Seq2seq

(SEQUENCE TO SEQUENCE)

Winter Vacation Capstone Study

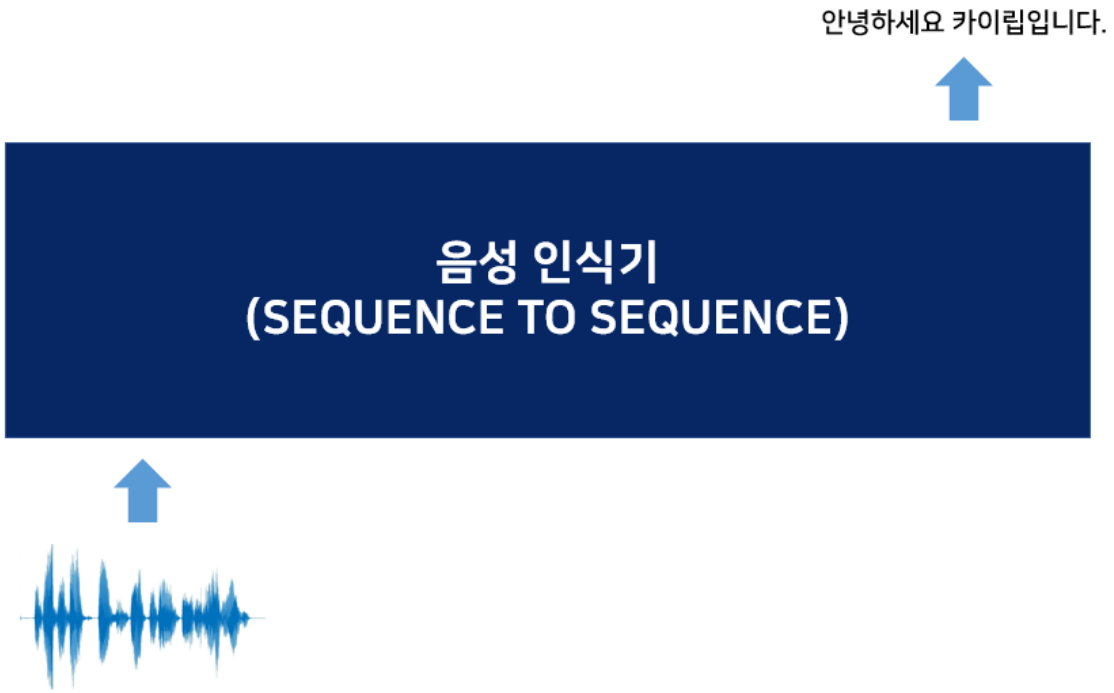
TEAM Kai.Lib

발표자 : 김수환

2020.01.13 (MON)

Seq2seq

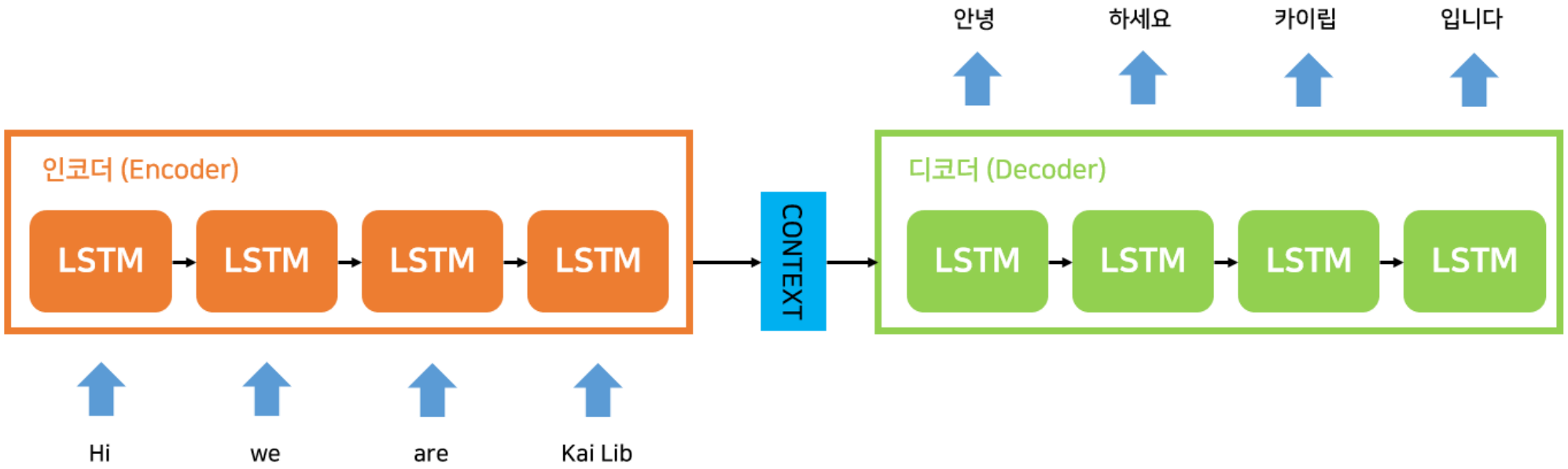
- **SEQUENCE TO SEQUENCE**



세상에는 시계열 데이터가 넘쳐난다. 언어 데이터, 음성 데이터, 동영상 데이터 등 많은 종류의 시계열 데이터가 존재한다. 그리고 이러한 **시계열 데이터를 또 다른 시계열 데이터로 변환**하는 문제도 숱하게 생각할 수 있다. 예컨대 기계 번역이나 음성 인식을 들 수 있다. 이를 위한 기법으로, 여기에서는 2개의 RNN을 이용하는 **seq2seq**sequence to sequence라는 방법을 살펴보자.

