8장 어텐션

윤 예 준

INDEX

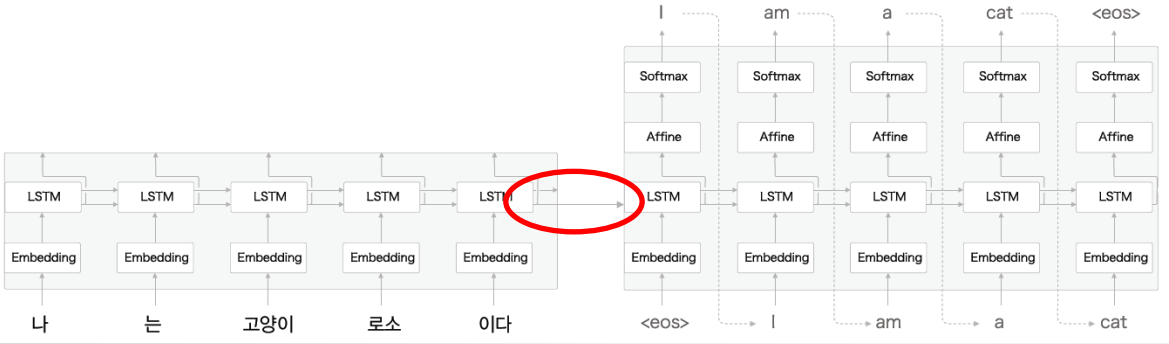
01

어텐션 구조

Seq2seq의 문제점

1. 하나의 고정된 벡터에 모든 정보를 압축하다보니 정보 손실이 발생한다.
2. RNN의 고질적인 문제인 기울기 손실 문제가 존재한다.

그림 7-9 seq2seq의 전체 계층 구성



Encoder 개선

그림 8-2 Encoder의 시각별(단어별) LSTM 계층의 은닉 상태를 모두 이용(**hs**로 표기)

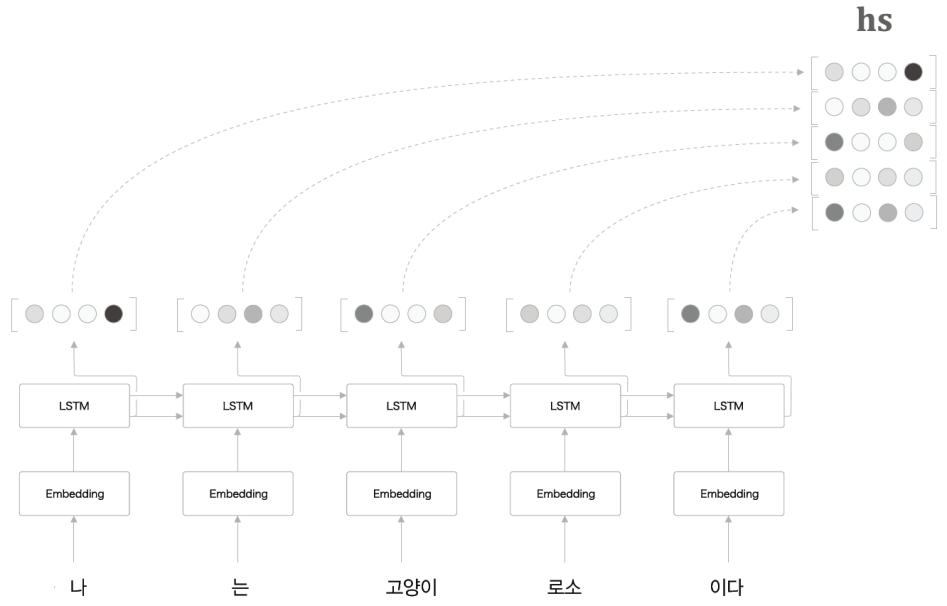
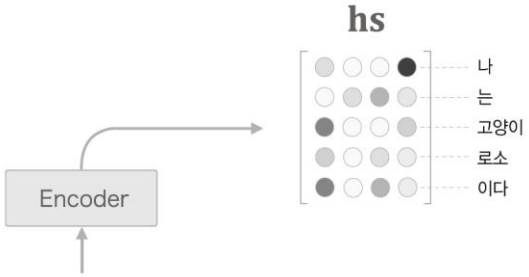


그림 8-3 Encoder의 출력 **hs**는 단어 수만큼의 벡터를 포함하며, 각각의 벡터는 해당 단어에 대한 정보를 많이 포함한다.



