phát triển của mô hình Transformer.

3.1. Mạng nơ-ron hồi quy (Recurrent Neural Network)

Mạng nơ-ron hồi quy (Recurrent Neural Networks - RNN) là một mô hình học sâu được thiết kế để xử lý dữ liệu tuần tự bằng cách duy trì thông tin từ các trạng thái trước đó. Điều này giúp RNN có khả năng ghi nhớ và sử dụng các mối liên hệ giữa các phần tử trong chuỗi dữ liệu, tạo ra đầu ra có ngữ cảnh và ý nghĩa hơn.

Khác với các mạng nơ-ron truyền thống xử lý dữ liệu đầu vào độc lập, RNN có một vòng lặp bên trong, cho phép lưu trữ trạng thái và truyền tải thông tin qua từng bước thời gian.

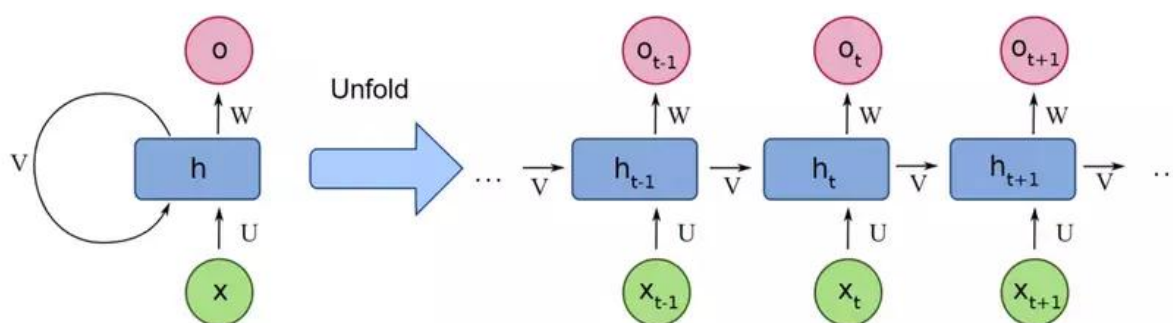
Ý tưởng

Con người đọc văn bản: Tôi → đang → học → AI

Mỗi từ mới được hiểu dựa trên:

- từ hiện tại
- thông tin đã đọc trước đó

RNN mô phỏng chính điều này.



Công thức

$$h_t = f(W_h h_{t-1} + W_x x_t + b)$$

Trong đó:

- h_{t-1} : bộ nhớ quá khứ

- x_t : dữ liệu hiện tại

RNN thường được biểu diễn theo thời gian.

$$x_1 \rightarrow h_1 \rightarrow y_1$$

$$x_2 \rightarrow h_2 \rightarrow y_2$$

$$x_3 \rightarrow h_3 \rightarrow y_3$$

$$x_4 \rightarrow h_4 \rightarrow y_4$$

hay

$$h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow h_4$$

Thông tin được truyền dọc chuỗi.

Ưu điểm của RNN: Ghi nhớ ngữ cảnh

Ví dụ:

“Nam sống ở Hà Nội.”

“Anh ấy thích AI.”

RNN có thể liên hệ: Anh ấy = Nam

Vấn đề vanishing gradient: Đây là lý do lớn nhất khiến RNN thất bại.

Ví dụ

Câu:

Tôi sinh ra ở Hà Nội.

...

...

...

Tôi hiện đang làm việc tại đây.

Đề hiểu: “đây” = “Hà Nội”

Mô hình phải nhớ thông tin rất xa.

Lan truyền ngược

Gradient phải đi qua:

$$h_{100} \rightarrow h_{99} \rightarrow h_{98} \rightarrow \dots \rightarrow h_1$$

Khi gradient nhỏ

Ví dụ: $0.5 \times 0.5 \times 0.5 \times 0.5 \times \dots$

Sau nhiều lần: ≈ 0

Gradient biến mất.

Hậu quả

RNN quên:

- chủ ngữ
- ngữ cảnh xa
- thông tin đầu câu

3.2. LSTM (Long Short-Term Memory)

LSTM là một dạng đặc biệt của mạng nơ-ron hồi quy (RNN), được thiết kế chuyên biệt để xử lý và ghi nhớ các chuỗi dữ liệu dài. Bằng cách sử dụng các cơ chế "cổng" (gating mechanisms), LSTM giải quyết triệt để vấn đề phụ thuộc xa (long-term dependency) mà các mạng RNN truyền thống thường gặp phải.

