Nguồn: Transformers for Machine Learning: A Deep Dive - Wael Emara, Kenneth Graham, Uday Kamath

**I. CƠ CHẾ ATTENTION**

**1. Background**

**a. Khái quát về Attention Mechanism**

Cơ chế chú ý (attention mechanism) là một phương pháp chọn lọc có chủ đích nhằm tập trung vào những yếu tố quan trọng và bỏ qua những yếu tố ít liên quan hơn. Dây thần kinh thị giác con người có khả năng nhận tín hiệu lên tới hàng tỷ bit mỗi giây, tuy nhiên não bộ chỉ có thể xử lý một phần nhỏ trong đó. Do đó, cần có một hệ thống giúp não chọn lọc thông tin để xử lý hiệu quả hơn.

Chú ý thị giác – một dạng đặc biệt của attention – giúp định hướng và duy trì sự tập trung vào một tác nhân cụ thể (có thể là người, vật vô tri hay nhiệm vụ), từ đó nâng cao hiệu quả xử lý thông tin. Điều này cho phép não bộ chỉ tập trung vào một phần nhỏ của dữ liệu, góp phần vào việc sử dụng tài nguyên tối ưu hơn và thúc đẩy sự phát triển cá nhân.

**b. Khung lý thuyết của William James về Attention**

William James – người được mệnh danh là “cha đẻ của tâm lý học Mỹ” – đã đề xuất một khung lý thuyết gồm hai thành phần cho cơ chế chú ý:

* **Chú ý không chủ đích (Nonvolitional)**: Là loại chú ý xảy ra một cách tự động, không cần nỗ lực của cá nhân. Nó được kích hoạt bởi độ nổi bật của các tác nhân trong môi trường. Ví dụ: sự chú ý đến một em bé đang khóc hoặc một vật thể có màu sắc khác biệt.
* **Chú ý có chủ đích (Volitional)**: Là sự chú ý dựa trên ý chí và sự lựa chọn của cá nhân. Ví dụ: trả lời một câu hỏi cụ thể hoặc tập trung vào một chi tiết nhất định trong một nhiệm vụ phức tạp.

Trong học máy, khái niệm này được ánh xạ như sau:

* **Query** ~ chú ý có chủ đích
* **Key** ~ chú ý không chủ đích
* **Value** ~ đầu vào cảm giác (sensory input)

Cơ chế attention trong deep learning chính là quá trình lựa chọn thông tin phù hợp (values) dựa trên mức độ tương đồng giữa queries và keys.

**c. Mô hình hóa Attention bằng toán học**

A diagram of a memory

AI-generated content may be incorrect.

Cơ chế attention có thể được coi như một bộ nhớ chứa các cặp khóa – giá trị (key-value pairs) và một lớp xử lý đầu ra. Khi truy vấn một key nào đó, cơ chế attention sẽ sinh ra kết quả dựa trên giá trị (value) tương ứng.

Giả sử ta có một bộ nhớ gồm n cặp key-value:

Trong đó:

* là key
* là value
* là query
* là output

Lớp attention sẽ tính độ tương đồng giữa query và mỗi key thông qua một hàm điểm số :

Sau đó, các điểm số này được chuẩn hóa bằng hàm softmax để thành trọng số attention:

Cuối cùng, output là tổng có trọng số của các giá trị :

**2. Các Loại Attention Dựa Trên Hàm Điểm Số (Score-Based Attention)**

Như đã thảo luận trước đó, hàm điểm số có thể được xây dựng theo nhiều cách khác nhau. Chính sự đa dạng này đã dẫn đến sự ra đời của nhiều cơ chế attention khác nhau, mỗi loại lại có những ưu điểm riêng biệt.

**a. Dot Product**

Hàm điểm số dựa trên tích vô hướng là dạng đơn giản nhất, không yêu cầu tham số nào cần học.  
Cách tính:

Đây là cơ chế nhanh và dễ thực hiện trong thực tế.

**b. Scaled Dot Product**

Biến thể này được đề xuất trong mô hình Transformer của Vaswani et al. Hàm điểm số được điều chỉnh bằng cách chia kết quả của tích vô hướng cho căn bậc hai của kích thước vector key ​.

