



Seq2Seq with Attention

Effective Approaches to Attention-based Neural Machine Translation


<https://arxiv.org/abs/1508.04025>

부스트캠프 AI tech 5기 NLP 김태민






목차

- Introduction
 - **Attention-based Models**
 - Attention Layer
 - Learning techniques
 - Experiments
 - Question
- 



Introduction

기존 연구의 한계점

- Seq2Seq 모델에서의 Context Vector가 병목 현상을 발생시킨다.
 - NMT에서 Attention기반의 유용한 아키텍처를 가진 모델이 거의 없다.
 - 기존의 Attention을 사용하지 않는 모델은 LSTM,GRU의 사용에도 불구하고 긴 문장을 잘 처리하기가 어렵다.
 - 기존의 soft attention은 연산량이 매우 많으며 hard attention은 미분이 불가능하여 어려운 테크닉이 들어가게 된다.
- 




Introduction

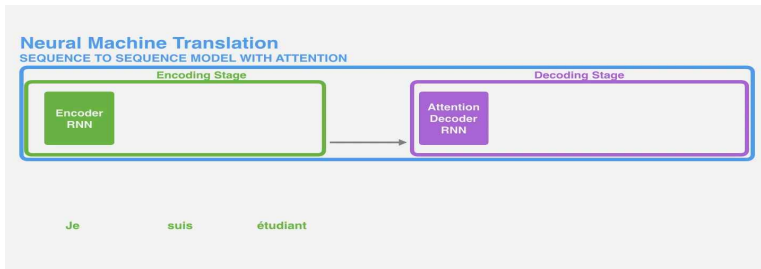
전체 개요

- 기존 Encoder-Decoder에서 Decoder 구조에 Attention 메커니즘을 도입 하였다.
- 기존의 Encoder 마지막 hidden state만 사용 되었던거와 달리 모든 Encoder hidden state를 사용하여 attention 메커니즘을 적용 하였다.
- 기계 번역 작업에서 영어를 프랑스어로 영어를 독일어(WMT'15)로 번역하는 테스트에서 SOTA를 달성
- Attention 메커니즘의 도입으로 보다 긴 문장을 잘 처리한다.

soft attention과 hard attention의 조합인 Local attention이라는 새로운 attention 방법을 제시한다.



Attention-based Models



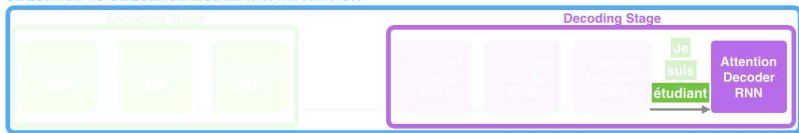
- 기존의 Seq2Seq모델이 Encoder의 마지막 hidden state만 사용했던거와 달리 Encoder의 모든 step에서 hidden state를 Decoder에 전달해주어 이를 통하여 output을 통과시킨다.
- 기존의 일반 Decoder와 달리 Attention Decoder로 실제로는 Decoder의 hidden state 위에 Attention Layer가 추가된다.

Attention-based Models

Time step: 7

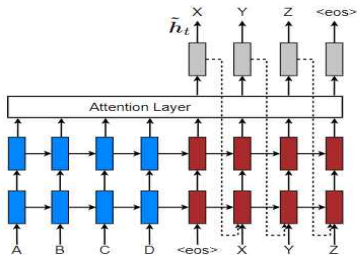
Neural Machine Translation

SEQUENCE TO SEQUENCE MODEL WITH ATTENTION



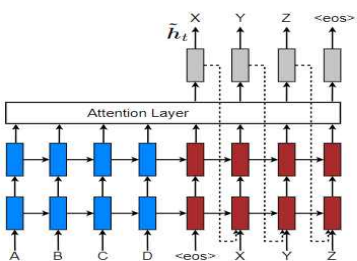
- 만약 입력이 위 그림과 같이 Je -> suis -> etudiant의 순서로 들어가게 된다면 디코더의 입력으로는 <eos> -> I -> am -> a가 입력으로 들어가게 되고 Time step7인 a가 입력으로 들어갔을때 attention에 의해 a와 같은 포지션인 etudiant가 가장 밝게 보이는 것을 볼 수 있다.

