



Transformer model in Sequence to Sequence translation

Nhóm 1:

Trần Thanh Trà.

Phạm Thanh Đạt

Nguyễn Kim Đức

Hoàng Gia Anh Đức

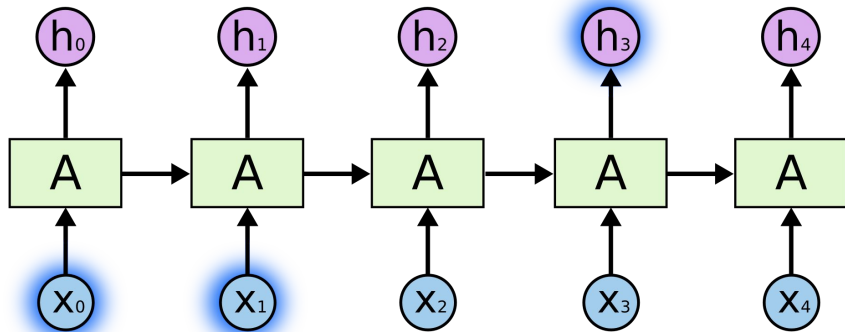


1. Đặt vấn đề

Sequence-to-Sequence Learning (phương pháp học chuỗi liên tiếp) sử dụng Neural Networks là một kỹ thuật rất mạnh được sử dụng để giải quyết rất nhiều vấn đề trong Machine Learning, trong đó có vấn đề về dịch thuật từ một đoạn văn bản trong ngôn ngữ này sang đoạn văn bản tương ứng trong ngôn ngữ khác.

2. Cách tiếp cận

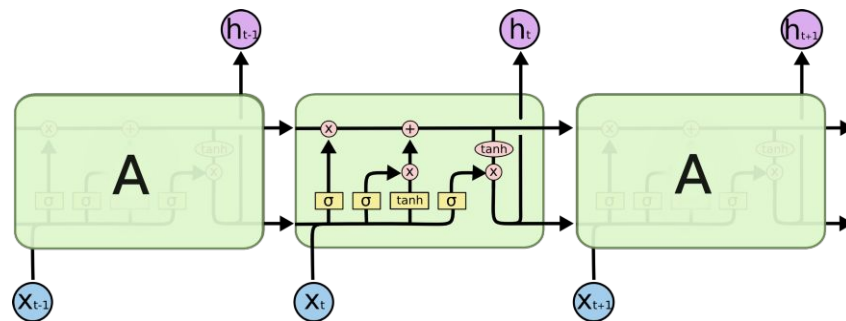
- RNN (Recurrent Neural Networks):



→ Vấn đề xử lý không tốt với những câu dài do hiện tượng Gradient Vanishing/Exploding và train rất chậm

2. Cách tiếp cận

- LSTM (Long-Short Term Memory)



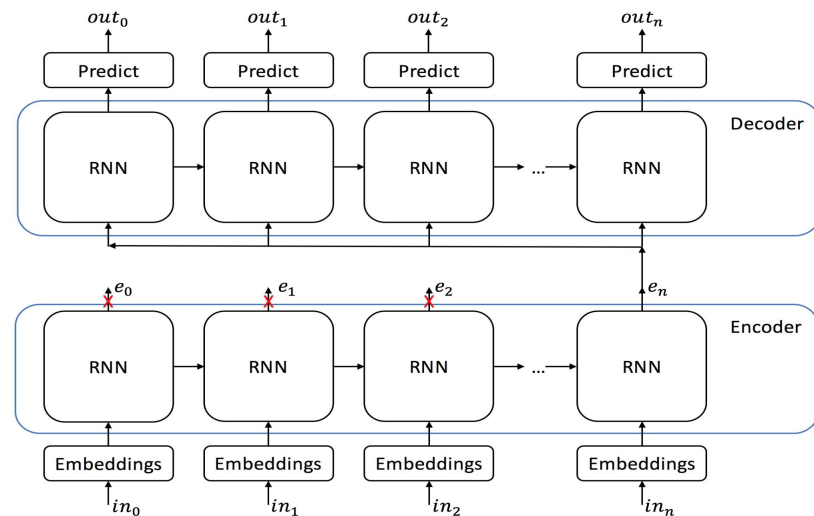
→ Nhưng LSTM vẫn còn tồn tại vấn đề như đã nêu ở RNN, là khi câu quá dài thì LSTM vẫn sẽ đưa ra kết quả không chính xác.

- Một vấn đề khác của RNN và LSTM là chỉ dịch word by word mà không thể xử lý song song được. Từ đó không tận dụng được GPU.

3. Phân tích thiết kế Transformer

- Cơ chế Attention:
 - Giúp mô hình tập trung vào các từ cụ thể
 - Mô hình RNN khi không áp dụng Attention:

→ Mô hình encoder phải nén tất cả thông tin của một câu lại thành một vector biểu diễn duy nhất, chứa toàn bộ thông tin cần thiết để mô hình decoder có thể dịch thành câu đích

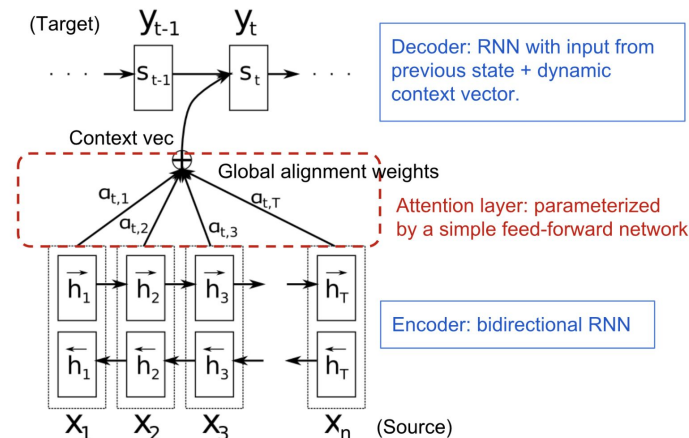


3. Phân tích thiết kế Transformer

- Cơ chế Attention:
 - Mô hình RNN khi áp dụng Attention

→ chúng ta sẽ sử dụng cơ chế attention, cho phép mô hình có thể chú ý vào từng phần của câu hoặc bức ảnh một cách rõ ràng. Từ đó, thông tin không cần phải nén vào một vector biểu diễn duy nhất.

Ngoài ra, cơ chế attention cho phép mình có thể hiểu được những từ hay phần ảnh nào quyết định đến kết quả hiện tại.

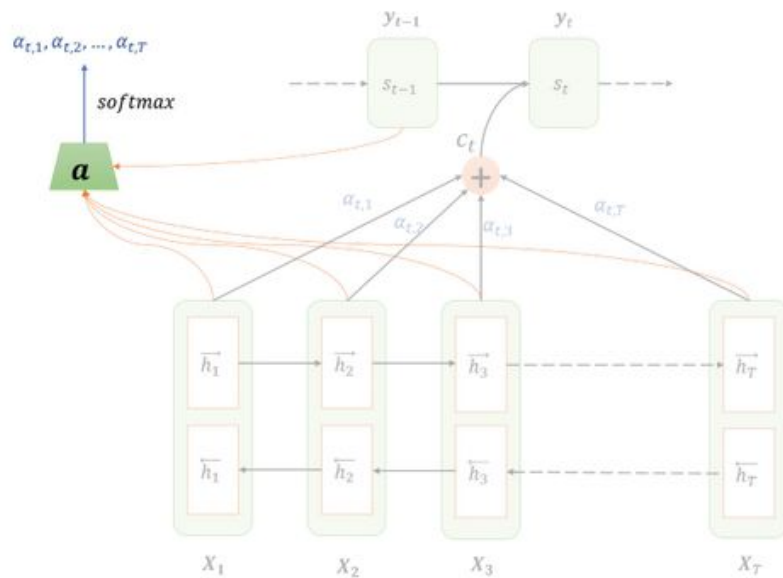


Additive Attention

$$\alpha_{t'} = \text{NeuralNet}([s_{t-1}, h_{t'}]), \quad t' = 1 \dots T_x$$

$$\text{context} = \sum_{t'=1}^{T_x} \alpha(t') h(t')$$

Calculating attention weights and creating the context vector using those attention values with encoder state outputs