Decoder 개선 ①

그림 8-4 Encoder와 Decoder의 관계

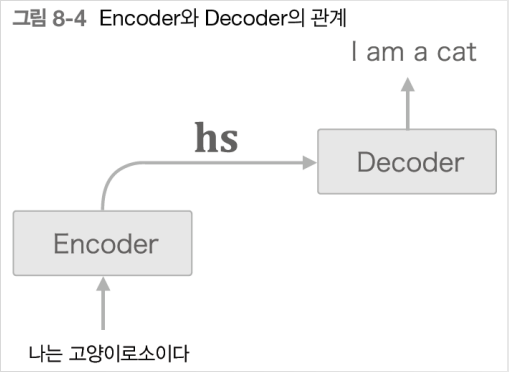
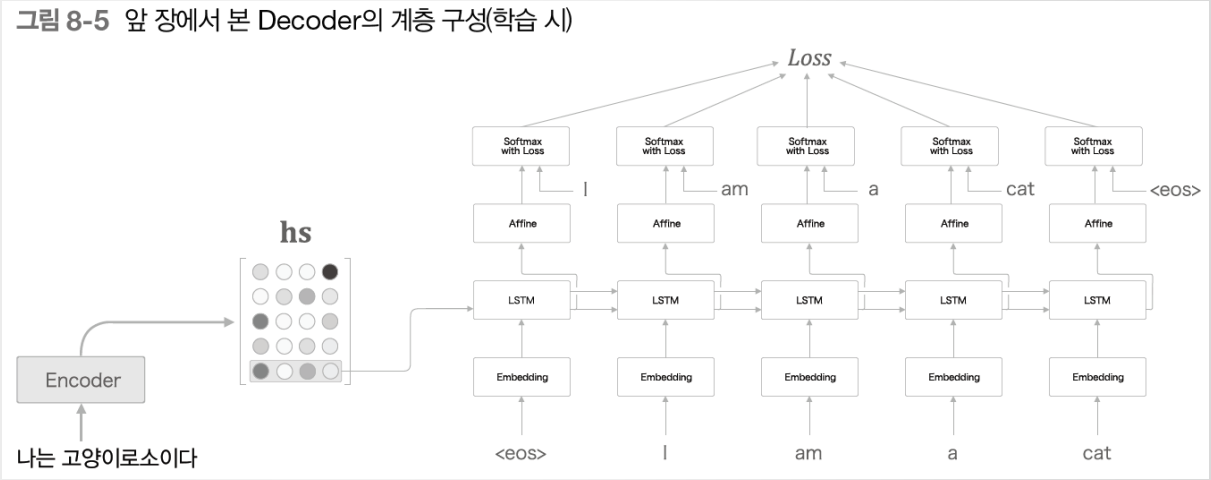


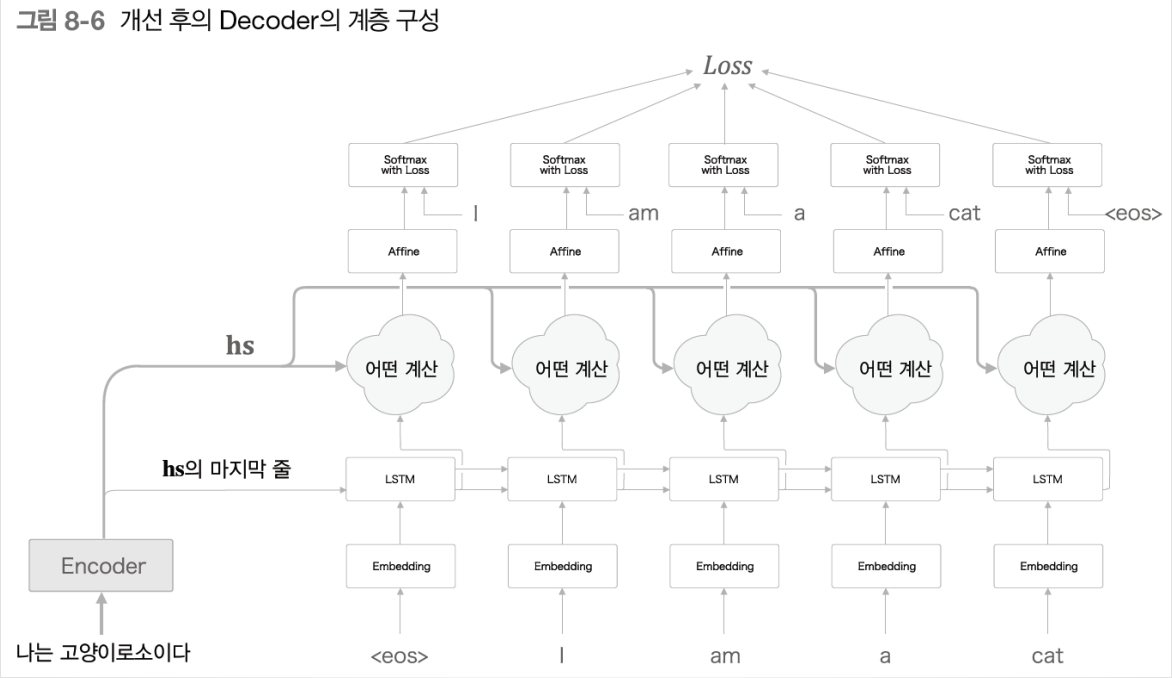
그림 8-5 앞 장에서 본 Decoder의 계층 구성(학습 시)



얼라인먼트 : 단어(혹은 문구)의 대응관계를 나타내는 정보

Decoder 개선 ①

그림 8-6 개선 후의 Decoder의 계층 구성



Decoder 개선 ①

그림 8-7 각 단어에 대해서 그것이 얼마나 중요한지를 나타내는 '가중치'를 구한다(구하는 방법은 뒤에서 설명).

