1.2 Cấu trúc Transformer

1.2.1 Encoder và decoder

A diagram of a sublaying process

Description automatically generated

* Kiến trúc máy biến áp bao gồm một bộ mã hóa (encoder) và một bộ giải mã(dencoder), mỗi bộ được tạo thành từ nhiều mulit-head attention và feed forward

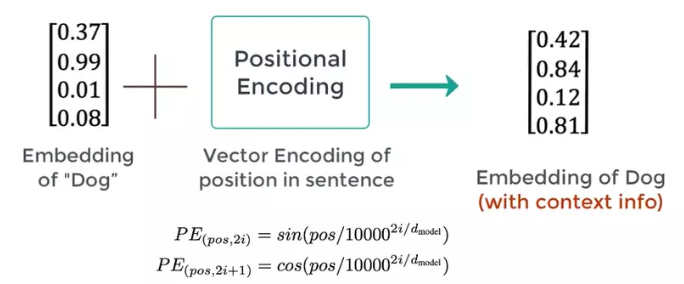
**Encoder**: là tổng hợp xếp chồng lên nhau của 6 layers xác định. Mỗi layer bao gồm 2 layers con (sub-layer) trong nó. Sub-layer đầu tiên là multi-head self-attention mà lát nữa chúng ta sẽ tìm hiểu. Layer thứ 2 đơn thuần chỉ là các fully-connected feed-forward layer

**Decoder:** cũng là tổng hợp xếp chồng của 6 layers. Kiến trúc tương tự như các sub-layer ở Encoder ngoại trừ thêm 1 sub-layer thể hiện phân phối attention ở vị trí đầu tiên.

