Attention

이현재

1. Attention 등장 배경 (seq2seq 한계)

RNN에 기반한 seq2seq 모델의 한계는 다음과 같습니다.

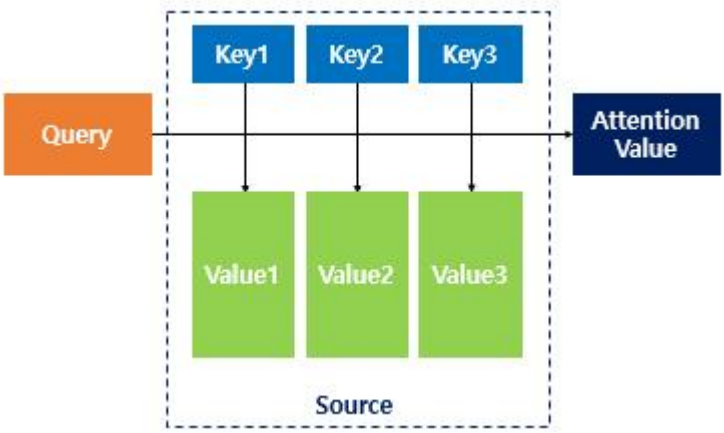
1. 하나의 고정된 크기의 벡터(Context Vector)에 대한 모든 정보를 압축하려고 하니 정보 손실이 발생
2. Gradient Vanishing 문제 발생

이를 위한 대안으로 입력 시퀀스가 길어지면 출력 시퀀스의 정확도가 떨어지는 것을 보정해 주기 위한 Attention이 등장하게 됩니다.

2. Attention 아이디어

어텐션의 기본 아이디어는 디코더에서 출력 단어를 예측하는 매 시점(time step)마다, 인코더에서의 전체 입력 문장을 다시 한 번 참고한다는 점입니다. 단, 전체 입력 문장을 전부 다 동일한 비율로 참고하는 것이 아니라, 해당 시점에서 예측해야할 단어와 연관이 있는 입력 단어 부분을 좀 더 집중(attention)해서 보게 됩니다.

Attention(Q, K, V) = attention value



<그림 1> Attention 구조

위의 함수는 Attention의 기본적인 구조입니다. 자세히 설명하면, Query(Q)에 대하여 모든 Key(K)와의 유사도를 구하고, 해당 유사도를 Key(K)에 매핑된 Value(V)에 반영하는 것입니다. 이 모든 Value 값들을 더하여 attention value를 얻게 됩니다.

3. Encoder 개선

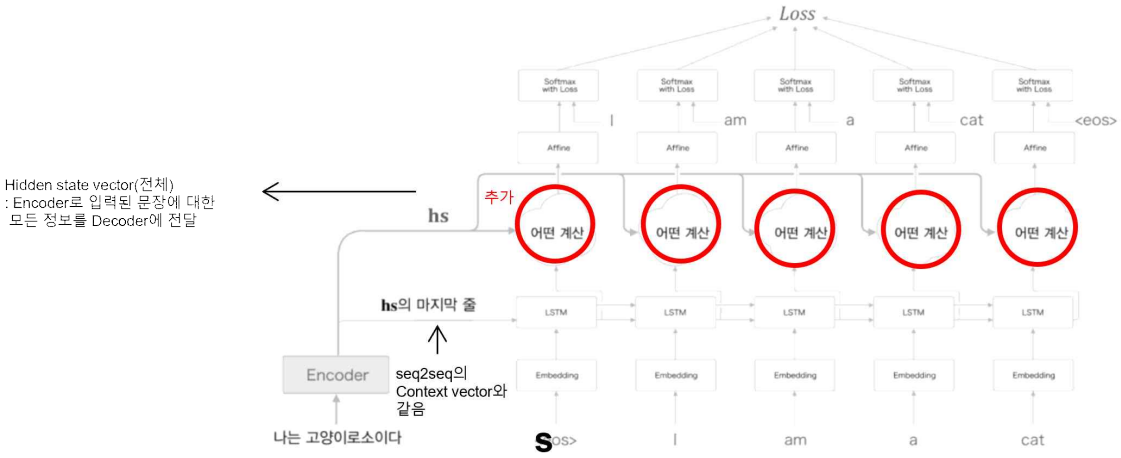
인코더가 만드는 Context Vector는 고정된 길이가 아닌 입력 문장의 길이에 맞추는 것으로 개선되었습니다. 즉, 각 Time step마다 hidden state를 만들어 stack으로 쌓는 것입니다.