Seq2seq

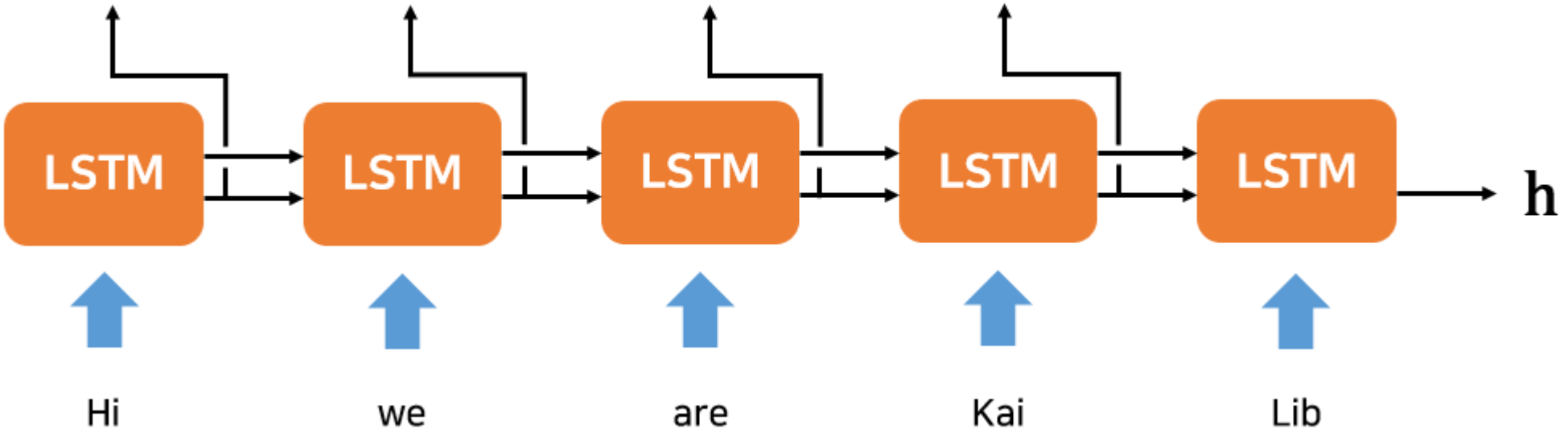
- **Seq2seq의 원리**



seq2seq를 Encoder-Decoder 모델이라고도 한다. 이름이 말해주듯이 여기에는 2개의 모듈, Encoder와 Decoder가 등장한다. 문자 그대로 Encoder는 입력 데이터를 인코딩하고, Decoder는 인코딩된 데이터를 디코딩한다. 위의 예를 보면, 먼저 어떠한 음성 신호를 적절하게 인코딩한다. 이어서 그 인코딩한 정보를 Decoder에 전달하고, Decoder가 도착어 문장을 생성한다. 이때 Encoder가 인코딩한 정보에는 번역에 필요한 정보가 조밀하게 응축되어 있는데, 이 벡터를 **컨텍스트 벡터** (context vector)라고 한다. Decoder는 조밀하게 응축된 이 정보를 바탕으로 도착어 문장을 생성하는 것이다.