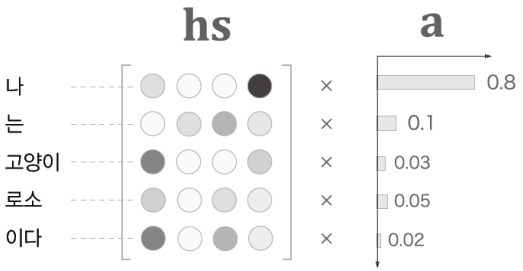
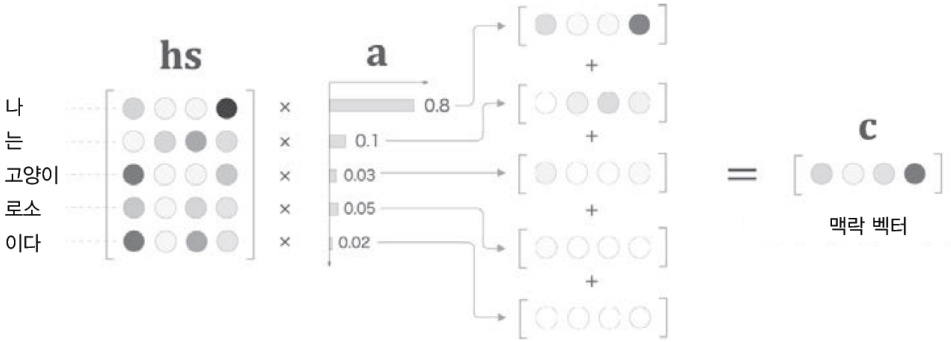
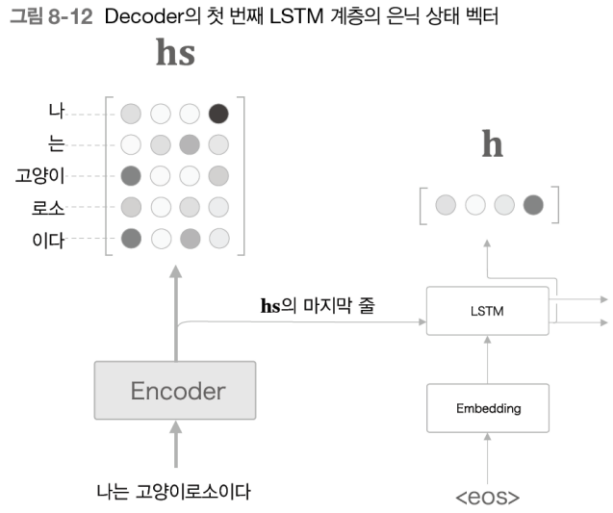


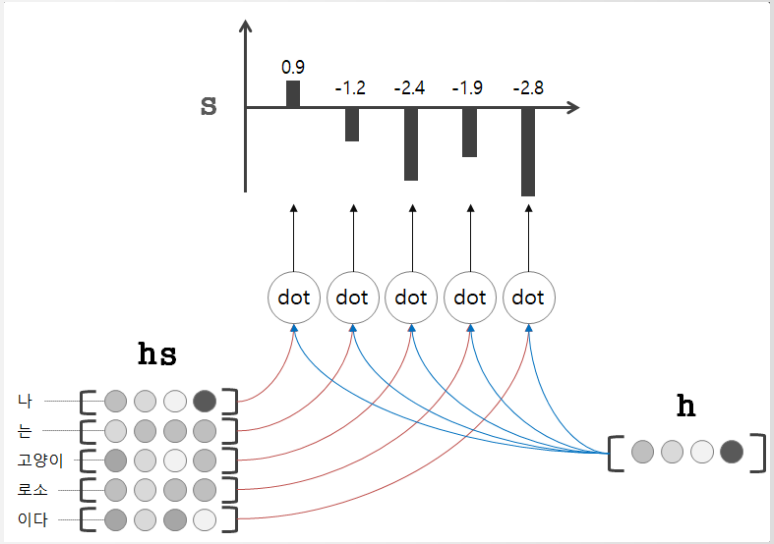
그림 8-8 가중합을 계산하여 '맥락 벡터'를 구한다.