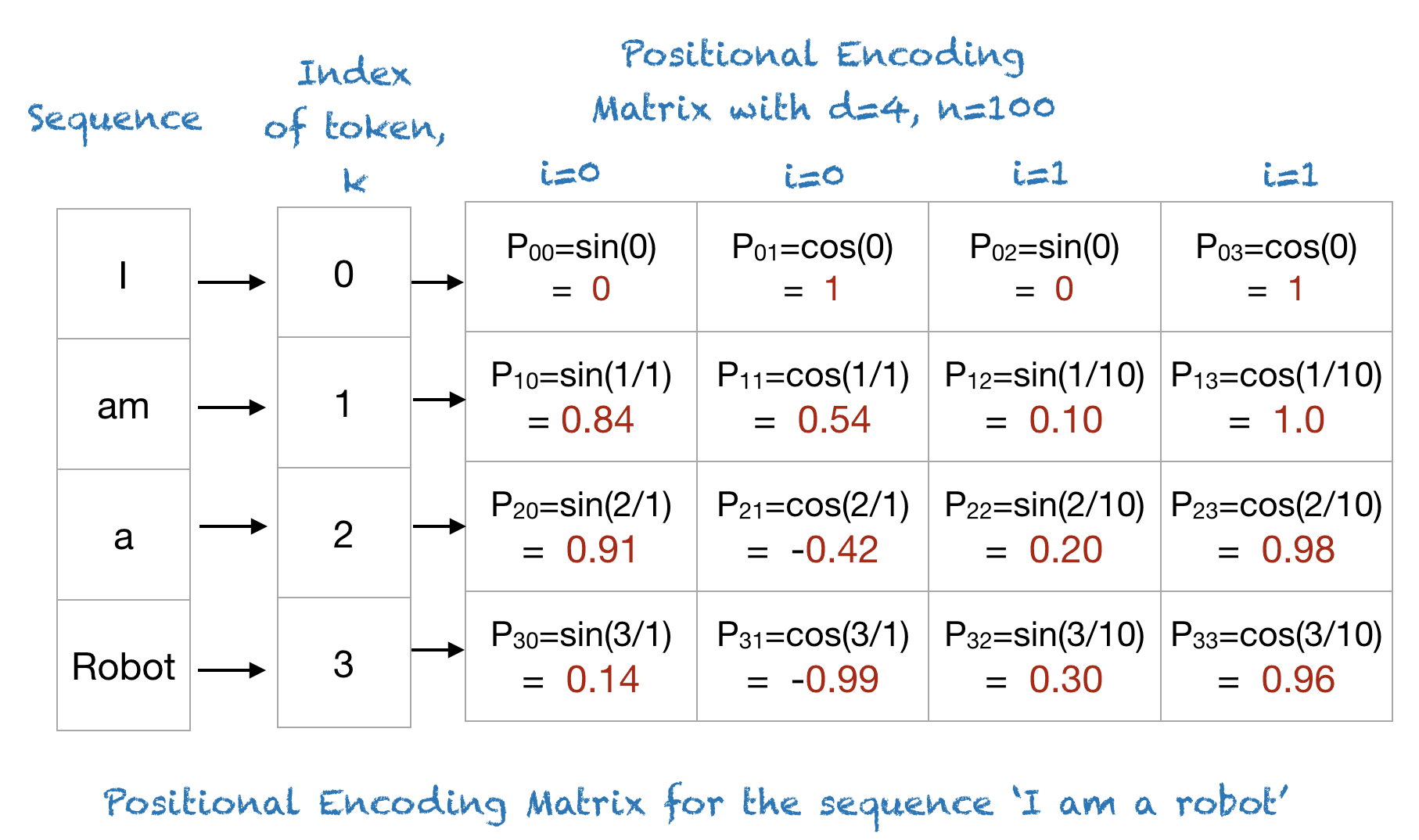
1.2.2 Positional encoding

Việc nhúng từ một phần giúp chúng ta thể hiện ngữ nghĩa của một từ, nhưng cùng một từ ở các vị trí khác nhau của câu có ý nghĩa khác nhau. Đó là lý do tại sao Transformers có thêm phần **Positional encoding** để đưa thêm thông tin. Về vị trí của một từ

Pos là vị trí của từ này trong câu, PE là giá trị của phần tử thứ được nhúng và có độ dài . Sau đó chúng ta tổng hợp vectơ PE và vectơ nhúng



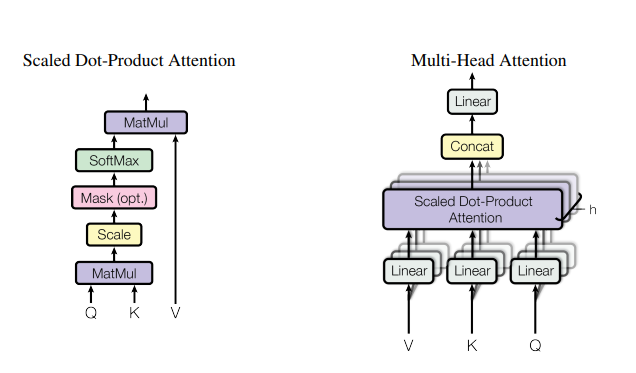
Để hiểu cách diễn đạt trên, hãy lấy ví dụ về cụm từ “I am a robot,” với n=100 và d=4. Bảng sau đây hiển thị ma trận mã hóa vị trí cho cụm từ này.



1.2.3 Attention

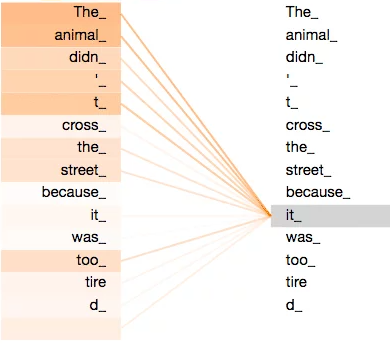
Sự chú ý (attention) trong mô hình Transformer cho phép mô hình tập trung có chọn lọc vào các phần cụ thể của chuỗi đầu vào, thay vì xử lý toàn bộ chuỗi đồng đều. Điều này giúp cải thiện hiệu quả và độ chính xác trong việc xử lý thông tin.

Trong Transformer, sự chú ý được sử dụng để ánh xạ giữa một truy vấn (query) và một tập hợp các cặp khóa-giá trị (key-value) để tạo ra một đầu ra. Cụ thể, truy vấn, khóa và giá trị đều là các vector. Đầu ra của cơ chế attention được tính như một tổng trọng số của các giá trị, trong đó trọng số được xác định bởi độ tương thích giữa truy vấn và các khóa tương ứng.