Seq2seq

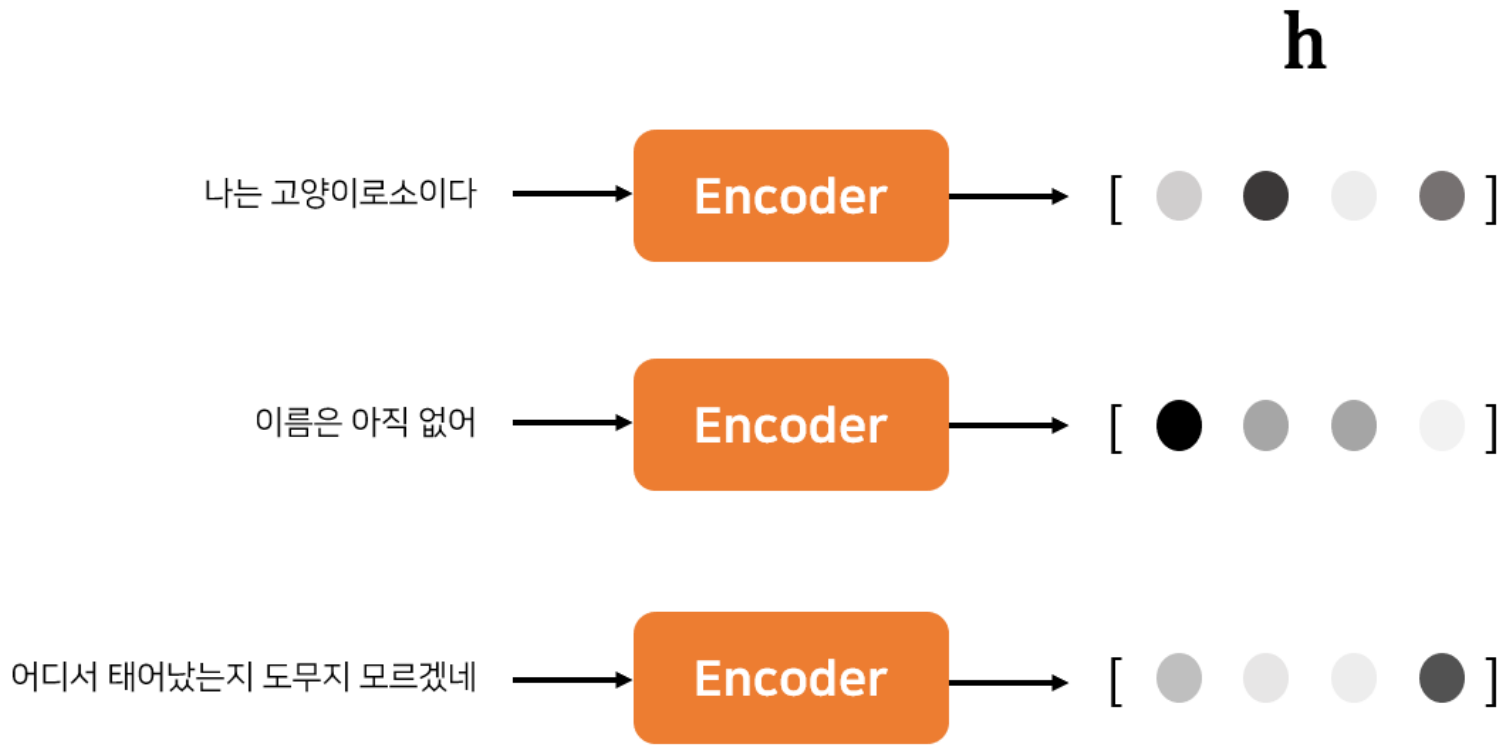
- **Encoder**



우선 Encoder의 처리에 집중해보자. Encoder의 계층은 위의 그림처럼 구성된다. 위의 그림처럼 Encoder는 RNN을 이용하여 데이터를 $\mathbf{h}$라는 Hidden State Vector로 변환한다. 지금 예에서는 RNN으로 LSTM을 이용했지만, '단순한 RNN'이나 GRU 등도 물론 이용할 수 있다. Encoder가 출력하는 벡터 $\mathbf{h}$는 LSTM 계층의 마지막 Hidden State이다. 이 마지막 은닉 상태 $\mathbf{h}$에 디코딩 하는데 필요한 정보가 인코딩 된다. 여기서 중요한 점은 Hidden State $\mathbf{h}$는 **고정 길이 벡터**라는 사실이다. 그래서 인코딩 한다라는 말은 결국 임의 길이의 데이터를 고정 길이 벡터로 변환하는 작업이 된다.

Seq2seq

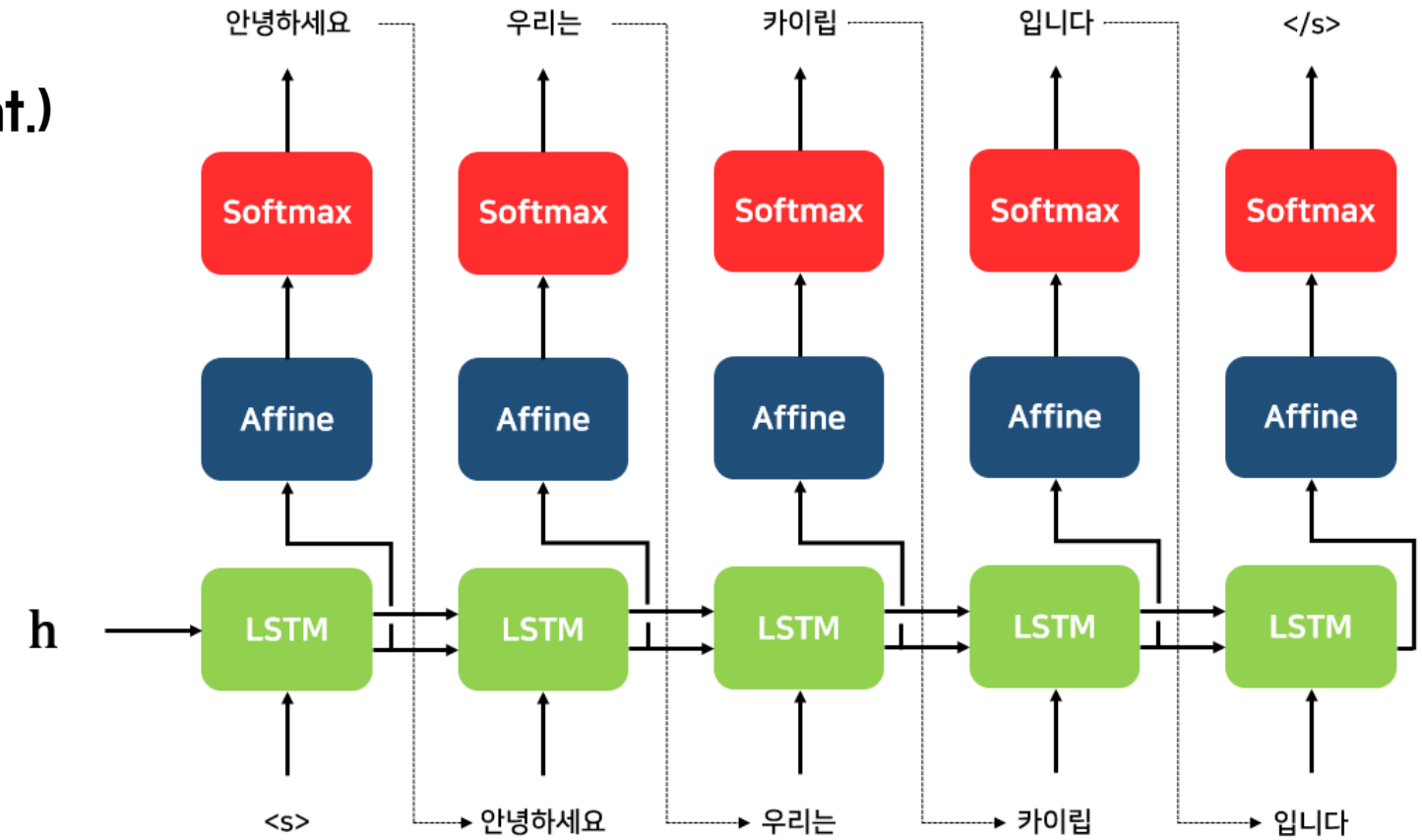
- **Encoder**



위의 그림에서 보듯 Encoder는 입력을 고정 길이 벡터로 변환한다. 어떤 길이의 문장이 오더라도 고정 길이로 변환하는 것이다.
즉, Encoder의 역할은 컨텍스트 벡터를 생성하는 것임을 알 수 있다.

