# Sequence to Sequence Learning with Neural Networks

Ilya Sutskever, Oriol Vinyals, Quoc V.Le

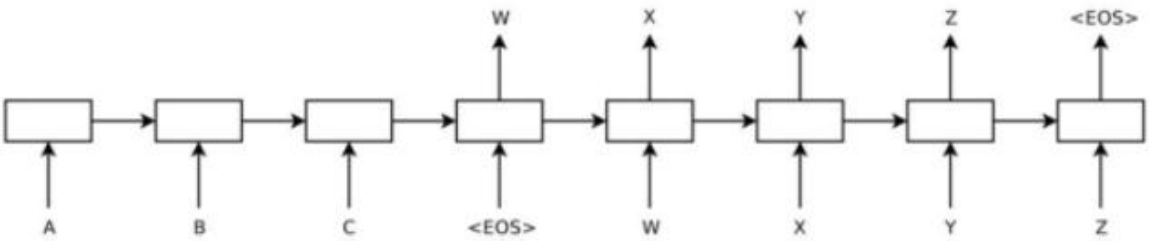
## Ⅰ. Introduction

Deep Neural Network은 speech 인식, 시각적 물체 인식과 같은 어려운 문제에 뛰어난 성능을 보였고 적당한 수의 단계로 임의의 병렬 계산, 충분한 네트워크의 파라미터를 가지고 있는 labeled training set을 supervised backpropagation을 수행할 수 있기 때문에 강력하였다. 하지만 input과 target이 고정된 dimension의 vector인 경우의 문제에만 사용할 수 있었다. 중요한 문제 중 많은 것들은 input과 target이 길이에 대한 정보가 미리 주어지지 않은 sequence이다. 예를 들어 QA(Question Answering) 문제도 주어진 가변 길이의 sequence를 answer에 대한 가변길이 sequence로 바꾸는 문제이다. 이 논문은 LSTM을 활용해서 Seq-to-Seq로 가는 구조를 소개한다.

하나의 LSTM이 input sequence를 읽고 하나의 timestep하고, large fixed-dimensional vector representation을 얻고 다른 LSTM을 사용하여 그 vector로부터 output sequence를 추출해낸다. 두 번째 LSTM은 그것이 input sequence에 조건되어있는 것을 제외하고는 필수적으로 recurrent neural network language model이다.

LSTM은 문장의 순서를 역순으로 하여 긴 문장에 어려움을 겪지 않았다. 그렇게 함으로써 optimization problem을 매우 간단하게 만드는 매우 짧은 term dependencies를 도입하였다. 그 결과, SGD는 LSTM을 문제 없이 학습할 수 있었다. 문장의 단어를 역순으로 하는 것은 이 작업의 key technical contributions 중 하나이다.

LSTM의 가장 유용한 특성은 가변 길이의 입력 문자을 고정된 차원의 벡터 표현으로 매핑하는 것을 학습한다는 것이다. 번역이 원본 문장을 의역한다는 경향이 있다는 점을 감안할 때 번역 목표는 LSTM이 유사한 의미를 가진 문장으로 의미를 포착하는 문장 표현을 찾도록 권장한다.



## Ⅱ. Model

Recurrent Neural Network는 sequence의 일반적인 feedforward neural network이다. input sequence $x_1,\ldots,x_n$에 대해 RNN은 아래의 수식을 반복하면서 output sequence $y_1,\ldots,y_T$를 계산한다.

$$h_t = \sigma(\mathbf{W}^{hx}x_t + \mathbf{W}^{hh}h_{t-1})$$
$$y_t = \mathbf{w}^{yh}h_t$$

RNN을 통해 input sequence를 앞선 시간 정보를 포함한 output sequence로 쉽게 매핑할 수 있다. 하지만 input의 길이와 output 길이가 다르고 두 길이가 간단한 관계로 이루어지지 않은 경우에는 RNN을 적용하기 어렵다.

이를 해결하기 위해 input sequence를 RNN을 통해 고정된 길이의 vector로 만든 후에 다시 그 vector를 RNN을 통해 우리가 원하는 target sequence를 구하는 방법이다. 단순한 RNN을 통해서도 할 수 있지만 RNN은 long-term에 대한 정보를 포함하지 않으므로 LSTM을 사용한다.

LSTM의 목표는 input sequence에 대한 output sequence의 조건부 확률을 구하는 것이다. Input sequence를 통해 고정된 길이의 vector representation v를 구하고 v는 첫 RNN의 마지막 hidden state가 된다. 그리고 다시 LSTM을 통해 $y_1,\ldots,y_T'$에 대한 확률을 아래와 같이 계산한다.

$$p(y_1,\ldots,y_{T'}|x_1,\ldots,x_T) = \prod_{t=1}^{T'} p(y_t|v,y_1,\ldots,y_{t-1})$$

위 식에서 우항은 전체 vocabulary의 단어를 통해 softmax 값을 계산해서 구한다. 문장이 끝난 지점에는 <EOS>라는 토큰을 사용해 문장이 끝났다는 정보를 주고 그 지점부터 다시 output의 계산을 시작한다.

