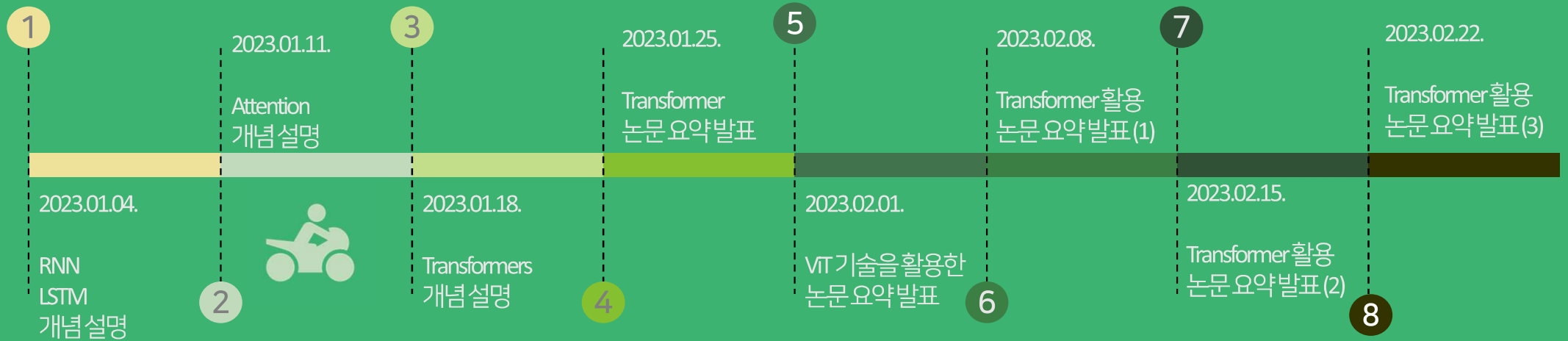
Attention

CGVM Transformer study in 2023 winter

>> Transformer 주차별 계획



>> Table of contents

1

What is Attention?

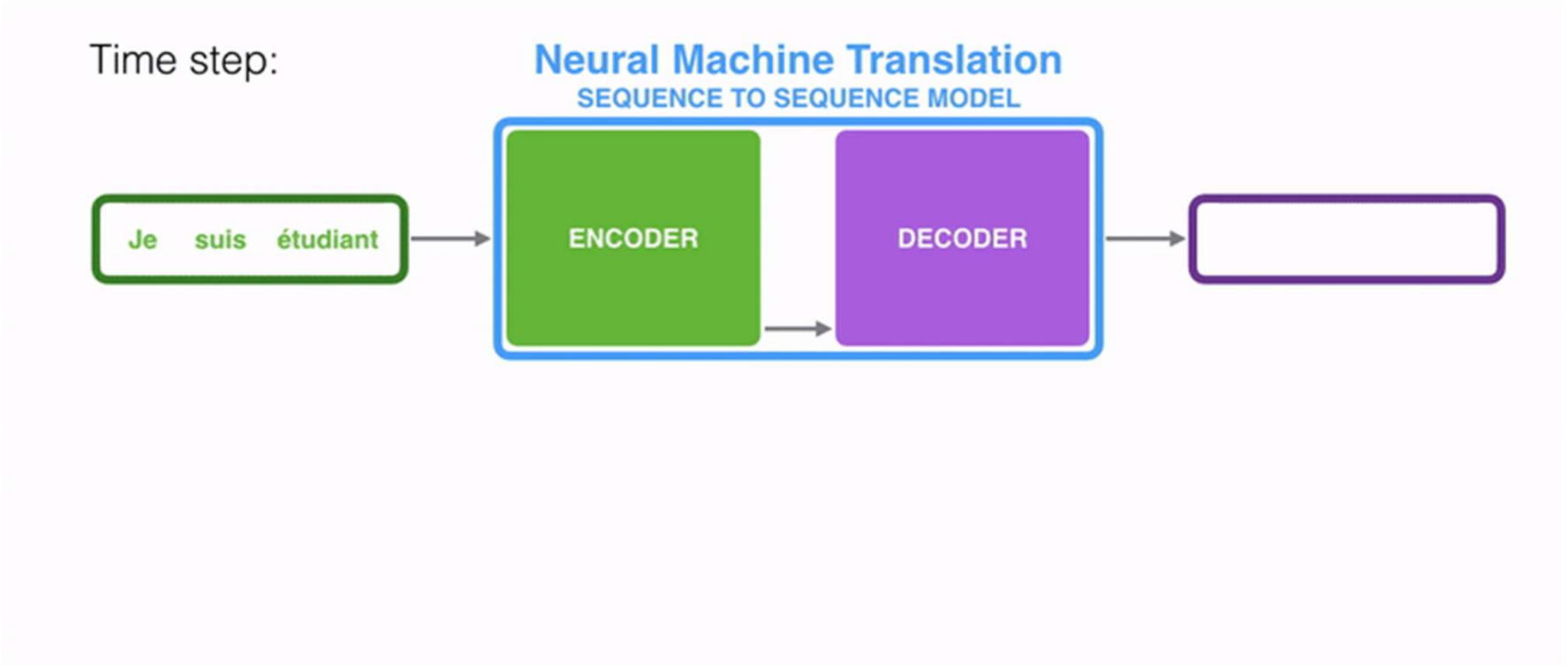
Part 1 >> **What is Attention?**

RNN에 기반한 seq2seq 모델이 갖고 있는 두 가지 문제점

1. 고정된 크기의 벡터(context vector)에 모든 정보를 압축하기 때문에 정보손실 발생

2. RNN의 고질적인 문제인 gradient vanishing 문제가 존재

Part 1 >> **What is Attention?**



https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Part 1 >> **What is Attention?**

Attention 이란?

- 디코더에서 출력단어를 예측하는 매 시점(time step)마다 인코더에서의 전체 입력 문장을 다시 참고하게 하자.

- 단 예측해야할 단어와 연관이 있는 입력 단어 부분을 좀 더 **집중**해서

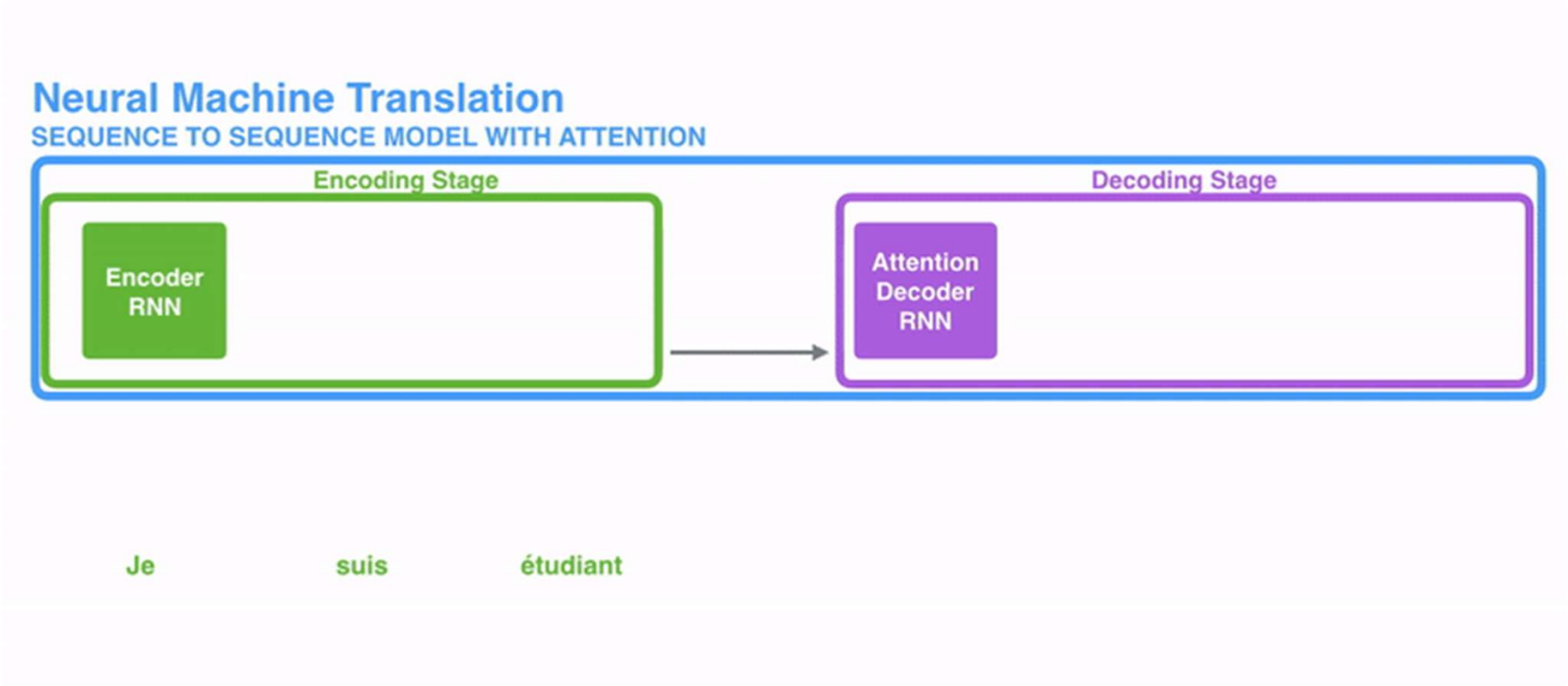
Part 1 >> **What is Attention?**

Attention 이란?

- 디코더에서 출력단어를 예측하는 매 시점(time step)마다 인코더에서의 전체 입력 문장을 다시 참고하게 하자.

