1. attention

Attention이라는 메서드는 seq2seq의 long-dependency problem 때 있을 수 있는 문제들을 보완하기 위해 탄생했습니다.

문제1) seq2seq에서 encoder의 모든 정보는 context vector라는 finite space 안에 담겨야하기 때문에 긴 input sequence들의 모든 정보를 담기 힘들 수 있음

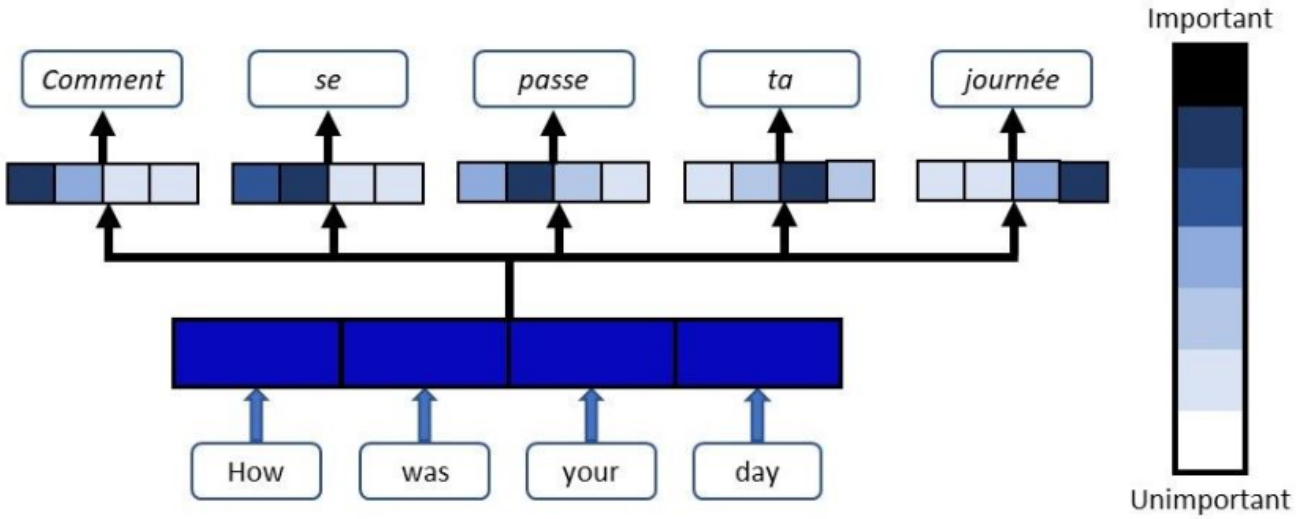
문제2) Long-dependency problem에 의해 input 초기에 등장한 내용들은 최종 context vector에서 해당 의미가 많이 퇴색되거나 미미할 수 있음

Attention은 위에서 언급한 문제들을 해결하기 위해 다음과 같이 seq2seq의 Encoder랑 Decoder를 개선한다.

-Encoder : 각 time 별 hidden state vector들을 모두 사용할 수 있도록 저장

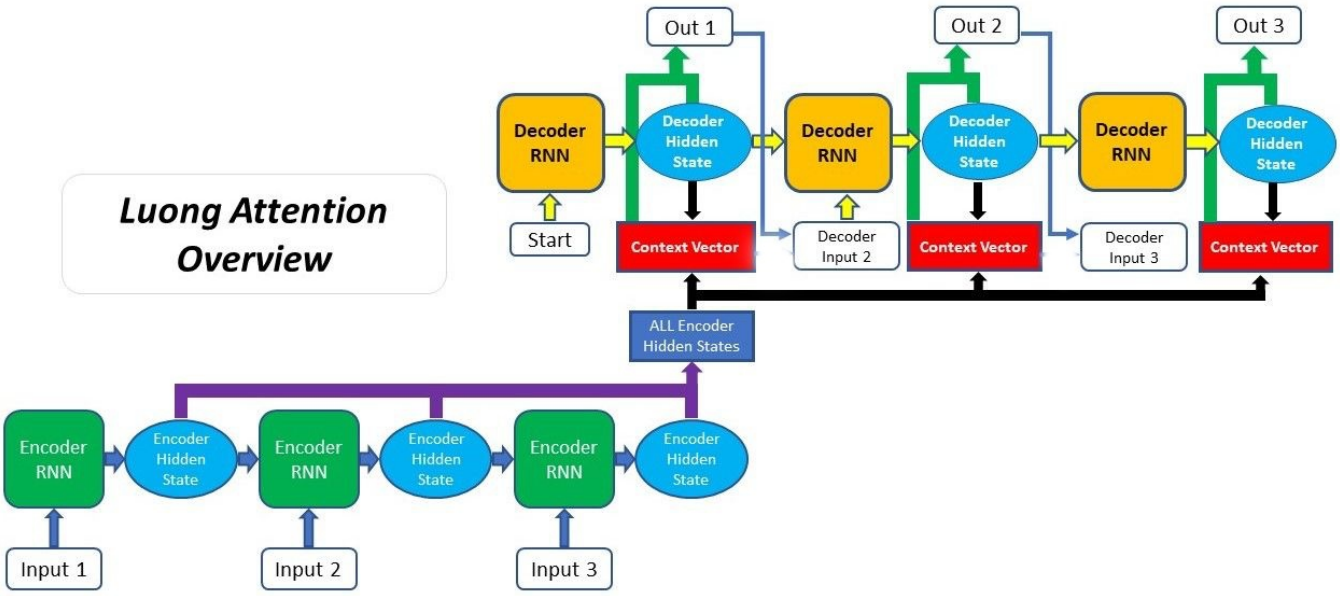
-Decoder : 1) 각 time 마다 출력되는 hidden state vector를 Encoder에서 구한 모든 time 별 hidden state vector랑 joint interaction을 시켜 Encoder에서 어떤 hidden state vector가 중요한지에 대한 weight를 구한다.