Hình 2: (trái) Scaled Dot-Product Attention. (phải) Chú ý nhiều đầu bao gồm một số

Giả sử câu sau là câu đầu vào mà chúng ta cần dịch: ***”The animal didn't cross the street because it was too tired”.*** Từ "it" trong câu trên đại diện cho điều gì? "**animal**" hay "**street**"?

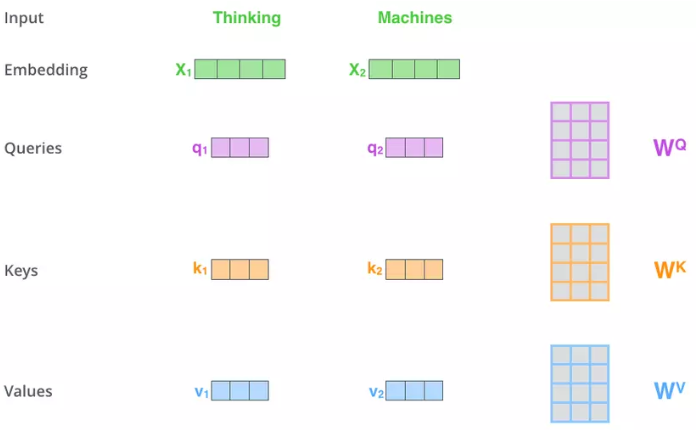
****

Câu hỏi này đơn giản đối với con người nhưng không đơn giản đối với các thuật toán. Khi mô hình xử lý từ "it", sự tự chú ý cho phép nó liên kết "it" với "animal ". Khi mô hình xử lý từng từ (mỗi vị trí trong cầu nối đầu vào), sự tự chú ý cho phép nó quan sát các vị trí khác trong câu để tìm ý tưởng để mã hóa tốt hơn từ hiện tại. Tự chú ý là những gì Transformer sử dụng để duy trì kiến thức về các từ khác có liên quan đến từ hiện tại.

1.2.3.1 Công thức self-attention

* **Trong đó:**
* Q (Query) là vector truy vấn.
* K (Key) là vector khóa.
* V (Value) là vector giá trị.
* là kích thước của vector khóa (Key).

Công thức này khá đơn giản, nó được thực hiện như sau. Đầu tiên, để có được 3 vectơ Q, K, V, các nhúng đầu vào được nhân với 3 ma trận trọng lượng tương ứng (được điều chỉnh trong quá trình đào tạo) .



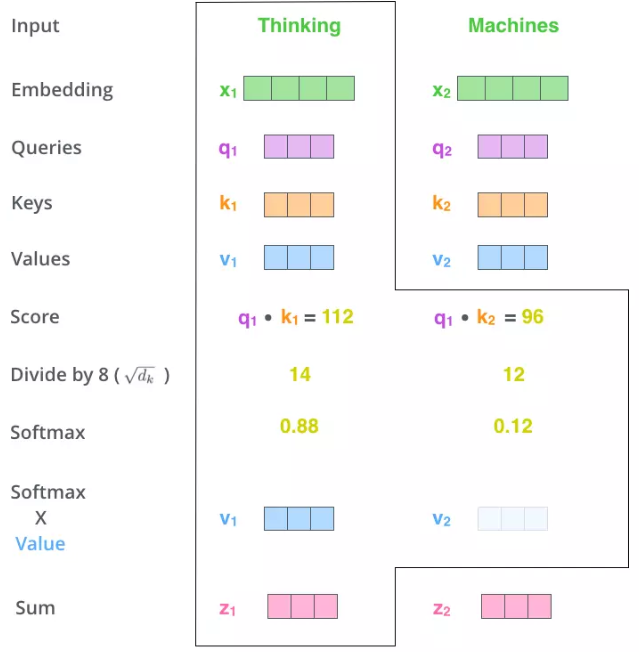
Tại thời điểm này, vectơ K hoạt động như một khóa đại diện cho từ và Q sẽ truy vấn các vectơ K của các từ trong câu bằng cách tích chập với các vectơ này. Mục đích của phép chập là để tính toán mối quan hệ giữa các từ. Theo đó, hai từ liên quan đến nhau sẽ có kết quả lớn và ngược lại.