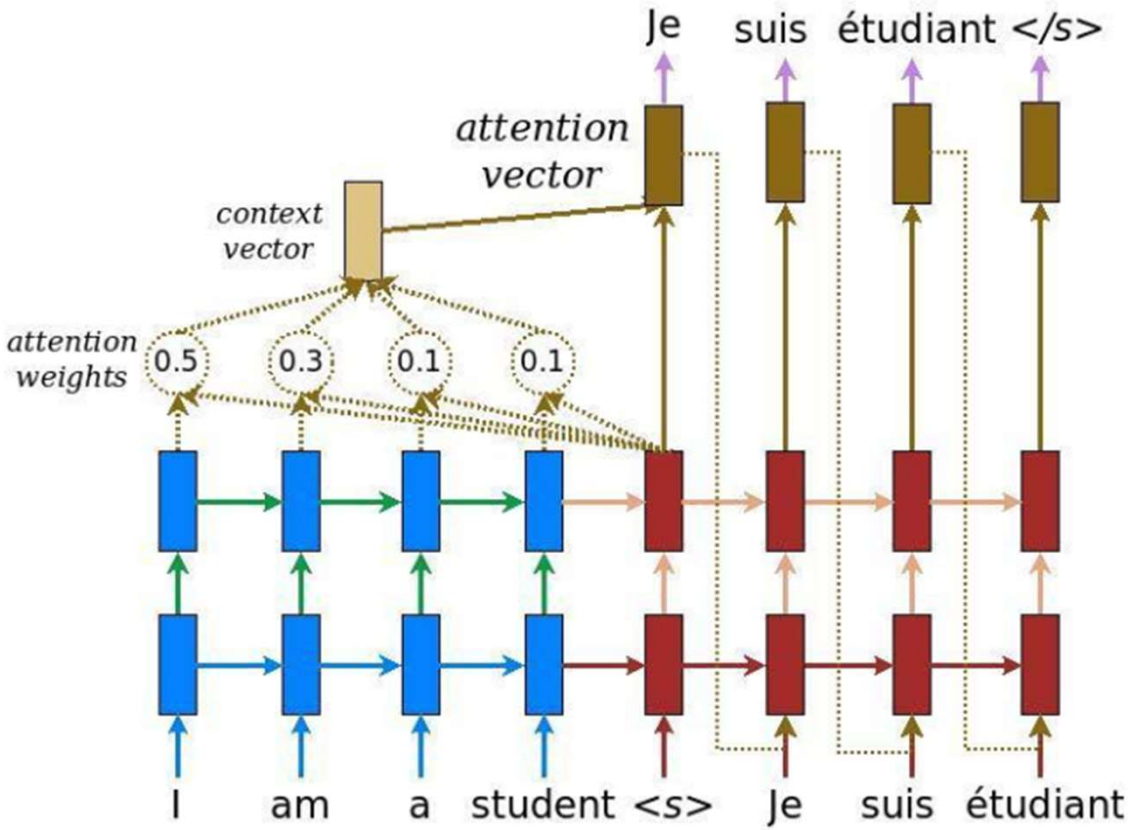
- 단 예측해야할 단어와 연관이 있는 입력 단어 부분을 좀 더 **집중**해서

Part 1 >> **What is Attention?**

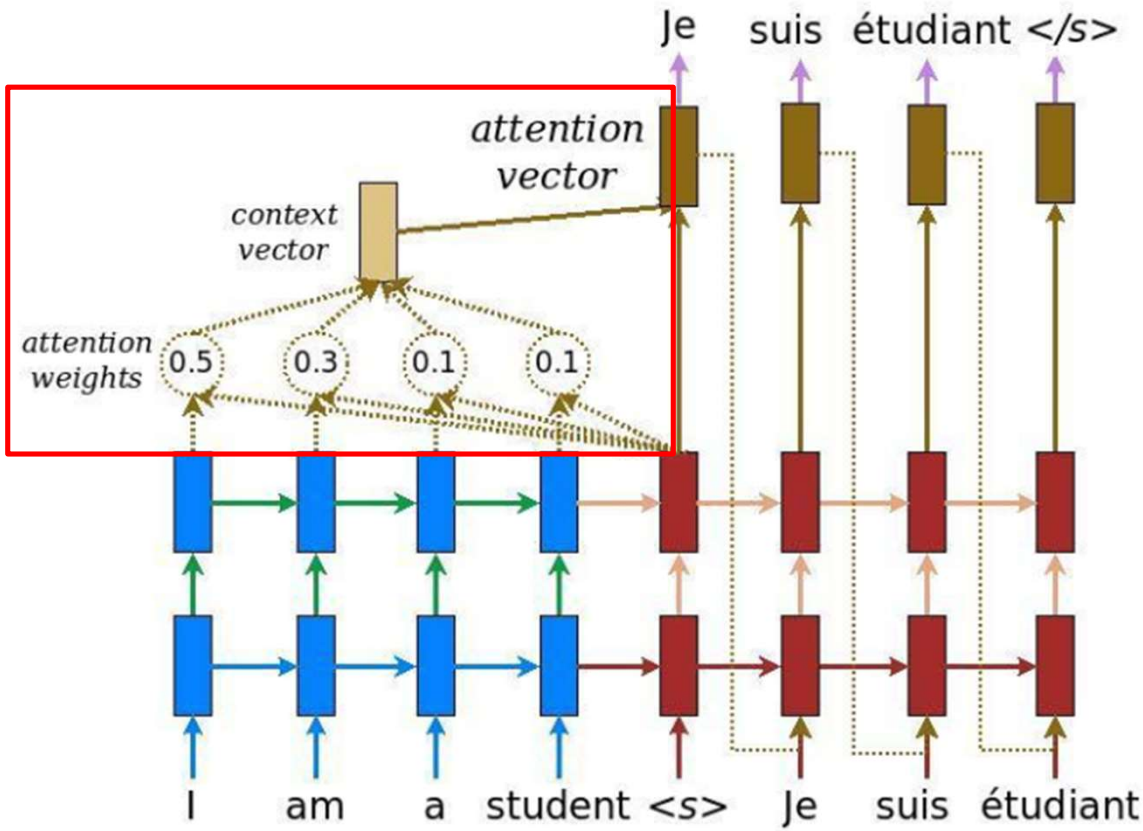


https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Part 1 >> **What is Attention?**



Part 1 >> # What is Attention?



Part 1 >> **What is Attention?**

Query :

Key :

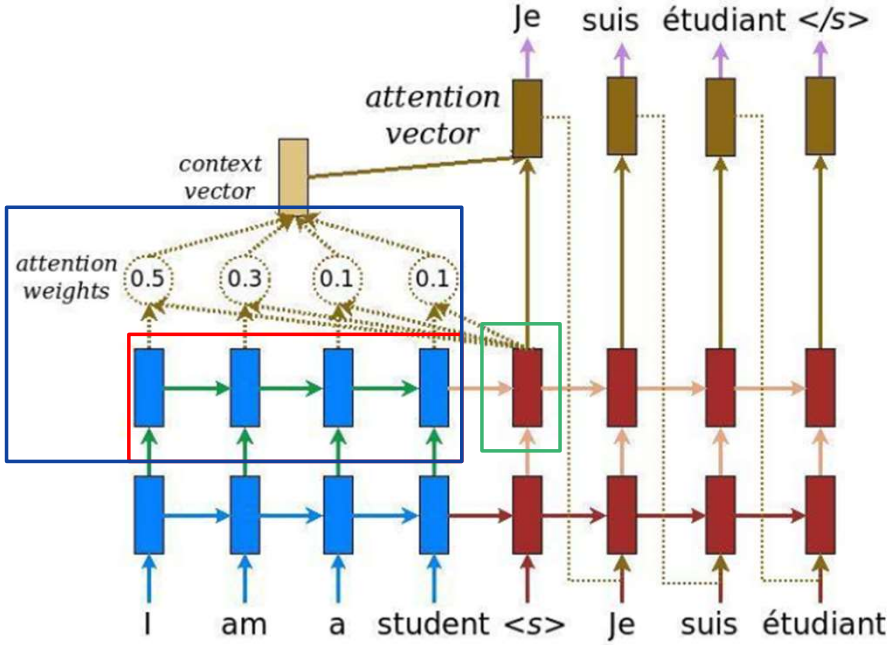
Value :

Part 1 >> **What is Attention?**

Query : t 시점의 디코더 셀에서의 은닉 상태들
Key : 모든 시점의 인코더 셀의 은닉 상태들
Value : 모든 시점의 인코더 셀의 은닉 상태들(유사도를 반영할)

Part 1 >> **What is Attention?**

Query : t 시점의 디코더 셀에서의 은닉 상태들
Key : 모든 시점의 인코더 셀의 은닉 상태들
Value : 모든 시점의 인코더 셀의 은닉 상태들(유사도를 반영할)