Dot product có xu hướng tăng cường độ lớn, có xu hướng làm cho hàm softmax trả về những giá trị rất nhỏ (Hàm softmax có xu hướng tăng cường giá trị lớn, giảm nhẹ giá trị nhỏ)

Việc chia này nhằm giảm ảnh hưởng tiêu cực của việc dot product tăng quá lớn khi kích thước vector tăng.

**c. Linear, MLP, or Additive**

Luồng chú ý dạng additive được đề xuất bởi Luong và cộng sự, trong đó truy vấn (query) và khóa (key) được chiếu qua một tầng ẩn có kích thước , sử dụng các ma trận trọng số học được , và sau đó áp dụng hàm tanh và nhân với vector . Công thức:

Mặc dù dạng attention này có khả năng biểu đạt phức tạp hơn, nhưng nó lại tiêu tốn nhiều tài nguyên bộ nhớ và thời gian tính toán hơn so với các phương pháp dot product.

**3. Mô Hình Attention Trong Sequence-to-Sequence**

Trong các mô hình sequence-to-sequence truyền thống như Encoder-Decoder, việc bổ sung lớp attention đã mang đến một bước cải tiến lớn. Cụ thể, khi tích hợp attention, quá trình ánh xạ giữa query, key và value diễn ra như sau:

1. **Key và Value**: Đầu ra từ các trạng thái của encoder được dùng làm **keys** (k) và **values** (v).
2. **Query**: Đầu ra của decoder tại thời điểm được sử dụng làm **query** (q).
3. **Context vector**: Kết quả đầu ra từ lớp attention (gọi là , hay biến ngữ cảnh) được đưa vào để tạo ra trạng thái tiếp theo của decoder tại thời điểm .

**Ví dụ cụ thể với dịch máy**

**A diagram of a algorithm

AI-generated content may be incorrect.**

Xét một ví dụ dịch từ tiếng Pháp sang tiếng Anh.

* **Encoder** nhận đầu vào là chuỗi tiếng Pháp: *{J'aime, le, thé, .}*. Các trạng thái encoder tương ứng sẽ đóng vai trò là **keys** và **values**.
* **Decoder** đã sinh ra chuỗi tiếng Anh: *{I, love}* tại thời điểm

Tại thời điểm hiện tại , từ **love** sẽ được đưa vào lớp attention làm **query**. Query này sẽ kết hợp với các keys để tạo ra điểm số chưa chuẩn hóa theo công thức: .

Các điểm số này sau đó được chuẩn hóa bằng hàm softmax để tạo thành trọng số attention.

Tiếp theo, các trọng số này kết hợp với các **values** (đầu ra encoder) để tạo ra **vector ngữ cảnh** .

Cuối cùng, vector được kết hợp với trạng thái trước đó của decoder để tạo ra từ tiếp theo — trong ví dụ này là từ *tea* tại thời điểm .

**II. Transformer**

Transformer được thiết kế dựa trên kiến trúc encoder – decoder, kết hợp những điểm mạnh của hai dòng mạng nơ-ron:

* Mạng nơ-ron tích chập (CNN): giúp thực hiện tính toán song song, tăng tốc độ huấn luyện mô hình.
* Mạng nơ-ron hồi tiếp (RNN): giúp nắm bắt thông tin tuần tự dài hạn, có độ dài biến đổi.

Kết hợp hai đặc điểm này, Transformer trở thành một công cụ mạnh mẽ trong việc xử lý thông tin chuỗi, đặc biệt là trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP).

**A diagram of a software architecture

AI-generated content may be incorrect.**

**1. Biểu diễn nguồn và đích (Source and Target Representation)**

Trong mô hình xử lý ngôn ngữ tự nhiên (NLP), cả câu đầu vào (source) và đầu ra (target) đều được mã hóa bằng cách:

1. Tách từ thành các token.
2. Chuyển mỗi token qua word embedding.
3. Kết hợp thêm thông tin vị trí bằng mã hóa vị trí (positional encoding).

Như vậy, mỗi câu được biểu diễn bằng một chuỗi vector có thông tin ngữ nghĩa và vị trí.