Cấu trúc và Cơ chế hoạt động

Kiến trúc cốt lõi của LSTM xoay quanh trạng thái ô (cell state), hoạt động như một "băng chuyền" mang thông tin xuyên suốt toàn bộ chuỗi thời gian. Dòng chảy thông tin được kiểm soát và cập nhật thông qua ba cổng chính:

- **Cổng Quên (Forget Gate):** Quyết định thông tin nào từ trạng thái ô trước đó sẽ bị loại bỏ hoặc giữ lại.
- **Cổng Đầu Vào (Input Gate):** Quyết định thông tin mới nào từ đầu vào hiện tại sẽ được thêm vào và cập nhật trong trạng thái ô.
- **Cổng Đầu Ra (Output Gate):** Quyết định giá trị đầu ra dựa trên trạng thái ô hiện tại.

Mặc dù tốt hơn RNN rất nhiều nhưng vẫn còn các vấn đề:

- Vấn đề 1: Xử lý tuần tự. Từ 1 $\rightarrow$ Từ 2 $\rightarrow$ Từ 3 ... Không thể xử lý song song.
- Vấn đề 2: Chuỗi dài rất chậm. Ví dụ 1000 token phải chạy 1000 bước
- Vấn đề 3: Bộ nhớ dài hạn vẫn chưa hoàn hảo.
- Ví dụ: Từ đầu câu $\rightarrow \dots \rightarrow$ Từ cuối câu, Thông tin vẫn suy giảm.

Các hạn chế của RNN và LSTM bao gồm:

- **Vấn đề về phụ thuộc dài hạn:** Mặc dù LSTM được thiết kế để giải quyết vấn đề biến mất gradient trong RNN truyền thống, chúng vẫn gặp khó khăn khi học các phụ thuộc dài hạn trong văn bản. Điều này là do bản chất tuần tự của chúng, khiến việc lưu giữ thông tin qua một chuỗi dài trở nên khó khăn.
- **Khó khăn trong việc song song hóa:** Các mô hình RNN, bao gồm LSTM, phải xử lý dữ liệu theo trình tự từng bước một thời điểm. Điều này hạn chế khả năng song song hóa quá trình huấn luyện, làm tăng thời gian cần thiết để huấn luyện mô hình trên các chuỗi dài và các tập dữ liệu lớn.
- **Hiệu suất xử lý không hiệu quả:** Việc xử lý tuần tự khiến RNN không thể tận dụng hiệu quả các GPU và TPU hiện đại, những thiết bị được thiết kế để xử lý đồng thời nhiều tác vụ. Điều này dẫn đến một hiệu suất xử lý không tối ưu.

Transformer đã được phát triển để giải quyết các hạn chế này bằng cách sử dụng cơ chế "attention", cho phép mô hình tập trung vào các phần quan trọng của đầu vào mà không cần xử lý tuần tự.

Các điểm nổi bật chính của Transformer là:

- **Xử lý song song:** Transformer xử lý đầu vào theo từng khối, không tuần tự. Điều này cho phép việc huấn luyện mô hình được thực hiện song song, giảm đáng kể thời gian cần thiết để huấn luyện mô hình.
- **Self-Attention:** Cơ chế này giúp Transformer xác định được mối quan hệ giữa tất cả các từ trong một câu, bất kể khoảng cách giữa chúng trong văn bản, giải quyết vấn đề về phụ thuộc dài hạn.
- **Hiệu suất và mở rộng:** Với khả năng xử lý đồng thời, Transformer tận dụng tối đa sức mạnh của phần cứng hiện đại, như GPU và TPU, để xử lý các tác vụ NLP một cách hiệu quả.

Do những ưu điểm nổi bật này, Transformer đã trở thành một tiêu chuẩn mới trong việc xây dựng các mô hình NLP, dẫn đến sự phát triển của nhiều kiến trúc hiện đại hơn như BERT, GPT-3, và các mô hình tương tự khác.