실제 모델은 세가지 방법이 다르다. 첫번째는 input sequence와 output sequence에 대해 두 개의 다른 LSTM을 사용했다. 두번째는 deep한 LSTM의 성능이 더 뛰어난 것을 확인한 후 4개의 layer를 사용하는 LSTM을 사용했다. 세번째는 input sequence의 순서를 뒤집어서 사용했다. 이러한 점들이 LSTM의 성능을 많이 올려준다.

## Ⅲ. Experiments

WMT '14 English to French 기계번역 태스크에 이 모델을 적용시켰다. 그리고 다른 SMT 시스템을 참고하지 않고 직접 이 모델을 통해 번역했다.

1. Dataset details

   총 12M 개의 sentence 중 일부를 학습시켰는데 여기에는 348M개의 프랑스어 단어와 304M개의 영어 단어가 포함되어 있다. 두 언어에 대해서 각각 일정한 단어를 사용했는데

여기에는 160000개의 가장 많이 사용되는 단어가 포함되어 있다. 단어에 포함되지 않은 단어는 UNK라는 토큰으로 대체하였다.

2. Decoding and Rescoring

   이 실험의 핵심은 크고 깊은 LSTM 모델을 많은 문장 쌍으로 학습시키는 것이다. 학습은 주어진 문장 S에 대해 정확히 번역된 문장 T의 log확률값을 최대화하는 것이다. Objective 함수는 다음과 같다. 학습이 끝난 후에는 주어진 문장에 대해서 가장 높은 확률을 가진 문장 T를 찾는다.

$$1/|S| \sum_{(T,S)\in S} \log p(T|S) \qquad \hat{T} = \arg\max_{T} p(T|S)$$

   실제 예측 과정에서는 간단한 left-to-right beam search decoder를 사용했다. 어떤 특정 B개의 문장을 정하고 각 timestep마다 다른 문장들을 추가한 후 위의 log확률이 높은 B개를 제외하고 나머지는 모두 버린다. 그리고 EOS 토큰이 나오면 문장에 이 토큰을 더해 완성된 문장을 만든다. 디코더의 경우 근사시키는 방법인데 구현하기 매우 간단하다. Beam size를 1로 해도 잘 동작한다.

3. Reversing the Source Sentence

   LSTM에서 source sentence를 거꾸로 사용했을 때 더욱 좋은 결과를 만들어냈다. Perplexity는 5.8에서 4.7로 떨어졌고 BLEU score는 25.9에서 30.6까지 올랐다. 이유는 정확하지는 않지만 input sentence를 뒤집으면 각 대응되는 단어끼리의 평균 길이는 변하지 않고 처음 단어에 대해 대응되는 거리가 가까워지기 때문에 조금 더 효율이 올라갔을 것이라고 추측했다.

4. Training details

   - Layer 4개
   - Cell 1000개
   - 단어는 100 dimension으로 embedding
   - 160000개 input vocabulary
   - 80000개 output vocabulary
   - 모든 파라미터는 -0.08~0.08사이에서 uniform distribution 이루도록 초기화
   - SGD 사용하였으며 learning rate는 0.7
   - Epoch5 이후에는 0.5epoch마다 learning rate를 절반으로 줄임. 총 7.5epoch으로 학습

- 128 sequence 크기의 배치를 사용함.
- Vanishing gradient는 발생하지 않았지만 exploding gradient는 발생하여 gradient의 L2 norm 값이 5를 넘어가면 gradient를 L2 norm 값으로 나눠준다.
- 대부분의 문장은 20~30 정도의 길이로 짧았지만 몇 개는 길이가 100 이상이었다. 128의 mini batch에는 주로 짧은 문장이 대부분이였고 긴 문장 몇 개가 포함되었다. 따라서 mini batch로 학습할 시 길이가 너무 달라 학습이 잘 되지 않는 문제를 위해 mini batch 안에서 길이를 고정시켰고 학습속도는 2배가 되었다.

5. Parallelization

단일 GPU에서 이전 섹션의 구성을 사용하는 깊은 LSTM의 C++ 구현은 초당 약 1700 단어의 속도를 처리한다. 이것은 너무 느려서 8-GPU machine을 사용하여 우리의 모델을 병렬화하였다. LSTM의 각 계층은 다른 GPU에서 실행되고 즉시 다음 GPU/layer에 계산되자마자 활성화를 전달하였다. 이 논문의 모델은 LSTM들의 4개의 layer를 가지고 있고 각각 분리된 GPU에서 처리한다. 남은 4개의 GPU는 softmax 계산 병렬화에 사용되고 각각의 GPU는 1000*2000 matrix를 곱하는 것을 맡는다. 그 결과 128 mini batch로 초당 6300 단어의 속도를 이루었다. 훈련은 이것으로 대략 10일이 걸렸다.