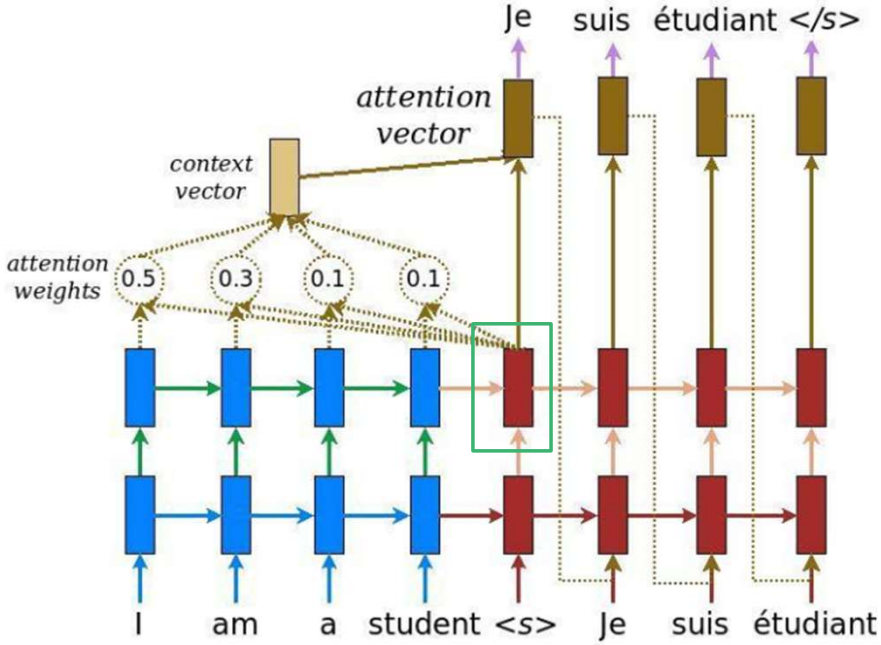


Part 1 >> **What is Attention?**

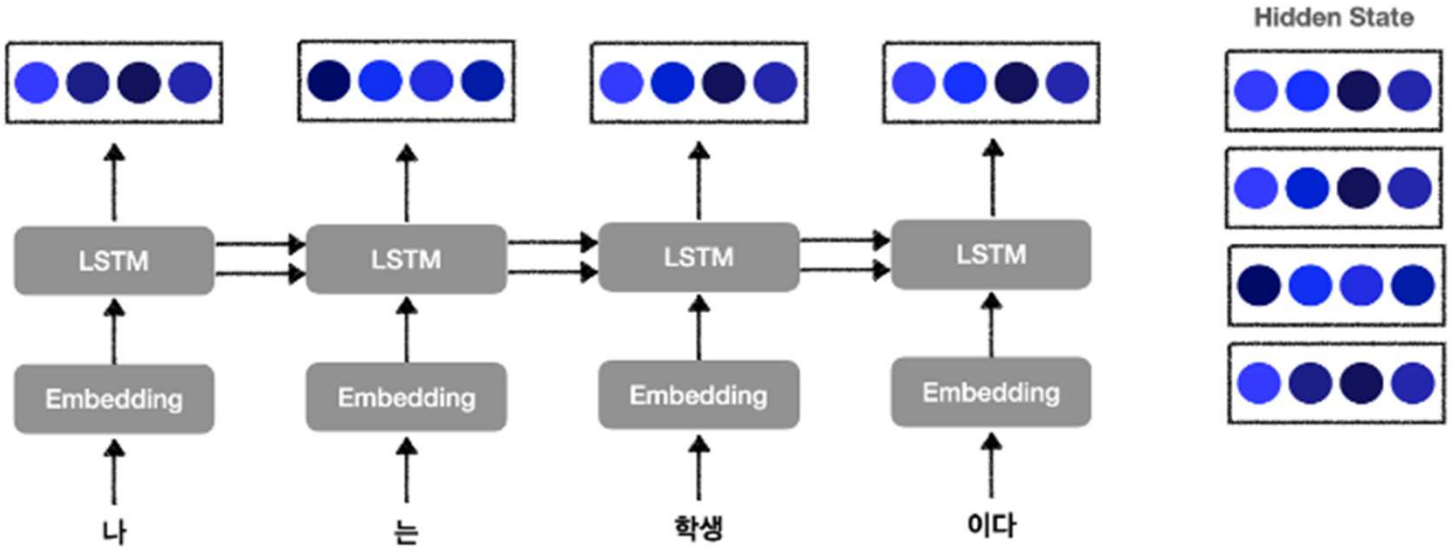
Query : t 시점의 디코더 셀에서의 은닉 상태들
Key : 모든 시점의 인코더 셀의 은닉 상태들
Value : 모든 시점의 인코더 셀의 은닉 상태들(유사도를 반영할)

Attention score

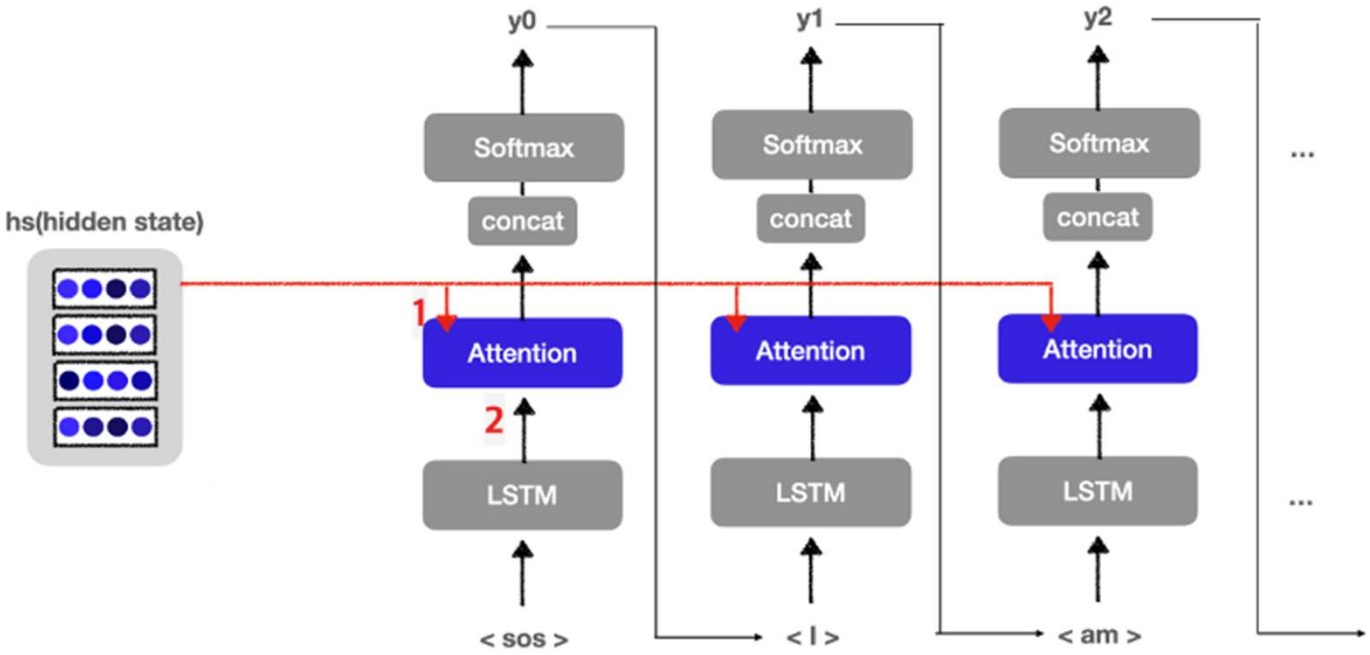
$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$



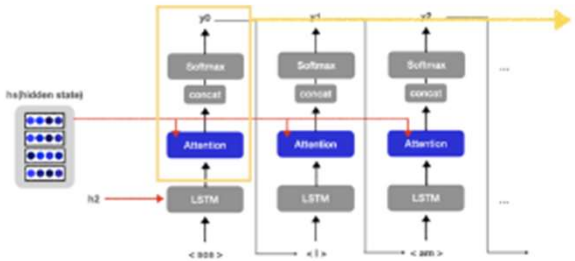
Part 1 >> **What is Attention?**

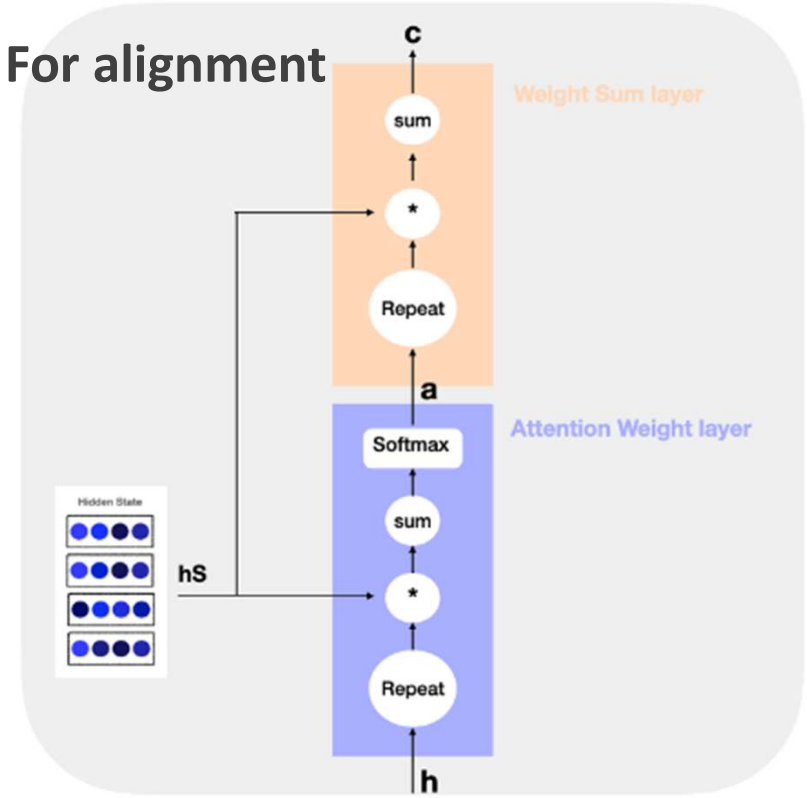


Part 1 >> # What is Attention?

