Abstract

언어 모델로 RNN이 제안된 이후, gradient vanshing problem을 해결하여 긴 Sequence에 대해서도 잘 동작하는 LSTM이 연구되고, 딥러닝과 LSTM을 활용한 Seq2Seq 모델이 제시되었다. Seq2Seq 모델의 한계점을 극복하기 위해 attention 메커니즘이 등장하였다. attention 방법을 잘 활용한 모델로 Transformer 모델이 제시되었고, 이는 최신 모델은 GPT, BERT에서 사용되고 있다. 우리는 자연어 처리 분야에서 근간이 되는 Seq2Seq와 Attention에 대해 알아볼 필요가 있다.

1 GRU-Gated Recurrent Unit

[3]에서 언급된 GRU는 LSTM의 변형으로, 초기 LSTM에 비해 상대적으로 간단하게 디자인되었다. 성능적으로 동일하게 훌륭할 뿐만 아니라, 학습 파라미터가 줄어든 것이 장점이라고 할 수 있다. LSTM의 state는 input state, cell state, hidden state가 존재하였지만, GRU에서는 cell state가 존재하지 않으며, 총 3개의 gate를 사용한 LSTM과 달리 reset gate와 update gate 2개의 gate를 사용한다. GRU의 구조에 대해서 자세히 알아본다.

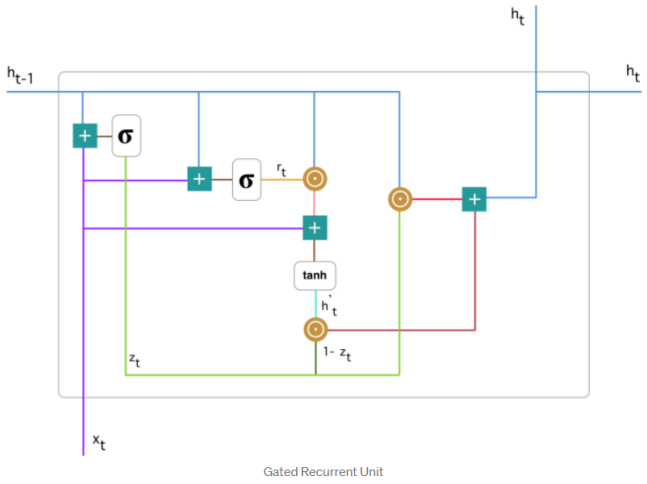

Figure 1: An architecture of GRU

먼저, GRU에 등장하는 4가지 formulae를 먼저 살펴본다.

$$z_t = \sigma(W^z x_t + U^z h_{t-1}) \quad (1)$$

$$r_t = \sigma(W^r x_t + U^r h_{t-1}) \quad (2)$$

$$h'_t = tanh(W x_t + r_t \odot U h_{t-1}) \quad (3)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \quad (4)$$

(3)는 current memory content로 현재의 입력을 반영한 정보이며, 이전 step의 정보인 h_{t-1}에 (2)가 Hadamard product(point wise)되어 현재 입력과 더해진 후 activation function(tanh)가 취해진다. (2)는 reset gate로 이전 step의 정보를 얼마나 지울지 결정하며, 현재 입력과 hidden state의 문맥 정보에 따라 0 1의 값으로 결정된다. (1)은 Update gate로, (2)과 동일한 수식을 갖는다. Update gate를 통과한 값 z_t는 과거정보 h_{t-1}에 곱해지고, $(1 - z_t)$만큼 현재 정보를 담은 current memory content에 곱해져 최종출력인 h_t를 (4)와 같이 출력한다.
GRU역시 LSTM과 마찬가지로 어떤 input의 gradient가 0이 되어 없어지지 않고 관련정보가 유지되어 다음 step으로 넘어가기 때문에 vanishing gradient문제를 해결하는 것을 확인할 수 있다.

