Sequence-to-Sequence
Seq2Seq

Contents

1. Overview(모델의 개요)
2. seq2seq의 동작과정
3. seq2seq의 다양한 변형

ppt 설명 방식: 내부가 보이지 않는 커다란 블랙박스에서

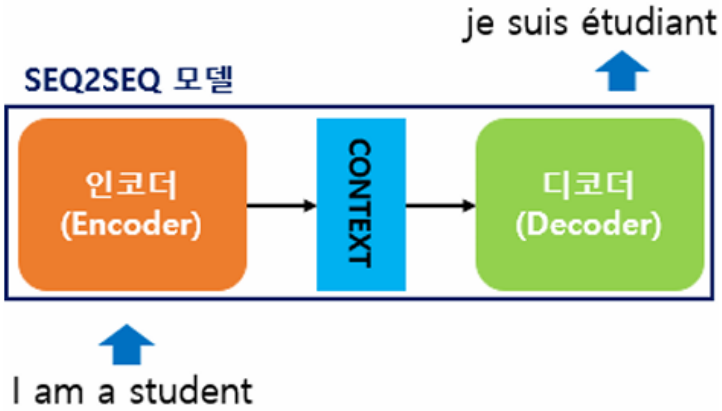
-> 점차적으로 확대해가는 방식으로 설명을 진행함

1. Overview(모델의 개요)

seq2seq 모델로 만든 번역기를 생각해보자.

- 입력: ‘I am a student’
- 출력: ‘je suis étudiant’

<seq2seq 모델의 외부 모습>



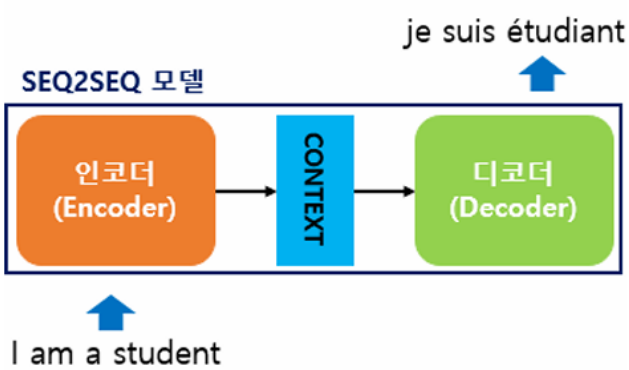
<seq2seq 모델의 내부 모습>

1. Overview(모델의 개요)

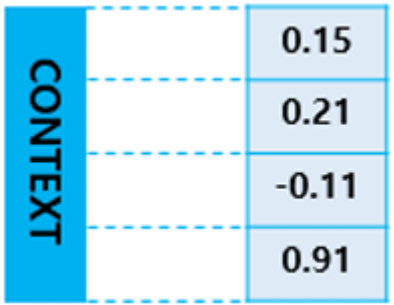
seq2seq는 크게 Encoder(인코더)와 Decoder(디코더)라는 2개의 아키텍처로 구성됨

Encoder: 입력 문장의 모든 단어들을 순차적으로 입력 받은 뒤, 이 모든 단어 정보들을 압축하여 1개의 context vector를 생성하고, context vector를 디코더로 전송