Attention-based Models



- 실제 입력으로는 토큰 A~D가 Encoder의 입력으로 들어가게 되고 <eos>토큰이 디코더의 입력으로 들어가 각각의 output을 출력하게 된다. 이때 Encoder의 각 hidden state가 Attention Layer에 입력되고 Decoder에선 매 Time step 마다의 Decoder hidden state 1개가 Attention Layer에 입력 된다.
- 최종적으로 모든 Encoder hidden state와 각 Time step의 Decoder hidden 1개가 입력되어 attention layer를 통과하고 FFN를 거쳐 \tilde{h}_t 이를 통해 output이 나오게 되고 두 번째 Decoder의 입력부터는 output과 \tilde{h}_t 그리고 t-1의 hidden state가 함께 들어가게 되어 이를 반복한다.

Attention-based Models



$$p(y_t|y_{<t}, x) = \text{softmax}(W_s \tilde{h}_t)$$

$$\tilde{h}_t = \tanh(W_c[c_t; h_t])$$

$$J_t = \sum_{(x,y) \in \mathbb{D}} -\log p(y|x)$$

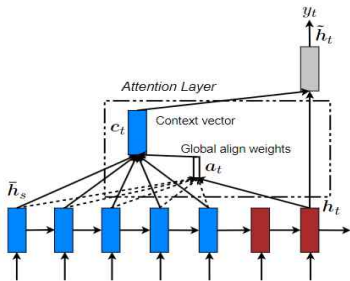
- 최종 목적식 J_t 은 위와 같으며 D는 우리의 training corpus이며 x가 주어졌을때 y에 log를 씌우고 -를 붙힌후 이를 다 더하는 공식이다.

-Attention Layer을 통과한 후의 $h(\sim)_t$ 는 Attention Layer의 내부에 있는 Context vector c_t 와 현재 time step의 hidden state h_t 와의 각종 연산으로 계산 되며 이에 가중치를 곱하고 tanh 활성화 함수를 적용시켜 $h(\sim)_t$ 를 구하게 된다.

- Time step의 output인 y_t 는 $h(\sim)_t$ 와 가중치를 곱하여 softmax 함수를 통과하여 결정된다.

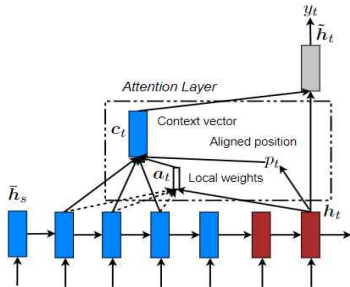
Attention Layer

Global Attention



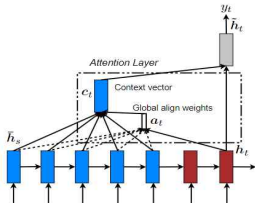
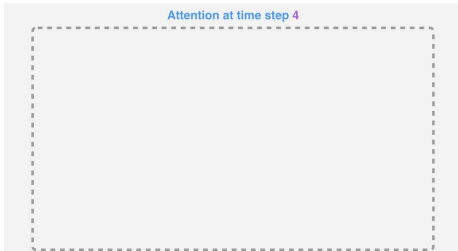
- Attention Layer는 총 2가지의 종류로 구분된다.

Local Attention



- Global Attention은 모든 Encoder의 hidden state를 보는 것이며
- Local Attention은 Windows 사이즈 만큼의 주변 Encoder의 hidden state를 본다.

Global Attention



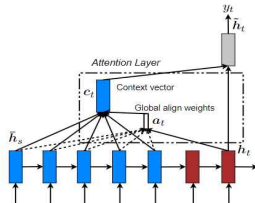
- 위 영상처럼 Encoder의 모든 h 와 현재 Decoder의 h_t 를 통하여 score를 계산한다.
- 이후 score에 softmax함수를 적용하고 나온 벡터를 다시 Encoder의 모든 h 와 weight sum을 하게되어 최종적인 Context vector인 c_t 가 나오게 된다.

Global Attention

$$\text{score}(h_t, \bar{h}_s) = \begin{cases} h_t^\top \bar{h}_s & \text{dot} \\ h_t^\top W_a \bar{h}_s & \text{general} \\ v_a^\top \tanh(W_a[h_t; \bar{h}_s]) & \text{concat} \end{cases}$$

$$a_t = \text{softmax}(W_a h_t) \quad \text{location}$$

$$\begin{aligned} a_t(s) &= \text{align}(h_t, \bar{h}_s) \\ &= \frac{\exp(\text{score}(h_t, \bar{h}_s))}{\sum_{s'} \exp(\text{score}(h_t, \bar{h}_{s'}))} \end{aligned}$$



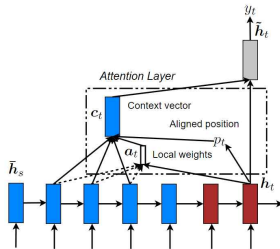
- 실제 a_t 를 연산하는데 score값은 위와 같은 방식으로 구한다. 논문에서는 단순히 초기에는 location을 사용하였다고 한다.