Seq2seq

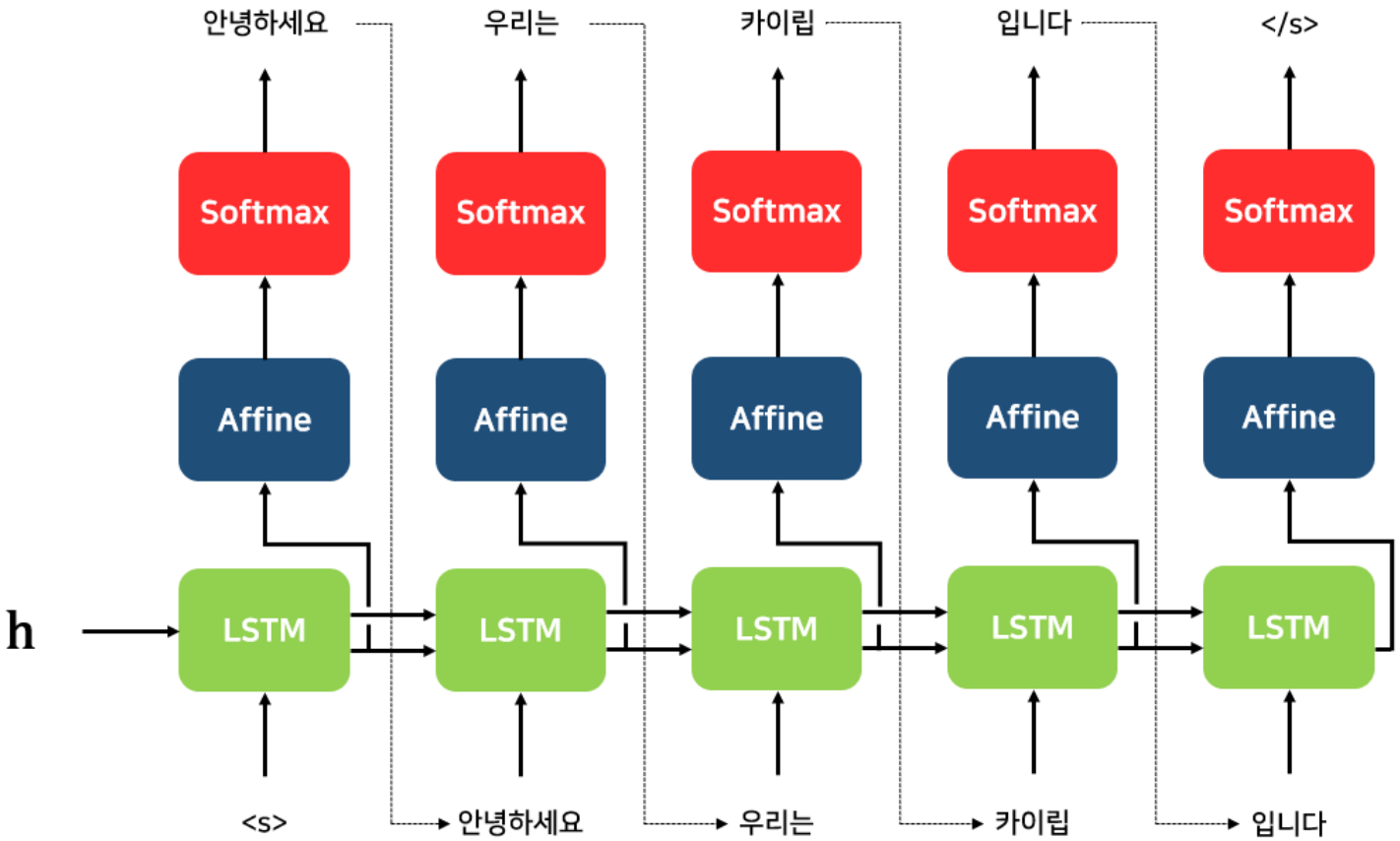
- **Decoder (Cont.)**



Decoder는 기본적으로 RNNLM(RNN Language Model)이다. Decoder는 초기 입력으로 문장의 시작을 의미하는 심볼인 <s>가 들어간다. Decoder는 <s>가 입력되면, 다음에 등장할 확률이 높은 단어를 예측한다. Decoder RNN 셀은 다음 등장 단어로 '나는'을 예측했다. 첫번째 Decoder RNN은 예측된 단어 '나는'을 다음 시점의 RNN 셀의 입력으로 입력한다. 그리고 다음 RNN 셀은 '나는'으로부터 다음에 올 단어를 예측하고, 그 예측한 단어를 다음 시점의 RNN 셀의 입력으로 넣는 행위를 반복한다. 그리고 이 과정은 문장의 끝을 의미하는 심볼인 </s>가 다음 단어로 예측될 때까지 반복된다.