6. Experimental Results

| Method | test BLEU score (ntst14) |
|---|---|
| Bahdanau et al. [2] | 28.45 |
| Baseline System [29] | 33.30 |
| Single forward LSTM, beam size 12 | 26.17 |
| Single reversed LSTM, beam size 12 | 30.59 |
| Ensemble of 5 reversed LSTMs, beam size 1 | 33.00 |
| Ensemble of 2 reversed LSTMs, beam size 12 | 33.27 |
| Ensemble of 5 reversed LSTMs, beam size 2 | 34.50 |
| Ensemble of 5 reversed LSTMs, beam size 12 | **34.81** |

| Method | test BLEU score (ntst14) |
|---|---|
| Baseline System [29] | 33.30 |
| Cho et al. [5] | 34.54 |
| State of the art [9] | **37.0** |
| Rescoring the baseline 1000-best with a single forward LSTM | 35.61 |
| Rescoring the baseline 1000-best with a single reversed LSTM | 35.85 |
| Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs | **36.5** |
| Oracle Rescoring of the Baseline 1000-best lists | ~45 |

가장 좋은 결과는 무작위 mini batch 순서와 무작위 초기화가 다른 LSTM들의 ensemble를 이용한 것이였다. LSTM ensemble의 디코딩 된 번역이 최고의 WNT'14 시스템을 능가하지는 못하지만 순수 신경 변역 시스템이 대규모 MT 작업에서 vocabulary words를 벗어남에도 불구하고 구문 기반 SMT 기준선보다 성능이 능가한 것은 처음이다. LSTM은 기준 시스템의 1000개의 최고 목록을 rescore하는데 사용되는 경우 WMT'14 결과의 0.5BLEU 점 이내이다.

7. Performance on long sentences

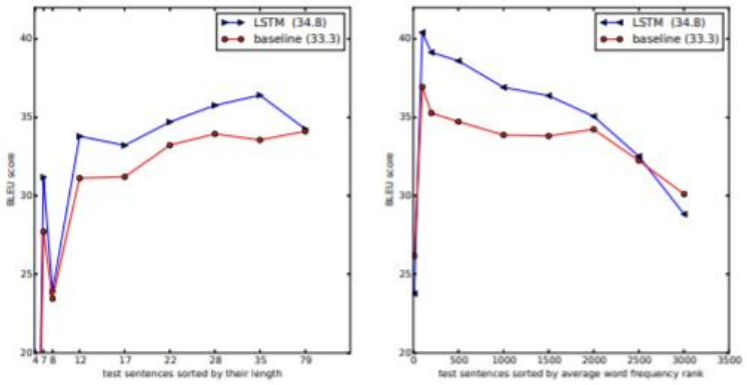
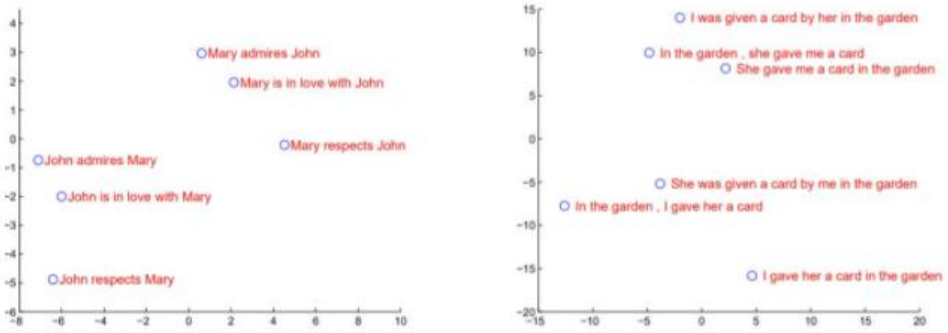



Figure 3: The left plot shows the performance of our system as a function of sentence length, where the x-axis corresponds to the test sentences sorted by their length and is marked by the actual sequence lengths. There is no degradation on sentences with less than 35 words, there is only a minor degradation on the longest sentences. The right plot shows the LSTM's performance on sentences with progressively more rare words, where the x-axis corresponds to the test sentences sorted by their "average word frequency rank".

LSTM이 긴 문장에서 잘 작동하였다.

8. Model Analysis




이 모델에서 중간의 input에서 output으로 가는 hidden state의 값인 고정된 벡터를 PCA를 통해 2차원 좌표상에 나타낸 결과는 위와 같다. 문장 순서에 따라 값이 매우 달라지고 능동/수동은 크게 상관이 없었다.

https://arxiv.org/pdf/1409.3215.pdf