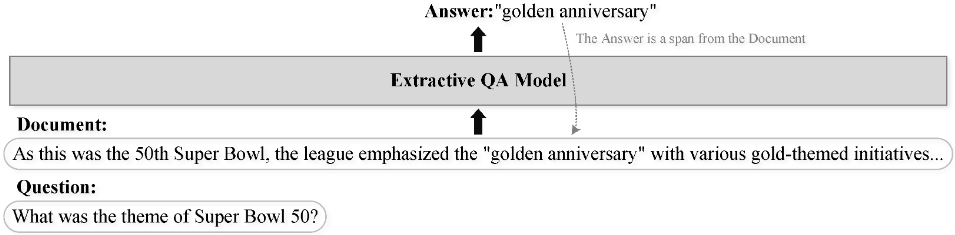
**1. Lý thuyết chung:**

**1.1. Tổng quan:**

**Giới thiệu**: Trả lời câu hỏi trích xuất (Extractive Question Answering), còn được biết đến như là một phần của khả năng hiểu ngôn ngữ tự nhiên của máy tính, có thể được sử dụng để đánh giá mức độ hiểu ngôn ngữ của người dùng. Đây là một chủ đề có giá trị với nhiều ứng dụng, từ chatbot cho đến hỗ trợ cá nhân. Các mô hình mạng nơ-ron dựa trên end-to-end đã đạt được hiệu suất xuất sắc trong các nhiệm vụ này. Cách tiếp cận thường được sử dụng nhất để trích xuất câu trả lời bằng cách sử dụng mạng nơ-ron là dự đoán vị trí bắt đầu và kết thúc của câu trả lời trong tài liệu, có thể là độc lập hoặc cùng nhau.

****

**Hoạt động như thế nào**: Chúng ta được cung cấp 2 dữ liệu: một câu hỏi và một ngữ cảnh, ví dụ như một bài viết đầy đủ trên Wikipedia có thể chứa câu trả lời cho câu hỏi. Mục tiêu của chúng ta là tạo ra một câu trả lời cho câu hỏi dựa trên ngữ cảnh. Trong trường hợp trả lời câu hỏi trích xuất, chúng ta giả định rằng câu trả lời là một tập hợp con của ngữ cảnh, vì vậy chúng ta có thể định nghĩa nó như là dự đoán khoảng (span prediction), ví dụ như một dải các ký tự hoặc token. Chúng ta cũng có thể định dạng điều này như là một nhiệm vụ trừu tượng, nơi chúng ta chỉ muốn lấy câu trả lời, có thể được diễn đạt khác với ngữ cảnh của chúng ta.

**1.2. Các phương pháp có thể sử dụng để giải quyết vấn đề:**

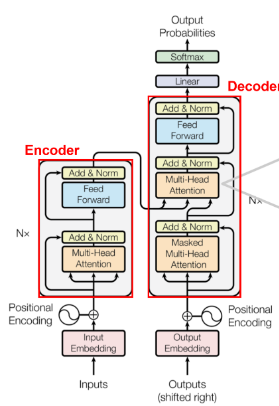
**1.2.1. Seq2Seq**

Trong cách tiếp cận sequence to sequence, liên quan đến trả lời câu hỏi trừu tượng, chúng ta sử dụng mô hình ngôn ngữ tự hồi quy để tạo ra câu trả lời.

Chúng ta có thể sử dụng chỉ câu hỏi, hoặc kết hợp ngữ cảnh và câu hỏi làm đầu vào. Chúng ta huấn luyện một mô hình để nó học cách tạo ra câu trả lời như là một phần tiếp nối của đầu vào đã cho.

Trong quá trình suy luận, chúng ta tạo ra câu trả lời từng bước, sử dụng mô hình ngôn ngữ để dự đoán token có khả năng cao nhất cho đến khi nó tạo ra token đại diện cho kết thúc của chuỗi.

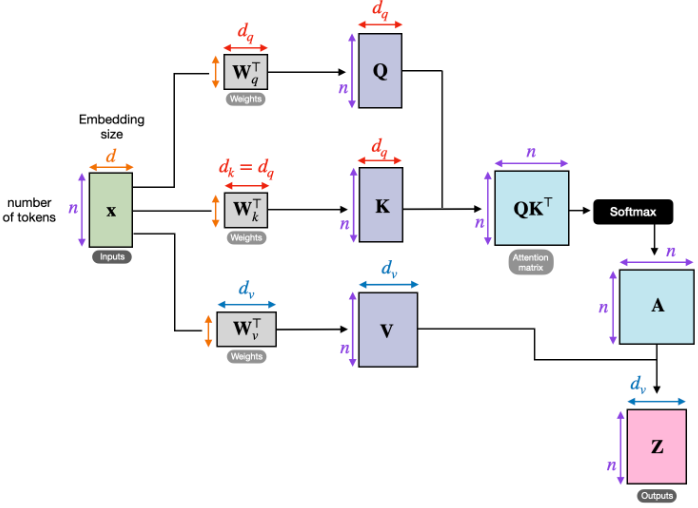
**Encoder**:

* **Đầu vào**: là một chuỗi dữ liệu đầu vào. Phần đầu tiên của kiến trúc Transformer là embedding, với hai thành phần là Word Embedding và Positional Encoding.
  + **Word Embedding**: là một phương pháp biểu diễn từ trong câu thành các vector đặc trưng thông qua ánh xạ hợp lý của các từ trong từ vựng vào không gian vector, sao cho các vector này đại diện cho mối quan hệ giữa các từ. (text -> word tokenized -> word vectors)
  + **Positional Encoding**: Vì mô hình Transformer không tính toán tuần tự thứ tự của các từ trong câu, phương pháp Positional Encoding cung cấp thông tin về vị trí của các từ trong câu cho mô hình.

Thông tin thu được từ Positional Encoding sẽ được thêm vào các vector embedding từ của các từ trong câu nguồn và câu đích. Sau khi đi qua Word Embedding và Positional Encoding, chúng ta nhận được ma trận X.

**Encoder**: là sự kết hợp của Word Embedding, Positional Encoding và một loạt các lớp Encoder liên tiếp N. Trong đó, lớp Encoder bao gồm nhiều thành phần như Multi-Head Attention, Feed Forward, skip connection và Layer Normalization.

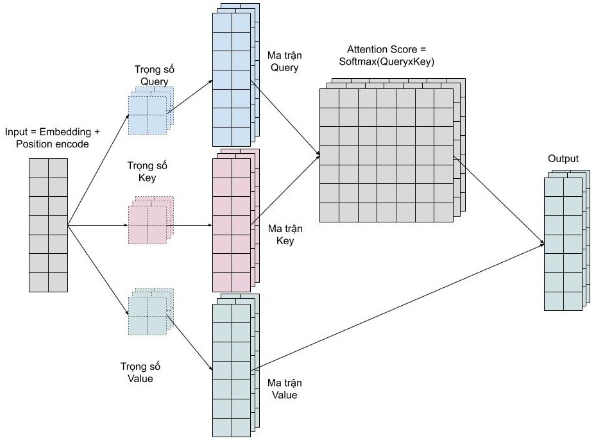
**+ Multi-Head Attention** thực chất là self-attention, nhưng để mô hình có thể chú ý đến nhiều mẫu khác nhau, đơn giản bằng cách sử dụng nhiều self-attention. Self-Attention có 4 bước:



1. Tạo một bộ 3 vector từ các vector đầu vào của encoder: Tại encoder đầu tiên, vector đầu vào là word embedding của từ. Vì vậy, đối với mỗi từ, chúng ta sẽ có 3 vector Query, Key và Value. Những vector này được tạo ra bằng cách nhân ma trận giữa vector đầu vào và 3 ma trận trọng số tương ứng với Query, Key và Value mà chúng ta sử dụng trong quá trình huấn luyện. Ba vector này đóng vai trò khác nhau và đều quan trọng đối với attention.
2. Tính điểm: Đối với mỗi từ, chúng ta cần tính điểm của các từ khác trong câu so với từ này. Giá trị này giúp quyết định từ nào cần được chú ý và chú ý bao nhiêu khi mã hóa một từ. Điểm được tính bằng tích chấm giữa vector Query của từ đang xét và các vector Key của các từ trong câu. Ví dụ, khi chúng ta tính self-attention trên một từ ở vị trí 1, điểm của nó với chính nó là q1.k1, điểm của nó với từ thứ hai là q1.k2, v.v.
3. Chuẩn hóa điểm: Trong bài viết gốc, điểm được chia cho 8 (căn bậc hai của 64 – kích thước của vector Key). Điều này làm cho độ dốc ổn định hơn. Tiếp theo, giá trị này được đưa qua hàm softmax để đảm bảo rằng các giá trị điểm đều dương và tổng không vượt quá 1.
4. Nhân vector Value với mỗi giá trị điểm đã tính ở trên và sau đó cộng chúng lại. Ý tưởng là bảo toàn giá trị vector của các từ cần chú ý và loại bỏ vector của các từ không liên quan (bằng cách nhân nó với một số rất nhỏ, chẳng hạn như 0.001).

**Multi-Head Attention** cho phép mô hình chú ý đến các mẫu dễ quan sát đồng thời như sau:

* Chú ý đến từ trước đó của một từ.
* Chú ý đến từ tiếp theo của một từ.
* Chú ý đến các từ liên quan của một từ.