3. Phân tích thiết kế Transformer

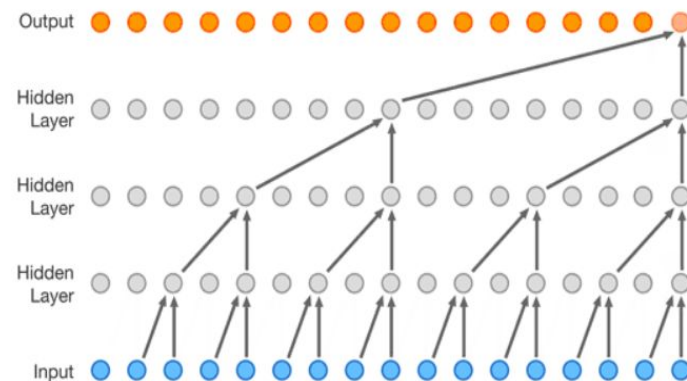
- Convolutional Neural Network

Mạng tích chập CNN là giải pháp cho tính toán song song.

Một số mạng nổi tiếng trong biến đổi chuỗi sử dụng CNN có thể kể đến như Wavenet và Bytenet.

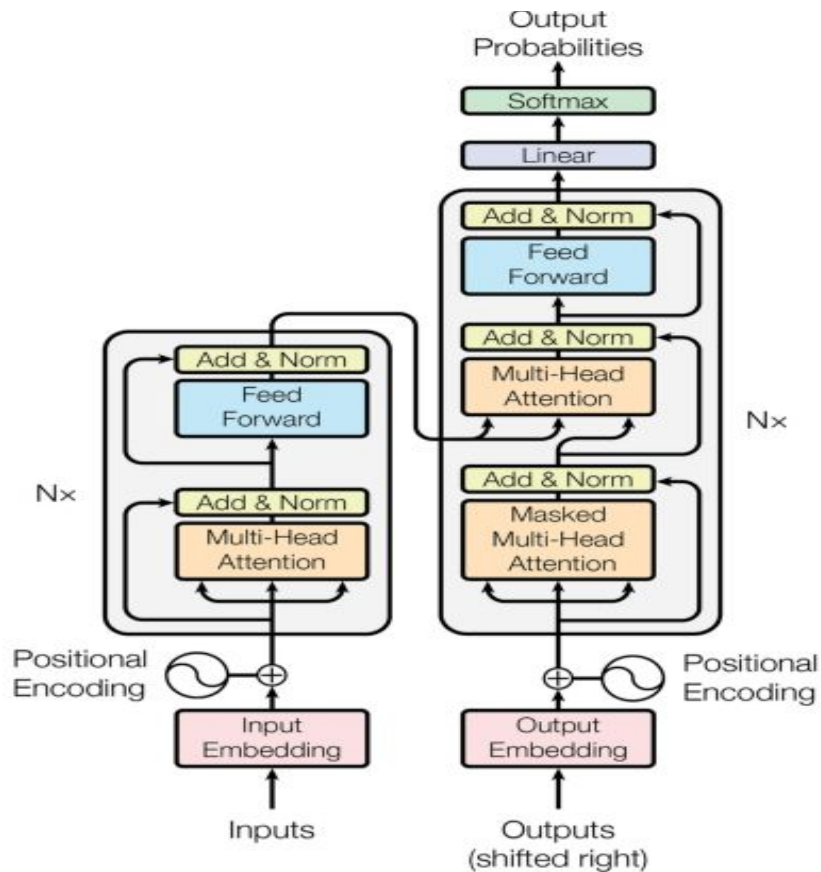
Lý do CNN có thể tính toán song song là các từ được xử lý cùng lúc và không cần chờ đợi nhau. Không chỉ có vậy,

khoảng cách giữa một từ đầu ra với một từ đầu vào là $\log(N)$, thay vì N như trong RNN.



3. Phân tích thiết kế Transformer

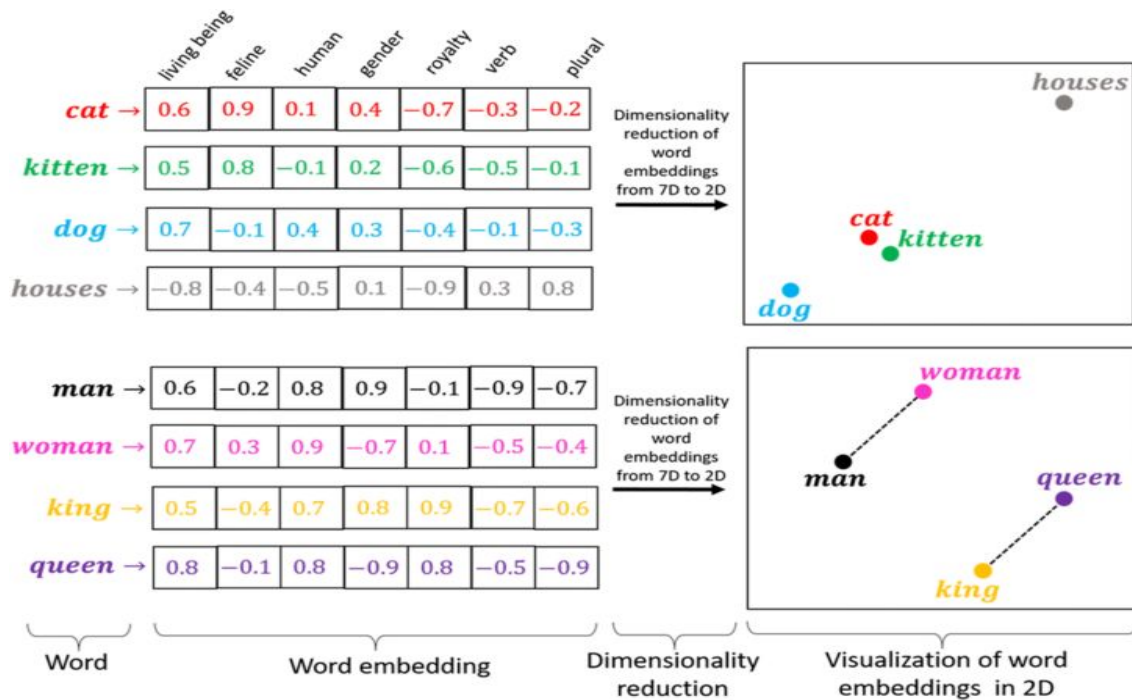
- Kiến trúc mô hình Transformer



3. Phân tích thiết kế Transformer

- Word Embedding

- Đầu tiên, các từ được biểu diễn bằng một vector sử dụng một ma trận word embedding có số dòng bằng kích thước của tập từ vựng. Sau đó các từ trong câu được tìm kiếm trong ma trận này, và được nối nhau thành các dòng của một ma trận 2 chiều chứa ngữ nghĩa của từng từ riêng biệt.