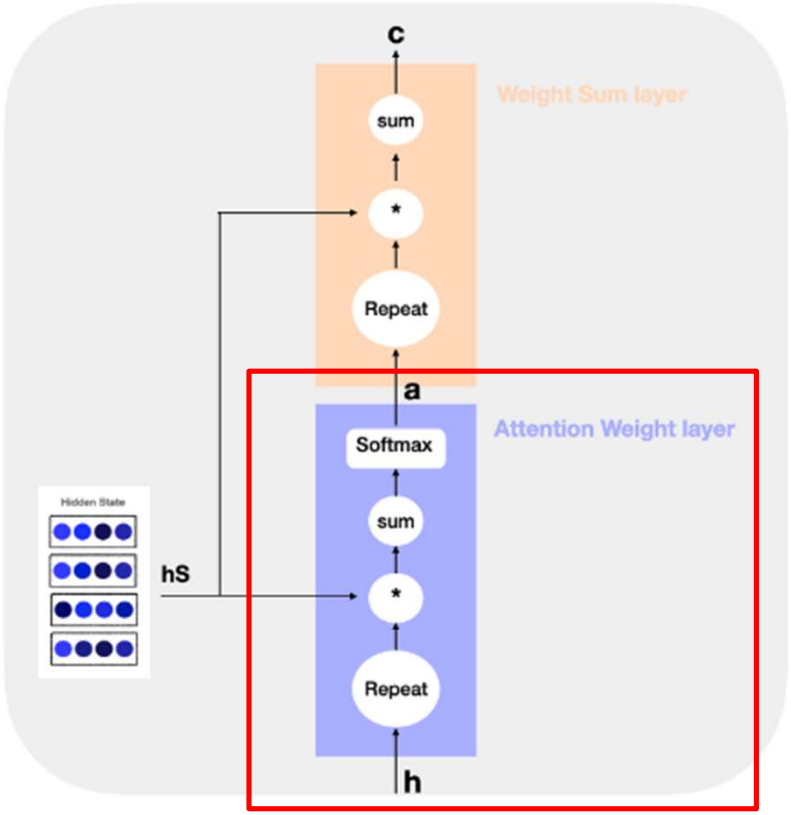


Part 1 >> **What is Attention?**





Part 1 >> **What is Attention?**



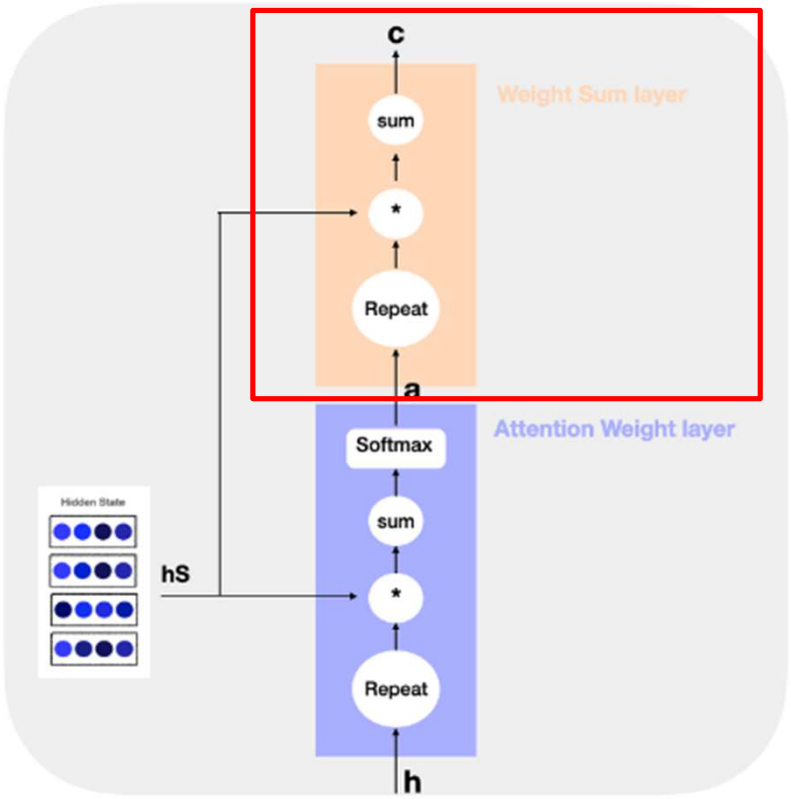
나	h0	=	1	0	0	1	*	0.1	=	0.1	0	0	0.1
는	h1	=	1	0	0	2	*	0.055	=	0.055	0	0	0.11
학생	h2	=	1	1	2	0	*	0.8	=	0.8	0.8	1.6	0
이다	h3	=	1	1	0	0	*	0.045	=	0.045	0.045	0	0

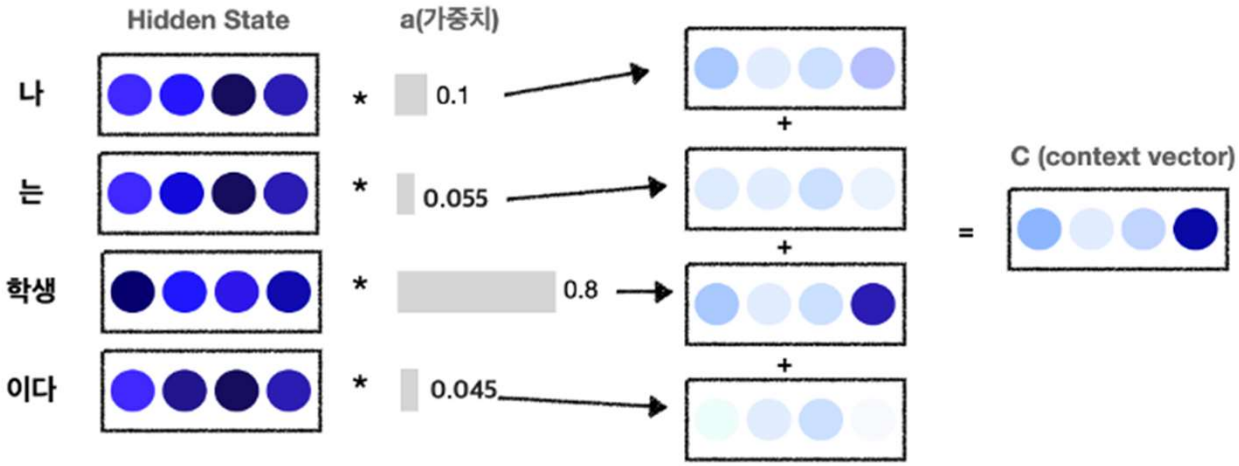
*이때, **Attention Weight layer에서 출력되는 값**은*

= (0.1 0 0 0.1) + (0.055 0 0 0.11) + (0.8 0.8 1.6 0) + (0.045 0.045 0 0)

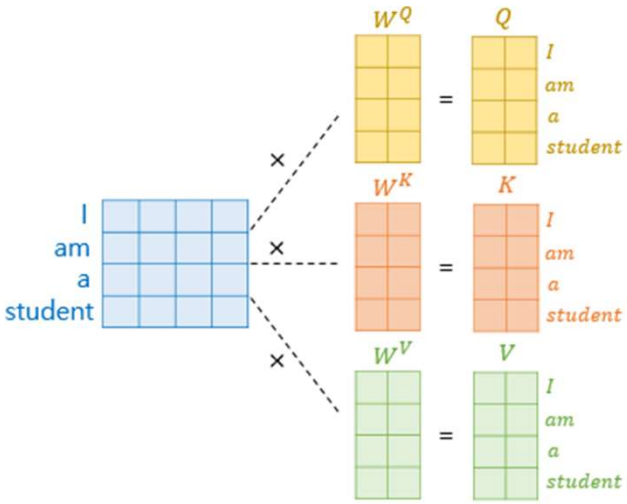
으로 계산하면 된다.

Part 1 >> # What is Attention?



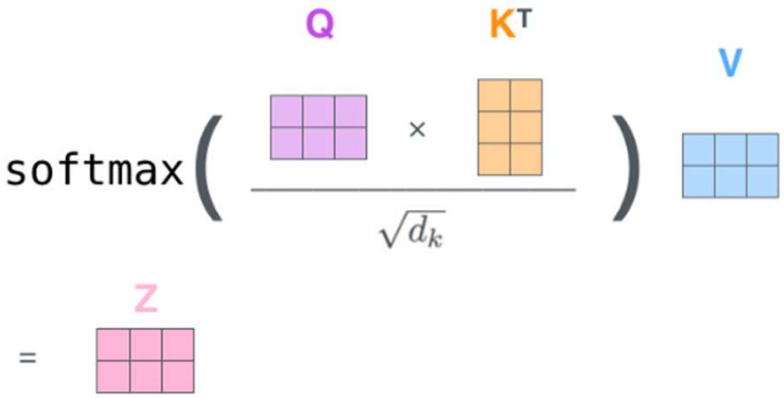


Part 1 >> **What is Attention?**



- 쿼리, 키, 벨류를 생성하기 위해 가중치행렬 W^Q, W^K, W^V 를 곱해준다.

- W^Q, W^K, W^V는 학습 과정을 통해 갱신해야 할 값이다.



- 위 과정으로 생성된 쿼리와 키^{T}를 곱해 attention score를 구한다.

- attention score의 값이 증대되는 문제를 보안하기 위해 sqrt(키 벡터의 차원) 값으로 나누어 준다.