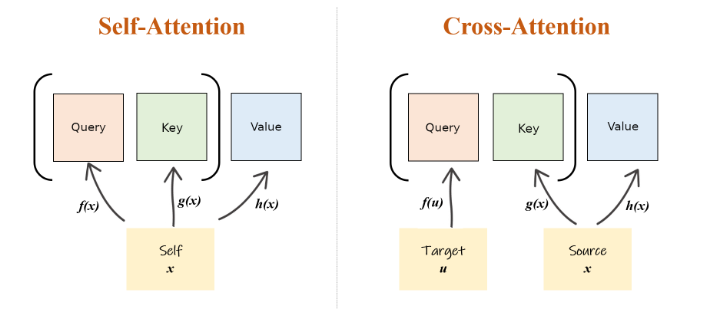
 *Multi Head Attention*

**Feed Forward Network (FFN)** trong Transformer sử dụng 2 lớp fully connected với số lượng đơn vị và hàm kích hoạt ReLU trong lớp đầu tiên. Đầu vào của FFN cũng là một ma trận.

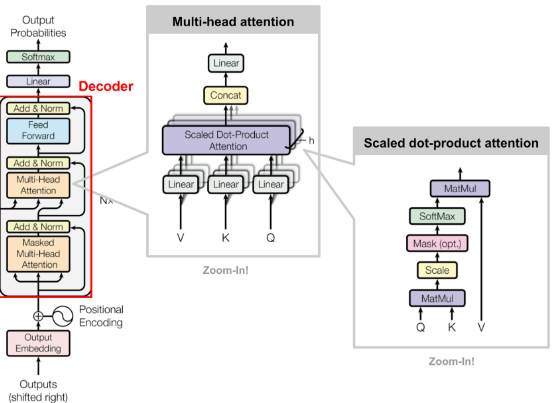
* Đầu ra của encoder sẽ là một ma trận có cùng hình dạng với ma trận đầu vào X của Encoder. Như vậy, chúng ta có thể tưởng tượng rằng Encoder đang thực hiện nhiệm vụ thêm các đặc trưng quan trọng vào vector embedding gốc của các từ trong câu.

**Decoder**: cũng có tổ chức tương tự như Encoder:

1. Bắt đầu với Word Embedding và Positional Encoding.
2. Sau đó là một loạt các lớp Decoder liên tiếp N. Trong lớp Decoder, chúng ta sẽ thực hiện cả Self-Attention và Cross-Attention. Cụ thể:



* **Self-Attention** được thực hiện đầu tiên để thực hiện Attention đến các vị trí trước vị trí hiện tại trong câu, có nghĩa là chúng ta đang sử dụng Masked Multi-Head Attention.
* Sau đó, **Cross-Attention** được thực hiện với hai thành phần K và V được tính từ đầu ra của Encoder. Lúc này, chúng ta đang thực hiện Attention đến tất cả các vị trí trong đầu ra của Encoder, do đó Multi-Head Attention được sử dụng.



**Lưu ý**:

- Trước khi đi đến lớp fully connected cuối cùng của Decoder để thực hiện phân loại, ma trận đầu ra mà chúng ta nhận được cũng chính là ma trận đầu ra của lớp Encoder.

- Kích thước của vector đầu ra của lớp fully connected cuối cùng sẽ bằng kích thước của từ điển của ngôn ngữ đích.

- Lớp Add&Norm: Lớp này đơn giản là chuẩn hóa đầu ra của multi-head attention, hiệu quả cải thiện sự hội tụ.

3. Phần cuối cùng của Decoder là một lớp fully connected với hàm softmax.

- Lớp tuyến tính ánh xạ vector ngữ cảnh vào không gian có cùng kích thước với kích thước từ vựng. Hàm softmax sau đó chuyển đổi các giá trị kết quả thành xác suất, đại diện cho khả năng mỗi token trong từ vựng là token tiếp theo.

- Cuối cùng, từ dự đoán là từ có xác suất cao nhất trong phân phối xác suất.

- Token dự đoán được thêm vào chuỗi đầu vào và quá trình này lặp lại cho bước mã hóa tiếp theo cho đến khi đạt đến độ dài mong muốn hoặc điều kiện dừng.

**1.2.2. Vị trí Bắt đầu/Kết thúc**

Phương pháp này liên quan đến trả lời câu hỏi trích xuất vì chúng ta đang cố gắng xác định khoảng - các chỉ số token bắt đầu và kết thúc - đại diện cho câu trả lời cho câu hỏi của chúng ta.

Để làm điều này, chúng ta kết hợp câu hỏi và câu trả lời với nhau, và đưa nó qua một mô hình mà thông thường đó là một transformer với 2 đầu ra tuyến tính tương ứng với START và END logits.

Đối với mỗi token, chúng ta thu được xác suất rằng nó là token BẮT ĐẦU của khoảng câu trả lời (START logits) và nó là token KẾT THÚC của khoảng câu trả lời (END logits). Dựa trên các xác suất đó, chúng ta có thể chọn khoảng có khả năng nhất và sử dụng nó làm dự đoán câu trả lời của chúng ta.