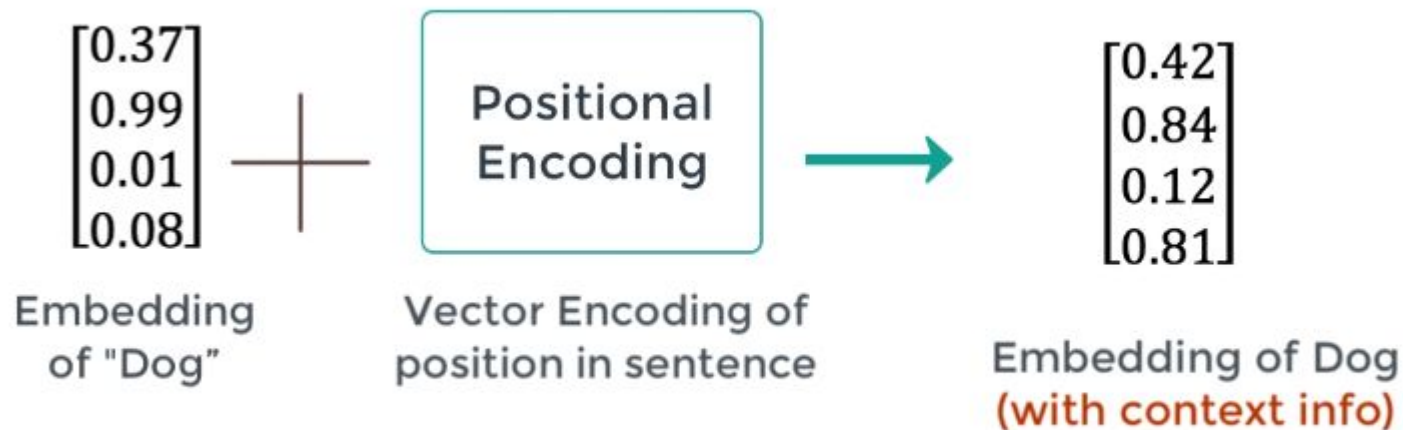
Positional Encoder : vector that gives context based on position of word in sentence

AJ's **dog** is a cutie  Position 2

AJ looks like a **dog**  Position 5

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

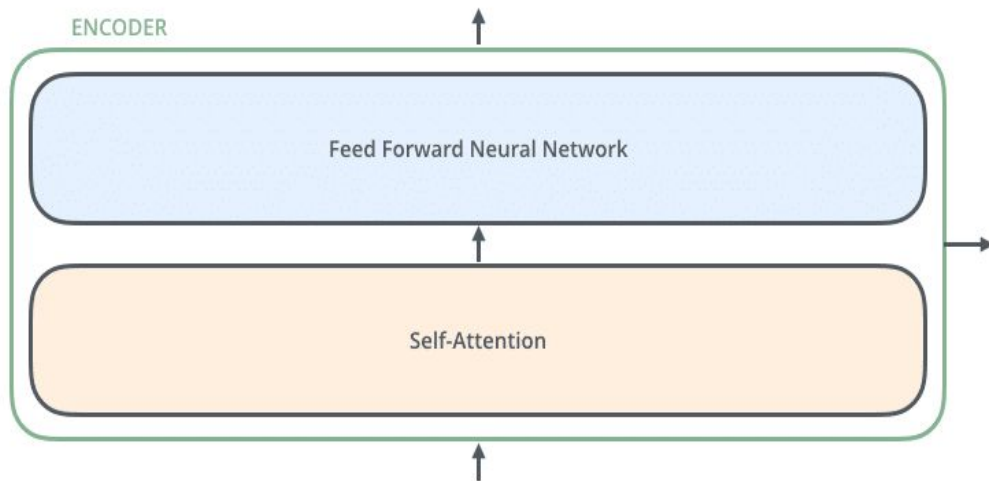
$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$



$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

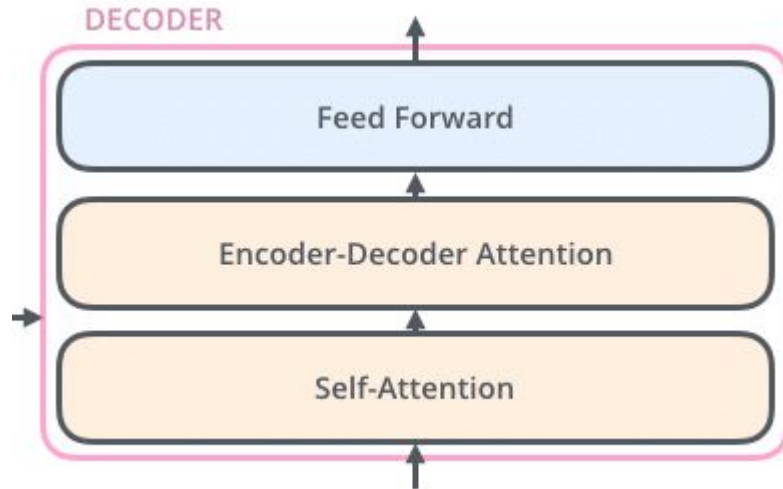
3. Phân tích thiết kế Transformer

Các encoder đều rất giống nhau, có cùng kiến trúc. Mỗi encoder chứa hai lớp: Self-attention và mạng truyền thẳng (FNN).



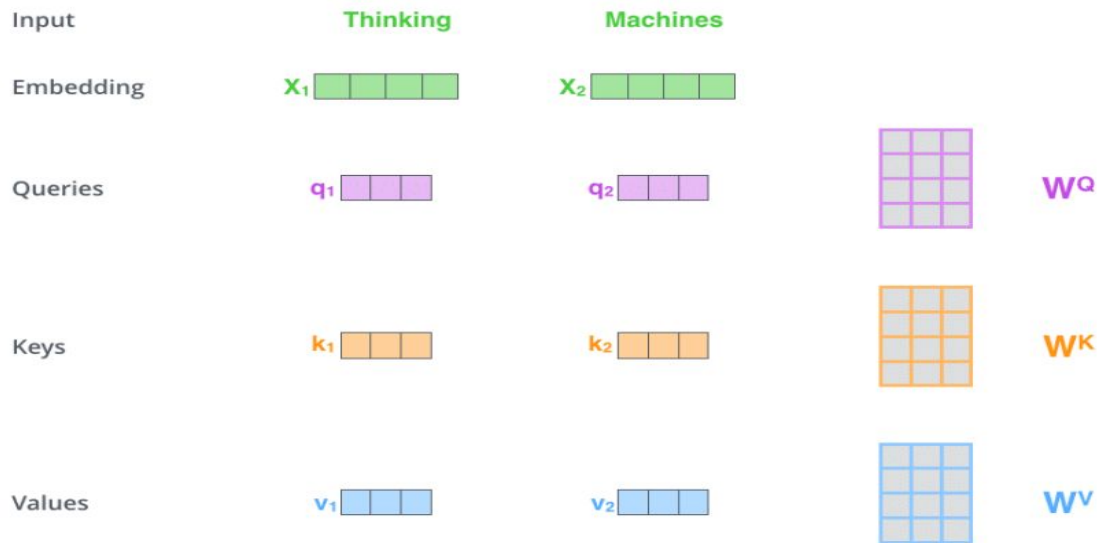
3. Phân tích thiết kế Transformer

Các decoder cũng có kiến trúc giống như vậy nhưng giữa chúng có một lớp attention để nó có thể tập trung vào các phần liên quan của đầu vào.