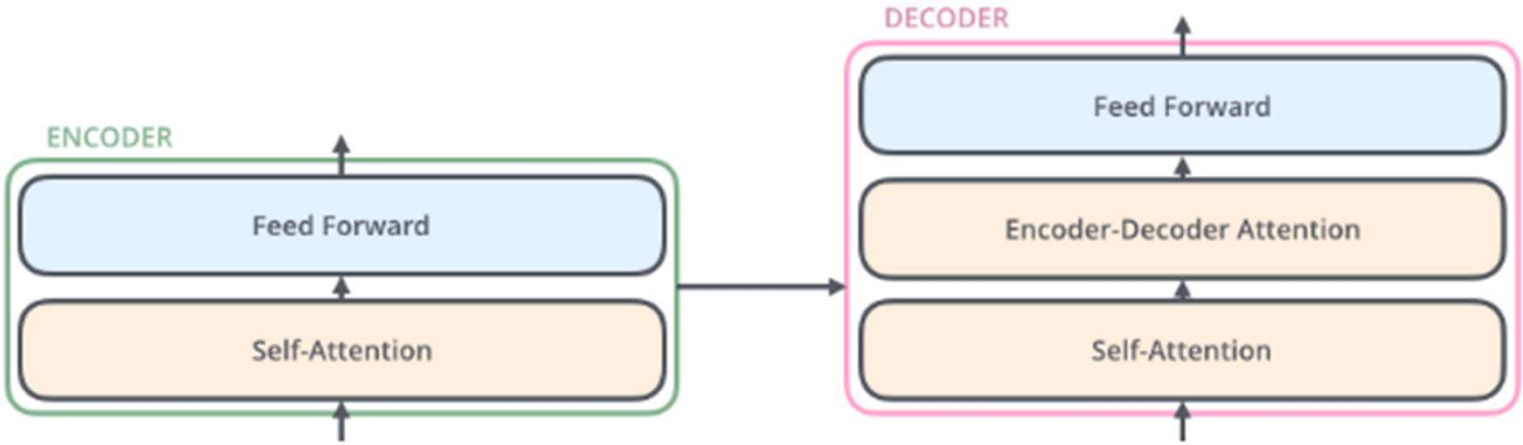
- 이를 벨류에 곱해주면 최종적인 context vector Z 가 구해진다

2

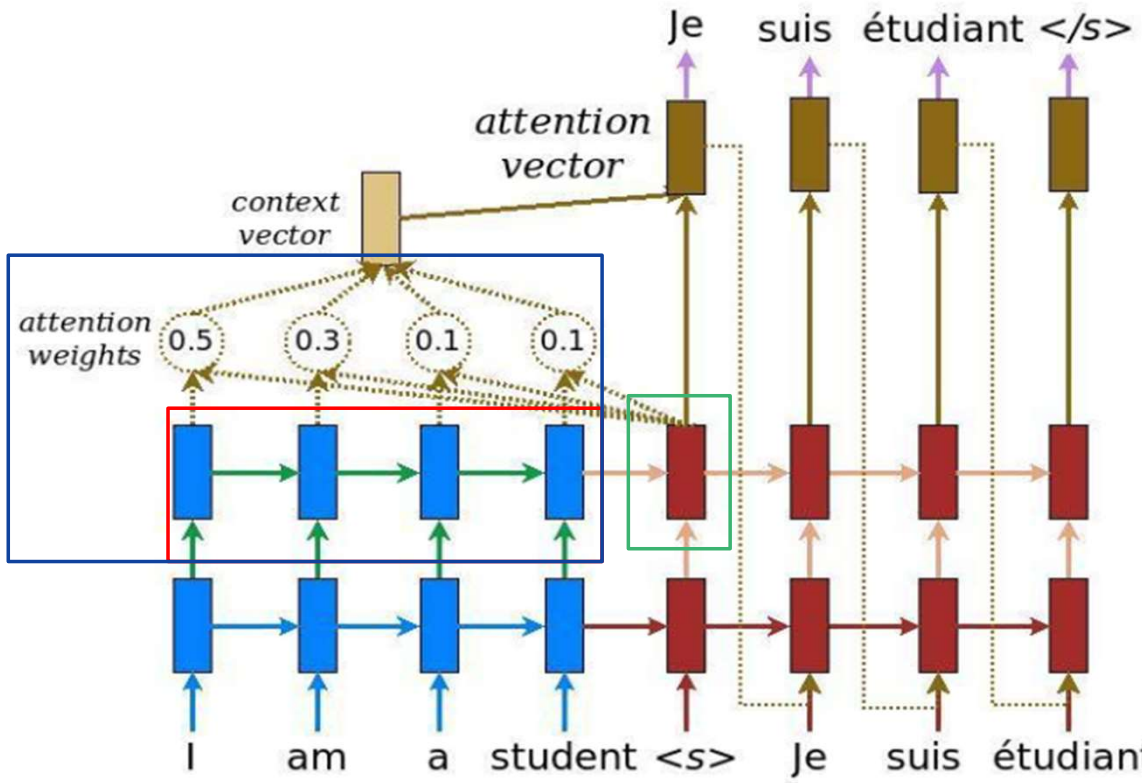
Self Attention

“Self Attention”

Part 2 >> **Self Attention**

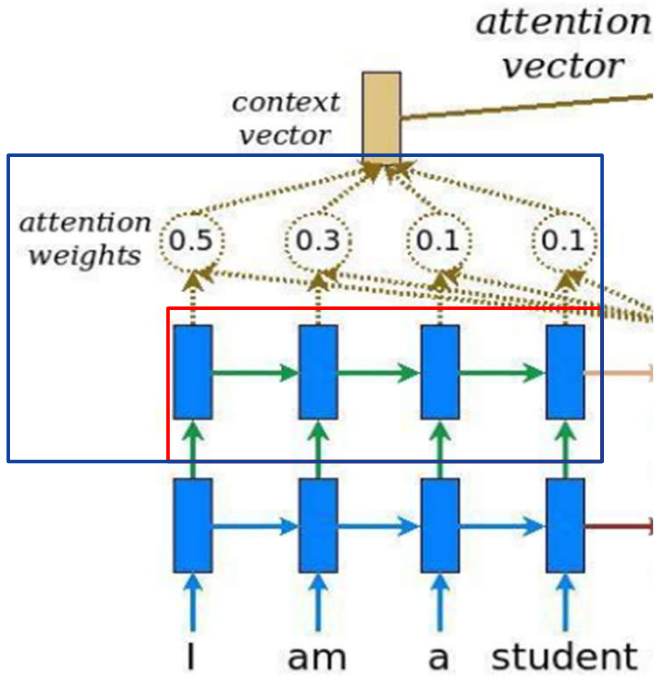


Part 2 >> **Self Attention**



Query
Key
Value

Part 2 >> **Self Attention**

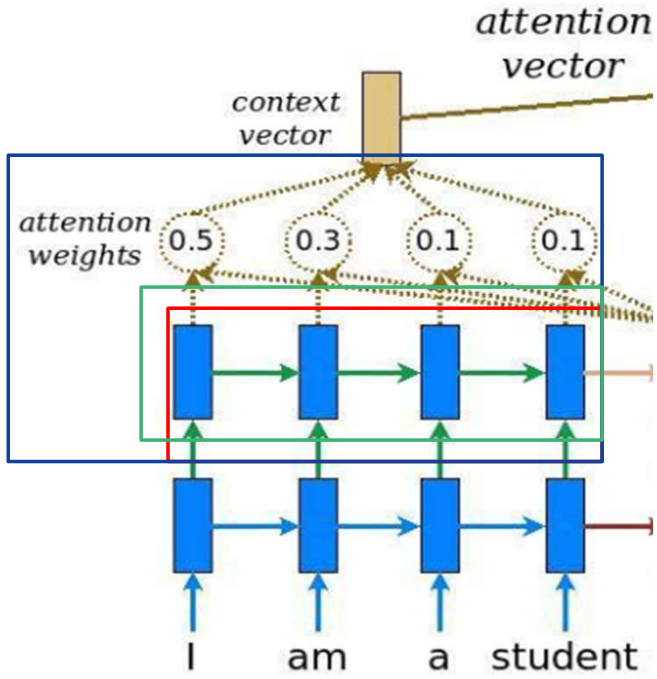


Query

Key

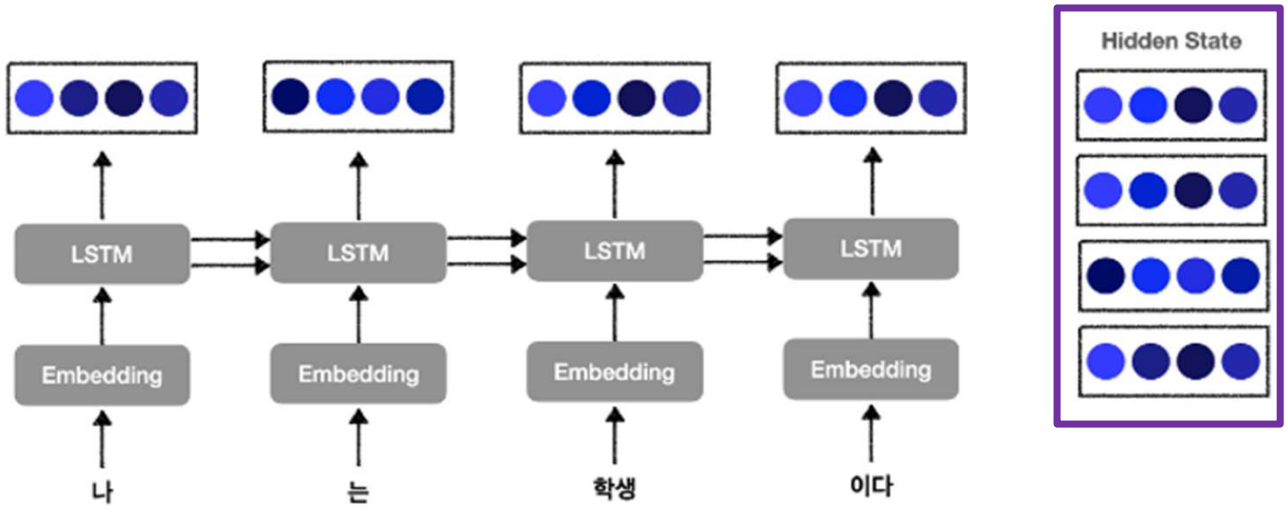
Value

Part 2 >> **Self Attention**



Query
Key
Value

Part 2 >> **Self Attention**



Query

Key

Value

Part 2 >> **Self Attention**

Attention VS Self-Attention

Q,K,V
- Attention : Q는 Decoder Cell에서 도출되고, K와 V는 Encoder Cell에서 도출됨
- Self-Attention : Q, K, V는 모두 동일한 Vector(Embedding Vector)에서 도출됨

Time step
- Attention : Time-Step을 활용함.
- Self-Attention : Time-Step을 활용하지 않음.