Decoder: context vector를 받아서 번역된 단어를 1개씩 순차적으로 출력

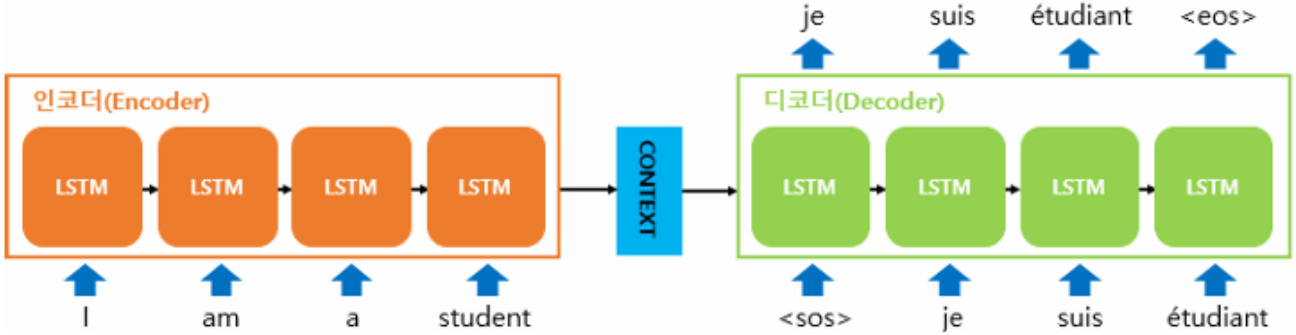


<Encoder와 Decoder>



<size=4인 context vector 예시>

2. Seq2Seq의 동작 과정



Encoder 아키텍처와 decoder 아키텍처의 내부는 사실 2개의 RNN 아키텍처임.
Encoder: 입력 문장을 받는 RNN 셀
Decoder: 출력 문장을 출력하는 RNN 셀
다만, 성능 문제로 인해 실제로는 vanila RNN이 아니라, LSTM 셀 또는 GRU 셀들로 구성됨.

process

1. 입력문장이 단어 토큰화를 통해서 단어 단위로 쪼개지고, 단어 토큰 각각은 Encoder RNN 셀의 각 시점의 입력이 됨
2. Encoder RNN 셀은 모든 단어를 입력 받은 뒤, 마지막 시점의 hidden state(Context Vector)를 Decoder RNN셀로 넘겨줌
3. Context vector가 Decoder RNN 셀의 첫번째 hidden state로 사용됨

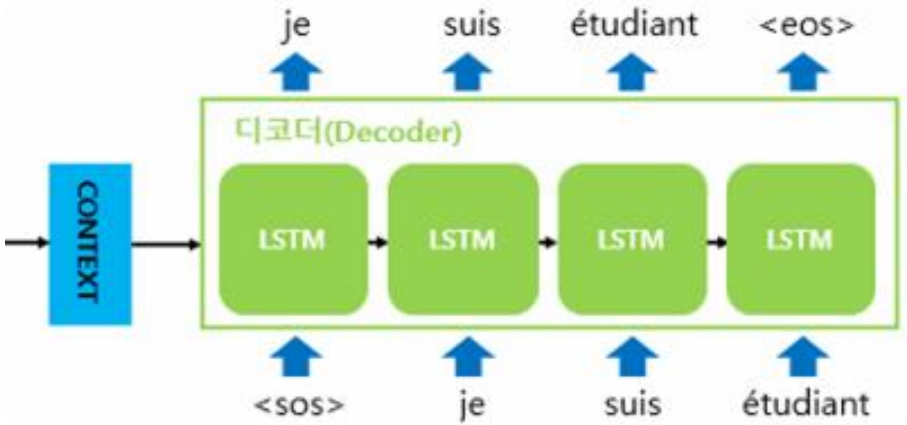
2-1. Seq2Seq의 동작 과정 - 테스트 단계

Example

1. Decoder의 초기 입력: 문장의 시작을 의미하는 심볼 '<sos>'가 들어감.
- Decoder에 <sos>가 입력되면, 첫번째 시점의 Decoder RNN 셀은 다음에 등장할 확률이 높은 단어('je')를 예측함.
- 첫번째 시점의 Decoder RNN 셀은 예측된 단어 je를 다음 시점 RNN 셀의 입력으로 보냄

2. 두번째 시점의 Decoder RNN셀은 입력된 단어 je로부터 다음에 올 단어인 suis를 예측하고, 이것을 다음 시점 RNN 셀의 입력으로 보냄

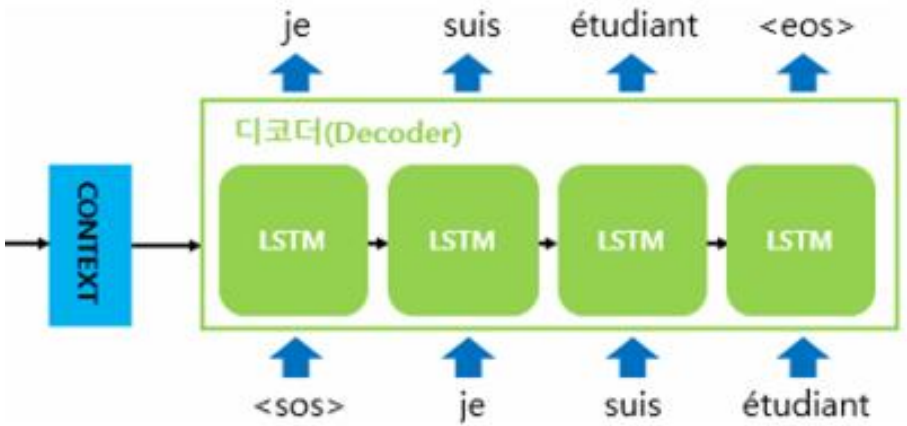
... (중략)



2-1. Seq2Seq의 동작 과정 - 테스트 단계

즉, **테스트 과정**동안, 문장의 끝을 의미하는 '<eos>'가 다음단어로 올 때까지 기본적으로

1. **다음에 올 단어를 예측**하고,
2. 그 예측한 단어를 **다음시점의 RNN 셀의 입력으로 넣는 행위**를 반복함.