3. Phân tích thiết kế Transformer

- Self Attention:
 - Bước đầu tiên để tính self-attention là tạo ra bộ 3 vector từ các vector đầu vào của encoder



3. Phân tích thiết kế Transformer

- Self Attention:
 - Bước thứ hai là tính điểm

Input

Embedding

Queries

Keys

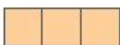
Values

Score

Thinking

x_1 

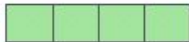
q_1 

k_1 

v_1 

$q_1 \cdot k_1 = 112$

Machines

x_2 

q_2 

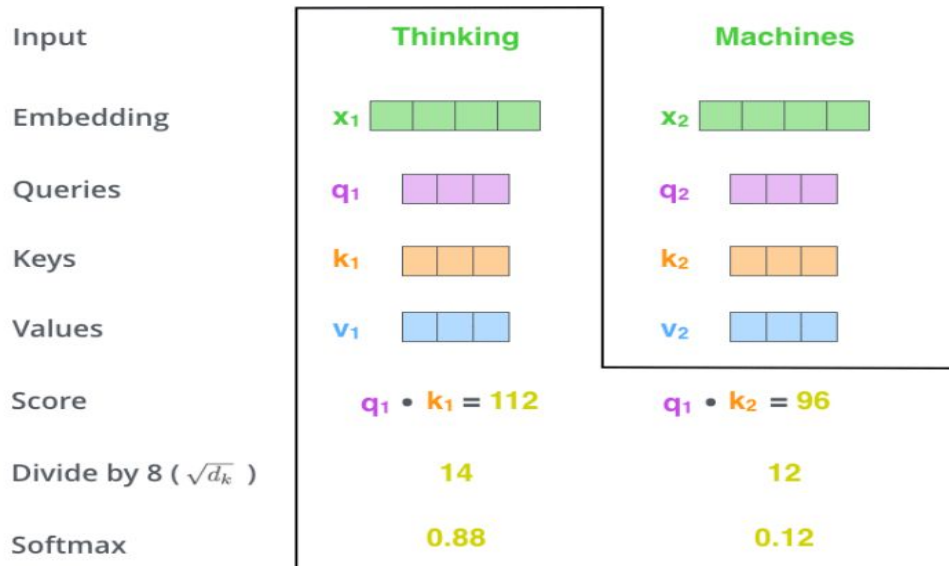
k_2 

v_2 

$q_1 \cdot k_2 = 96$

3. Phân tích thiết kế Transformer

- Self Attention:
 - Bước tiếp theo là chuẩn hóa điểm



3. Phân tích thiết kế Transformer

- Self Attention
 - Bước tiếp theo là nhân vector *Value*

với mỗi giá trị điểm đã tính phía trên rồi

cộng lại với nhau

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 ($\sqrt{d_k}$)

Softmax

Softmax
X
Value

Sum

Thinking

x_1 [] [] [] []

q_1 [] [] []

k_1 [] [] []

v_1 [] [] []

$q_1 \cdot k_1 = 112$

14

0.88

v_1 [] [] []

z_1 [] [] []

Machines

x_2 [] [] [] []

q_2 [] [] []

k_2 [] [] []

v_2 [] [] []

$q_1 \cdot k_2 = 96$

12

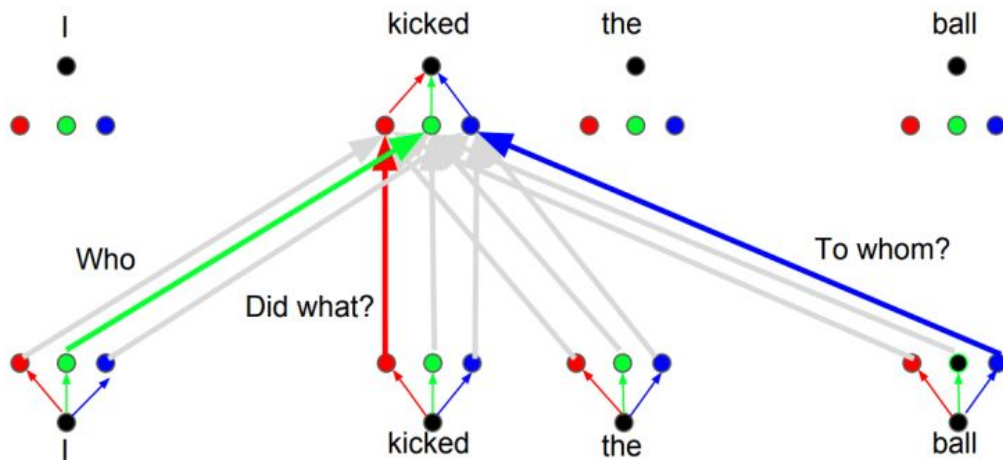
0.12

v_2 [] [] []

z_2 [] [] []

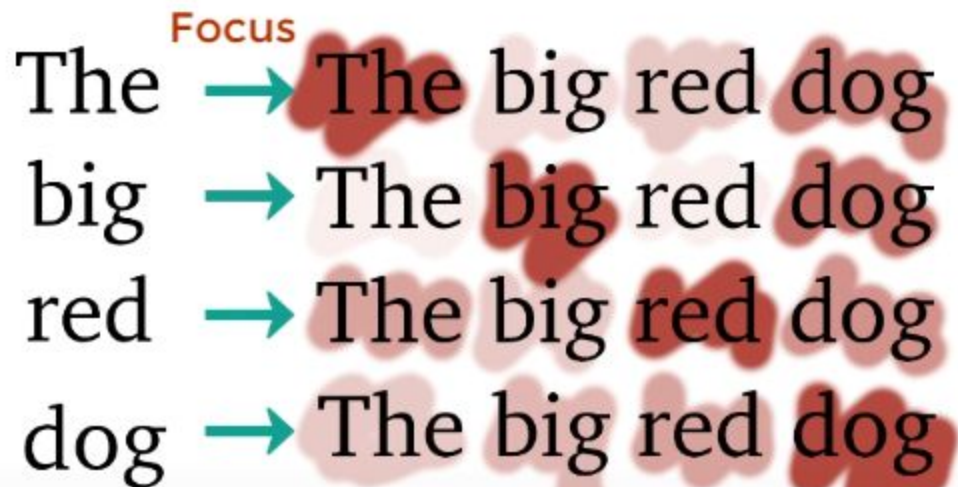
3. Phân tích thiết kế Transformer

- Multihead attention:
 - Ý tưởng đằng sau kỹ thuật này là một từ có thể có nhiều nghĩa hoặc nhiều cách thể hiện khác nhau khi dịch ra một ngôn ngữ khác



Focus

The → The big red dog
big → The big red dog
red → The big red dog
dog → The big red dog

The diagram illustrates the attention mechanism for the sentence "The big red dog". On the left, the words "The", "big", "red", and "dog" are listed vertically. To their right, the full sentence "The big red dog" is repeated four times, corresponding to each word on the left. A teal arrow points from each word on the left to the first word "The" in the repeated sentence. Red, semi-transparent blobs of varying intensity are overlaid on the words of the repeated sentences, representing attention weights. The word "The" in each repetition has the highest attention weight, followed by "big", "red", and "dog". The word "Focus" is written in red above the first "The".

