**<Sequence to Sequence Learning with Neural Networks>**

1. **Introduction**

Deep Neural Network (DNN) 은 적당한 수의 단계에 대해 임의의 병렬 계산을 수행할 수 있어서 강력하다. 그러나 DNN은 input과 target이 고정된 차원의 벡터로 인코딩 할 수 있는 문제에만 적용될 수 있다. 예를 들어서, speech recognition과 machine translation은 길이를 알 수 없는 시퀀스로 가장 잘 표현 가능하다. 따라서 sequence to sequence는 domain-independent method가 적합하다.

본 논문에서는 두 개의 LSTM을 사용하여 sequence to sequence (seq2seq) 문제를 해결할 수 있음을 보인다. 먼저 하나의 LSTM이 한 번에 한 타임 스텝으로 input 시퀀스를 읽고, 큰 고정된 차원의 벡터 표현을 얻은 다음, 다른 LSTM을 사용하여 해당 벡터에서 output 시퀀스를 출력한다. 이때 두번째 LSTM은 input 시퀀스에 대해 조절된다는 점을 제외하면 본질적으로 Recurrent Neural Network Language Model이다.

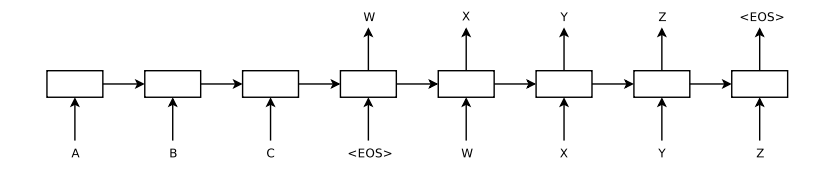


Figure 1. 이 논문의 모델은 input 시퀀스 “ABC”를 읽고, output 시퀀스 “WXYZ”를 만들어낸다. 모델은 end-of-sentence token(<EOS>)를 출력한 다음에 예측을 멈춘다. LSTM은 input 시퀀스를 역으로 읽음에 주의해야 한다. 이렇게 해서 데이터의 최적화 문제를 훨씬 쉽게 만드는, 많은 short term dependency를 만들어낸다.

1. **The model**

Recurrent Neural Network (RNN) 은 시퀀스에의 feedforward neural network의 natural generalization이다. Input 시퀀스 에 대해, standard RNN은 output 시퀀스 를 계산한다. RNN은 input과 output 사이의 시간에 대한 alignment를 알고 있다면, seq2seq를 쉽게 매핑할 수 있다. 그러나, input 시퀀스와 output 시퀀스가 복잡하고 단조롭지 않은 관계를 가지면서 길이가 서로 다르다면, RNN을 적용하는 것이 쉽지 않다.

일반적인 시퀀스 학습에 가장 간단한 방법은 input 시퀀스를 하나의 RNN으로 고정된 길이의 vector로 매핑한 다음, 그 벡터를 또다른 RNN을 사용하여 target sequence로 매핑하는 것이다.

LSTM의 목표는 조건부 확률 을 추정하는 것이다. 이때 과 는 다를 수 있다. LSTM은 마지막 hidden state에 의해 주어진 input 시퀀스의 고정 차원 표현 벡터 를 얻은 다음, 의 확률을 계산함으로써 조건부확률을 계산한다.

이 논문의 실제 모델은 다음의 세 가지 차별성을 갖는다.

1. 두 개의 다른 LSTM을 사용했다.

Input 시퀀스를 위한 것과, output 시퀀스를 위한 것이다. 왜냐하면 이렇게 하는 것은 계산 비용을 무시할 수 있을 정도로 모델의 파라미터 수를 증가시키기 때문이다. 그리고 multiple language pair에서 동시에 LSTM을 학습시키는 것을 자연스럽게 만든다.

1. deep LSTM이 shallow LSTM보다 상당히 우수한 성능을 내는 것을 알고 있으므로, 저자는 4개의 layer를 갖는 LSTM을 선택했다.
2. input 시퀀스의 순서를 뒤집는 것이 아주 중요함을 발견했다.