Decoder 개선 ②

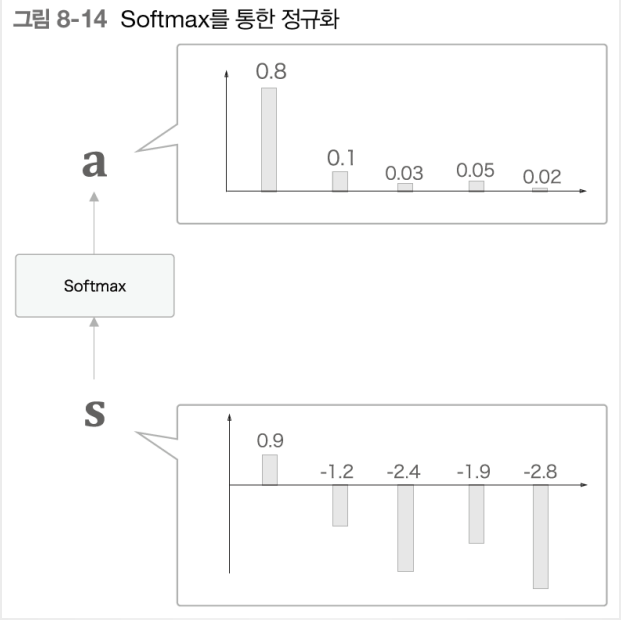
그림 8-12 Decoder의 첫 번째 LSTM 계층의 은닉 상태 벡터





Decoder 개선 ②

그림 8-14 Softmax를 통한 정규화



Decoder 개선 ③

그림 8-16 맥락 벡터를 계산하는 계산 그래프

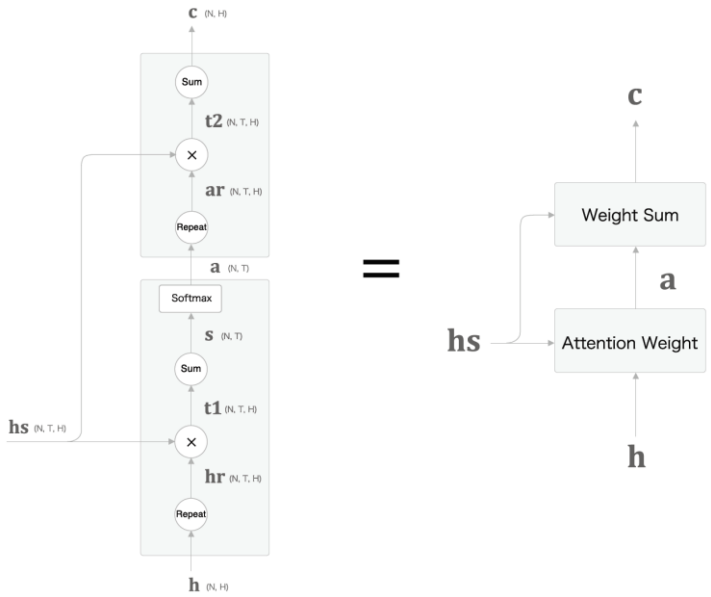
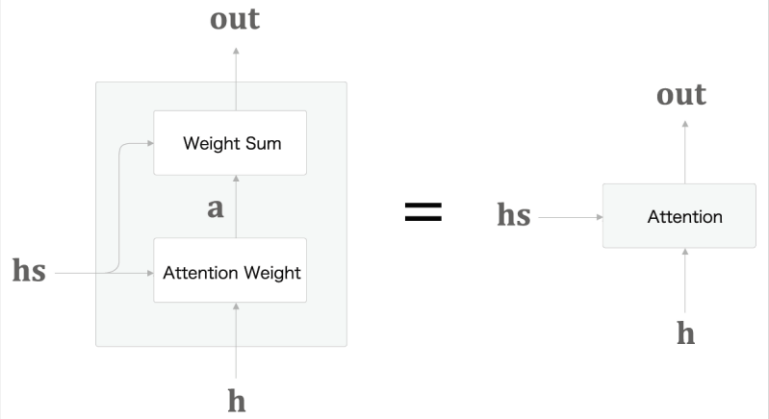


그림 8-17 왼쪽 계산 그래프를 Attention 계층으로 정리



02

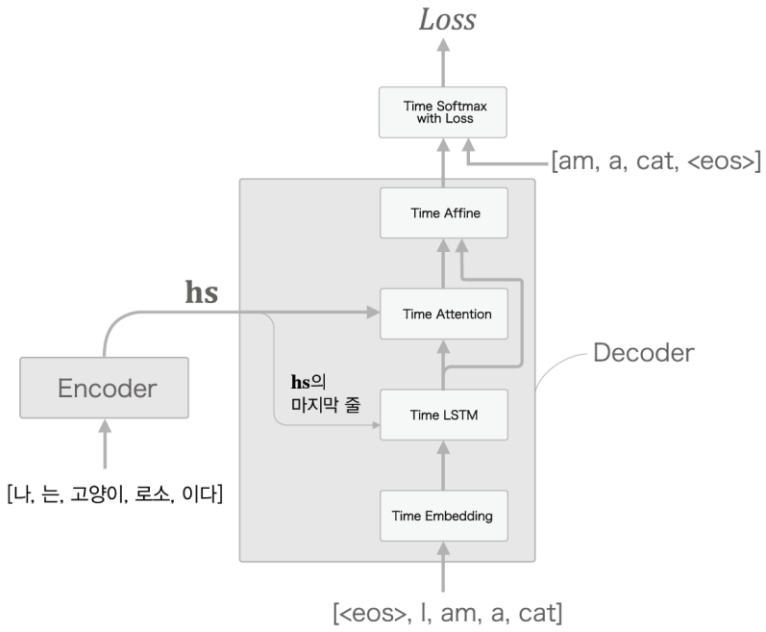
어텐션을 갖춘 seq2seq 구현

Encoder 구현

```python
class AttentionEncoder(Encoder):
    def forward(self, xs):
        xs = self.embed.forward(xs)
        hs = self.lstm.forward(xs)
        # return hs[:, -1, :]
        return hs

    def backward(self, dhs):
        dout = self.lstm.backward(dhs)
        dout = self.embed.backward(dout)
        return dout
```

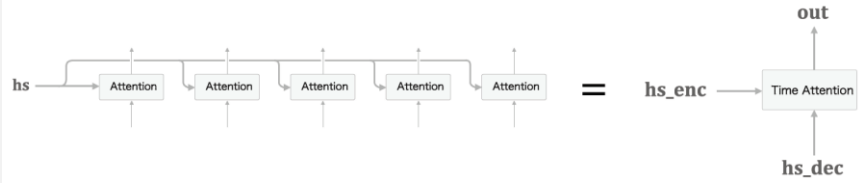
Decoder 구현

그림 8-21 Decoder의 계층 구성




```python
class AttentionDecoder:
    def __init__(self, vocab_size, wordvec_size, hidden_size):
        V, D, H = vocab_size, wordvec_size, hidden_size
        rn = np.random.randn

        embed_W = (rn(V, D) / 100).astype('f')
        lstm_Wx = (rn(D, 4 * H) / np.sqrt(D)).astype('f')
        lstm_Wh = (rn(H, 4 * H) / np.sqrt(H)).astype('f')
        lstm_b = np.zeros(4 * H).astype('f')
        affine_W = (rn(2*H, V) / np.sqrt(2*H)).astype('f')
        affine_b = np.zeros(V).astype('f')

        self.embed = TimeEmbedding(embed_W)
        self.lstm = TimeLSTM(lstm_Wx, lstm_Wh, lstm_b, stateful=True)
        self.attention = TimeAttention()
        self.affine = TimeAffine(affine_W, affine_b)
        layers = [self.embed, self.lstm, self.attention, self.affine]

        self.params, self.grads = [], []
        for layer in layers:
            self.params += layer.params
            self.grads += layer.grads

    def forward(self, xs, enc_hs):
        h = enc_hs[:,-1]
        self.lstm.set_state(h)

        out = self.embed.forward(xs)
        dec_hs = self.lstm.forward(out)
        c = self.attention.forward(enc_hs, dec_hs)
        out = np.concatenate((c, dec_hs), axis=2)
        score = self.affine.forward(out)

        return score
```