**Bước thứ hai** là bước "Scale ", chỉ đơn giản là chia "Score" cho căn bậc hai của chiều Q / K / V (trong hình chia cho 8 vì Q / K / V là vectơ 64-D). Điều này làm cho giá trị "Score" không phụ thuộc với độ dài của vectơ Q/K/V

**Bước thứ ba** là softmax các kết quả trước đó để có được phân phối xác suất trên các từ.

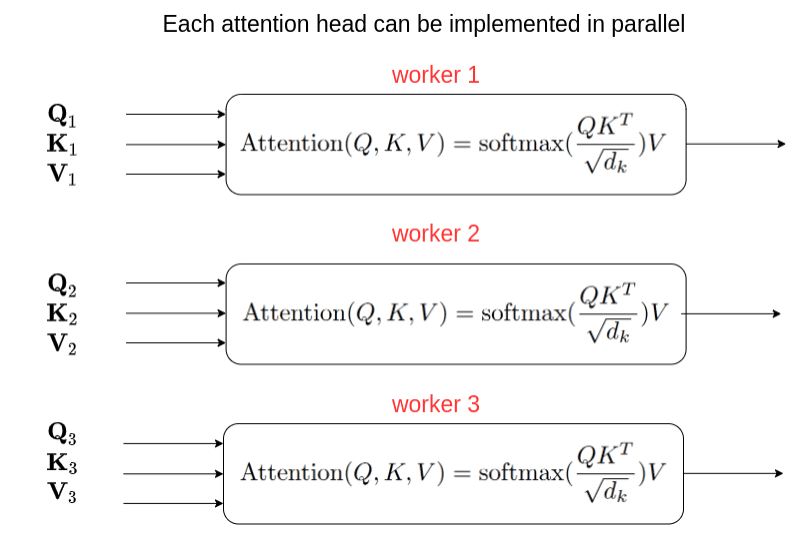
**Bước thứ tư** là nhân phân bố xác suất đó với vectơ V để loại bỏ các từ không cần thiết (xác suất nhỏ) và giữ lại các từ quan trọng (xác suất lớn).

Trong bước **cuối cùng**, các vectơ V (nhân với đầu ra softmax) được cộng lại với nhau, tạo ra vectơ chú ý Z cho một từ. Lặp lại quy trình trên cho tất cả các từ, chúng ta có được ma trận chú ý cho 1 câu.



1.2.3.2 Multi-head attention

Multi-Head Attention trong Transformer cho phép mô hình học và tích hợp thông tin từ nhiều khía cạnh khác nhau của chuỗi đầu vào một cách hiệu quả. Thay vì thực hiện một phép tính attention duy nhất, Multi-Head Attention chia quá trình này thành nhiều **"đầu" attention song song.** Các truy vấn (query), khóa (key) và giá trị (value) được chiếu tuyến tính thành nhiều không gian khác nhau bằng các trọng số khác nhau, tạo ra nhiều phiên bản. Trên mỗi phiên bản này, hàm Attention được thực hiện song song, thu được các giá trị đầu ra. Các đầu ra này sau đó được nối lại và chiếu tuyến tính một lần nữa để tạo ra giá trị đầu ra cuối cùng.



1.2.4 Residuals

Residuals, hay còn gọi là kết nối bỏ qua (skip connections), là một kỹ thuật được sử dụng trong nhiều mô hình học sâu, bao gồm cả Transformer. Ý tưởng chính của kỹ thuật này là cho phép thông tin từ đầu vào đi qua các lớp (layers) mà không bị biến đổi, bằng cách thêm đầu vào trực tiếp vào đầu ra của lớp đó.

