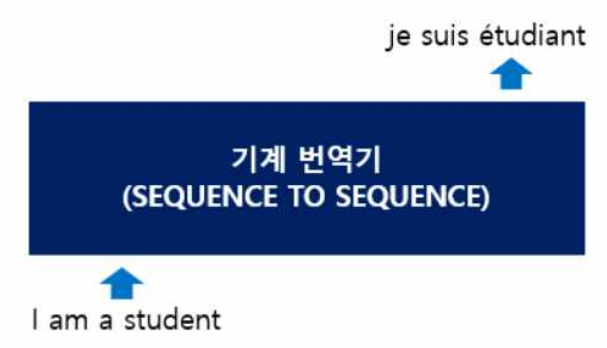
Sequence to Sequence (seq2seq)

이현재

∴ seq2seq의 내부 구조에서 하나의 Cell에서 사용되는 모델은 RNN, LSTM, GRU 등이 사용되지만 RNN으로 고정하시면 이해하는데 편합니다.

1. seq2seq 모델이란?

Sequence to Sequence(seq2seq) 모델은 Sequence(한 문장 혹은 하나의 시계열)을 다른 Sequence로 변환하는 모델입니다. 예를 들어서, <그림 1>은 seq2seq 모델을 통해 한 문장을 다른 언어로 변환시키는 과정입니다.



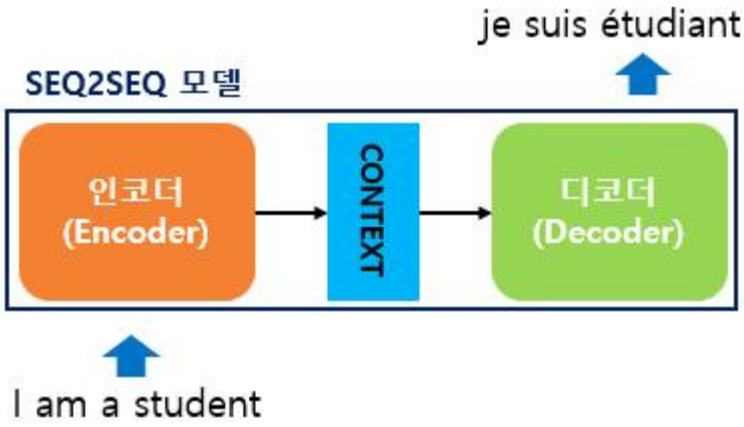
<그림 1> seq2seq 모델

"I am a student" -> [Seq2Seq model] -> "je suis étudiant"(input -> model -> output)의 순서로 seq2seq 모델이 작동합니다. 영어 문장을 input으로 seq2seq 모델에 넣으면 불어 문장인 output으로 출력하게 됩니다. 이 과정에서 input의 수와 output의 수가 같을 필요는 없습니다. 이 예시에서도 input은 4개, output은 3개가 나온 것을 확인할 수 있습니다.

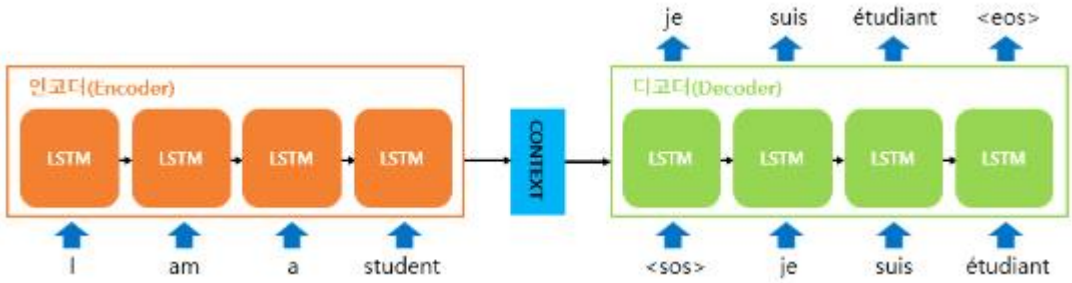
2. seq2seq 내부 구조

seq2seq 모델은 Encoder-Decoder 모델이라고도 합니다. <그림 2>와 같이 seq2seq 모델은 인코더(Encoder)와 디코더(Decoder)로 구성되어 있기 때문입니다. 문자 그대로 인코더는 입력 데이터를 인코딩(부호화)하고, 디코더는 인코딩된 데이터를 디코딩(복호화)합니다. 즉, 인코더는 입력을 처리하고 디코더는 결과를 생성합니다.