그림 8-20 다수의 Attention 계층을 Time Attention 계층으로서 모아 구현



seq2seq 구현

```
class AttentionSeq2seq(Seq2seq):
    def __init__(self, vocab_size, wordvec_size, hidden_size):
        args = vocab_size, wordvec_size, hidden_size
        self.encoder = AttentionEncoder(*args)
        self.decoder = AttentionDecoder(*args)
        self.softmax = TimeSoftmaxWithLoss()

        self.params = self.encoder.params + self.decoder.params
        self.grads = self.encoder.grads + self.decoder.grads
```

03

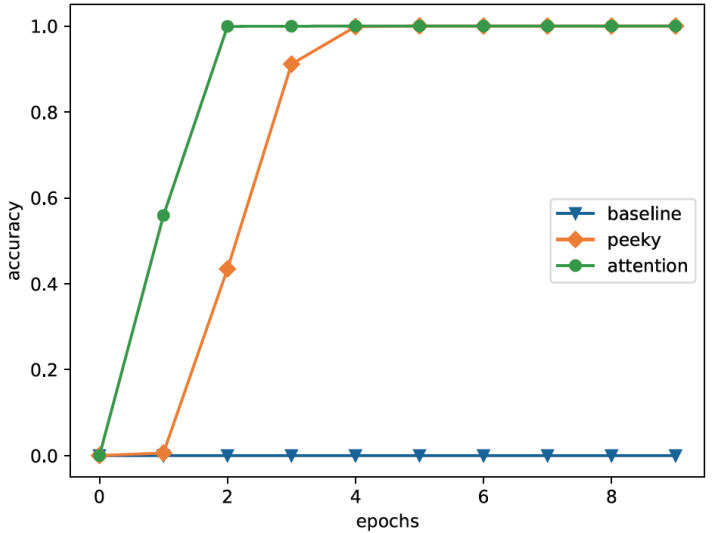
어텐션 평가

날짜 형식 변환 문제

그림 8-22 날짜 형식 변환의 예

september 27, 1994	→	1994-09-27
JUN 17, 2013	→	2013-06-17
2/10/93	→	1993-02-10

그림 8-26 다른 모델과의 비교: 'baseline'은 앞 장의 단순한 seq2seq, 'peeky'는 엿보기를 적용한 seq2seq(입력 문장 반전은 모든 모델에서 사용)

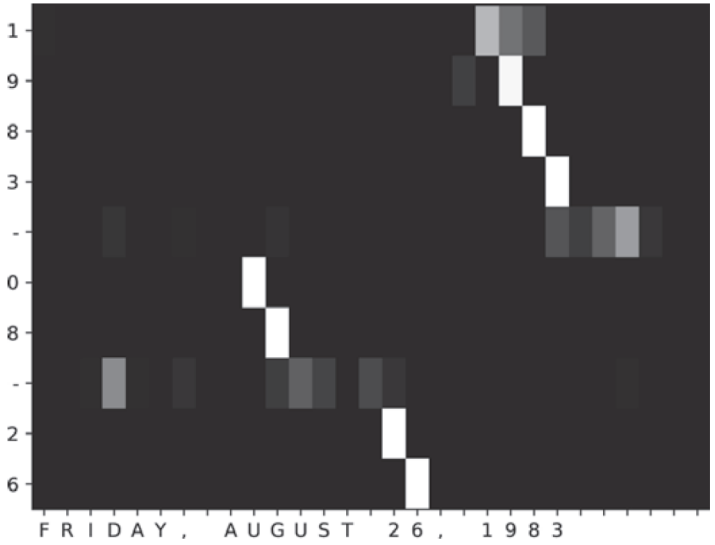


04

어텐션 시각화

어텐션 시각화

그림 8-27 학습된 모델을 사용하여 시계열 변환을 수행했을 때의 어텐션 가중치 시각화: 가로축은 입력 문장, 세로축은 출력 문장, 맵의 각 원소는 밝을수록 값이 크다(1.0에 가깝다).