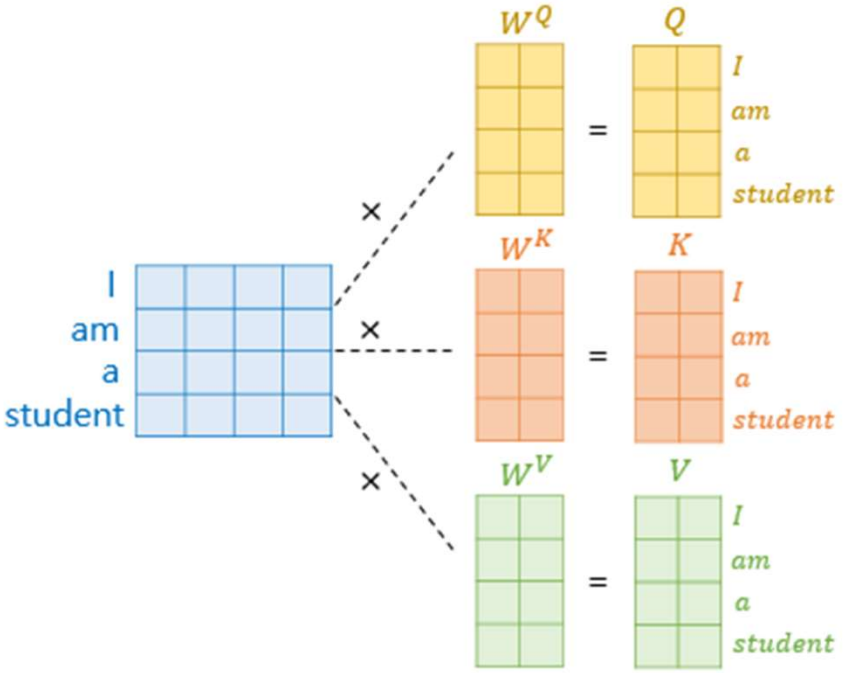
방향성
- Attention은 왼쪽->오른쪽, 혹은 오른쪽->왼쪽의 방향으로 해석이 진행되는 Unidirectional한 Model이다
- Self-Attention은 (Encoder측에서) 오른쪽과 왼쪽에 있는 모든 단어를 활용하는 Bidirectional한 Model이다.

3

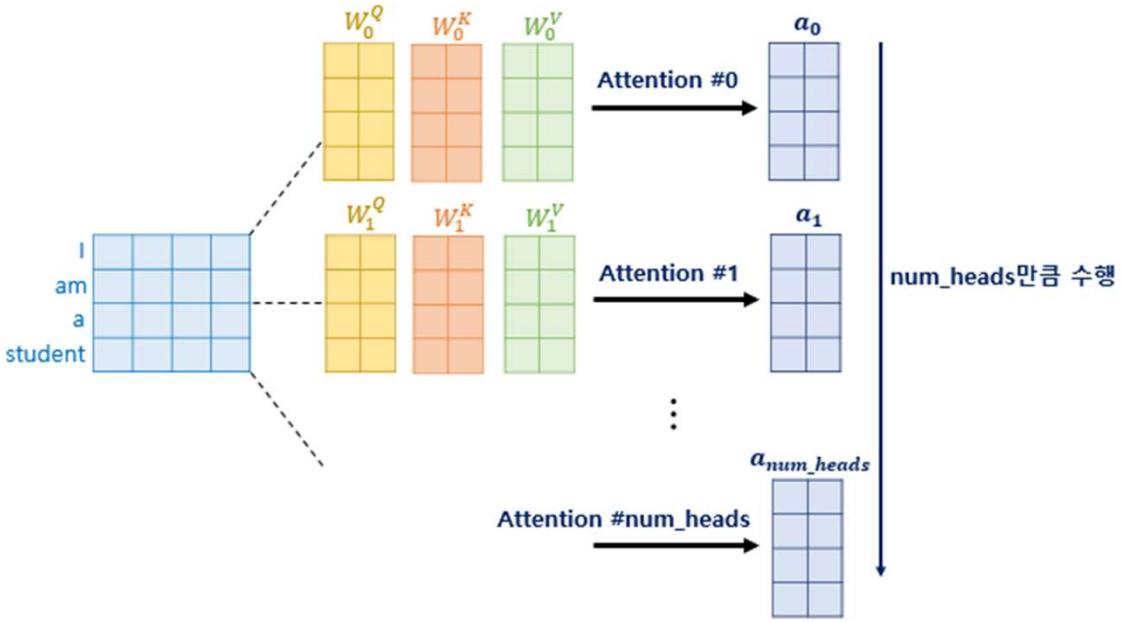
Multi-head Attention

“Multi-head Attention”