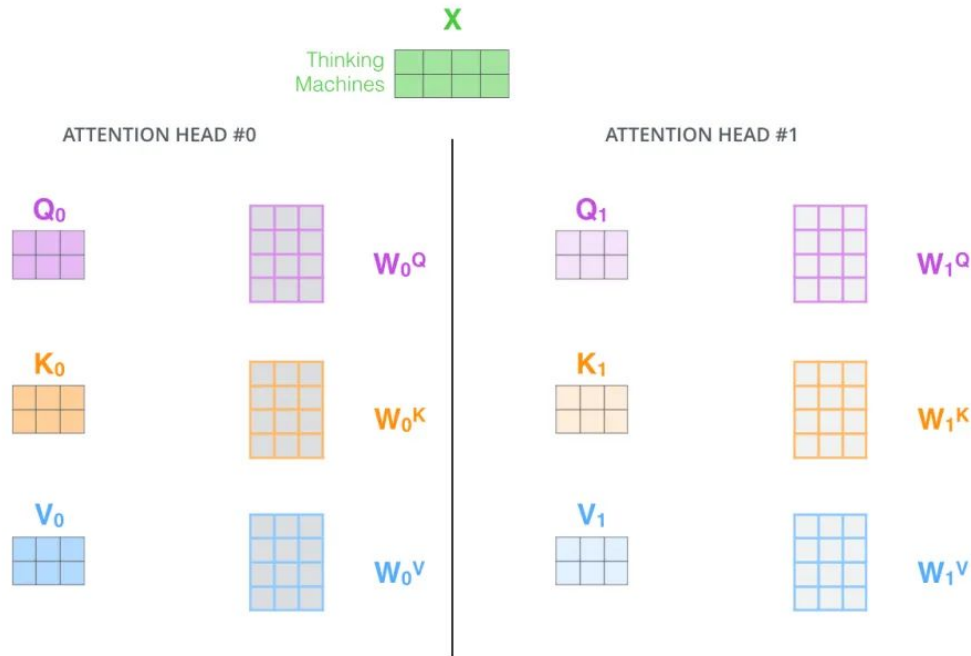
Attention Vectors

$[0.71 \quad 0.04 \quad 0.07 \quad 0.18]^T$
 $[0.01 \quad 0.84 \quad 0.02 \quad 0.13]^T$
 $[0.09 \quad 0.05 \quad 0.62 \quad 0.24]^T$
 $[0.03 \quad 0.03 \quad 0.03 \quad 0.91]^T$

3. Phân tích thiết kế Transformer

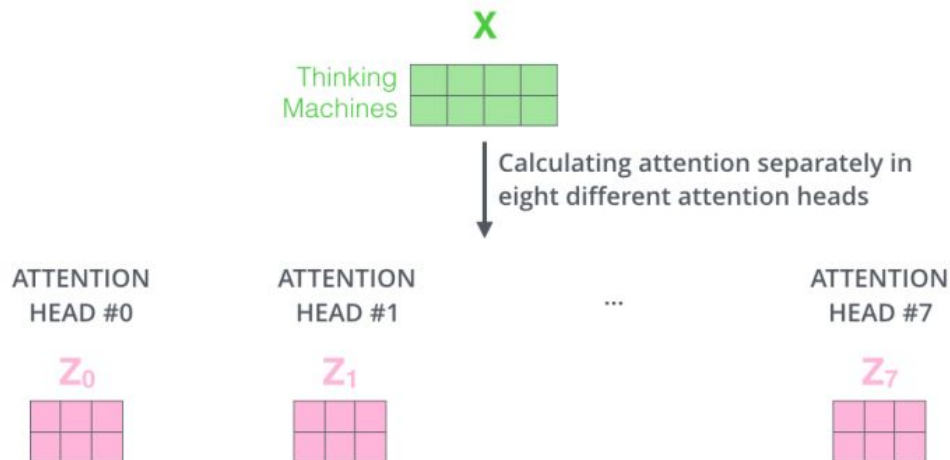
- Multihead attention:

Với multi-headed attention chúng ta không chỉ có một mà nhiều bộ ma trận trọng số Query/Key/Value



3. Phân tích thiết kế Transformer

- Multihead attention:
 - Với 8 lần tính với các ma trận khác nhau, ta có 8 ma trận Z khác nhau.



3. Phân tích thiết kế Transformer

- Multihead attention:

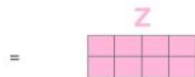
1) Concatenate all the attention heads



2) Multiply with a weight matrix W^O that was trained jointly with the model

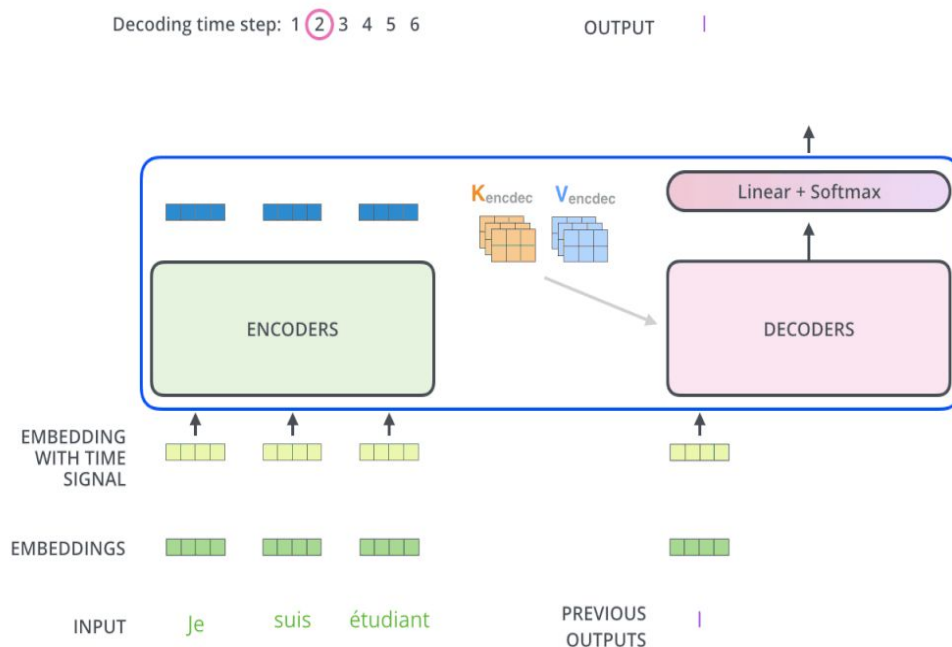
X

3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



3. Phân tích thiết kế Transformer

Kết quả của encoder trên cùng được chuyển thành một bộ các véc tơ attention K và V. Chúng được sử dụng bởi mỗi decoder trong lớp “encoder-decoder attention” để giúp decoder tập trung vào phần quan trọng trong chuỗi đầu vào.