05

어텐션에 관한 남은 이야기

양방향 RNN

그림 8-29 LSTM 계층에 의한 **hs** 출력

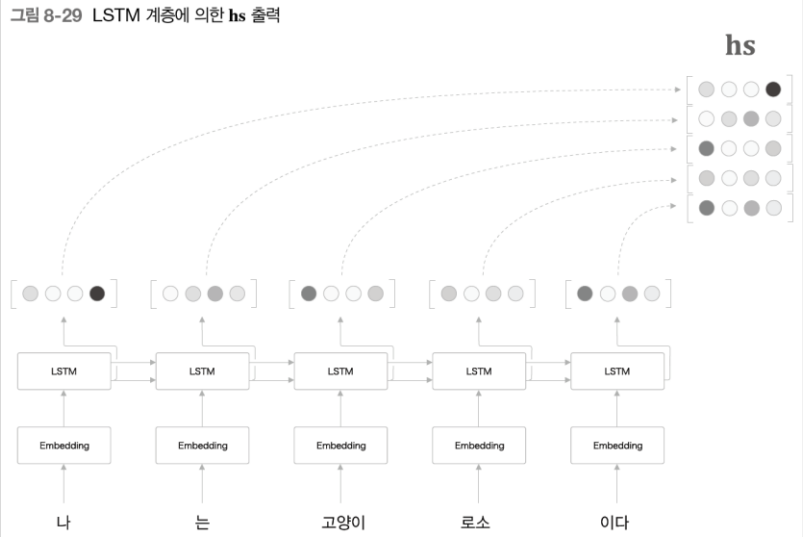
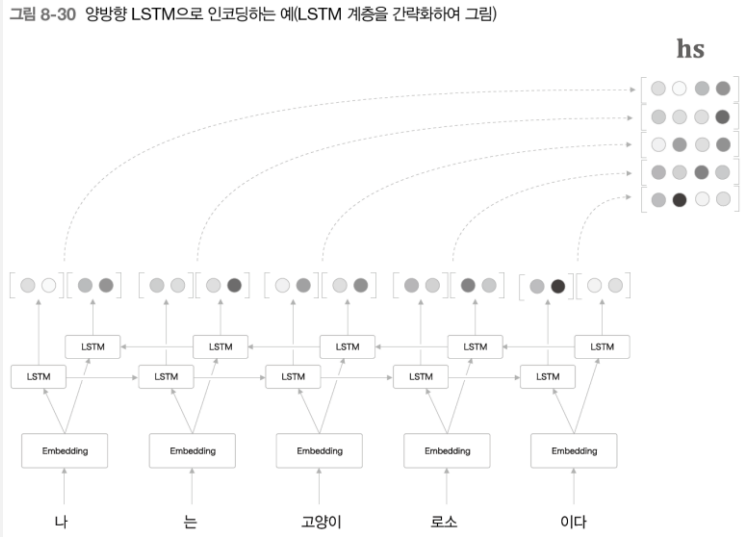


그림 8-30 양방향 LSTM으로 인코딩하는 예(LSTM 계층을 간략화하여 그림)



Attention 계층 사용 방법

그림 8-31 앞 절까지 사용해온 어텐션을 갖춘 seq2seq의 계층 구성

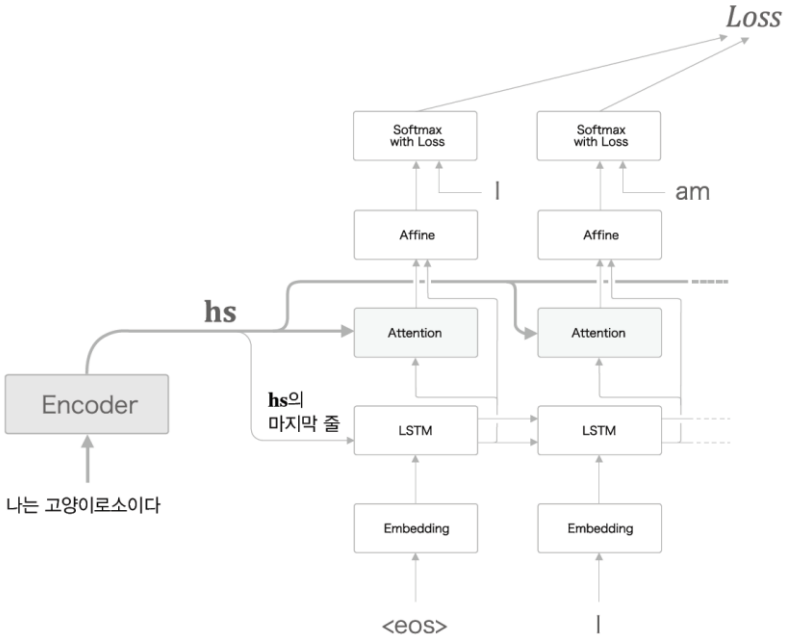
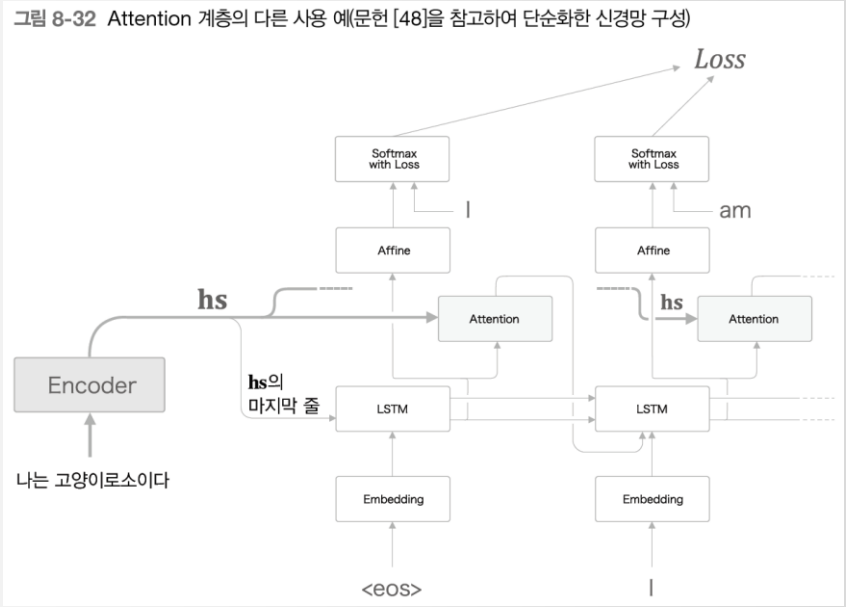


그림 8-32 Attention 계층의 다른 사용 예(문헌 [48]을 참고하여 단순화한 신경망 구성)