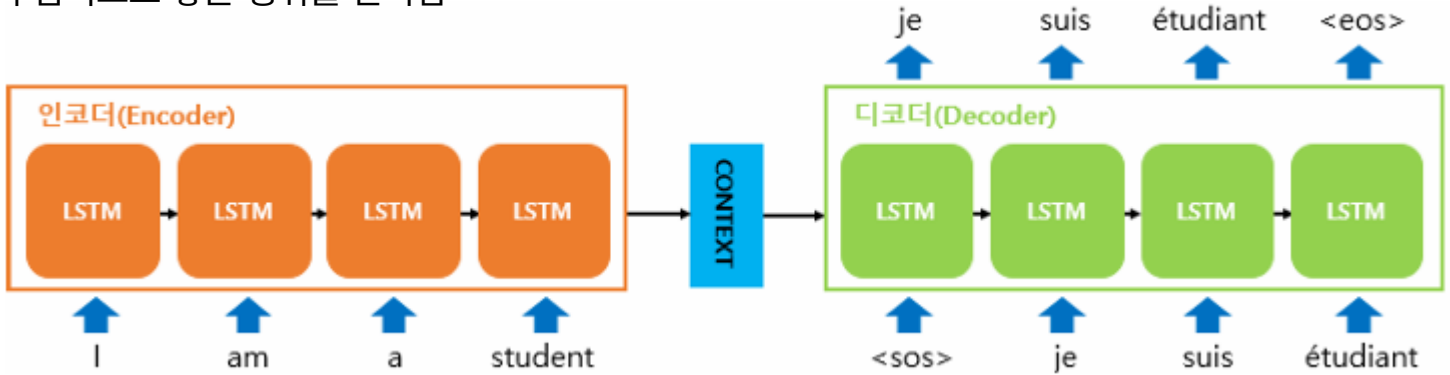


2-2. Seq2Seq의 동작 과정 - 훈련 단계 & teacher forcing

seq2seq의 훈련 과정은 테스트 과정의 작동방식과는 차이가 존재.

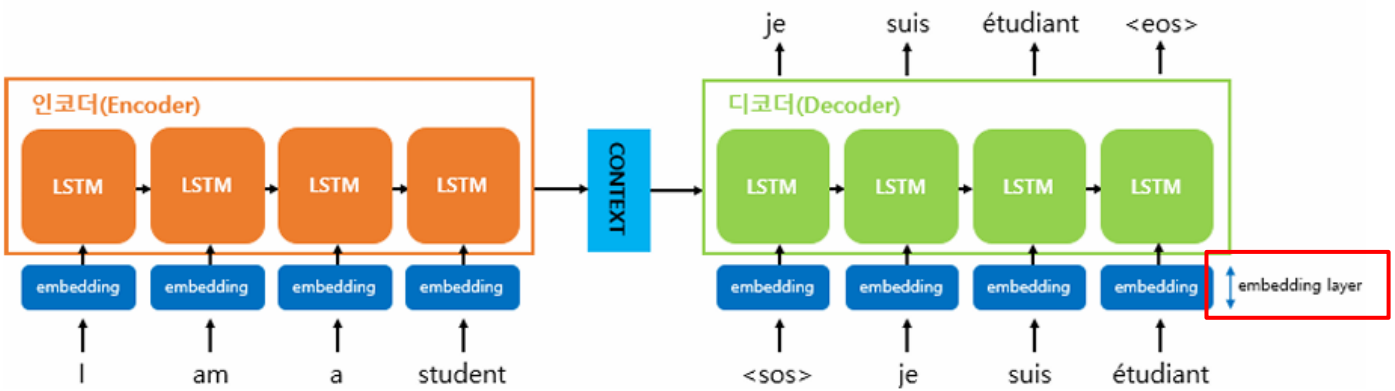
훈련 과정: Encoder가 Decoder에게 보낸 Context Vector와 실제 정답인 상황인 '<sos> je suis étudiant'를 입력받았을 때 -> 'je suis étudiant <eos>'가 나와야한다는 정답을 알려주면서 훈련함. (이를 **teacher forcing**(교사강요)이라고 함)