<그림 2> 개선된 Encoder 구조

<그림 2>의 hs는 hidden state 행렬(혹은 벡터)로 이해하셔도 좋을 것 같습니다. 이 은닉 행렬의 행의 개수는 단어의 개수가 되며 열의 개수는 고정적인 값을 가지게 됩니다.

4. Decoder 개선 ① - Alignment 추출

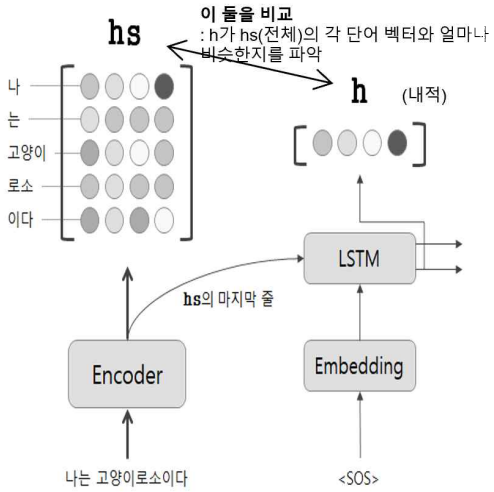
Encoder 개선에서 하나의 hidden state이 아닌 hidden state vector(Stack)를 만들어서 Decoder로 보내주었으므로, Decoder에서도 이를 모두 사용할 수 있도록 개선해야 합니다. 이 과정에서 단어의 대응 관계를 나타내는 정보인 Alignment를 추출하게 됩니다. '특정 단어'와 대응 관계가 있는 '특정 단어'의 정보를 골라내는 것이 Attention의 핵심이기 때문에 필요한 정보에 주목하여 그 정보로부터 문장을 변환하게 됩니다. Alignment를 추출하는 방법으로는 Decoder의 각 Time step에서 출력하고자 하는 단어와 대응 관계인 단어의 vector를 hidden state vector(전체)에서 **가중치**를 사용하여 선택하게 됩니다. 예를 들어, Decoder가 'I'를 출력하려고 할 때, 전체인 hidden state vector에서 'I'에 대응되는 '나'에 대한 정보가 담겨 있는 hidden state를 가중치를 조절함으로써 찾아내겠다는 것입니다. 마지막으로, <그림 3>처럼 '어떤 계산'이 추가되는데 이는 전체인 hidden state vector와 각 time step에서 Decoder의 hidden state를 **내적(Dot-Product)**하여 필요한 정보만을 넘겨주게 됩니다.