, 참조문헌1 / 참조문헌2 / 참조문헌3

2 Seq2Seq

[4]의 등장이후, 딥러닝은 많은 task에서 높은 성과를 성취하고 있지만, 번역에서는 사용되고 있지 않다. [6]에서는 딥러닝을 기반으로 한 language model을 제안한다. 이 논문이 나온 시점에서 Deep neural network는 다양한 task에서 높은 성능을 보여주었지만, 입력과 출력의 차원이 고정되어 있는 경우에만 적용되어 왔다. 하지만 이는 speech recognition이나 번역과 같은 sequential data를 처리하는 작업에 있어 매우 큰 한계점이다. Seq2Seq는 encoder-decoder와 context vector의 개념을 제시하여 이를 해결하였다. 이를 통해 딥러닝을 기반으로 한 기계번역에서 state-of-the-art를 달성하였고, 이는 기계번역의 BLEU점수에서 가장 좋았던 결과(37.0)와 유사한 성능(36.5)을 보여주었다. Seq2Seq가 어떠한 방식을 사용하였는지 자세히 알아본다.

2.1 main concept - context vector

Translation task를 위한 Language model를 제안하는데 있어 2가지 문제가 존재하였다. 위에서 언급한 것과 같이 DNN에서는 입력과 출력의 차원이 고정되어 있는 문제가 있으며, 기존의 RNN 기반의 번역도 한계점이 존재하였다. RNN의 prediction은 이전의 값과 입력을 통해 결정되는데, 만약 한국어와 영어처럼 어순이 다르다면 정확한 결과를 유추하기 어려운 상황이 발생한다. 이 문제를 해결하기 위해 'context vector'의 개념을 제시한다. 뒤에서 언급할 인코더에서 LSTM의 hidden state는 입력이 들어올 때마다 갱신되는데, 그렇다면 마지막 hidden state는 앞의 모든 문장에 대한 정보를 포함하는 vector가 된다. 이 마지막 hidden state를 context vector로 하여 디코더에 넘겨주고, 디코더는 context vector로 부터 번역 결과를 추론하게 된다.

2.2 architecture

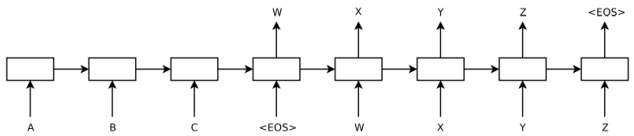

Figure 2: An architecture of Encode-Decode model

Figure2는 Seq2Seq model의 architecture이다. Figure2의 상황은 어느 한 나라의 언어인 "ABC"라는 문장을 받아 다른 나라의 언어인 "WXYZ"로 번역하는 것이다. 먼저 번역하고자 하는 문장인 "ABC" (Sequential data)를 받아 LSTM이나 RNN모델을 이용하여 context vector를 만들어 내는 인코더가 있으며, 이 context vector를 받아서 이를 토대로 번역 대상의 나라 언어를 이용한 문장 "WXYZ"로 번역을 진행하는 디코더로 이루어져 있다. 이때, 디코더에서 prediction 값을 다음 step의 입력으로 사용하는 것을 알 수 있다.
모델의 구현, 학습에서 또 다른 특징은 다음과 같다. 인코더와 디코더에서 사용하는 LSTM은 서로 다른 파라미터를 갖는다. LSTM을 하나만 사용하는 것이 아니라 총 4개를 겹쳐서 사용할 수 있다. 또한 학습 과정에서 입력 문장의 단어 순서를 뒤집어 학습한다. 입력 문장의 단어 순서를 뒤집는 것은 첫 번째로 등장하는 단어끼리 높은 연관성을 가질 수 있도록 할 수 있기 때문에 모델의 학습 난이도를 낮추고, 성능을 향상 시킬 수 있다.

2.3 conclusion

Seq2Seq의 알고리즘을 정리해보면 입력 문장은 단어 단위로 쪼개어 임베디드 layer로 들어가고, 출력된 임베딩 벡터를 입력으로 하여 인코딩을 거치게 된다. 입력 벡터와 이전 hidden state로 현재의 hidden state를 만들어 내는 과정을 반복하여, 최종 hidden state(context vector)를 디코더에 넘겨준다. context vector는 입력 문장의 모든 단어들의 정보가 요약하여

담겨있다고 할 수 있고, 이를 통해 디코더에서 번역을 진행한다. 예측된 벡터는 softmax되어 사용하기로 정해둔 단어 내에서 확률 값을 반환한다.