- 그 후 a_t 는 softmax를 적용하여 최종적인 a_t 를 얻어내게 된다. 마치 self attention처럼 softmax를 적용하게 되면 관계성을 확률로 변환시킨다.

- c_t 는 Encoder의 모든 h 와 a_t 와의 weight sum을 하여 Context vector c_t 를 완성한다.

- 추후 c_t 와 h_t 를 concat 시키고 FFN을 통과시켜 output을 뽑아낸다.

Local Attention



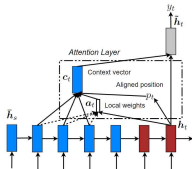
- 이 논문에서 주장한 Local Attention은 soft attention과 hard attention의 어딘가 그 중간이라고 설명한다.
- Global attention과 다른 점은 모든 Encoder의 h 만 쓰는것이 아니라 window 사이즈를 지정해 위 사진 처럼 현 t 의 위치의 윈도우 사이즈(1)로 양 옆의 encoder h 만 사용한다는 점이다.
- 또 다른 점으로는 p_t 라는 벡터를 만들게 되는데 이는 다음 슬라이드에서 설명하도록 하겠다.

Local Attention

$$p_t = S \cdot \text{sigmoid}(v_p^\top \tanh(W_p h_t)),$$

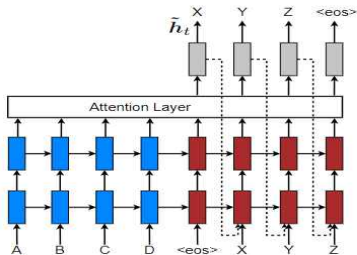
$$a_t(s) = \text{align}(h_t, \bar{h}_s) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right)$$

$$a_t = \text{softmax}(W_a h_t) \quad \text{location}$$



- 우선 p_t 는 현재 대상 단어의 위치를 단일로 계산하는 역할을 하게 된다. 이후 c_t 에서 a_t 와 함께 연산할때 두번째 수식처럼 가우시안 분포의 평균에 들어가게되는데
- 이를 통하여 a_t 는 윈도우 사이즈 만큼의 h 와 현재 h_t 의 값이 들어가 있는데 정규분포에서 임의로 샘플링 된 값을 곱하여 새로운 벡터를 얻어낸다.
- 이후 c_t 와의 계산은 Global Attention과 동일하다.

Feed input

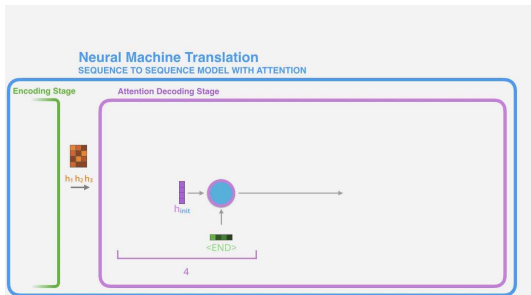


- feed input 은 단순히 $h(\sim)$ t가 next-step의 input으로 들어가는 부분이다
위 사진에선 점선으로 표시되었다.

- feed input으로 인해 다음 스텝을 생성할때 이전에 번역된 단어를 고려할수 있게된다. 또한 긴 문장을 고정 길이 벡터로 인코딩할때 발생하는 정보 손실을 완화하는 역할 또한 한다.

- 전체 문장의 맥락과 의미를 고려하는데 정확성을 향상 시킨다.


Attention Layer



- 실제 연산을 GIF로 보게되면은 편리하다.
- encoder의 h 와 현재 h_t 를 Attention하여 나온 벡터와 현재 h_t 를 concat 시켜서 FFN을 통과시키는 것을 볼 수 있다.



Learning techniques

- Dataset : WMT'14 (1억 1600만 영어단어, 1억 1천만 독일어 단어)
 - vocab : 자주 사용되는 단어 50000개 사용(없는 토큰은 <unk>로 대체)
 - 훈련 시 길이가 50단어 초과하는 문장 쌍은 필터링하며 진행하였다.
 - model : 4_Layers의 LSTM을 사용하며 1000 cell과 1000차원의 임베딩을 진행하였다.
 - 매개변수는 -0.1 ~ 0.1로 균일하게 초기화 하며 10 epochs로 훈련하였다.
 - optimizer : SGD를 사용하였으며 $lr = 1$ 이다.
 - 스케줄러 : 5epoch 이후에는 매 에포크마다 lr 을 절반으로 감소시킨다.
 - mini-batch size = 128
 - normalized gradient는 norm이 5를 초과할때마다 다시 조정한다.
 - Dropout은 0.2로 지정하여 사용하는데 Dropout을 적용한 Model은 12epoch로 훈련시키며 lr 은 8epoch 이후에 감소시킨다.
 - Local attention 모델같은 경우 window size $D = 10$
- 