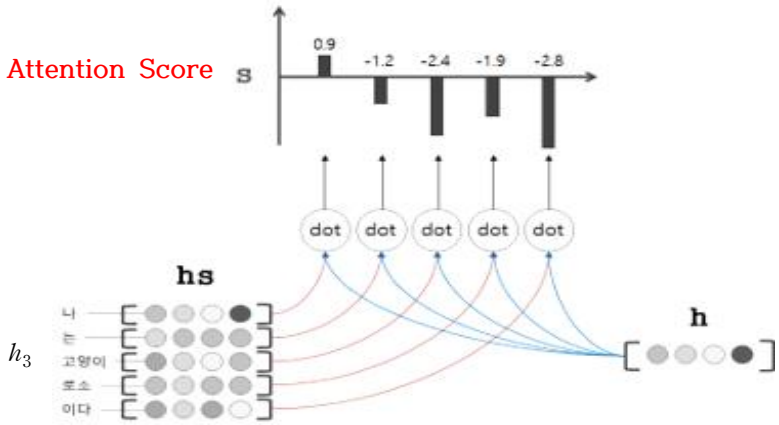
<그림 3> 개선된 Decoder를 가진 Attention 구조

5. Decoder 개선 ② - 내적

벡터에서 내적이란, 두 벡터가 얼마나 같은 방향을 향하고 있는지 판단할 수 있는 방법입니다.(B라는 기준 벡터에서 A라는 벡터가 얼마나 B 벡터의 성분에 들어있나로 해석할 수 있습니다.) 모든 원소를 다 곱하고 더해서 나오는 스칼라값으로 이를 판단할 수 있게 합니다. <그림 4>에서 Encoder를 통해서 나온 hidden state vector와 특정 시점의 Decoder의 hidden state를 내적하게 됩니다.



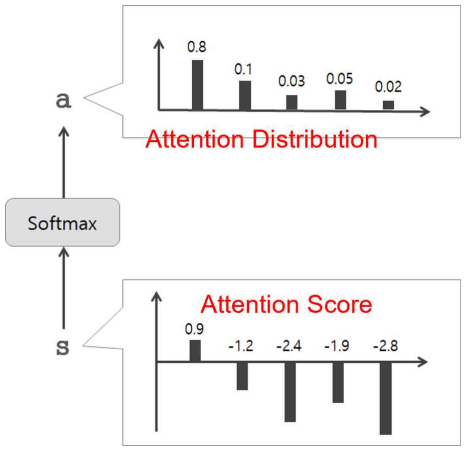
<그림 4> Encoder의 전체 hidden state와
Decoder의 hidden state 내적



<그림 5> 내적 과정

지금부터 <그림 4>의 자세한 과정을 설명하도록 하겠습니다.

Encoder의 전체 hidden state vector의 i 번째 hidden state를 h_i라고 할 때, Decoder의 hidden state와 모든 $h_{i(i=1,2,...,5)}$를 내적해줍니다. (실제 Attention에서는 Decoder의 hidden state를 Transpose해주게 됩니다.) 이 내적해서 나온 스칼라 값을 **Attention score**라고 합니다. <그림 5>에서 [0.9, -1.2, -2.4, -1.9, -2.8] 값들이 모두 Attention score입니다. 내적을 통해서 유사도 값을 구하였지만, 내적은 음수값이 나올 수 있기 때문에 <그림 6>처럼 이를 softmax 함수를 거쳐서 정규화시켜줍니다.



<그림 6> softmax 함수

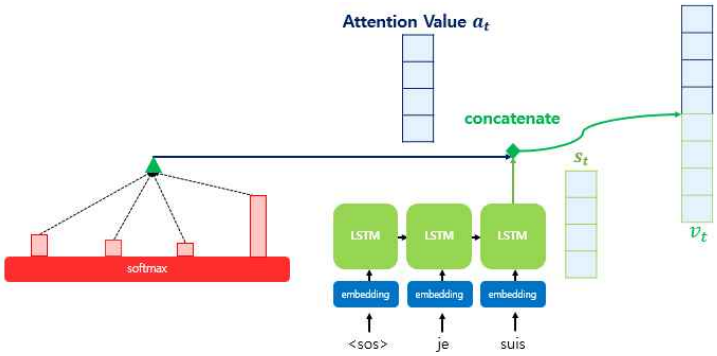
Attention score를 softmax해준 것을 **Attention distribution**이라고 합니다. 이 Attention distribution의 각각의 값을 **Attention weight**라고 합니다. 예시를 통해서 '나'에 대한 정보를 담고 있는 hidden state와 h 벡터의 Attention weight가 가장 높은 것을 확인할 수 있습니다.