Decoder

 $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

Le

 $\begin{bmatrix} 0.1 \\ 0.9 \\ 0 \\ 0 \end{bmatrix}$

gros

 $\begin{bmatrix} 0.05 \\ 0.40 \\ 0.55 \\ 0 \end{bmatrix}$

chien

 $\begin{bmatrix} 0.16 \\ 0.09 \\ 0.15 \\ 0.66 \end{bmatrix}$

rouge

 $\begin{bmatrix} 0.71 \\ 0.04 \\ 0.07 \\ 0.18 \end{bmatrix}$

The

 $\begin{bmatrix} 0.01 \\ 0.84 \\ 0.02 \\ 0.13 \end{bmatrix}$

big

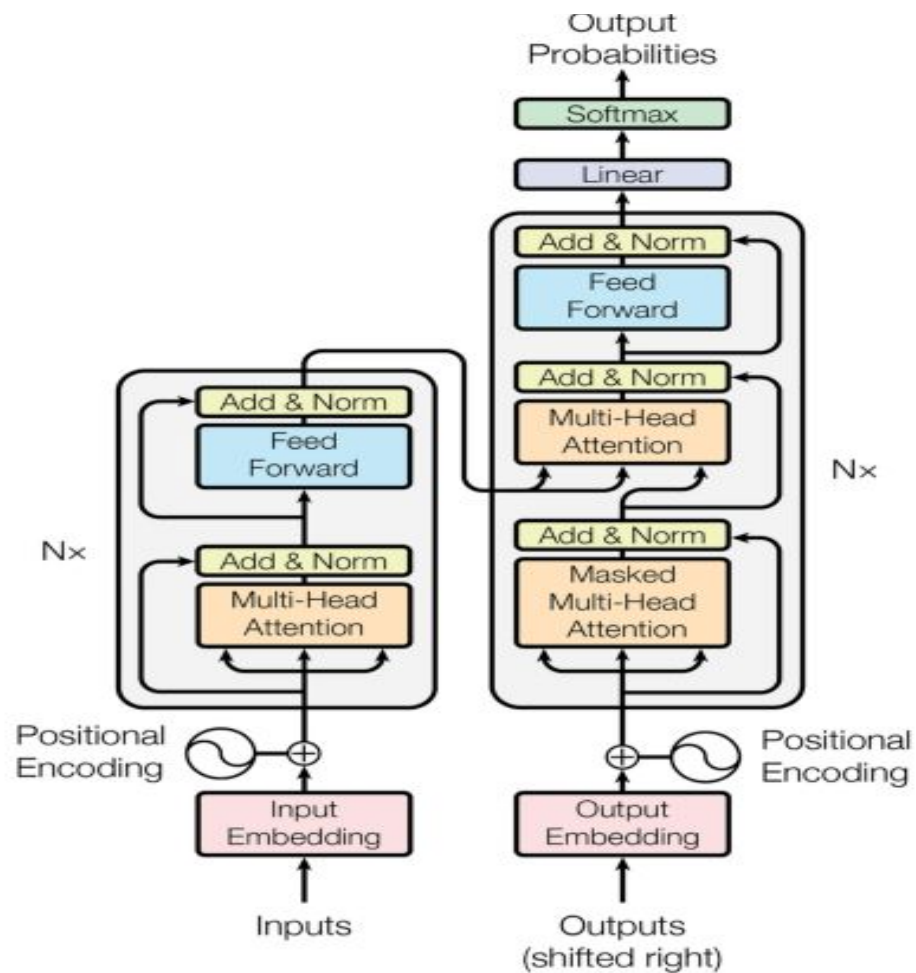
 $\begin{bmatrix} 0.09 \\ 0.05 \\ 0.62 \\ 0.24 \end{bmatrix}$

red

 $\begin{bmatrix} 0.03 \\ 0.03 \\ 0.03 \\ 0.91 \end{bmatrix}$

dog

Encoder-
Decoder
Attention





4. Ứng dụng và đánh giá.

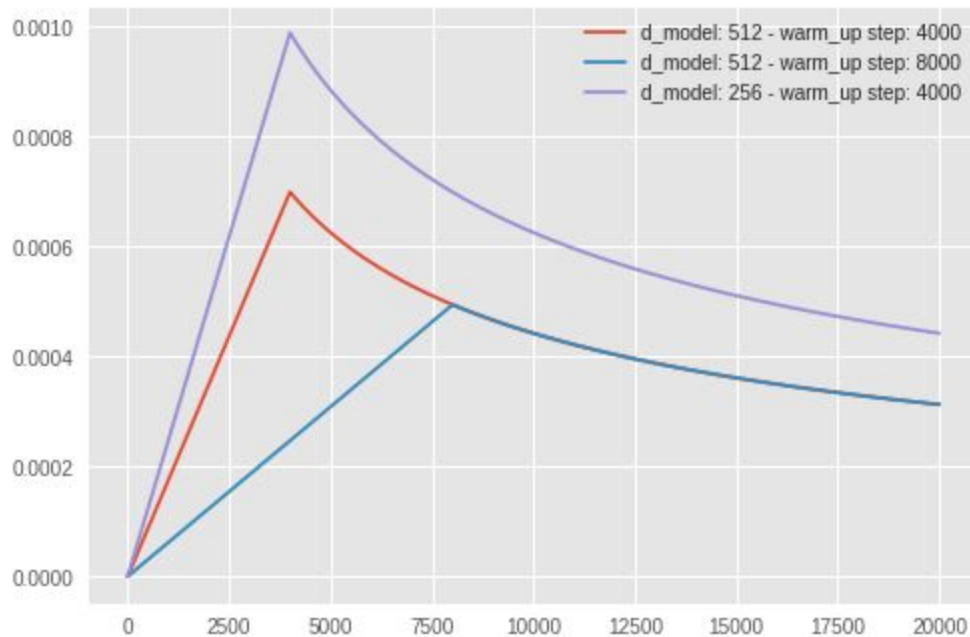
```
[ ] model = model.to(opt['device'])
    model.eval()
    sentence="this is the first book i've ever read."
    trans_sent = translate_sentence(sentence, model, SRC, TRG, opt['device'], opt['k'], opt['max_strlen'])
    trans_sent
```

'đây là cuốn sách đầu tiên mà tôi đã từng đọc.'

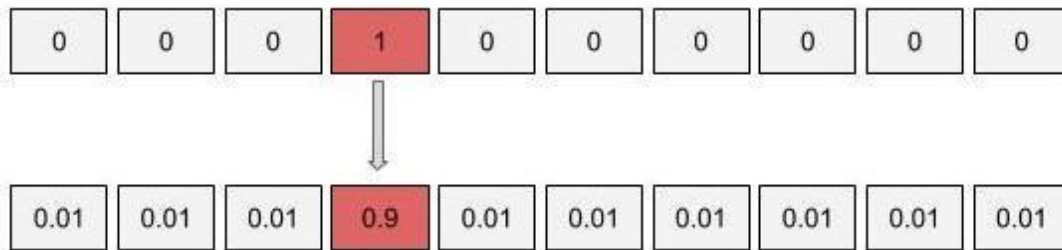
Optimizer

$$lr_rate = d_{model}^{-0.5} * \min(step_num^{-0.5}, step_num * warmup_steps^{-1.5})$$

Cơ bản thì learning rate sẽ tăng dần trong các lần cập nhật đầu tiên, các bước này được gọi là warm up step, lúc này mô hình sẽ ‘chạy’. Sau đó learning rate lại giảm dần, để mô hình hội tụ

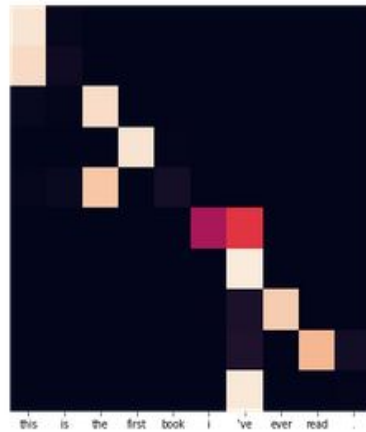
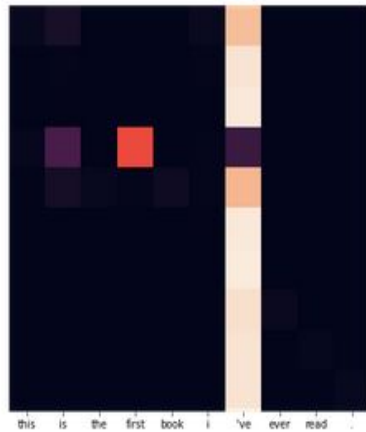
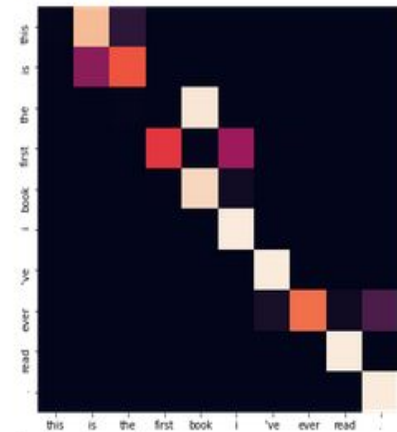