Seq2seq

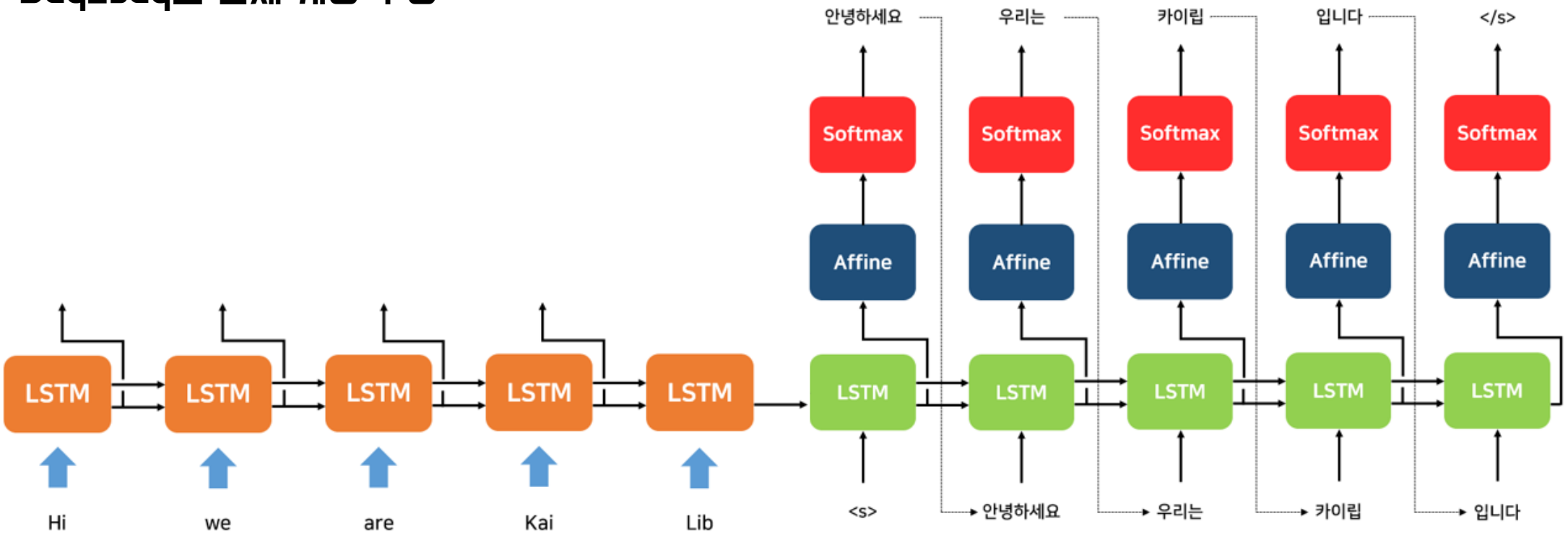
- **Decoder**



다시 한번 말하자면 Decoder는 기본적으로 RNNLM(RNN Language Model)이다. 하지만 기본적인 RNNLM과의 유일한 차이점은 바로 LSTM 계층이 벡터 **h**를 입력받는다는 점이 다르다. 즉, Encoder가 압축해서 표현한 **컨텍스트 벡터**를 입력으로 받는다는 사소한 차이점이 평범한 언어 모델도 번역과 같은 복잡한 문제도 풀 수 있는 Decoder로 탈바꿈시킬 수 있다.

Seq2seq

- **Seq2seq의 전체 계층 구성**



다음은 Encoder와 Decoder를 연결한 계층 구성이다. 위의 그림에서 보듯 seq2seq는 크게 LSTM 두 개로 구성된다. 이때 Encoder의 마지막 Hidden State가 Encoder와 Decoder를 이어주는 '가교'가 된다. 순전파 때는 Encoder에서 Decoder로 인코딩 된 정보를 보내고, 역전파 때는 이 '가교'를 통해 기울기가 Decoder로부터 Encoder로 전해진다.

Decoder

▪ Decoder의 Hidden State 초기화 - (1) 0으로 초기화 (행복코딩팀)

1. Decoder의 Hidden State 초기화

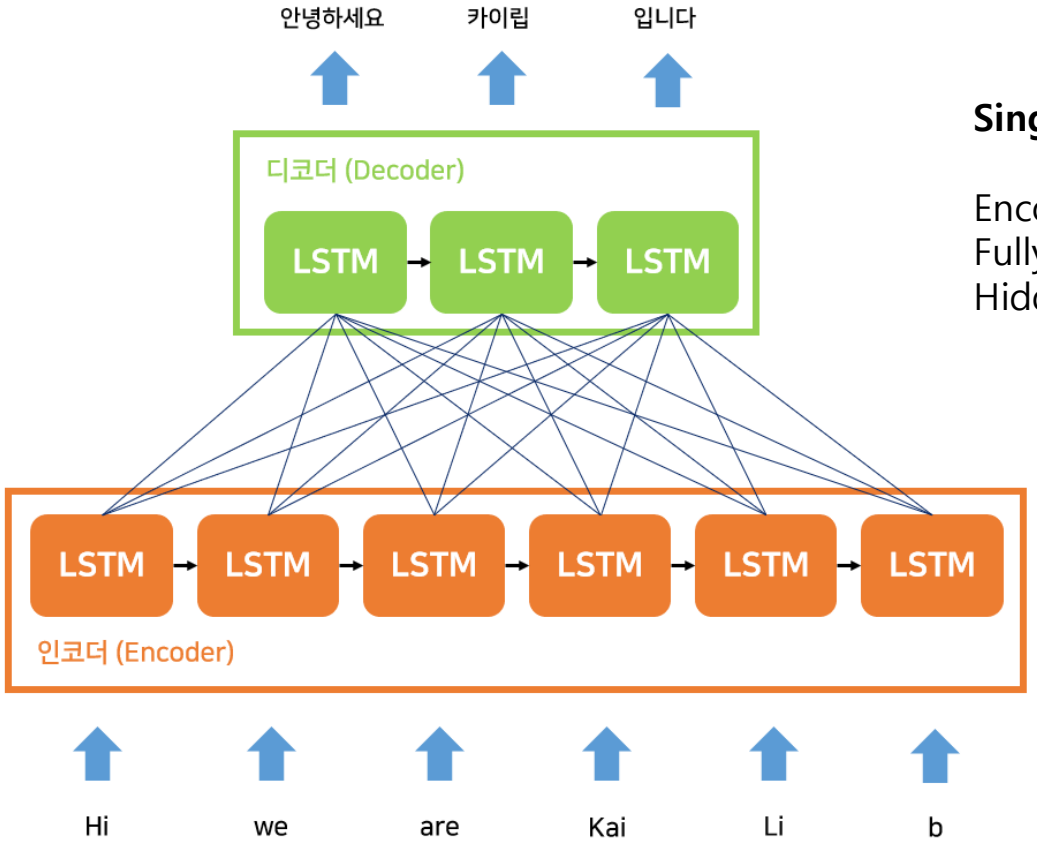
처음에는 Encoder의 Hidden State로 초기화 시키는 방법을 사용했지만, 그러면 Encoder와 Decoder의 레이어 사이즈가 같아야 한다는 제약이 있었다. 또한 Bidirectional-RNN에서 마지막 Hidden State가 Decoder에서 첫 단어를 뽑는데 유의미한 정보를 가지고 있을지 결론을 잘 내릴 수 없어서 결국 0으로 초기화 하는 방법을 시도했고, 성능이 비슷하거나 더 높게 나왔다.
그 이후 쭉 0으로 초기화하는 방법을 사용했다.