4. Transformer hoạt động như nào?

Cốt lõi của transformer là attention mechanism (cơ chế tập trung), giúp mô hình tập trung vào các phần quan trọng của văn bản để đưa ra dự đoán chính xác hơn.

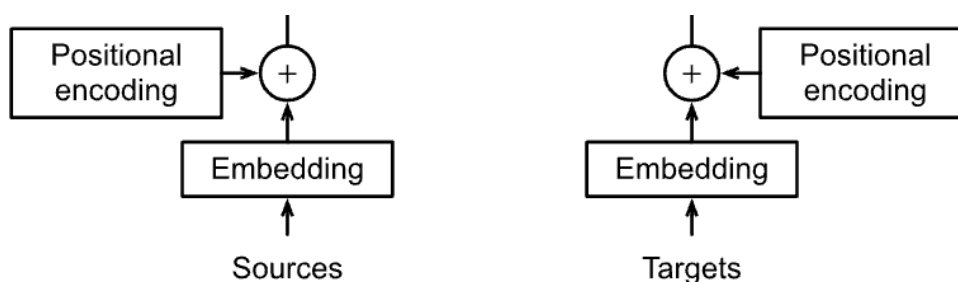
Transformer được cấu trúc thành hai phần chính là **encoder** và **decoder**.

- **Encoder:** Encoder xử lý dữ liệu đầu vào (gọi là "Source") và nén dữ liệu vào vùng nhớ hoặc context mà Decoder có thể sử dụng sau đó.
- **Decoder:** Decoder nhận đầu vào từ đầu ra của Encoder (gọi là "Encoded input") kết hợp với một chuỗi đầu vào khác (gọi là "Target") để tạo ra chuỗi đầu ra cuối cùng.

Mỗi encoder và decoder đều bao gồm nhiều lớp, mỗi lớp chứa các thành phần self-attention và feed-forward neural networks.

Về cơ bản, quy trình hoạt động của Transformer gồm các bước sau:

Bước 1: Preprocessing (Tiền xử lý)



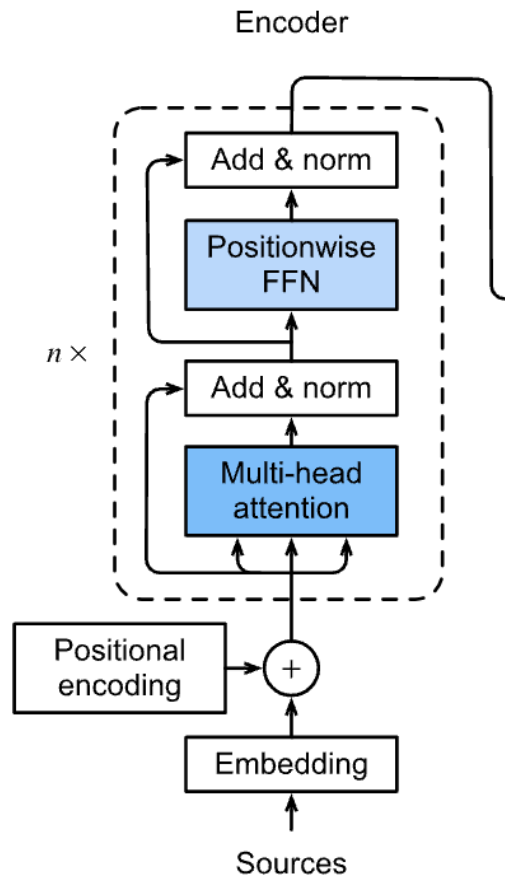
Hình mô tả Preprocessing

- **Tokenization:** Dữ liệu đầu vào được biến đổi thành token. Ví dụ, câu "ChatGPT là gì?" sẽ được mã hoá thành 4 token ["ChatGPT", "là", "gì", "?"].
- **Embedding:** Biến token thành vector. Mỗi token được chuyển thành một vector.
- **Positional Encoding:** Vì Transformer không xử lý tuần tự nên cần một cách để hiểu vị trí của từng từ trong câu. Điều này được thực hiện thông qua positional encodings, được cộng trực tiếp vào input embeddings. Các positional encodings có thể dùng các hàm sin và cos với bước sóng khác nhau để mỗi vị trí có một encoding duy nhất.