테스트 과정: 오직 Context Vector와 <sos>만을 입력으로 받은 후에 다음에 올 단어를 예측하고, 그 단어를 다음 시점의 RNN 셀의 입력으로 넣는 행위를 반복함



2-3. Seq2Seq의 동작 과정 - 임베딩 층(Embedding Layer)

기계는 텍스트보다 숫자를 더 잘 처리하기에, 텍스트를 Vector로 바꾸는 Word Embedding을 사용함.
즉, **seq2seq에서 사용되는 모든 단어들은** Word Embedding을 통해 표현된 Embedding Vector임



Example: Encoder의 입력 단어들에 대한 Embedding Vector

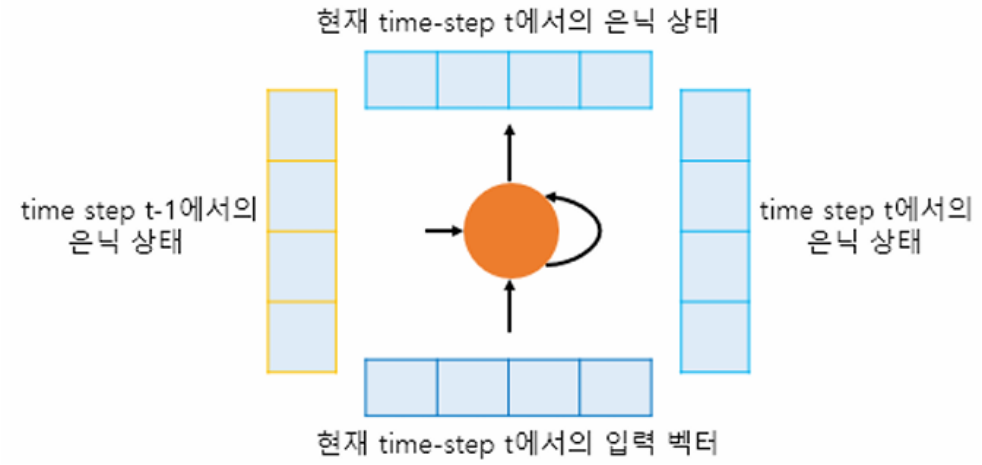
I		am		a		student	
0.157		0.78		0.75		0.88	
-0.25		0.29		-0.81		-0.17	
0.478		-0.96		0.96		0.29	
-0.78		0.52		0.12		0.48	

2-4. Seq2Seq의 동작 과정 - RNN 셀

하나의 RNN 셀은 각 시점(time step)마다 2개의 입력을 받는다.
t를 현재 시점이라고 하자.

입력: t-1에서의 hidden state 및 t에서의 입력벡터
출력: 현재 시점 t에서의 hidden state

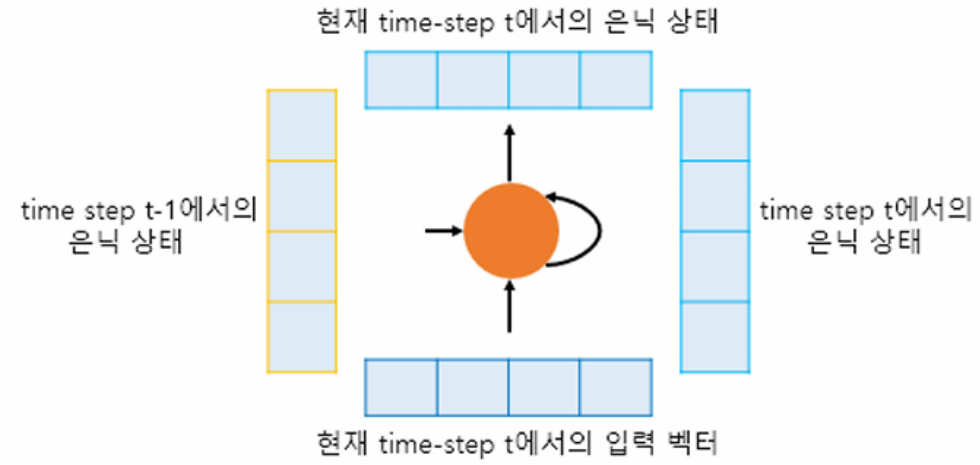
이후, RNN셀은 **다음 시점에 해당하는 t+1에서의 RNN 셀**에 현재 시점 t에서의 hidden state를 입력으로 보낸다.