**a. Word Embedding**

Word embedding sẽ chuyển đổi mỗi từ trong câu thành một vector có kích thước cố định. Nếu câu có độ dài , và chiều không gian embedding là , ta có thể biểu diễn toàn bộ câu bằng một ma trận .

**b. Positional Encoding**

Vì mô hình Transformer không có tính tuần tự như mạng RNN, nên cần có một cơ chế để cung cấp thông tin về vị trí của từ trong câu — đó chính là mã hóa vị trí (positional encoding). Mã hóa này giúp mô hình nhận biết được vị trí tương đối và tuyệt đối của từng từ.

**Giải pháp bằng hàm lượng giác**

Một phương pháp đơn giản đáp ứng các yêu cầu trên là sử dụng các hàm và .

Cho độ dài câu là , và số chiều embedding là , ma trận positional encoding sẽ có kích thước .

**Công thức tính như sau:**

Trong đó:

* : vị trí của từ trong câu (từ đến ).
* : chỉ số chiều (từ đến )

Hàm sin áp dụng cho các chiều chẵn, và cos cho chiều lẻ.

**c. Kết hợp Embedding và Positional Encoding**

Cuối cùng, hai ma trận word embedding và positional encoding được cộng lại để tạo thành biểu diễn đầu vào cuối cùng:

**2. Lớp Attention (Attention Layers)**

Lớp attention là khối xây dựng cơ bản trong mô hình Transformer, xuất hiện ở cả encoder và decoder. Thành phần cốt lõi trong các lớp này là **self-attention**. Tùy theo vị trí sử dụng (ở encoder hay decoder), sẽ có một số khác biệt nhỏ, tuy nhiên nguyên lý chung vẫn giữ nguyên.

**a. Self-Attention**

A diagram of a machine

AI-generated content may be incorrect.

**Nguyên lý hoạt động:**

Tại mỗi thời điểm, mỗi vector đầu vào ​ sẽ được chiếu qua ba ma trận trọng số (có thể học được):

* ​ để sinh ra **query vector** .​
* ​ để sinh ra **key vector** .
* ​ để sinh ra **value vector** .

Các vector này sẽ được kết hợp để tính điểm attention giữa các phần tử trong chuỗi đầu vào.

**Vai trò:**

1. **Query vector ​**: được dùng để tính điểm liên kết với tất cả key vectors trong câu:
2. **Key vector ​**: đóng vai trò xác định mức độ liên quan giữa các query và key để tạo điểm attention.
3. **Value vector ​**: mang thông tin thực tế, được kết hợp với trọng số attention để tạo ra output vector .

A diagram of a computer algorithm

AI-generated content may be incorrect.

**Công thức tính attention:**

Với input ma trận , đầu ra được tính bằng:

**b. Multi-Head Attention**

Thay vì chỉ có một self-attention, mô hình Transformer sử dụng nhiều self-attention chạy song song (gọi là các “head”).

**Lợi ích của Multi-head Attention:**

* Mỗi head có thể học các biểu diễn khác nhau trong không gian đặc trưng
* Nắm bắt các khía cạnh khác nhau của thông tin
* Tăng khả năng mô hình hóa các phụ thuộc dài hạn trong chuỗi

**Cách hoạt động:**

Mỗi head sử dụng các ma trận riêng biệt , , sau đó các đầu ra được nối lại và nhân với ma trận ​ để tạo ra output cuối cùng:

A diagram of a diagram

AI-generated content may be incorrect.

**c. Masked Multi-head Attention**

Trong decoder, cần đảm bảo rằng mô hình không được “nhìn thấy” các từ tương lai khi dự đoán từ hiện tại. Điều này được xử lý bằng cơ chế **masking**, tạo thành **masked multi-head attention**.