Bước 2: Encoder

Encoder bao gồm nhiều layer (lớp) xếp chồng lên nhau (stack). Mỗi layer có 2

sublayer là Multi-Head Attention và Positionwise Feed-forward Neural Network (FNN). Output của 2 sublayer sẽ đi qua một lớp gọi là Add & Norm.

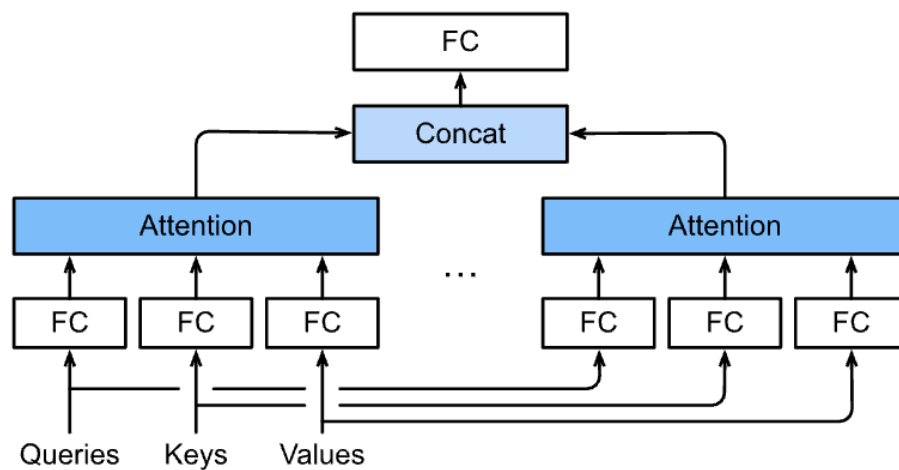


Hình mô tả Encoder

- **Multi-Head Attention:** Dữ liệu đầu vào được truyền vào nhiều head để tạo một tập các vector đầu ra. Multi-head attention được thiết kế để cho phép mô hình xử lý thông tin đồng thời ở các không gian không gian (subspaces) khác nhau. Thay vì có một module attention duy nhất, mô hình có nhiều "heads", mỗi head có thể tập trung vào các phần khác nhau của đầu vào.
- **Positionwise Feed-forward Neural Network (FNN):** Các vector đầu ra của attention head được truyền qua positionwise feed-forward neural network (mạng thần kinh chuyển tiếp theo vị trí). Mỗi head trong khối có một mạng FNN riêng, cho phép mô hình học được các biểu diễn dựa trên nhiều không gian không gian (subspaces) khác nhau của dữ liệu. Mục tiêu là tạo một lớp biểu diễn phi tuyến tính (non-linear representation) của dữ

liệu đầu vào, giúp mô hình có khả năng học được các mối quan hệ phức tạp và phi tuyến tính của dữ liệu.