2) 구해진 weight들을 기반으로 Encoder에서 구한 hidden state vector matrix들을 elementwise multiplication을 해 Encoder hidden state vector들의 weighted sum을 구한다. 이를 context vector라고 칭한다.

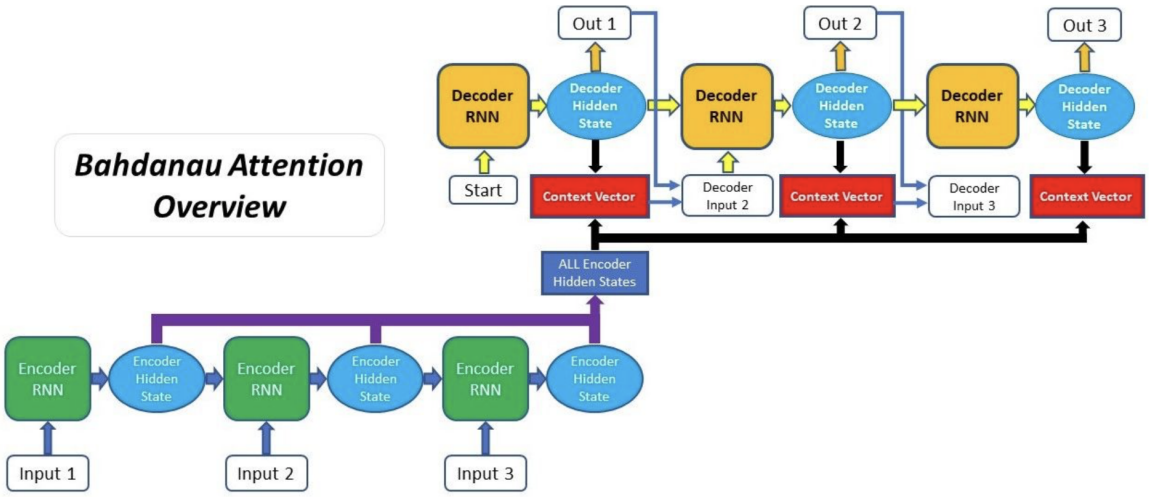


Weights are assigned to input words at each step of the translation

이렇게 구해진 context vector를 어떻게 활용하느냐에 따라 다양한 attention을 활용한 모델 구조들이 생긴다.



예를 들어 다음과 같이 "현재 input 단어 다음으로 나올 output 단어는 input sequence의 어디어디에 attention을 가진다"의 의미를 가지는 구조(Luong attention)도 만들 수 있으며



여기서 보이는 attention의 구조는 "현재 input 단어 다음으로 나올 output 단어를 찾기 위해 현재 input 단어의 attention 정보를 사용하겠다"의 의미를 가지는 구조(Bahdanau attention)를 가지고 있다.

이처럼 attention을 어떻게 활용하느냐에 따라 다양한 구조가 생길 수 있다.

이렇게 attention이 활용되는 구조만큼 중요한 것이 attention을 계산하는 방법이며 이 또한 다양하게 존재한다.

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

위의 표가 attention을 계산하는 메커니즘들을 정리한 표다.

1) Bahdanau attention

$$score_{alignment} = W_{combined} \cdot tanh(W_{decoder} \cdot H_{decoder} + W_{encoder} \cdot H_{encoder})$$

해당 attention 식에는 learnable weight가 총 3개 등장하며, 하나의 non-linear layer가 들어간다. 이 non-linear layer가 존재하지 않았다면 해당 식은 learnable weight가 3개가 아닌 2개만 존재하는 구조를 가지게 되고 또, 이 layer를 사용함으로써 더 다양한 representation의 내용을 담을 수 있게 된다.

2) Luong attention

Luong attention의 경우 Bahdanau attention과는 다르게 총 3가지 종류의 attention 수식을 논문에서 언급한다.

- Dot : 내적을 통해 벡터 유사도를 구해 해당 값을 가중치로 사용

$$score_{alignment} = H_{encoder} \cdot H_{decoder}$$

- General : Dot이랑 비슷하지만 해당 식에 weight를 추가

$$score_{alignment} = W(H_{encoder} \cdot H_{decoder})$$

- Concat : Bahdanau랑 비슷한 구조를 가진 식 But decoder weight이랑 encoder weight이 따로 존재하지 않고 하나의 weight를 공유

$$score_{alignment} = W \cdot tanh(W_{combined}(H_{encoder} + H_{decoder}))$$