* Áp dụng ma trận mặt nạ , trong đó:
  + Các phần tử tương lai được gán giá trị
  + Các phần tử hiện tại và quá khứ được gán
* Công thức attention sẽ trở thành:

**d. Encoder-Decoder Multi-head Attention**

Ở phía decoder, cần học mối quan hệ attention giữa **tập đầu vào (source)** và **tập đầu ra (target)**. Cụ thể:

* Query vector được lấy từ chuỗi target (phía decoder)
* Key và Value được lấy từ chuỗi input (phía encoder)

Lớp attention này giúp decoder tham chiếu trực tiếp đến các thông tin quan trọng từ encoder để dự đoán chính xác hơn tại mỗi bước thời gian.

**3. Residuals và Layer Normalization**

Các đầu vào trong Transformer được kết nối tắt (shortcut) đến đầu ra , sau đó hai thành phần này được cộng lại và đưa qua một lớp chuẩn hóa tầng (layer normalization). Toàn bộ quá trình được biểu diễn bằng công thức:

Lớp **layer normalization** giúp đảm bảo rằng mỗi tầng trong mô hình có trung bình bằng 0 và phương sai bằng 1, nhằm ổn định quá trình huấn luyện.

**Công thức tính Layer Normalization**

Với mỗi đơn vị ẩn , đầu ra được chuẩn hóa theo công thức:

Trong đó:

* là gain variable (có thể đặt bằng )
* là giá trị trung bình:
* là độ lệch chuẩn:

**Ý nghĩa**

* Giảm hiện tượng lệch hiệp phương sai (covariance shift):  
  Đây là hiện tượng sự phụ thuộc giữa các tầng thay đổi, khiến việc huấn luyện chậm lại. Layer normalization làm giảm sự thay đổi này và giúp mô hình hội tụ nhanh hơn, giảm số lần lặp cần thiết.
* Khác biệt với Batch Normalization:  
  Batch normalization chuẩn hóa trên toàn bộ một batch, còn layer normalization chuẩn hóa trên từng đơn vị đầu ra của từng mẫu. Điều này khiến layer normalization không phụ thuộc vào kích thước batch, rất hữu ích khi mô hình xử lý một mẫu duy nhất hoặc batch nhỏ.

**4. Positionwise Feed-forward Networks**

Sau mỗi lớp attention trong encoder và decoder, Transformer sử dụng một mạng fully connected feed-forward network để xử lý riêng biệt từng vị trí trong chuỗi đầu vào.

Biến đổi này được mô tả bởi công thức:

Trong đó:

* là các ma trận trọng số
* là các vector bias
* là hàm kích hoạt ReLU

Mỗi vị trí (position) trong chuỗi đầu vào sẽ đi qua cùng một mạng feed-forward này, nhưng mỗi vị trí được xử lý độc lập.

**5. Bộ Mã Hóa (Encoder)**

Khối encoder trong Transformer bao gồm lớp lặp lại, mỗi lớp có cấu trúc gồm các thành phần sau:

**Nguyên lý hoạt động:**

* Mỗi lớp multi-head attention trong encoder thực hiện attention nội tại, tức là giữa các phần tử trong chính chuỗi đầu vào (source).
* Điều này giúp mô hình học được mối quan hệ giữa các từ trong câu đầu vào.

**Ứng dụng:**

Theo Tenney và các cộng sự, từng lớp trong encoder có thể tự học để thực hiện các tác vụ NLP truyền thống như:

* Gán nhãn từ loại (Part-of-speech tagging)
* Phân tích cấu trúc câu (Constituents)
* Phụ thuộc cú pháp (Dependencies)
* Nhận diện thực thể (Entity resolution)

**6. Bộ Giải Mã (Decoder)**

Khối decoder cũng bao gồm lớp, nhưng cấu trúc mỗi lớp gồm các thành phần sau:

**Nguyên lý hoạt động:**

* Lớp đầu tiên là **masked multi-head attention**, chỉ cho phép attention đến những token trước đó trong chuỗi đầu ra (target). Điều này đảm bảo mô hình không “nhìn trước” khi dự đoán token tiếp theo.
* Tiếp theo là **encoder-decoder attention**, kết nối thông tin giữa đầu vào và đầu ra. Ở đây, truy vấn (query) đến từ chuỗi đầu ra, trong khi khóa (key) và giá trị (value) đến từ chuỗi đầu vào.
* Sau đó là mạng FFN và các bước chuẩn hóa tương tự như trong encoder.