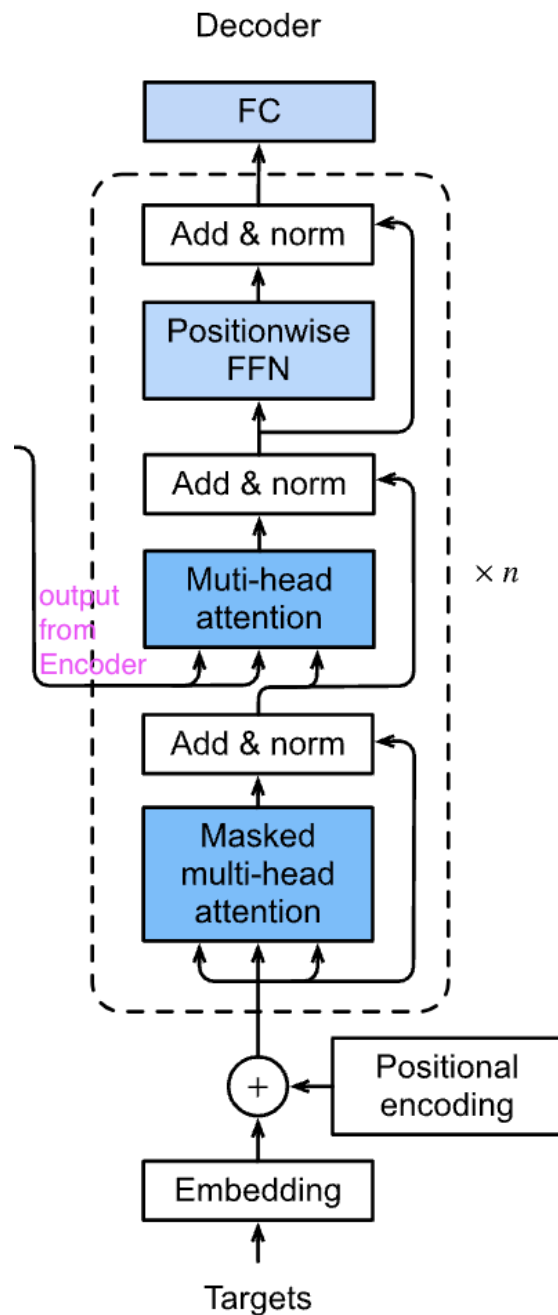
- **Add & Norm:** Bao gồm hai bước Add và Norm. Bước Add thêm *residual connection* (kết nối dư) nhằm giảm vấn đề vashing gradient (mất độ dốc) trong các deep network (mạng sâu). Ngay sau đó, bước Norm thực hiện *layer normalization* (chuẩn hoá lớp) giúp ổn định quá trình huấn luyện và giảm số lượng giai đoạn cần thiết để huấn luyện.



Multi-Head Attention

Bước 3: Decoder

Tương tự Encoder, Decoder cũng có 2 sublayer và có thêm 1 sublayer ở nằm giữa 2 sublayer tên là **Encode-decode Attention**.



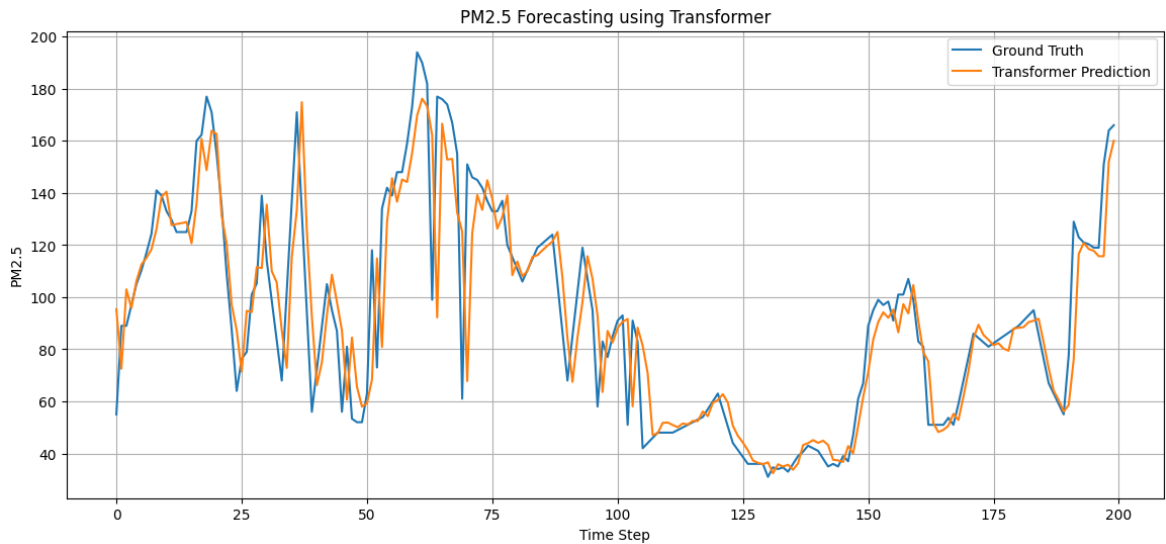
- **Masked Multi-Head Attention:** Là một biến thể của Multi-Head Attention, hoạt động tương tự Multi-Head Attention nhưng có "mask" để đảm bảo rằng các vị trí trong output chỉ có thể tập trung vào các từ trước đó mà không "nhìn thấy" các từ tiếp theo (điều này quan trọng cho quá trình dự đoán từ tiếp theo).
- **Multi-Head Attention:** Hoạt động giống với bước Encoder.
- **Encode-decode Attention:** Trong Encode-decode Attention, truy vấn tới từ đầu ra của lớp con Attention, còn key và value tới từ đầu ra của Encoder.