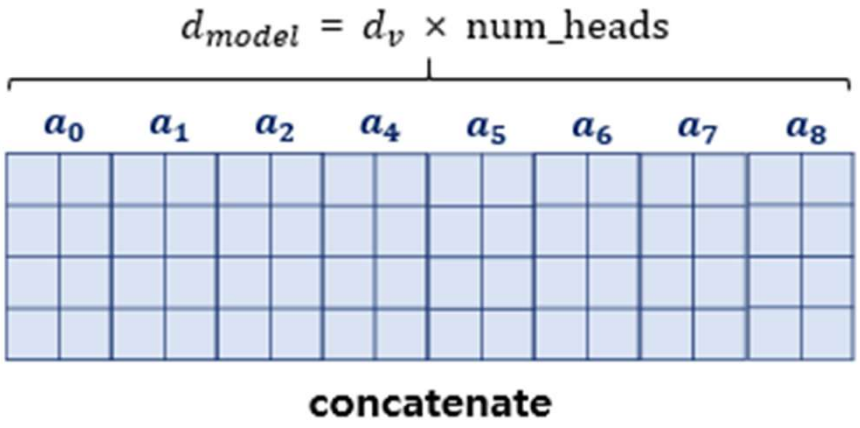
Part 3 >> **Multi-head Attention**



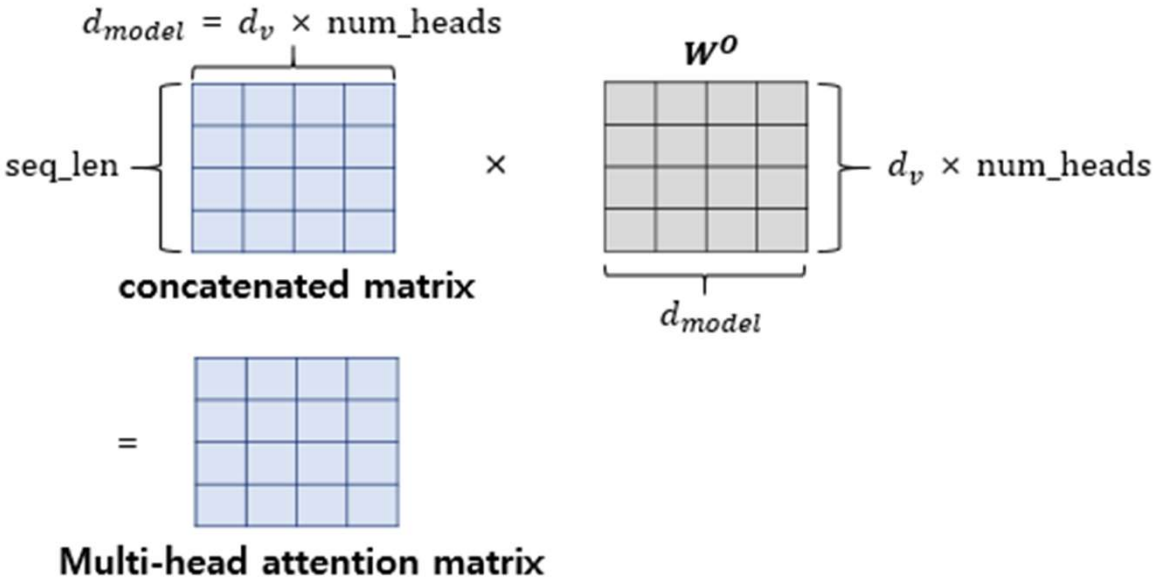
Part 3 >> # Multi-head Attention



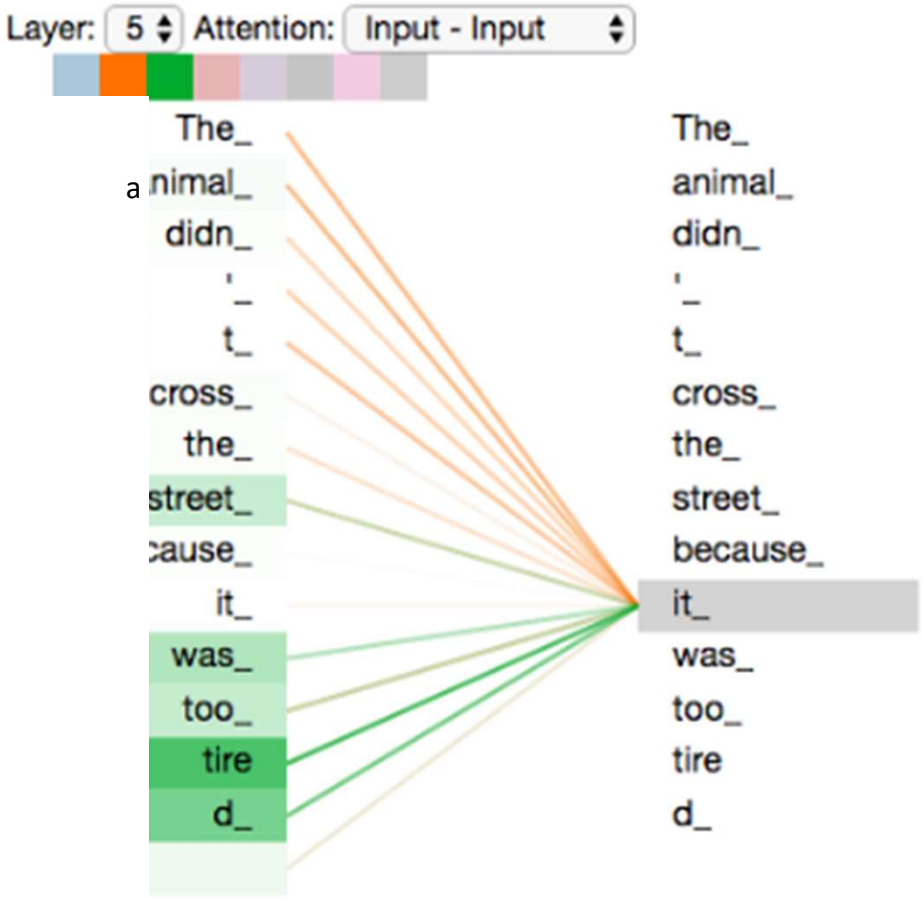
Part 3 >> **Multi-head Attention**



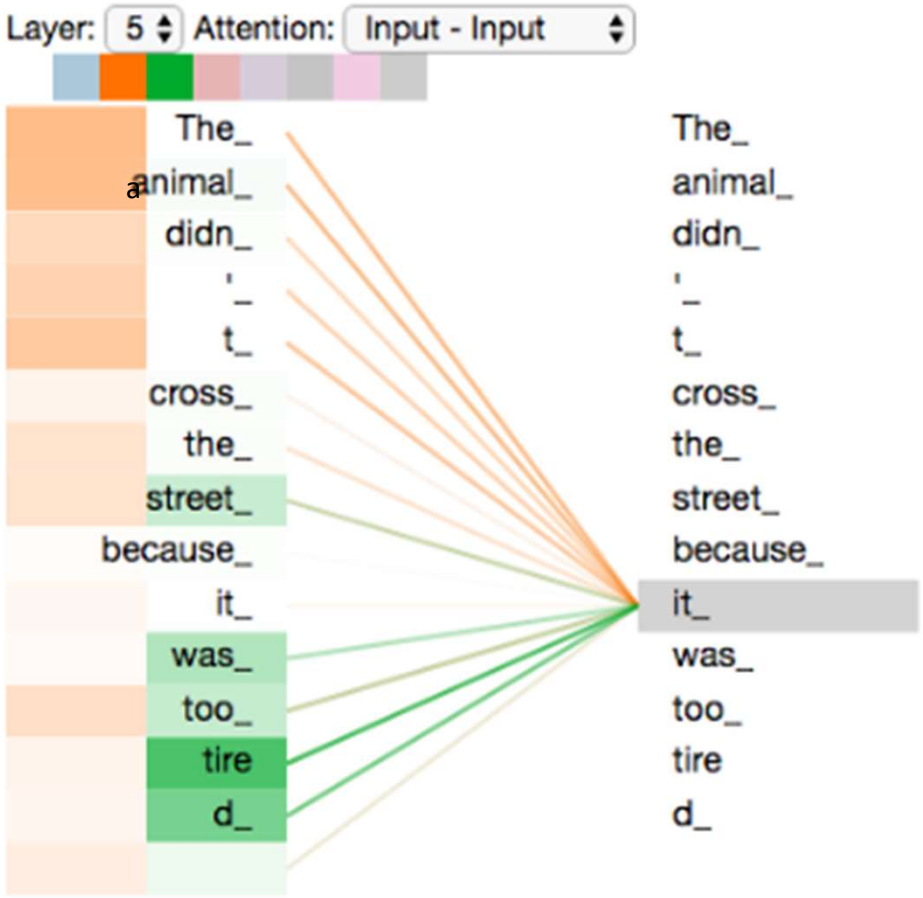
Part 3 >> # Multi-head Attention



Part 3 >> **Multi-head Attention**



Part 3 >> **Multi-head Attention**

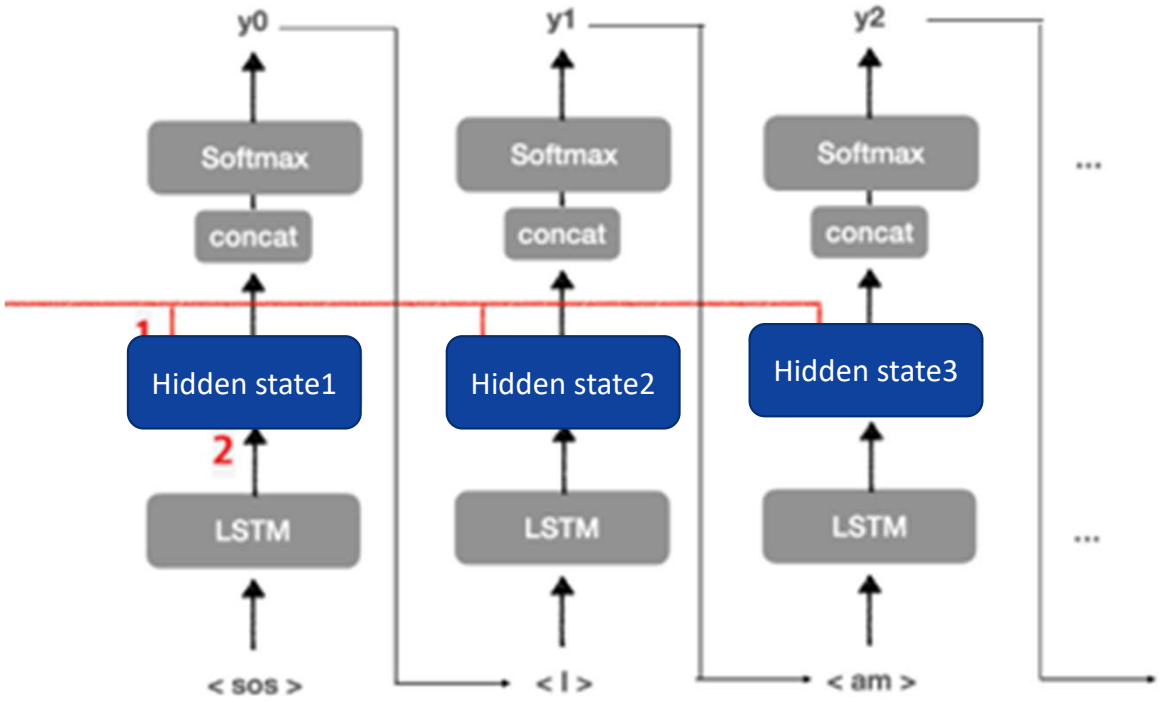


4

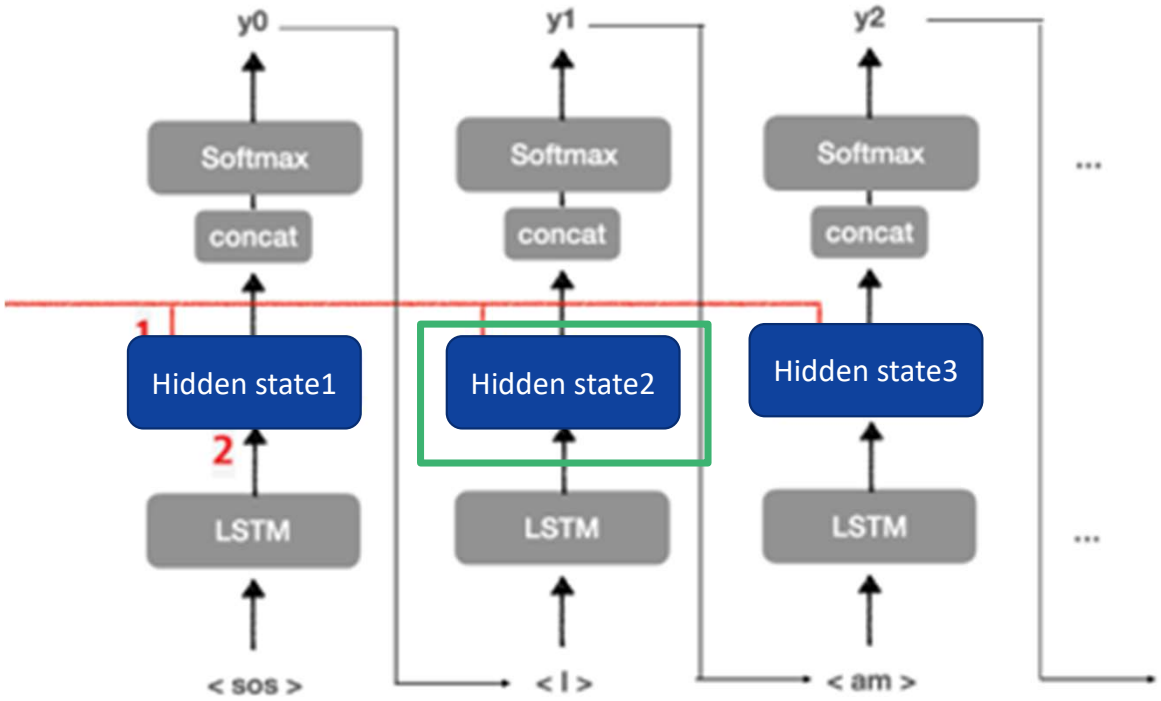
Masked Self Attention

“Masked self Attention”