2-4. Seq2Seq의 동작 과정 - RNN 셀

즉, 이 구조에서
현재 시점 t에서의 hidden state = 과거 시점의 동일한 RNN 셀에서의 모든 hidden state값들의 영향을 누적해서 받아온 값

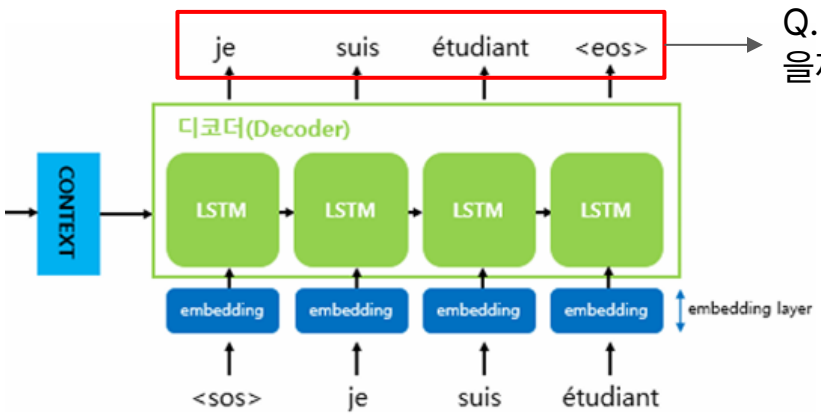
따라서, Context Vector는 **Encoder에서의 마지막 RNN 셀의 hidden state값**을 뜻하며,
입력 문장의 모든 단어 토큰들의 정보를 요약해서 담고있다고 할 수 있음



2-5. Seq2Seq의 동작 과정 - Decoder

Decoder는 Encoder의 마지막 RNN 셀의 hidden state(=Context Vector)를 첫번째 hidden state값으로 사용함

- 첫번째 RNN 셀: Context Vector와 현재 t에서의 입력벡터인 <sos>로부터
  => 다음에 등장할 단어(je) 예측
  => 이 예측된 단어가 다음시점 t+1에서의 RNN 셀의 입력값이 됨.
- 일반화(t+1에서의 RNN셀): t에서의 hidden state와 t+1에서의 입력벡터(임베딩벡터)로부터
  => t+1에서의 hidden state(다음에 등장할 단어)를 예측함



Q. 디코더가 다음에 등장할 단어를 어떻게 예측할 수 있을까?

2-5. Seq2Seq의 동작 과정 - Decoder

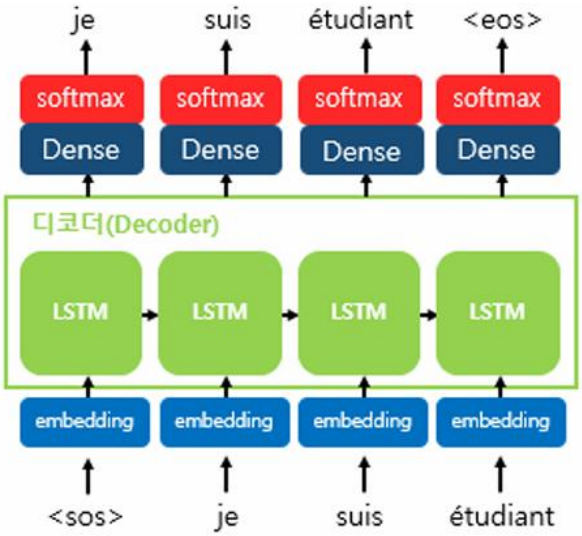
Goal: seq2seq 모델은 출력 단어로 나올 수 있는 다양한 단어들 중, 하나의 단어를 골라서 예측해야함

=> 이를 위해 softmax 함수 사용

=> Decoder에서 각 시점의 RNN 셀에서 출력 벡터가 나오면,

=> 해당 벡터는 softmax 함수를 통해 출력 시퀀스의 각 단어별 확률값을 반환하고,

=> Decoder가 출력단어를 결정함.



3. seq2seq의 다양한 변형

가장 기본적인 seq2seq에 대해 다루었음. 어떻게 구현하느냐에 따라 충분히 더 복잡해질 수 있음.

Attention 메커니즘

해당 방법을 이용하여 현재의 Context Vector보다 더욱 문맥을 반영할 수 있는 새로운 Context Vector를 구하여, 이 Context Vector를 매 시점마다 하나의 입력으로 사용할 수도 있다.