6. Decoder 개선 ③ - 가중합(weighted sum)

지금까지 얻은 정보를 하나로 합치는 단계입니다. Attention의 최종 결과값을 얻기 위해서

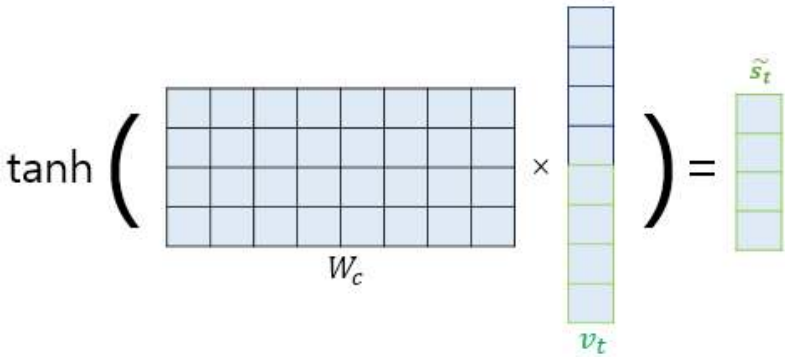
각 Encoder의 각 hidden state($h_{i(i=1,2,\ldots,5)}$)와 Attention weight를 곱하고, 최종적으로 모두 더합니다. 요약하면 가중합(Weighted sum)을 진행합니다. 이렇게 나온 가중합 값을 **Attention value**라고 합니다. 이러한 Attention value를 종종 인코더의 문맥을 포함하고 있다고 하여, Context vector라고도 불립니다. 앞서 배운 가장 기본적인 seq2seq에서는 인코더의 마지막 hidden state를 Context vector라고 부르는 것과 대조됩니다.

7. 예측



<그림 7> Concatenate

Attention의 최종값인 Attention value를 구했습니다. Attention 메커니즘은 <그림 7>처럼 Attention value와 Decoder의 hidden state를 결합(concatenate)하여 하나의 벡터로 만드는 작업을 수행합니다. 이 결합된 벡터인 v_t라고 하고 이 v_t를 $\hat{y}$ 예측 연산의 입력으로 사용함으로써 인코더로부터 얻은 정보를 활용하여 $\hat{y}$를 좀 더 잘 예측할 수 있게 합니다.



<그림 8> vector $\tilde{s_t}$ 생성

논문에서는 v_t를 바로 출력층으로 보내기 전에 신경망 연산을 한 번 더 추가하였습니다. 가중치 행렬과 곱한 후에 하이퍼볼릭탄젠트 함수를 지나도록 하여 출력층 연산을 위한 새로운 벡터인 $\tilde{s_t}$를 얻습니다. 어텐션 메커니즘을 사용하지 않는 seq2seq에서는 출력층의 입력이 t시점의 hidden state였다면, Attention에서는 출력층의 입력이 $\tilde{s_t}$입니다. 식으로 표현하면 W_c는 학습 가능한 가중치 행렬, b_c는 편향입니다.

$$\tilde{s_t} = \tanh(W_c[a_t;s_t] + b_c)$$

이 $\tilde{s_t}$를 출력층의 입력으로 사용하여 예측 벡터를 얻습니다.

$$\hat{y_t} = Softmax(W_y \tilde{s_t} + b_y)$$

8. 코드

코드는 따로 첨부하겠습니다!

9. 참고(Q, K, V) - self Attention

Q = Query : t 시점의 디코더 셀에서의 hidden state
K = Keys : 모든 시점의 인코더 셀의 hidden states
V = Values : 모든 시점의 인코더 셀의 hidden states

<그림 9> Q, K, V 예시

이 문장에서 'it'이 animal인지 street인지 알기 위해서 Q, K, V 개념을 도입합니다.

Query는 우리가 알고자 하는 객체, 이 경우에서는 it,
Keys는 이 문장 전체를 의미하고,
Values는 각 Keys를 통해서 얻어낸 유사도 값입니다!