2. Context Vector 활용

Context Vector를 Attention에 집어넣어서 사용했다.
LAS 논문을 참고하여 Context Vector를 Attention mechanism에 넣어 그 위 Decoder 레이어와 다음 스템의 아래 디코더 레이어에 인풋으로 넣어줬다. (현재 우리 코드 상에서도 Context Vector가 Attention mechanism에 들어감)

Decoder

- **Decoder의 Hidden State 초기화 - (2) 인코더-디코더 사이에 FC 삽입**



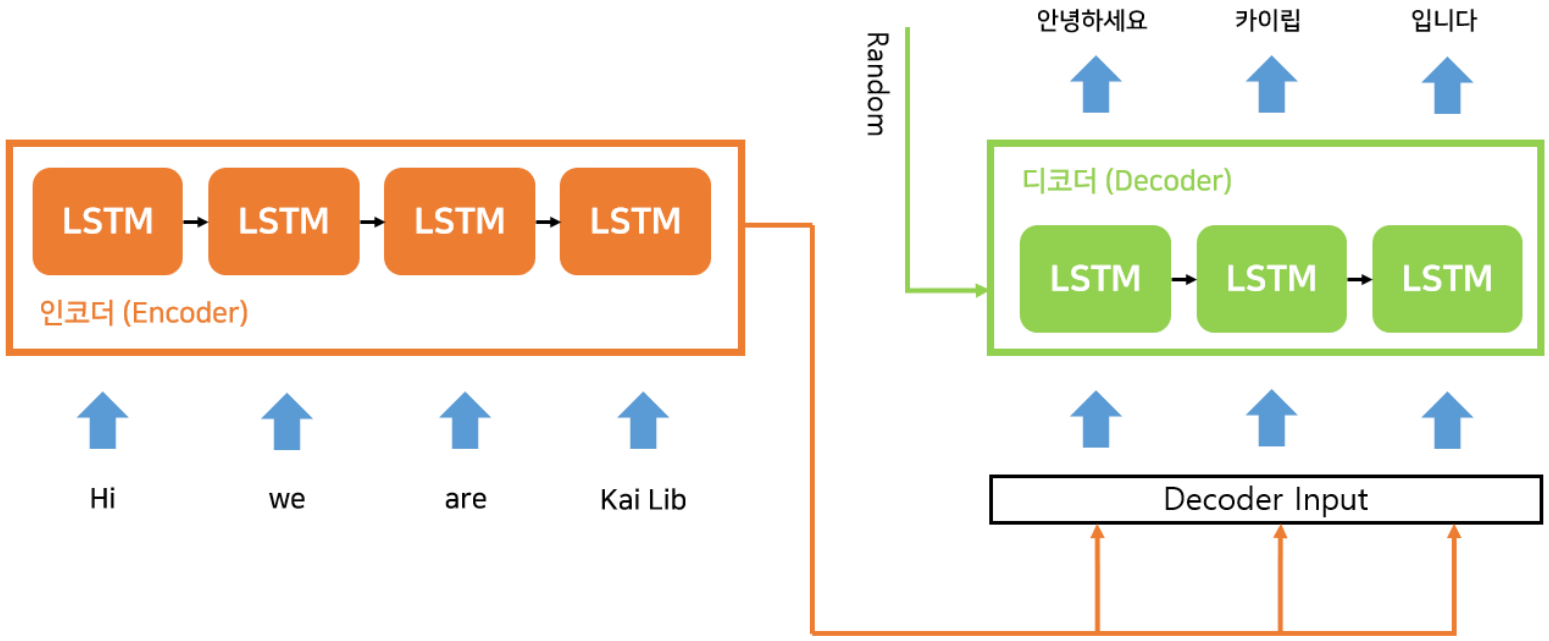
Single Fully Connected Network를 Encoder와 Decoder 사이에 배치

Encoder_layer_size의 인풋과 decoder_layer_size의 아웃풋을 가지는 Fully Connected Network를 배치함으로써 서로 다른 사이즈의 인코더의 Hidden State를 이용하여 디코더의 Hidden State를 초기화 할 수 있다.

참고문헌 : https://www.quora.com/In-seq2seq-models-how-does-one-initialize-the-states-of-a-decoder-when-it-has-a-different-number-of-layers-from-the-encoder

Decoder

- **Decoder의 Hidden State 초기화 - (3) 인코더의 아웃풋을 디코더의 인풋에 추가**



디코더의 Hidden State는 랜덤으로 초기화 한 후,

인코더의 Hidden State Output(Last Hidden State)을

디코더의 인풋에 **concatenate**한다.