Cụ thể, đầu vào (x) được thêm vào đầu ra của lớp attention hoặc lớp feed-forward network, và kết quả này được đưa qua một lớp Chuẩn hóa lớp (Layer Normalization). Công thức tổng quát cho quá trình này như sau:

A diagram of a process

Description automatically generated

Output=LayerNorm (x + SubLayer(x))

* Trong đó:
* x là đầu vào của lớp con
* SubLayer(x) là đầu ra của lớp con (có thể là attention hoặc feed-forward network).
* LayerNorm là lớp Chuẩn hóa lớp được áp dụng sau khi thêm đầu vào x và đầu ra SubLayer(x)
* Lợi ích:
* Giữ Lại Thông Tin Gốc: Bằng cách thêm trực tiếp đầu vào x vào đầu ra SubLayer(x), thông tin gốc từ đầu vào được giữ lại và truyền qua các lớp, giúp mô hình học được các đặc trưng quan trọng từ đầu vào một cách hiệu quả hơn.
* Giảm Gradient Vanishing: Residuals giúp giảm vấn đề biến mất gradient trong quá trình huấn luyện các mô hình sâu, giúp cải thiện hiệu quả học tập và tăng tốc độ hội tụ.

1.2.5 Feed forward

* Ngoài các lớp con chú ý, mỗi lớp trong bộ mã hóa và bộ giải mã của chúng tôi chứa một mạng chuyển tiếp nguồn cấp dữ liệu được kết nối đầy đủ, được áp dụng cho từng vị trí riêng biệt và giống hệt nhau. Điều này bao gồm hai phép biến đổi tuyến tính với kích hoạt ReLU ở giữa.
* Mặc dù các phép biến đổi tuyến tính giống nhau ở các vị trí khác nhau, nhưng chúng sử dụng các tham số khác nhau từ lớp này sang lớp khác. Một cách khác để mô tả điều này là hai kết cấu với kích thước hạt nhân 1. Kích thước của đầu vào và đầu ra là = 512 và lớp bên trong có chiều = 2048.
* A screenshot of a computer

  Description automatically generated
* Vì các vectors này không phụ thuộc vào nhau nên ta có thể tận dụng được tính toán song song cho cả câu.

1.2.6 Decoder

1.2.6.1 Masked Multi-head Attention

Giả sử bạn muốn Transformers thực hiện bài toán dịch Anh-Pháp, thì công việc của Bộ giải mã là giải mã thông tin từ Bộ mã hóa và tạo ra từng từ tiếng Pháp dựa trên PREVIOUS WORDS. Vì vậy, nếu chúng ta sử dụng Multi-head attention trên toàn bộ câu như trong Encoder, Decoder sẽ "nhìn thấy" từ tiếp theo mà nó cần dịch (Nó sẽ nhìn thấy từ tiếp theo mà không có học học để dịch ra). Để ngăn chặn điều đó, khi Bộ giải mã dịch sang từ từ tôi , phần sau của câu tiếng Pháp sẽ bị che và Bộ giải mã sẽ chỉ được phép "xem" phần mà nó đã dịch trước đó.A diagram of a group of words

Description automatically generated with medium confidence

1.2.6.2 Decode proccess

Quá trình giải mã về cơ bản giống như mã hóa, ngoại trừ Bộ giải mã giải mã từng từ một và đầu vào của Bộ giải mã (câu tiếng Pháp) được che giấu. Sau khi đầu vào ẩn được truyền qua **sub-layer#1** của Bộ giải mã, nó sẽ không còn nhân với 3 ma trận trọng số để tạo ra Q, K, V mà sẽ chỉ nhân với 1 ma trận trọng lượng WQ. K và V được lấy từ Bộ mã hóa cùng với Q từ Masked multi-head attention và được đưa vào các sub-layers # 2 và # 3 tương tự như Bộ mã hóa. Cuối cùng, các vectơ được đẩy vào lớp tuyến tính (là một mạng được kết nối đầy đủ) tiếp theo là Softmax để tạo ra xác suất của từ tiếp theo.

Hai hình dưới đây mô tả trực quan quá trình mã hóa và giải mã Transformers

A screenshot of a computer

Description automatically generated