Seq2seq 심층화와 skip 연결

그림 8-33 3층 LSTM 계층을 사용한 어텐션을 갖춘 seq2seq

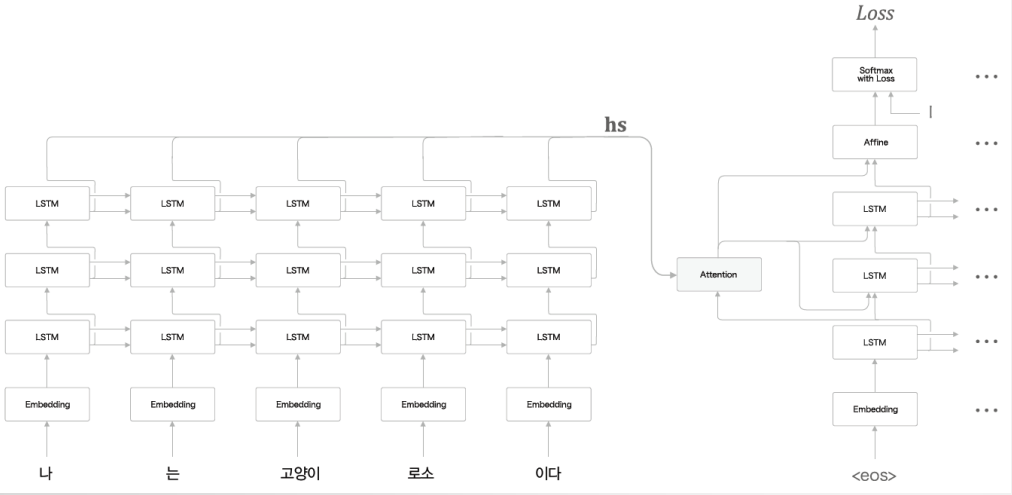
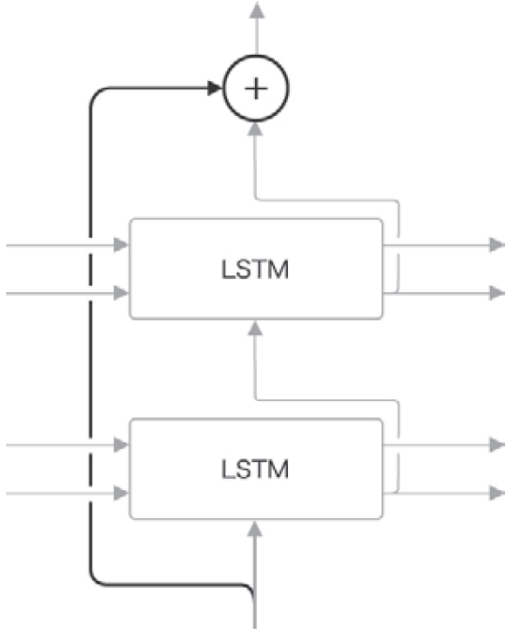


그림 8-34 LSTM 계층의 skip 연결 예



06

어텐션 응용

구글 신경망 기계 번역(GNMT)
트랜스포머
뉴럴 튜링 머신(NTM

그림 8-37 왼쪽이 일반적인 어텐션, 오른쪽이 셀프어텐션

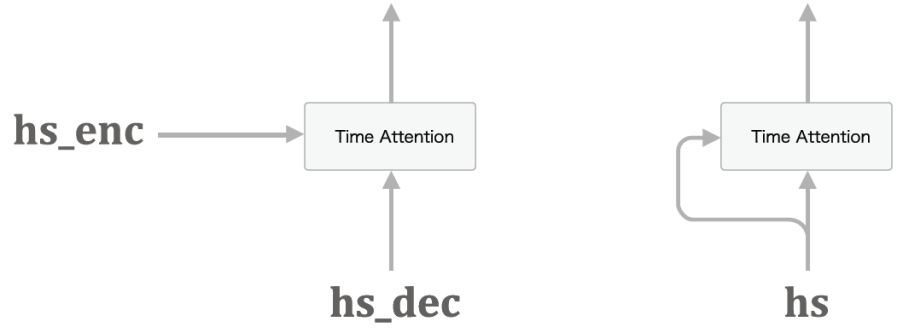
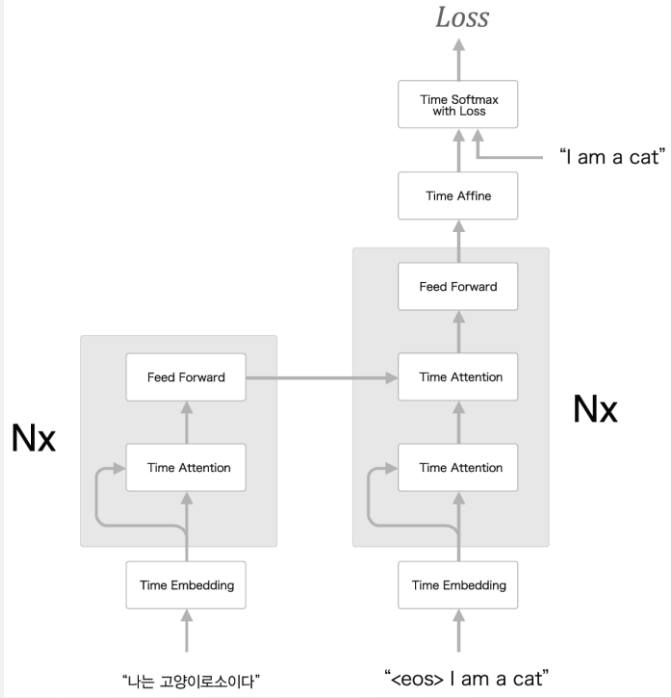