그래서 예를 들면, sentence 를 시퀀스 에 매핑하는 대신에, LSTM은 를 에 매핑한다. 는 여전히 에 매핑된다. 이러한 방식으로, 는 에 매우 가까워지고 는 에 상당히 가까워진다. SGD가 input과 output 사이의 communication을 설립하는 것을 쉽게 만든다. 이 간단한 데이터 변환이 LSTM의 성능을 대단히 향상시켰다.

1. **Experiments**

본 논문에서는 WMT’14 English to French MT task를 두 가지 방법으로 적용했다. 첫번째로, reference SMT system을 사용하지 않고 input 시퀀스를 직접적으로 번역하기 위해 사용했다. 두번째로, SMT baseline의 n-best list를 rescore 하기 위해 사용했다. 번역의 정확도와 sample translation을 제시하고, resulting sentence representation을 시각화한다.

3-1. Dataset details

WMT’14 English to French dataset을 사용했다. 348M개의 프랑스어 단어를 304M개의 영어 단어로 구성된 12M개의 문장 서브셋을 학습시켰다. 공개적으로 tokenized training 이용 가능하기 때문이다. 전형적인 neural language model이 각 단어의 vector 표현에 의존함에 따라, 두 언어 모두 fixed vocabulary를 사용하였다. Source language에서 가장 자주 사용된 160,000개의 단어와, target language에서 가장 자주 사용된 80,000개의 단어를 사용하였다. 모든 out-of-vocabulary word는 “UNK” 토큰으로 대체되었다.

3-2. Decoding and Rescoring

실험의 핵심은 많은 sentence pair에 대해 large deep LSTM을 훈련시키는 것이다. 주어진 source sentence 에 대해, 맞는 번역 의 log probability를 최대화시도록 학습시켰다. 따라서 훈련의 목표는

이때 는 훈련 집합이다. 일단 학습이 한 번 끝나면, LSTM에 따라 가장 확률이 높은 translation을 찾음으로써 번역을 만들어낸다. 이때 left-to-right beam search decoder를 사용했다. 이것은 적은 수의 부분 가정 를 유지시킨다. 부분 가설은 일부 번역의 prefix이다. 이것은 가설의 개수를 상당히 증가시키므로, 모델의 log probability에 따라 가장 가능성이 높은 를 제외한 나머지는 모두 제외시킨다. “<EOS>” 심볼이 hypothesis에 추가되면, beam에서 제거되고 complete hypothesis 집합에 추가된다. 이 디코더는 대략적인 반면에, 구현하기 쉽다. 흥미롭게도, 이 시스템은 심지어 beam size가 1일 때에도 잘 작동했고, beam size 2가 가장 잘 맞았다.

3-3. Reversing the Source Sentences

LSTM이 long term dependency 문제를 해결할 수 있으면서도, source sentence가 역순일 때훨씬 더 잘 배운다. (target sentence는 역순이 아니다.) 이렇게 함으로써, LSTM의 test perplexity는 5.8에서 4.7까지 떨어졌고, 디코딩된 번역의 test BLUE score는 25.9에서 30.6으로 증가했다.

이 현상에 대해 완전히 설명을 할 수 없지만, 이것이 데이터셋에 많은 short term dependency를 도입했기 때문이라고 믿고 있다. 보통, source sentence를 target sentence에 concatenate 할 때, source sentence의 각각의 단어는 target sentence에서 대응하는 단어에 멀리 떨어져있다. 결과적으로, 문제는 큰 “minimal time lag”을 갖는다. Source sentence의 단어들을 뒤집음으로써, source와 target에 대응하는 단어 사이의 평균 거리는 바뀌지 않는다. 하지만, source language의 첫 몇 단어들은 이제 target language의 첫 몇 단어들과 아주 가까워진다. 따라서, problem’s minimal time lag은 상당히 줄어든다. 그러므로, backpropagation은 source sentence와 target sentence 사이의 communication을 더 쉽게 만든다. 이것은 결국 전체적인 성능을 상당히 높인다.