- **Positionwise Feed-forward Neural Network (FNN):** Hoạt động giống với bước Encoder.
- **Add & Norm:** Hoạt động giống với bước Encoder.

Đầu ra của Decoder thường là một chuỗi các vector. Mỗi vector tương ứng với một từ trong chuỗi đầu ra. Các vector này sau đó được chuyển qua một lớp tuyến tính (linear layer) và một hàm softmax để tạo ra một phân phối xác suất trên tất cả các từ trong từ điển. Sau cùng, từ có xác suất cao nhất được chọn làm từ dự đoán tiếp theo trong chuỗi. Quy trình này lặp lại cho mỗi từ trong chuỗi đầu ra, cho tới khi mô hình dự đoán ra ký tự kết thúc câu hoặc đạt độ dài tối đa cho phép.

5. Ứng dụng mô hình Transformer để dự báo nồng độ bụi mịn PM2.5 tại Hà Nội trong giai đoạn 2022-2025

Ô nhiễm không khí do bụi mịn PM2.5 đang trở thành một vấn đề môi trường nghiêm trọng tại nhiều đô thị lớn, đặc biệt ở các khu vực có tốc độ đô thị hóa nhanh. Việc dự báo chính xác nồng độ PM2.5 có ý nghĩa quan trọng trong công tác quản lý chất lượng không khí và xây dựng các hệ thống cảnh báo ô nhiễm môi trường. Nghiên cứu này nhằm đánh giá khả năng ứng dụng của các phương pháp học máy và học sâu trong dự báo nồng độ PM2.5 dựa trên dữ liệu chuỗi thời gian kết hợp với các yếu tố khí tượng. Dữ liệu sử dụng trong nghiên cứu bao gồm nồng độ PM2.5 cùng các biến khí tượng như nhiệt độ, độ ẩm và tốc độ gió tại khu vực Hà Nội. Các bước tiền xử lý dữ liệu được thực hiện bao gồm phát hiện và xử lý ngoại lai bằng phương pháp khoảng tứ phân vị (IQR), chuẩn hóa dữ liệu theo phương pháp Z-score và xây dựng các đặc trưng chuỗi thời gian. Các mô hình dự báo được xem xét gồm ARIMA, Random Forest, LSTM, GRU và Transformer. Kết quả thực nghiệm cho thấy các mô hình học sâu đạt hiệu suất dự báo cao hơn so với các phương pháp truyền thống. Trong đó, mô hình Transformer cho kết quả tốt nhất với sai số dự báo thấp và khả năng tái hiện xu hướng biến động của PM2.5 hiệu quả hơn so với các mô hình còn lại. Kết quả nghiên cứu cho thấy tiềm năng ứng dụng của các mô hình học sâu trong dự báo chất lượng không khí và cung cấp cơ sở khoa học cho việc xây dựng các hệ thống cảnh báo sớm ô nhiễm không khí tại các đô thị lớn.



Hình 1: So sánh giữa giá trị thực và dự đoán PM2.5 sử dụng Transformer

6. Kết luận

Transformer đã chứng minh được sức mạnh và linh hoạt của mình trong lĩnh vực học sâu và NLP. Với khả năng xử lý song song dữ liệu và hiệu quả học tập vượt trội, transformer tiếp tục là một trong những công cụ hàng đầu giúp các nhà nghiên cứu và các kỹ sư AI giải quyết các thách thức ngày càng phức tạp trong thế giới ngôn ngữ tự nhiên và nhiều ứng dụng khác

TÀI LIỆU THAM KHẢO

- [1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, Deep Learning, MIT Press, (2016).
- [2] Mohit Sewak, Md. Rezaul Karim, Pradeep Pujari, “Practical Convolutional Neural Networks”, (2018).
- [3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, Attention Is All You Need, Submitted on 12 Jun 2017 (v1), last revised 6 Dec 2017 (this version, v5)