그림 8-38 트랜스포머의 계층 구성(문헌 [52]를 참고로 단순화한 모델)



구글 신경망 기계 번역(GNMT)
트랜스포머
뉴럴 튜링 머신(NTM

그림 8-37 왼쪽이 일반적인 어텐션, 오른쪽이 셀프어텐션

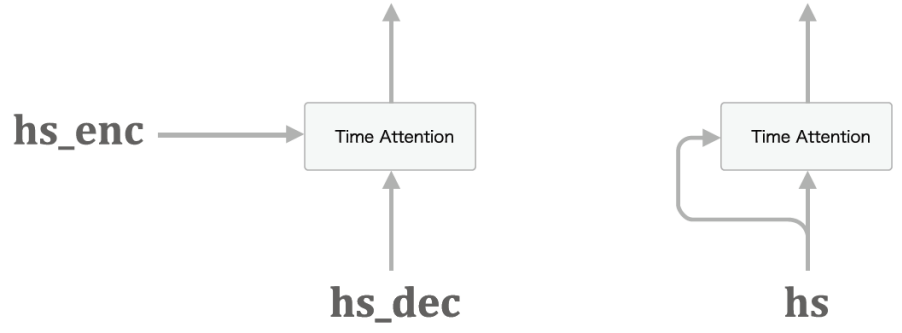
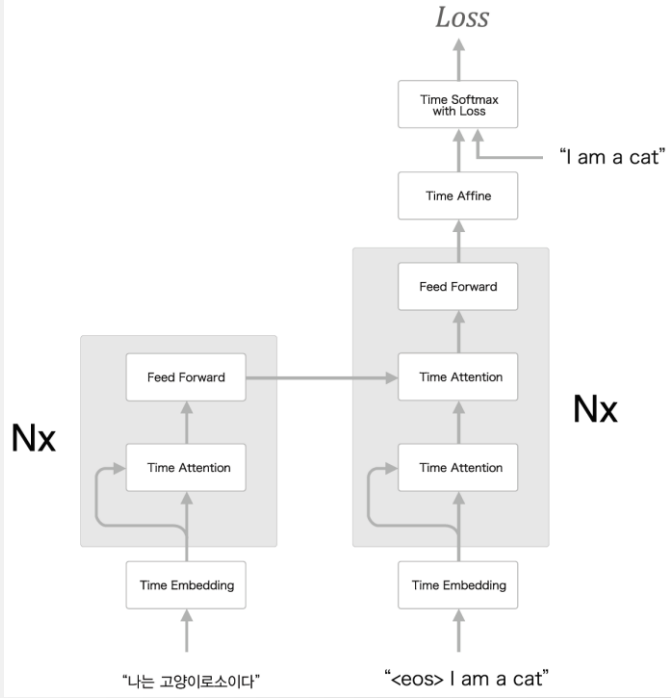


그림 8-38 트랜스포머의 계층 구성(문헌 [52]를 참고로 단순화한 모델)



구글 신경망 기계 번역(GNMT)
트랜스포머
뉴럴 튜링 머신(NTM

그림 8-37 왼쪽이 일반적인 어텐션

그림 8-38 트랜스포머의 계층 구성(문헌 [52]를 참고로 단순화한 모델)

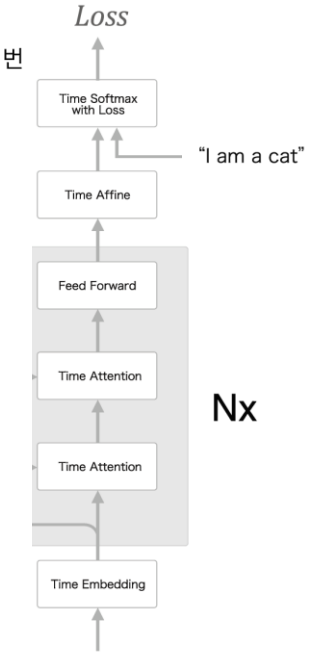
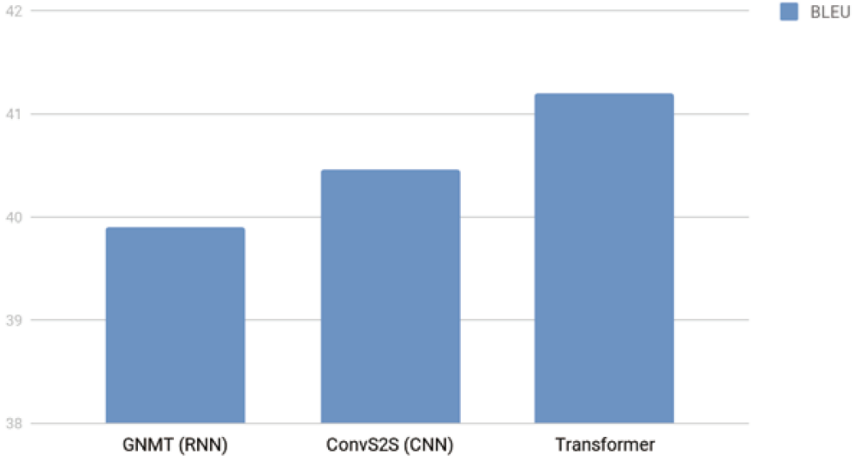


그림 8-39 벤치마크용 번역 데이터인 WMT를 사용하여 '영어 to 프랑스어' 번역 정확도를 평가한 결과: 세로축은 번역 정확도 척도인 BLEU 점수이며, 높을수록 좋다. (문헌 [53]에서 발췌)



THE END

감 사 합 니 다