초기에, input sentence를 뒤집는 것은 target sentence의 앞 부분에서 더 확실한 prediction을 만들고, 뒷부분에는 덜 확실한 prediction을 만들 것이라고만 생각했다. 그러나, reversed source sentence로 학습된 LSTM은 원래 문장을 학습한 LSTM보다 긴 문장에 대해 훨씬 더 잘 학습했다. 이것은 input sentence를 뒤집는 것이 LSTM의 memory utilization을 향상시킨다고 볼 수 있다.

3-4. Training details

LSTM 모델은 학습시키기 쉽다. 본 논문에서의 LSTM은 4개의 레이어로 구성되어 있고, 각 레이어는 1,000개의 cell을 갖는다. Input vocabulary로 160,000개를, output vocabulary로 80,000개를 사용했고, 1,000 차원의 word embedding을 진행했다. 그러므로, deep LSTM은 한 문장을 표현하기 위해 8,000개의 실수 값을 사용한다. deep LSTM이 shallow LSTM보다 상당히 우수한 성능을 낸다는 것을 발견했다. 각각의 추가적인 레이어는 perplexity를 거의 10%까지 감소시켰고, 아마도 훨씬 더 큰 hidden state 때문일 것이다. 각각의 output에서 80,000개의 단어에 naïve softmax를 적용시켰다. 결과적으로, LSTM은 384M개의 파라미터를 갖고, 그 중 64M은 순수하게 recurrent connection에 속한다. (32M은 encoder LSTM에, 32M은 나머지 decoder LSTM이다.) 전체 트레이닝 디테일은 아래와 같다.

* LSTM의 모든 파라미터를 -0.08과 0.08 사이에서 uniform distribution 형태를 띠도록 초기화시켰다.
* Momentum 없이 stochastic gradient descent (SGD) 를 사용하였고, learning rate는 0.7로 고정시켰다. 5 epoch 다음부터, epoch의 절반마다 learning rate를 절반으로 줄였다. 본 논문에서는 총 7.5 epoch 만큼 학습시켰다.
* Gradient 계산에서, batch size는 128이다. (batches of 128 sequences for the gradient)
* LSTM은 vanishing gradient problem을 잘 겪지 않지만, exploding gradient 문제를 갖는다. 그러므로, gradient의 norm이 threshold 값을 넘을 때마다 scaling 함으로써 강하게 제약을 두었다. 각각의 training batch에서, 를 계산했다. 이때 는 128로 나눈 gradient 값이다. 만약 이면, 로 세팅하였다.
* 서로 다른 문장끼리는 서로 다른 길이를 갖는다. 대부분의 문장은 짧다. (길이 20-30 정도) 하지만, 일부 문장들은 길다. (길이가 100이 넘음) 따라서, 128개의 랜덤하게 선택된 training sentence의 minibatch는 많은 short sentence를 가질 것이고, long sentence는 거의 갖지 않을 것이다. 결과적으로, minibatch에서의 computation 대부분은 낭비된다. 이 문제를 언급하기 위해, minibatch의 모든 문장은 대략적으로 같은 길이를 갖도록 했고, 결과적으로 속도를 2배 향상시켰다.

3-5. Parallelization

Deep LSTM은 C++로 구현이 되었고, 하나의 GPU는 초당 약 1,700개의 단어를 처리한다. 8개의 GPU 머신을 사용해 병렬 계산하였다. LSTM의 각 레이어는 다른 GPU에서 실행되었고, activation을 거친 결과를 다음 GPU/layer에 넘겼다. LSTM의 4개의 layer는 분리된 GPU에서 계산되었다. 나머지 4개의 GPU는 softmax를 병렬 연산하기 위해 사용되었고, 따라서 각각의 GPU는 1,000\*2,0000 크기의 행렬곱 연산을 진행했다. 결과적으로, 사이즈 128의 미니배치에서 영어와 프랑스어 모두에서 초당 6,300개의 단어를 처리했다. 이렇게 구현하여, 10일간 학습시켰다.

3-6. Experimental Results

테이블이(가) 표시된 사진

자동 생성된 설명

**Table 1.** WMT’14 English to French test set (ntst14) 에서의 LSTM의 성능이다. Beam size가 2인 5개의 LSTM의 앙상블이 beam size가 12인 1개의 LSTM보다 성능이 더 좋음에 주목하라.