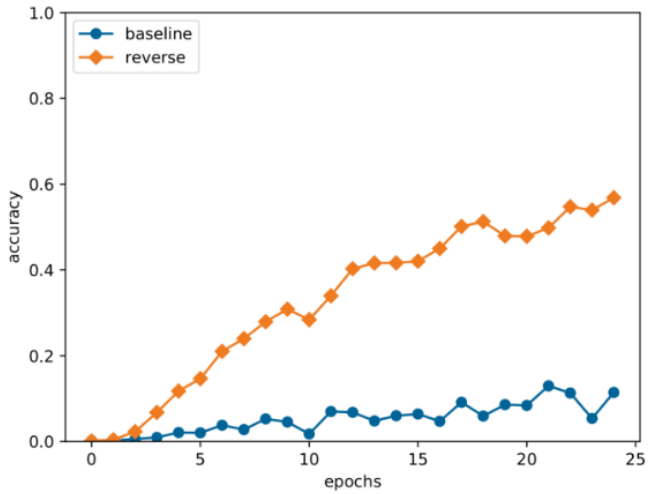
참고문헌 : https://www.quora.com/In-seq2seq-models-how-does-one-initialize-the-states-of-a-decoder-when-it-has-a-different-number-of-layers-from-the-encoder

Seq2seq 개선

▪ 입력 데이터 반전(Reverse)

이번에는 앞 절의 seq2seq를 세분화하여 조금 개선하고자 한다. 효과적인 기법이 몇 가지 있는데, 그중 두 가지를 살펴보자.
첫 번째 개선안은 아주 쉬운 트릭으로, 다음 그림에서 보듯이 입력 데이터의 순서를 반전시키는 것이다.

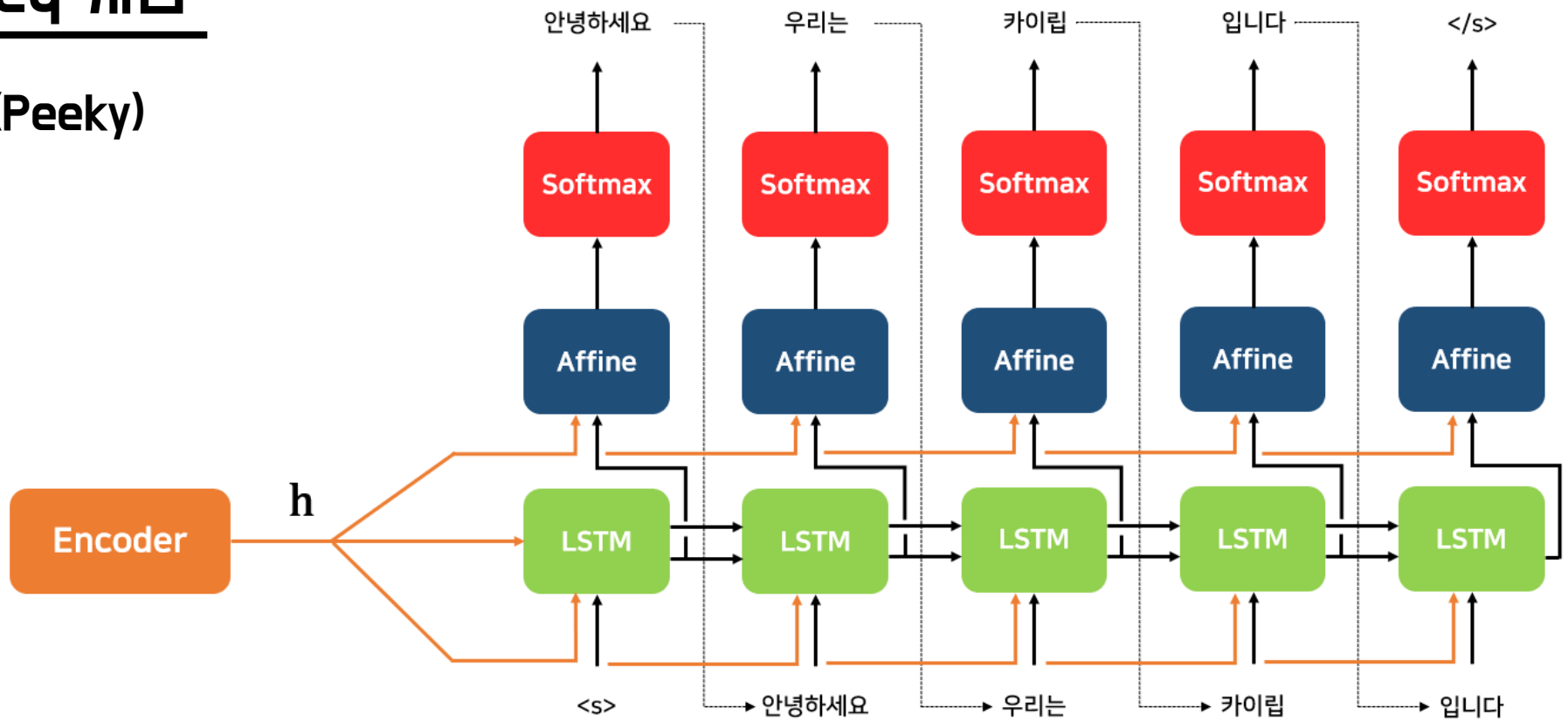
이러한 트릭은「"Sequence to sequence learning with neural networks." Advances in neural information processing system. 2014.」에서 제한했다. 이 트릭을 사용하면 많은 경우 학습 진행이 빨라져서, 결과적으로 최종 정확도도 좋아진다고 한다.



다음 그림에서 보듯, 입력 데이터를 반전시킨 것만으로 학습 진행이 개선되었다.
입력 데이터를 반전시켰을 뿐인데 이만큼 차이가 난다는 것은 놀라울 따름이다.
물론, 데이터를 반전시키는 효과는 어떤 문제를 다루느냐에 따라 다르지만, 대부분의 경우 더 좋은 결과로 이어진다고 한다. 그렇다면 왜 입력 데이터를 반전시키는 것만으로 학습이 빨라지고 정확도가 향상되는 걸까? 직관적으로는 기울기 전파가 원활해지기 때문이라고 생각된다. 예를 들어 "나는 고양이로소이다"를 "I am a cat"으로 번역하는 문제에서, "나"라는 단어가 "I"로 변환될 때 "나"로부터 "I"까지 가는 것보다 데이터를 반전시켰을 때 기울기 전파가 잘 될 것이다. 물론 평균적인 거리는 그대로이지만, 데이터에 따라 이러한 트릭이 더 좋은 효과를 낼 수 있을 것이다.