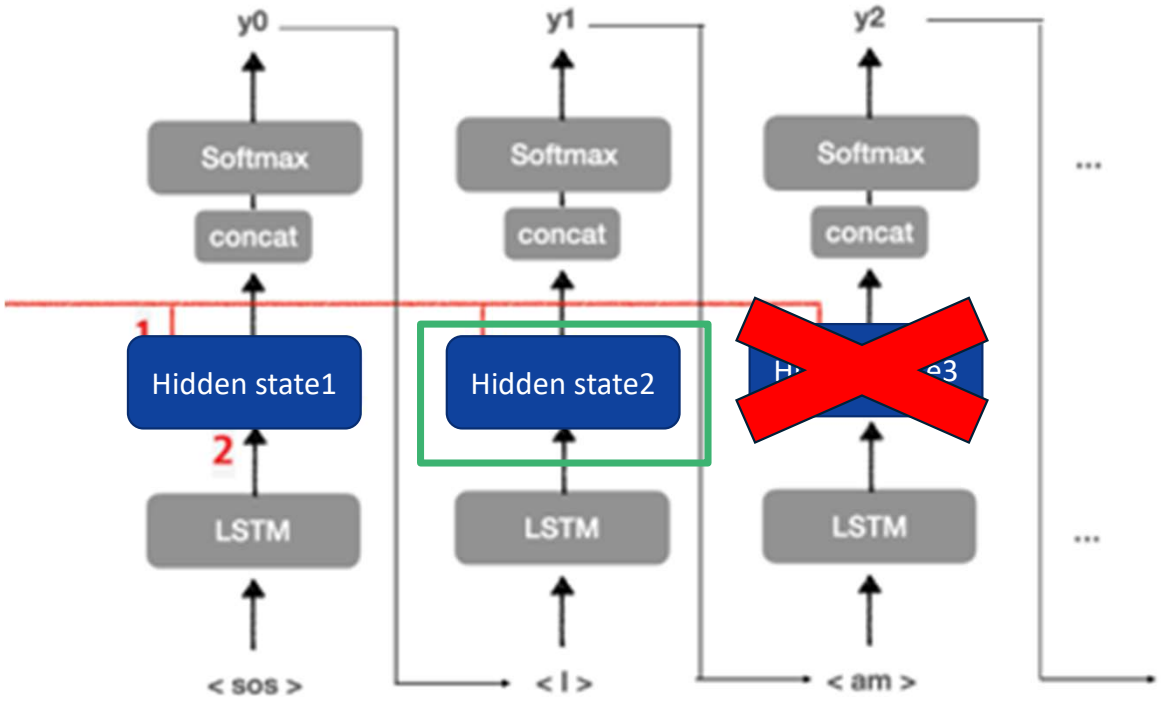
Part 4 >> # Masked Self Attention



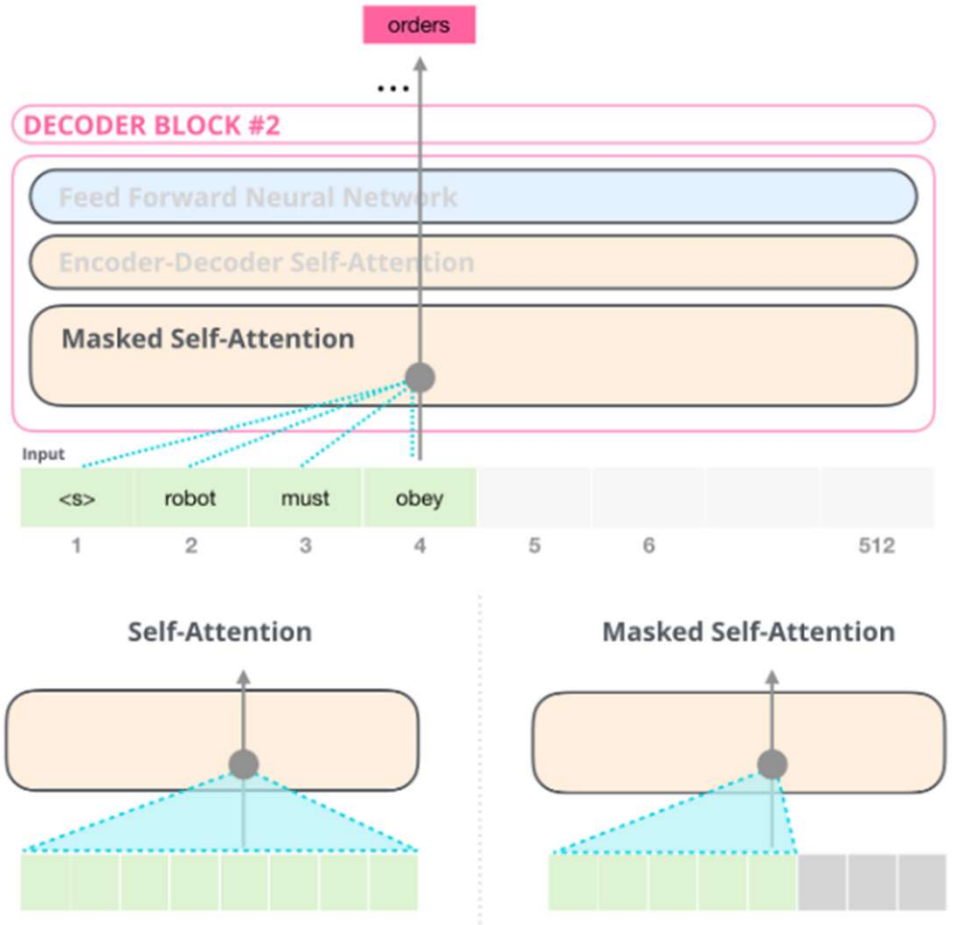
Part 4 >> # Masked Self Attention



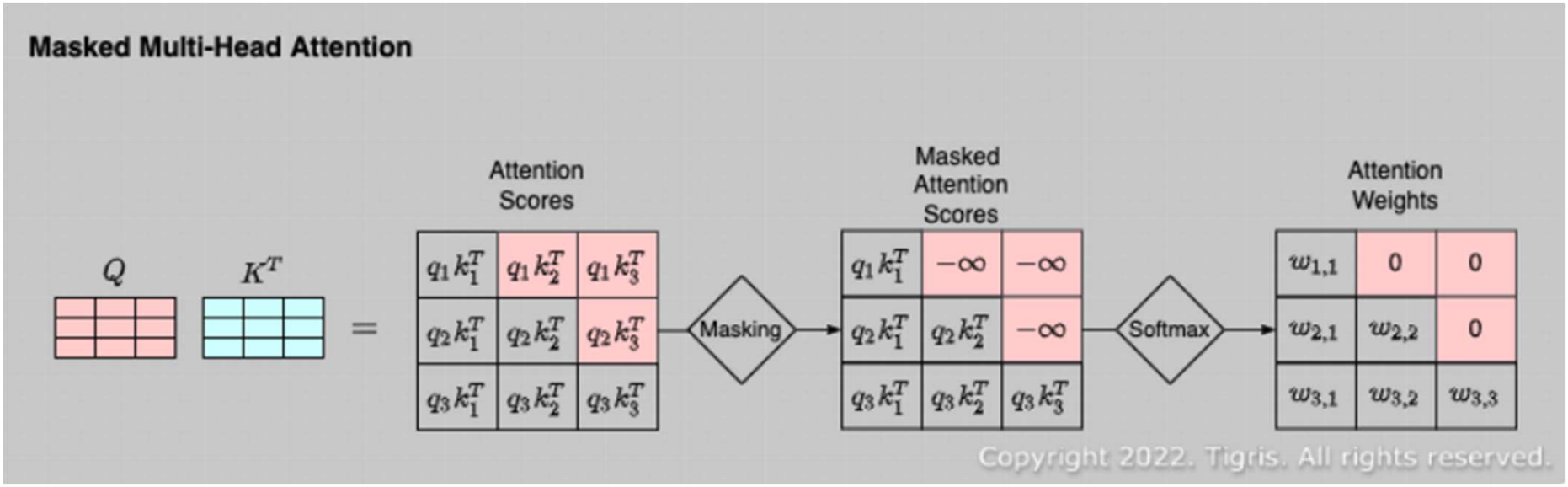
Part 4 >> # Masked Self Attention



Part 4 >> # Masked Self Attention



Part 4 >> # Masked Self Attention



Part 4 >> # Masked Self Attention

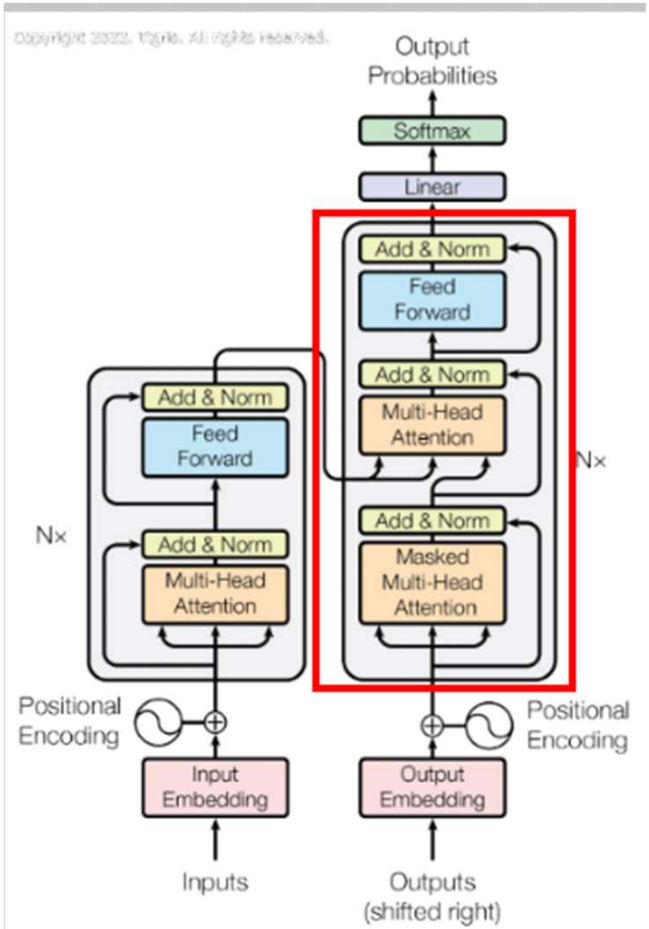


Figure 1: The Transformer - model architecture.

Thank you

Hyoungbum Kim