Model Architecture

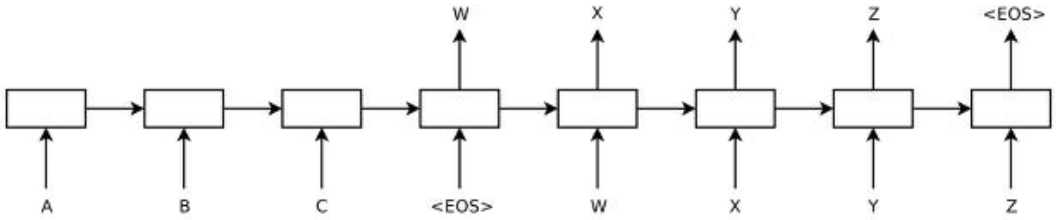


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

우선 구조를 설명하자면, Encoder 를 통해 Input Sentence 를 하나의 누적된 Cell State 로 계산하는 부분과 Encoder 에서 Input 으로 받은 Cell State 를 기반으로 Sequence 를 생성하는 Decoder 부분으로 나눠볼 수 있을 것이다.

일단 Encoder 부분에서는 위에서 언급했듯이 Input Sentence 를 누적된 Cell State 로 계산하는 역할을 하는데, 여기서는 Sequence 의 길이에 관계없이 <EOS>토큰이 나올때까지 Input 을 계속 넣어준다. (하지만 이론과는 다르게 실제로는 Training 의 편의성 내지 병렬처리를 위해서는 Fixed Length 를 가질 필요가 있을 것이며, 이는 Maximum Length 또는 원하는 길이를 설정해준 후 Padding/Clipping 을 해줘야 할 것이다) 이렇게 누적된 상태의 Cell State 는 이전 논문에서도 설명했듯이 하나의 Latent Vector 로 봐도 무방하다.

Decoder 는 Encoder 에서 마지막으로 나온 Cell State 만을 받아서 Sequence 를 생성하게 된다. 이렇게 생성하는 Sequence 는 Encoder 와 마찬가지로 <EOS>를 생성할때까지 작업을 반복하게 된다. (이전 논문에서도 언급했지만 Sequence 생성의 정확도를 높이기 위해서 Teacher Forcing 을 쓰게된다) 문장을 생성할 때에도 두가지 방법으로 접근할 수 있는데, 하나는 <EOS>토큰을 Input Sentence 의 끝부분으로 인식하는 방법, 다른 하나는 <BOS>토큰을 Prediction Sentence 의 시작으로 보는 방법이다. 하지만 두 방법 모두 위치는 사용하는 위치는 같으므로 용도는 동일할 것이다.

여기서 또 한가지 흥미로운 점은 Encoder 부분의 Input 의 순서만 역순으로 넣어줄 경우, 순서대로 넣은 결과보다 더 좋은 결과를 보인다는 점이다. 이러한

구조는 Bi-LSTM 에서 Backward 방향으로 진행되는 LSTM 과 유사하며, 조금만 더 생각해보면 Encoder 부분에 Bi-LSTM 을 사용할 경우 성능 개선을 할수 있지 않을까 하는 생각을 해볼 수도 있다. 하지만 이 논문이 나오는 당시에는 Bi-LSTM 이 쓰이기 이전이기 때문에 이러한 구조에 대한 제안은 하지 않고있다.

$$
\begin{aligned}
h_t &= \mathrm{sigm}\left(W^{\mathrm{hx}}x_t + W^{\mathrm{hh}}h_{t-1}\right) \\
y_t &= W^{\mathrm{yh}}h_t
\end{aligned}
$$

문장을 생성하는 연산은 위 연산을 따른다. h_t 는 Input Sentence 를 바탕으로 Latent Vector 로서 Cell State 를 계산하기 위함이고, y_t 는 Encoder 에서 나오는 Cell State 를 기반으로 Sequence generation 을 하기 위한 계산이다. Cell State 를 계산할 때는 사실 Sigmoid 와 Tanh 를 주로 사용하는데, 여기서는 Sigmoid 를 Cell State 의 activation 함수로 사용하고 있다.

또한 Gradient Vanishing 문제를 해결하기 위해서 Vanilla RNN 이 아닌 LSTM 을 사용하고 있다. 또한 여러층의 LSTM Layer 를 구성할 경우 성능이 높아진다고 언급하는데 4-Layer LSTM 이 가장 좋다고 언급하고 있다.