3 Attention

Seq2Seq는 인코더에서 입력의 정보를 함축한 context vector를 디코더에 넘겨주어 번역을 진행하는 아이디어를 제시하여 RNN기반 번역과정의 한계점을 해결하였다. 하지만 context vector는 크기가 고정되어 있어, 입력 Sequential data의 길이가 어떻든 문장에 대한 정보를 고정된 크기의 vector로 함축해야 한다. Sequential data의 정보를 고정된 크기의 vector로 담아내는 것에 대한 문제는 [2]에서 실험적으로 보여주었다. 그 문제는 학습된 모델을 text할 때 학습에서 보다 더 긴 문장을 사용하는 경우 성능이 떨어진다는 것이다. 이에 따라 [1]는 encoder-decoder model을 확장하여, 모델이 자동적으로 예측하려는 단어와 관련된 hidden state를 입력에서 찾아내는 concept을 제시한다. 이를 "attention"이라고 하는데, 입력 문장을 하나의 벡터가 아닌 vectors의 sequence로 인코딩하여, decoding시에 적절히 예측할 단어와 연관이 있는 subset을 vector의 sequence에서 선택하는 것이다. 이를 통해 긴 문장에 대해서도 모델이 잘 동작하는 것이 기대된다. 이제, Attention concept의 구현을 자세히 살펴본다.

3.1 decoding - Attention mechanism

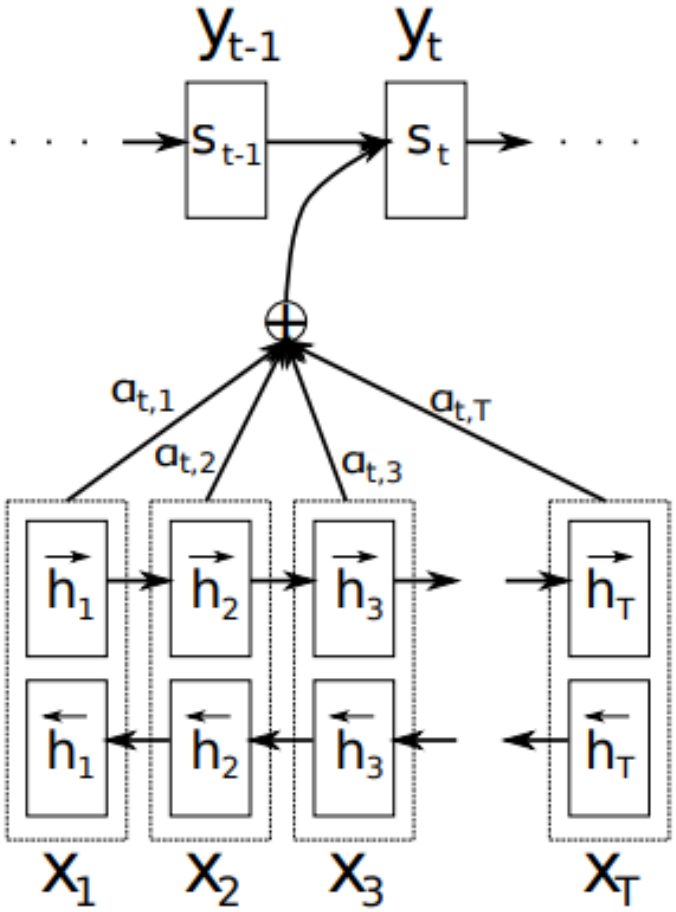

Figure 3: An architecture of Attention model

Figure3와 같이 인코더의 time step을 1, 2, ..., T라 하였을 때, 인코더의 hidden state는 $h_1, h_2, \cdots, h_T$가 된다. 그리고 디코더의 time step이 t일 때 구하고자 하는 hidden state는 s_t이다. 기존의 LSTM구조대로라면, 이전 time step의 hidden state h_{t-1}과 이전 시점의 예측 단어 y_{t-1}를 입력으로 하여 s_t를 결정한다. 즉 s_t는 $s_t = f(s_{t-1}, y_{t-1})$로 결정되는 것이다. 이때, attention mechanism은 아래 방정식을 통해 현재 시점의 hidden state를 구한다.

$$s_t = f(s_{t-1}, y_{t-1}, c_t) \tag{5}$$

기존과 달리 c_t의 값이 추가되었는데, 이를 attention value라 하며 이 어텐션 값이 구해지는 과정은 다음과 같다. time step의 index를 i라하고, 인코더의 time step의 index를 j라 하고 마지막 index를 T라 하였을 때, c_i는 아래의 방정식으로 구해진다.