Label Smoothing



Với mô hình nhiều triệu tham số của transformer, để hạn chế hiện tượng overfit, có thể sử dụng kỹ thuật label smoothing. Phạt mô hình khi nó quá confident (tự tin) vào việc dự đoán. Thay vì mã hóa nhãn là một one-hot vector, sẽ thay đổi nhãn một chút bằng cách phân bổ thêm xác suất vào các trường hợp còn lại. Từ đó để epoch lớn sẽ không bị overfit

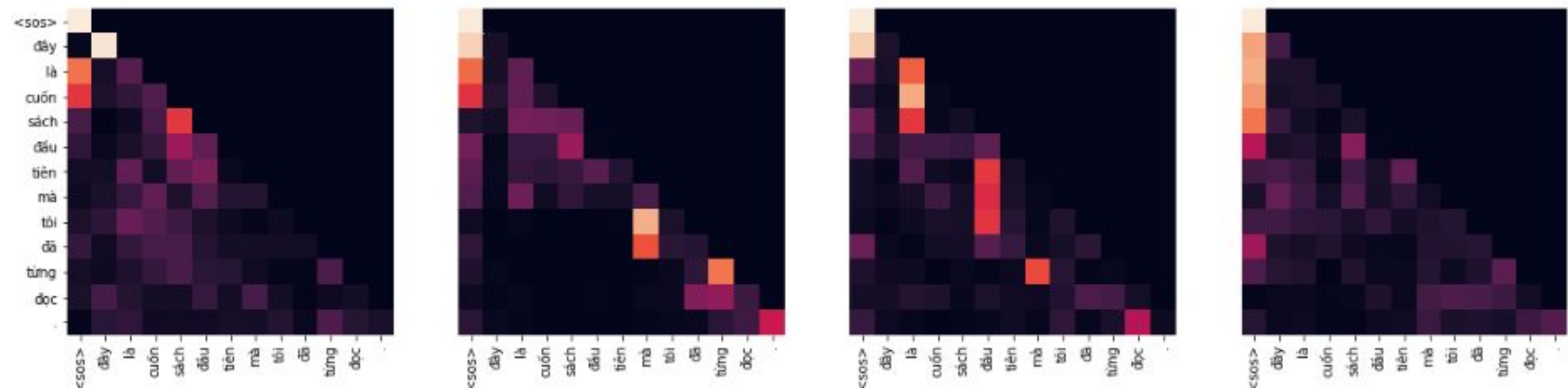
Encoder Layer 2



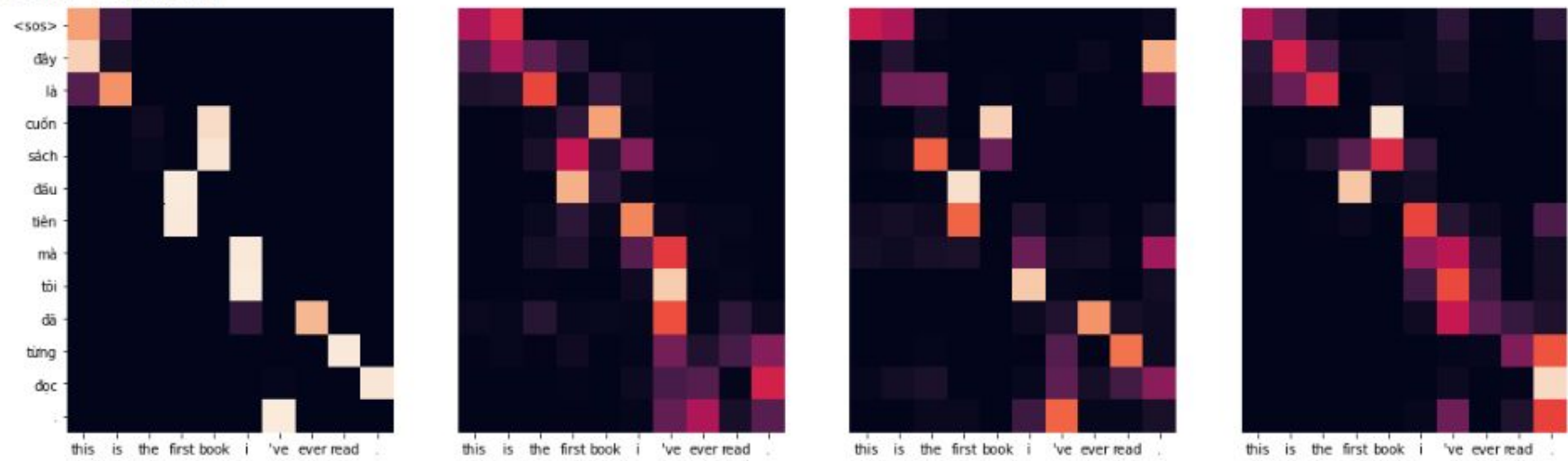
Encoder Layer 4



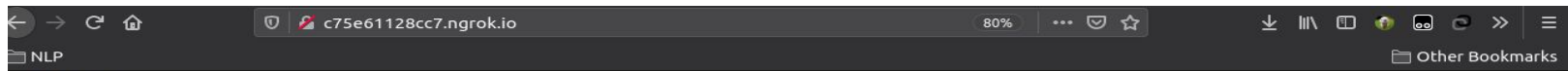
Decoder Self Layer 2



Decoder Src Layer 2



API

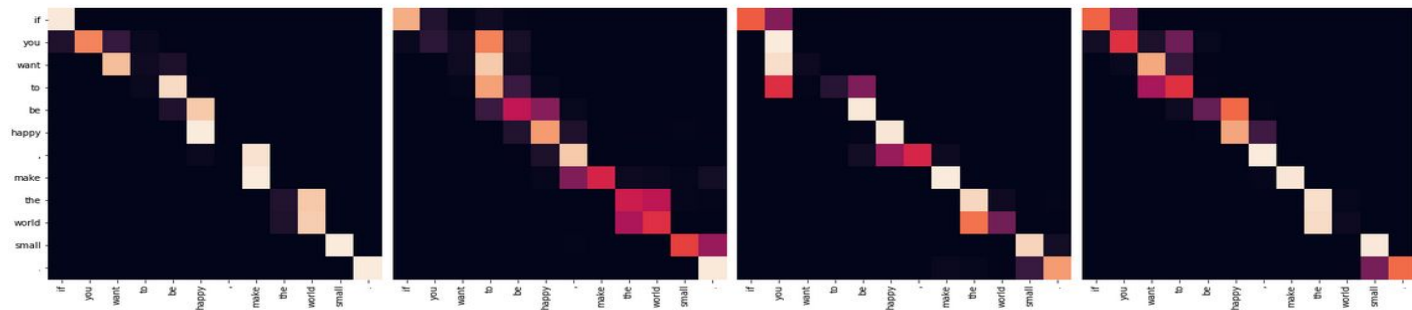


source sentence:

If You Want To Be Happy, Make The World Small |

Submit

Nếu bạn muốn hạnh phúc, hãy làm cho thế giới nhỏ bé.





Thử nghiệm với nhận diện tên riêng, ngày tháng.

Với những cái tên phổ biến và dễ nhận dạng thì model dễ dàng nhận diện.

Source sentence:

Jack is a good guy.

Submit

jack là một anh chàng tốt.



Thử nghiệm với nhận diện tên riêng, ngày tháng.

Nhưng với những cái tên trùng với ngày tháng, năm hoặc các từ có nghĩa, model không nhận dạng được và dịch luôn sang nghĩa.