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

수식을 보면 예측하고자 하는 것이 어떤 것인지 더 명확해진다. T 길이의 Input Sentence 가 주어질때 T' 길이의 Output 을 생성하고자 하는 것인데 이를 구하기 위해서는 Encoder 에서 나온 Cell State 인 v 와 y_1, ..., y_t-1 까지의 Input 이 주어졌을때 yt 일 확률을 모두 곱하는 방식이다. (여기서 가장 확률이 높은 yt 가 Output 이 될 것이다)

$$1/|\mathcal{S}| \sum_{(T,S)\in\mathcal{S}} \log p(T|S)$$

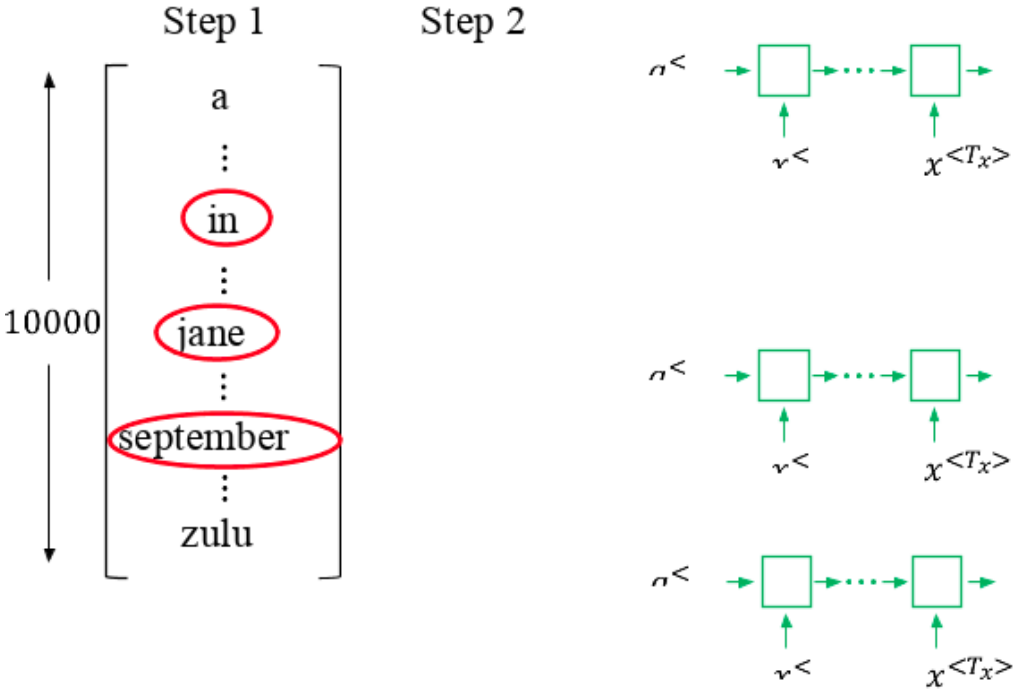
Training Objective 를 수식으로 나타내면 위와 같다. 위 수식을 살펴보자면, T 는 Sequence Prediction, S 는 Input Sequence, |S|는 Training Dataset 을 나타낸다. 즉, Input Sequence 가 주어질때의 Sequence Prediction 의 확률을 모두 더한 후 평균을 낸 값을 최대화 하고자 하는 것이 목적이다.