Seq2seq 개선

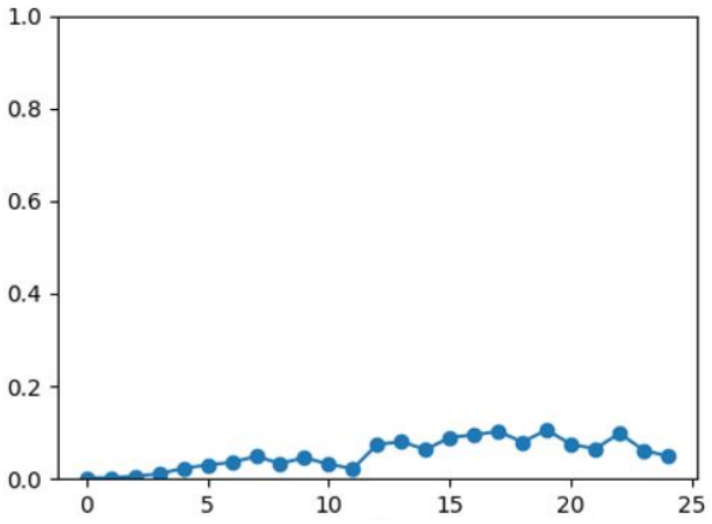
- 엿보기(Peeky)



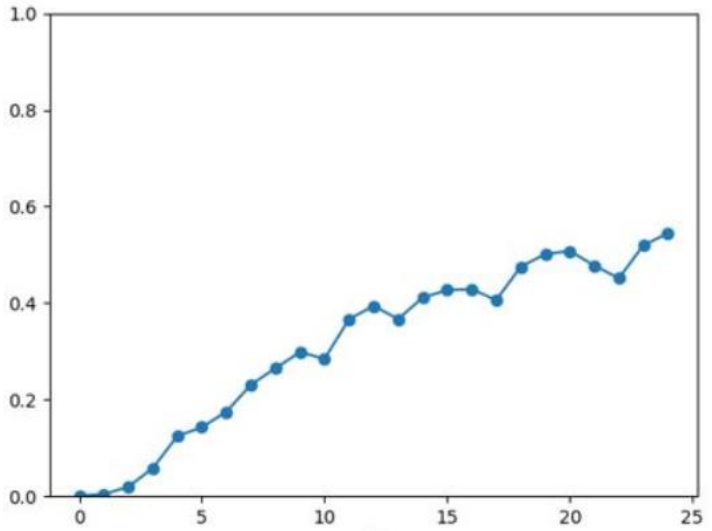
이어서 seq2seq의 두 번째 개선이다. 주제로 들어가기 전에 seq2seq의 동작을 다시 한번 살펴보게되면, Encoder는 입력 문장을 고정 길이의 **Context Vector**로 변환한다. 즉, 여기서는 이 Context Vector가 유일한 정보인 셈이다. 그러나 Baseline Seq2seq는 최초 시각의 LSTM 계층만이 이 Context Vector를 이용하고 있다. 이러한 점을 수정해서 중요한 정보가 담긴 Context Vector를 디코더의 다른 계층에도 전해주는 것이다. 이러한 아이디어는「"learning phrase representation using RNN encoder-decoder for statistical machine translation" Cho, Kyunhyun 2014.」논문에서 제안되었다.

Seq2seq 개선

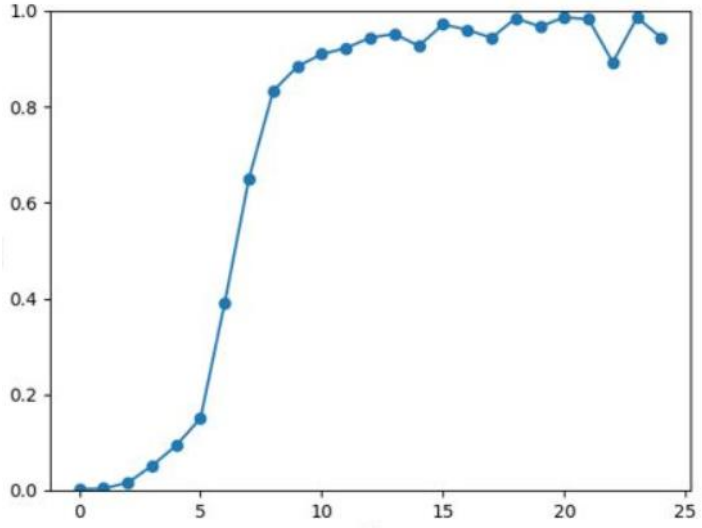
- **Reverse + Peeky**



:: baseline ::



:: reverse ::



:: reverse + peeky ::

다음은 Toy Problem으로 진행한 실험의 결과이다. reverse + peeky를 적용한 모델은 10에폭만에 정답률이 90%넘고, 최종적으로 100%에 가까워짐을 확인할 수 있다. 이상의 실험 결과에서 Reverse와 Peeky가 함께 효과적으로 작동하고 있음을 알 수 있다. 하지만 Peeky를 사용하게 되면 우리의 신경망은 가중치 매개변수가 커져서 계산량도 늘어나게 된다. 또한 seq2seq의 정확도는 하이퍼파라미터에 영향을 크게 받으므로, 실제 문제에서 어떤 효과를 낼지는 미지수이다.