Experiments

System	Ppl	BLEU
Winning WMT'14 system – <i>phrase-based + large LM</i> (Buck et al., 2014)		20.7
<i>Existing NMT systems</i>		
RNNsearch (Jean et al., 2015)		16.5
RNNsearch + unk replace (Jean et al., 2015)		19.0
RNNsearch + unk replace + large vocab + <i>ensemble 8 models</i> (Jean et al., 2015)		21.6
<i>Our NMT systems</i>		
Base	10.6	11.3
Base + reverse	9.9	12.6 (+1.3)
Base + reverse + dropout	8.1	14.0 (+1.4)
Base + reverse + dropout + global attention (<i>location</i>)	7.3	16.8 (+2.8)
Base + reverse + dropout + global attention (<i>location</i>) + feed input	6.4	18.1 (+1.3)
Base + reverse + dropout + local-p attention (<i>general</i>) + feed input	5.9	19.0 (+0.9)
Base + reverse + dropout + local-p attention (<i>general</i>) + feed input + unk replace		20.9 (+1.9)
Ensemble 8 models + unk replace		23.0 (+2.1)

- Base는 [Bahdanau et al.2015] D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. InICLR. 이 논문에 나온 기본 attention 모델이라고 볼 수있다.

- 실험 결과로 보았을때 마지막 가장 많은것을 적용한 모델(local-p)(general)이 가장 높은 점수를 나타내면서 앙상블을 하였을시 23.0 BLEU로 SOTA를 달성하였다.

Experiments

System	BLEU
Top – NMT + 5-gram rerank (Montreal)	24.9
Our ensemble 8 models + unk replace	25.9

System	Ppl.	BLEU
<i>WMT'15 systems</i>		
SOTA – <i>phrase-based</i> (Edinburgh)		29.2
NMT + 5-gram rerank (MILA)		27.6
<i>Our NMT systems</i>		
Base (reverse)	14.3	16.9
+ global (<i>location</i>)	12.7	19.1 (+2.2)
+ global (<i>location</i>) + feed	10.9	20.1 (+1.0)
+ global (<i>dot</i>) + drop + feed	9.7	22.8 (+2.7)
+ global (<i>dot</i>) + drop + feed + unk		24.9 (+2.1)

- 왼쪽 사진은 wmt'15 데이터셋인데 더 적은 wmt'14 데이터로 학습했음에도 불구하고 wmt'15에서 SOTA를 달성하였다.(English-German)

- 오른쪽은 WMT'15 German-English 테스트인데 높은 성능을 달성하였다.

Experiments

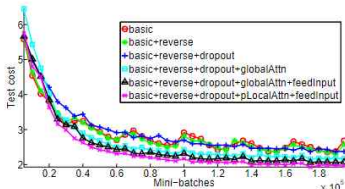


Figure 5: **Learning curves** – test cost (ln perplexity) on newstest2014 for English-German NMTs as training progresses.

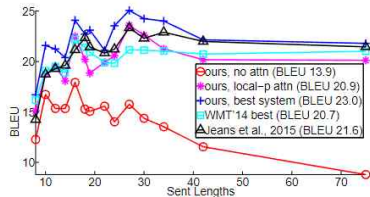


Figure 6: **Length Analysis** – translation qualities of different systems as sentences become longer.

- Learning curves를 보았을때도 Local에 저 방법을 적용한 것이 Test cost 가 가장 낮은 것을 볼수 있다.

- 우측 사진의 번역 품질을 보았을 때도 긴 문장에 대해 기존 방법은 잘 예측하지 못하였지만 저자가 주장한 논문에서는 긴 문장도 잘 번역되는것을 확인할 수 있다.

Question

Question

1. 앙상블에 대해 어떠한 다른 방식을 적용했는지에 대한 의문점
2. 기존의 soft attention과 hard attention의 문제점 그리고 방법론
3. 실제 모델의 구현에 대한 각종 연산들이 어떻게 이루어져서 최종 차원은 어떻게 나타나게 되는지
4. 아래 그림의 <eos> 토큰이 어떻게 바로 디코더로 들어가게 되는지에 대한 의문점