테이블이(가) 표시된 사진

자동 생성된 설명

**Table 2.** WMT’14 English to French test set (ntst14) 에서 SMT system과 neural network를 함께 쓴 방법이다.

3-7. Performance on long sentences

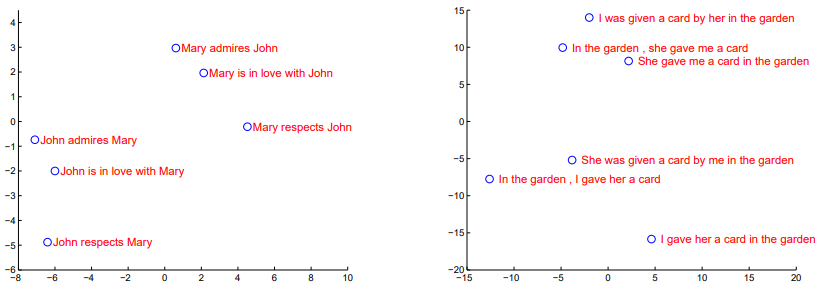
테이블이(가) 표시된 사진

자동 생성된 설명

**Table 3.** LSTM에 의해 만들어진 long translation과 ground truth의 몇 가지 예시이다. Google translate을 사용해서 translation이 잘 되었다는 것을 확인할 수 있다.

3-8. Model Analysis

본 논문에서 제시하는 모델의 흥미로운 특징 중 하나는 단어 시퀀스를 고정된 차원의 벡터로 바꾸는 능력이다.



**Figure 2.** 그래프에 보이는 phrase를 처리한 다음의 LSTM hidden state에서 2차원 PCA이다. phrase들은 단어로 cluster 되었고, 기본적으로 단어 순서의 기능이다. 이는 bag-of-words model로 잡아내기 어려운 특징이다. 양쪽 cluster가 비슷한 internal structure를 보임에 주목하라.

위의 Figure 2에서는 학습된 representation의 일부를 시각화 한 것이다. 이것은 representation이 단어의 순서에 민감함을 명백히 보여주고 있다. 반면 active voice를 passive voice로 대체하는 것에는 그렇지 않다.

1. **Related work**

(생략) 대충… 지금까지는 ~~ 방식으로 연구해왔고, 좀 더 최근에는 ~~ 방식으로 연구했고, 우리는 그 중에서 어떤 거랑 밀접한 관계가 있고… 이러이러한 연구들도 있었다. 라는 내용

1. **Conclusion**

이 연구에서, 제한된 vocabulary를 갖고, problem structure에 대한 거의 어떠한 assumption 없었던 large deep LSTM이 대규모의 MT task에서 vocabulary의 제한이 없는 SMT-based system보다 우수한 성능을 낼 수 있음을 보였다. 간단한 LSTM-based approach는, 충분한 training data를 갖는 많은 sequence learning problems에서도 좋은 성능을 낼 것이 분명하다.

Source sentence의 단어를 뒤집음으로써 어느 정도의 성능 향상을 이룬 것은 놀라운 결과이다. 따라서 저자는 가장 많은 short term dependency를 갖도록 problem을 encoding 하는 것이 중요하다고 결론 내리고 있다. 이렇게 함으로써, learning problem을 훨씬 더 간단하게 만들 수 있다. 특히, standard RNN에서도 source sentence를 reverse 함으로써 더 쉽게 학습시킬 수 있을 것이라고 주장한다.

LSTM은 매우 긴 문장에서도 잘 작동한다. 기존에는 비슷한 구조의 모델에서 긴 문장에 대해 성능이 그닥 좋지 않았지만, reversed dataset에서 학습시킨 LSTM은 long sentence를 학습하는 데 거의 어려움을 겪지 않았다.

가장 중요한 것은, 간단하고 복잡하지 않고, 상대적으로 덜 최적화된 이 접근 방법이 SMT system의 성능을 능가하였다는 것이다. 따라서 이후 연구는 훨씬 더 높은 accuracy를 이끌어낼 수 있을 것 같다. 저자는 어려운 seq2seq 문제에서도 잘 작동할 것이라고 주장한다.