$$\hat{T} = \arg\max_{T} p(T|S)$$

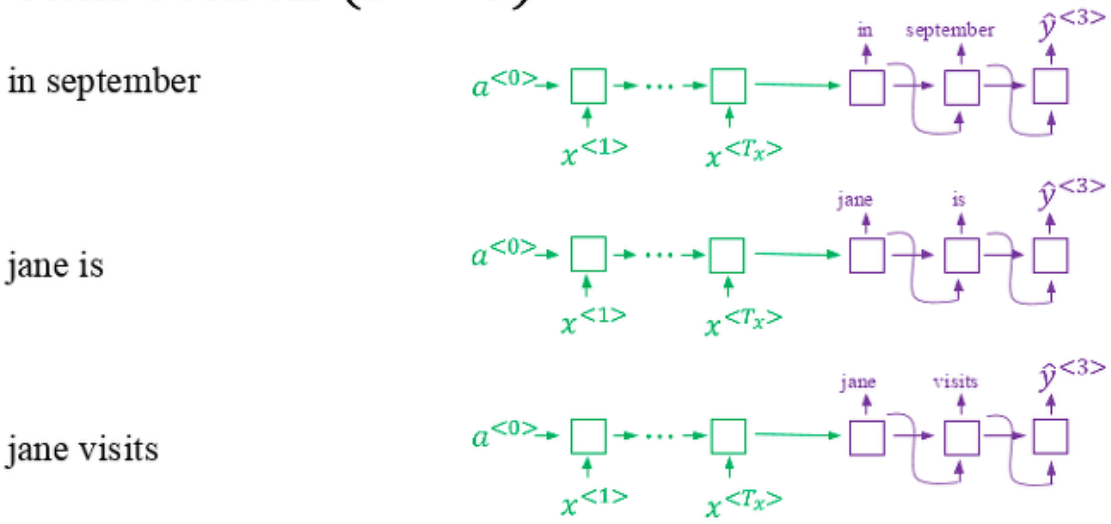
학습/훈련이 끝나면 위 식을 기반으로 Translation 결과를 생성한다. 여기서 번역 과정에는 Left-to-Right Beam Search Decoder 를 사용한다.

Beam search 알고리즘을 설명하자면, beam size 에서 가장 큰 확률을 계속 찾아나가는 식이다.

Beam search algorithm



Beam search ($B = 3$)



$P(y^{<1>}, y^{<2>} | x)$ jane visits africa in september. <EOS>

위 그림들을 살펴보자면, Bean Size 가 3 이라고 가정할때 Step 1 에서 가장 큰 확률을 n 개(Beam Size)를 고른다. 그 다음으로 Step 2 에서 나온 확률와 Step

1 에서 나온 가장 큰 확률 n 개와 각각 곱해서 n * (Vocab 갯수)중 가장 높은 확률 n 개를 취한다. 이와 같은 방식을 Sequence Prediction 이 끝날때까지 반복하는 방식이다.

학습에 사용된 Hardware 스펙도 주목할만하다. 논문에서 제안한 모델을 구현하기 위해 8 GPU 를 사용했다고 하며, Encoder/Decoder 의 LSTM Layer 마다 GPU 를 각각 할당했다. 이렇게 구성된 Hardware 로 10 일간 학습을 진행한다.

Result & Conclusion

Result

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**

Table 1: The performance of the LSTM on WMT'14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.

결과에서는 Sequence 를 Forward 방향과 Reverse 방향으로 넣은 결과를 보여주는데, Beam size 가 클수록, 그리고 Forward 대신 Reverse 순서로 넣을때 BLEU Score 가 더 높다고 언급한다. 한가지 신기한 점은 특히 문장을 역순으로 넣을때 Minimal Time Lag 가 줄어든다는 결과를 보여준다는 점이다. 그리고 여기서는 나와있지 않지만 Perplexity 의 개선도 이루어졌다는 언급을 하고 있다. (Perplexity: 5.8 → 4.7)

Method	test BLEU score (ntst14)
Baseline System [29]	33.30
Cho et al. [5]	34.54
Best WMT'14 result [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

Table 2: Methods that use neural networks together with an SMT system on the WMT'14 English to French test set (ntst14).

또한 SMT 와의 같이 사용할때의 성능에 대한 비교도 같이 하고 있는데, 가장 좋은 Result 를 보이고 있지는 않지만 SMT 와 같이 사용할 경우 전반적인 성능이 올라가는 것을 볼 수 있다.