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j \tag{6}$$

이때 α_{ij} vector는 현재 time step i에서 구하고자 하는 s_i와 관련이 있는 인코더의 hidden state에 대한 weight를 나타낸다. 다시 말하면, 현재 디코더의 i 시점에서 단어를 예측하는데 있어, 인코더의 모든 hidden state 각각이 s_i와 얼마나 유사한지를 나타내는 척도인 것이다. α_{ij}의 값은 아래와 같이 e 값의 softmax로 계산된다.

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T} exp(e_{ik})} \tag{7}$$

이때, e 값은 인코더 내 j 번째 position의 hidden state가 디코더 내 i 번째 position의 s_i와 얼마가 잘 맞는지 나타내는 score이다.

3.2 encoding - Bidirectional RNN

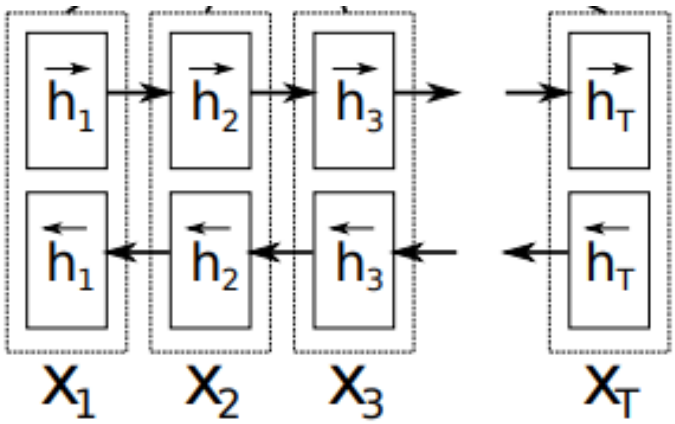

Figure 4: Bidirectional RNN

Figure4는 Figure3의 인코딩 부분으로, 인코딩 시 hidden state값이 정방향과 역방향에서 모두 구해지는 것을 확인할 수 있다. 이는 Bidirectional RNN 방식으로 [5]에서 제안되었다. RNN 방식은 제시된 Sequential data에서 차례로 단어를 입력으로 하여, 지금까지 주어진 것을 토대로 다음의 단어를 예측하는 것이다. 이와 달리 Bidirectional RNN는 시퀀스의 과거에 대한 정보뿐만 아니라, 이후 값의 정보까지 주어진 상태에서 prediction를 하자는 것이다. 하나의 예를 들어 설명하면, 직관적으로 이해가 된다. 첫 번째 입력으로 'She', 두 번째 입력으로 'is'가 들어왔다고 할 때, 그 뒤에 나올 단어는 무수히 많이 생각할 수 있다. 하지만, 그 뒤의 정보로 'beautiful'이라는 단어가 주어졌을 때, 'She is ( ) beautiful'에서 ( )에 맞는 단어는 이전보다 그 범위가 엄청나게 좁혀진다. 이렇게 문장의 양 끝에 대한 정보는 중간의 어떤 단어를 예측하는 것에 대해 상당히 중요한 정보로 작용하고, 이에 기반하여 Bidirectional RNN이 제시되었다.

4 Conclusion

Seq2Seq와 Attention paper에서는 영어와 불어 사이 번역 task를 다루어 연구를 진행하였다. 이 Language model은 각 나라의 언어 사이 번역 뿐만 아니라 다양한 곳에 사용될 수 있을 것이다. 이 모델을 초석으로 하여 프로그래밍 언어 사이 같은 작업을 수행하는 코드로 변환이 이루어지거나 자동으로 코드를 생성해주는 task도 할 수 있을 것이라 생각된다.

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[5] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[6] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.