Source sentence:

Marry like cream so much

Submit

cười giống như kem rất nhiều.



Nhận xét việc nhận diện tên:

Về việc nhận dạng tên riêng và ngày tháng năm thì model đã train không thể nhận diện hoàn toàn các tên với ngày tháng, lý do là bộ dữ liệu còn nhỏ chỉ 600k cặp câu và dữ liệu cũng không chuyên biệt cho nhiệm vụ này. Vì nhiệm vụ nhận tên riêng so với ngày tháng là nhiệm vụ khó, tùy thuộc hoàn cảnh, kinh nghiệm và ý định của con người. Mô hình cần bộ dataset riêng để nhận diện các tên hiệu quả hơn.

Thêm head để tăng độ hiệu quả

Mô hình ban đầu 8 head 6 layer

```
epoch: 027 - iter: 01999 - train loss: 2.4136 - time: 0.2397
epoch: 027 - iter: 02199 - train loss: 2.4136 - time: 0.2397
epoch: 027 - iter: 02399 - train loss: 2.4019 - time: 0.2405
epoch: 027 - iter: 02496 - valid loss: 2.3889 - bleu score: 0.2501 - time: 401.1084
epoch: 028 - iter: 00199 - train loss: 2.3805 - time: 0.2409
epoch: 028 - iter: 00399 - train loss: 2.3920 - time: 0.2385
epoch: 028 - iter: 00599 - train loss: 2.3563 - time: 0.2437
epoch: 028 - iter: 00799 - train loss: 2.3711 - time: 0.2361
epoch: 028 - iter: 00999 - train loss: 2.3572 - time: 0.2520
epoch: 028 - iter: 01199 - train loss: 2.3946 - time: 0.2450
epoch: 028 - iter: 01399 - train loss: 2.3939 - time: 0.2373
epoch: 028 - iter: 01599 - train loss: 2.3963 - time: 0.2486
epoch: 028 - iter: 01799 - train loss: 2.3998 - time: 0.2435
epoch: 028 - iter: 01999 - train loss: 2.3973 - time: 0.2402
epoch: 028 - iter: 02199 - train loss: 2.3773 - time: 0.2486
epoch: 028 - iter: 02399 - train loss: 2.4067 - time: 0.2458
epoch: 028 - iter: 02496 - valid loss: 2.3903 - bleu score: 0.2539 - time: 394.6476
epoch: 029 - iter: 00199 - train loss: 2.3618 - time: 0.2337
epoch: 029 - iter: 00399 - train loss: 2.3452 - time: 0.2281
epoch: 029 - iter: 00599 - train loss: 2.3492 - time: 0.2385
epoch: 029 - iter: 00799 - train loss: 2.3596 - time: 0.2501
epoch: 029 - iter: 00999 - train loss: 2.3678 - time: 0.2358
epoch: 029 - iter: 01199 - train loss: 2.3663 - time: 0.2478
epoch: 029 - iter: 01399 - train loss: 2.3640 - time: 0.2419
epoch: 029 - iter: 01599 - train loss: 2.3898 - time: 0.2386
epoch: 029 - iter: 01799 - train loss: 2.3912 - time: 0.2377
epoch: 029 - iter: 01999 - train loss: 2.3617 - time: 0.2353
epoch: 029 - iter: 02199 - train loss: 2.3887 - time: 0.2364
epoch: 029 - iter: 02399 - train loss: 2.3730 - time: 0.2407
epoch: 029 - iter: 02496 - valid loss: 2.3910 - bleu score: 0.2559 - time: 397.0753
chúng ta đã chứng kiến một thành viên trong suốt thời gian ăn mặc dù không đúng cách.
```

Thêm head để tăng độ hiệu quả

Mô hình 16 head 6 layer:

```
epoch: 027 - iter: 01199 - train loss: 2.4071 - time: 0.2506
epoch: 027 - iter: 01399 - train loss: 2.4203 - time: 0.2245
epoch: 027 - iter: 01599 - train loss: 2.4238 - time: 0.2290
epoch: 027 - iter: 01799 - train loss: 2.4265 - time: 0.2339
epoch: 027 - iter: 01999 - train loss: 2.4151 - time: 0.2609
epoch: 027 - iter: 02199 - train loss: 2.4036 - time: 0.2573
epoch: 027 - iter: 02399 - train loss: 2.4035 - time: 0.2511
epoch: 027 - iter: 02496 - valid loss: 2.4194 - bleu score: 0.2527 - time: 409.9355
epoch: 028 - iter: 00199 - train loss: 2.3310 - time: 0.2410
epoch: 028 - iter: 00399 - train loss: 2.3890 - time: 0.2257
epoch: 028 - iter: 00599 - train loss: 2.3667 - time: 0.2464
epoch: 028 - iter: 00799 - train loss: 2.3851 - time: 0.2548
epoch: 028 - iter: 00999 - train loss: 2.3850 - time: 0.2439
epoch: 028 - iter: 01199 - train loss: 2.3836 - time: 0.2370
epoch: 028 - iter: 01399 - train loss: 2.3931 - time: 0.2543
epoch: 028 - iter: 01599 - train loss: 2.4040 - time: 0.2444
epoch: 028 - iter: 01799 - train loss: 2.3844 - time: 0.2387
epoch: 028 - iter: 01999 - train loss: 2.4059 - time: 0.2499
epoch: 028 - iter: 02199 - train loss: 2.4218 - time: 0.2455
epoch: 028 - iter: 02399 - train loss: 2.4157 - time: 0.2436
epoch: 028 - iter: 02496 - valid loss: 2.4221 - bleu score: 0.2514 - time: 400.5117
epoch: 029 - iter: 00199 - train loss: 2.3479 - time: 0.2567
epoch: 029 - iter: 00399 - train loss: 2.3540 - time: 0.2485
epoch: 029 - iter: 00599 - train loss: 2.3606 - time: 0.2516
epoch: 029 - iter: 00799 - train loss: 2.3655 - time: 0.2545
epoch: 029 - iter: 00999 - train loss: 2.3706 - time: 0.2479
epoch: 029 - iter: 01199 - train loss: 2.3825 - time: 0.2462
epoch: 029 - iter: 01399 - train loss: 2.3761 - time: 0.2447
epoch: 029 - iter: 01599 - train loss: 2.3928 - time: 0.2543
epoch: 029 - iter: 01799 - train loss: 2.3842 - time: 0.2627
epoch: 029 - iter: 01999 - train loss: 2.3751 - time: 0.2434
epoch: 029 - iter: 02199 - train loss: 2.3847 - time: 0.2568
epoch: 029 - iter: 02399 - train loss: 2.3930 - time: 0.2315
epoch: 029 - iter: 02496 - valid loss: 2.4325 - bleu score: 0.2527 - time: 400.5555
gia đình tôi không nghèo, và bản thân tôi, tôi chưa bao giờ trải qua sự đói.
```



Nhận xét việc thêm head.

Mô hình khi train với 16 head và 8 head cho bleu score không thực sự quá khác biệt với chỉ số 0.2527 giảm so với 0.2559 ban đầu, nên có thể khi số head đủ thì tăng số head lên cũng không thực sự tăng độ hiệu quả của việc dịch.

Khi nghiên cứu thêm các tài liệu ngoài và thực nghiệm, ta có thể khẳng định trong trường hợp dịch máy không cần quá nhiều head và cần số lượng vừa đủ, phù hợp.



Đánh giá tổng thể:

Có khả năng thực hiện song song trong quá trình encoder

Cho kết quả tốt với dữ liệu câu dài. Mô hình chạy tương đối nhanh và hiệu quả.

Nhưng còn chưa xử lý tốt unknown word

THANKS FOR
WATCHING