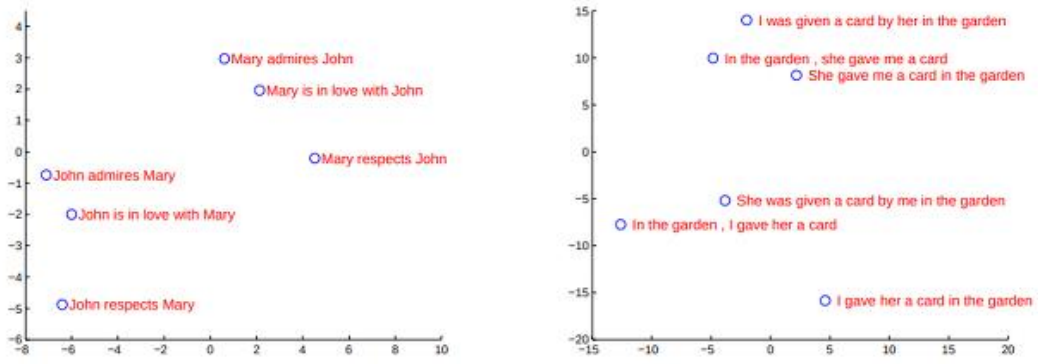


Figure 2: The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words model. Notice that both clusters have similar internal structure.

그리고 LSTM 에 대한 Hidden state 를 PCA 로 클러스터링 한 결과도 같이 보여주고 있다. 위 결과를 보면 유사한 문장 사이의 거리가 의미가 전혀 다른 문장과의 거리에 비해 더 가까운 것을 확인할 수 있다. 주목할 점은 단어의 주어/동사별 의미단위로 조밀하게 모여있고, 수/능동태 단위의 영향도는 거의 없음을 확인할 수 있다.

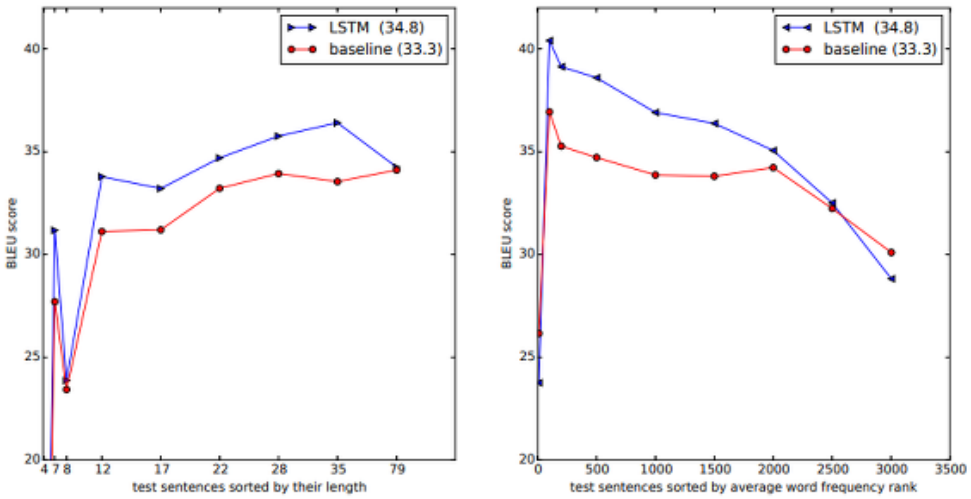


Figure 3: The left plot shows the performance of our system as a function of sentence length, where the x-axis corresponds to the test sentences sorted by their length and is marked by the actual sequence lengths. There is no degradation on sentences with less than 35 words, there is only a minor degradation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

문장의 길이에 따른 결과도 흥미롭다. 전반적으로 LSTM 이 Baseline 에 비해 높은 성능을 보여주지만 일정 길이(여기서는 35 단어 이상의 문장)를 넘기면 LSTM 기반 모델의 성능이 급격하게 감소하는 것을 확인할 수 있다.

Conclusion

해당 논문은 Seq2Seq구조의 Multi-Layer LSTM을 이용한 NMT를 제안한다. 이 모델의 Seq2Seq는 Encoder/Decoder 두 부분으로 구성되어있다. 여기서 아쉬운 점은 문장의 길이가 35개 단어를 넘기는 경우에는 정확도가 급격하게 하락하는 점인데, Abstract에서도 언급했듯이 이러한 점을 보완하기 위해 Attention 개념이 등장하게 된다. 하지만 결과를 분석하는데에서도 알수 있듯이 여전히 SMT에 대한 의존은 남아있는 것으로 보인다.

이렇게 문장의 길이가 길어질 경우 정확도가 급격하게 떨어지는 점 때문에 Character-level이 아닌 Word-level의 Input이 필요했던 것으로 보인다. Character-level로 진행할 시에는 문장의 길이가 조금만 길어도 35글자를 넘기는 경우가 빈번하게 나오기 때문이다.