인코더는 'I am a student'라는 입력 문장을 받아 Context Vector를 만듭니다. Context Vector는 'I am a student'에 대한 정보를 압축하고 있는 벡터입니다. Context Vector는 다시 디코더로 전달되며 디코더는 이를 활용하여 최종적으로 'je suis étudiant'라는 불어 문장을 생성합니다. 즉, 인코더는 문장을 가지고 Context Vector를 만들어주는데, 이 Context Vector에는 문장에 대한 정보가 응축되어 있습니다. 반면, 디코더는 정보가 응축되어 있는 Context Vector로부터 다른 문장을 생성해줍니다. 참고로, **간단한 seq2seq 모델에서 Context Vector는 인코더의 마지막 스텝이 출력한 은닉 상태와 같습니다**. 이제 seq2seq 모델의 내부 구조를 더 자세히 살펴보겠습니다.



<그림 2> seq2seq 내부 구조



<그림 3> seq2seq 자세한 내부 구조

인코더나 디코더는 RNN으로 구성되어 있습니다. 실제로는 성능 향상을 위해서 LSTM이나 GRU가 사용되는데 <그림 3>에는 LSTM을 예시로 사용했습니다. 인코더의 LSTM은 단어 순으로 입력을 받습니다.

우선 인코더를 자세히보면, 입력 문장은 단어 토큰화를 통해서 단어 단위로 쪼개지고 단어 토큰 각각은 LSTM 셀의 각 시점의 입력이 됩니다. 인코더 LSTM 셀은 모든 단어를 입력받은 뒤에 인코더 LSTM 셀의 마지막 시점의 은닉 상태를 디코더 LSTM 셀로 넘겨주는데 이를 Context Vector라고 합니다. Context Vector는 디코더 LSTM 셀의 첫번째 은닉 상태에 사용됩니다. 인코더의 LSTM 계층은 오른쪽(시간 방향)으로도 은닉 상태를 출력하고 위쪽으로도 은닉 상태를 출력합니다. 이 구성에서 위에는 다른 계층(layer)이 없으니 LSTM 계층의 위쪽 출력은 폐기됩니다. <그림 3>과 같이 인코더에서는 마지막 문자를 처리한 후 LSTM 계층의 은닉 상태 층인 Context Vector를 디코더로 전달합니다.

디코더는 입력 문장을 통해 출력 문장을 예측하는 언어 모델 형식입니다. 위 그림에서 <sos>는 문장의 시작(start of string)을 뜻하고 <eos>는 문장의 끝(end of string)을 뜻합니다. 인코더로부터 전달받은 Context 벡터와 <sos>가 입력되면 그다음에 등장할 확률이 가장 높은 단어('je')를 예측합니다. 다음 스텝에서는 이전 스텝의 예측 값인 'je'가 입력되고 'je' 다음에 등장할 확률이 가장 높은 단어('suis')를 예측합니다. 이런 식으로 문장 내 모든 단어에 대해 반복합니다. 하지만 이는 **Test 단계에서의 디코더 작동 원리**입니다.

Training 단계에서는 교사 강요(teacher forcing) 방식으로 디코더 모델을 훈련합니다.

3. 디코더 Train에서의 교사 강요

현재 시점(time step)을 t라고 할 때, RNN 셀은 t-1에서의 hidden state와 t에서의 입력 벡터를 input으로 받고, t에서의 hidden state을 만듭니다. 이때 t에서의 hidden state는 바로 위에 또 다른 은닉층이나 출력층이 존재하는 경우에는 위의 층으로 보내거나, 필요없으면 값을 무시할 수 있습니다. 그리고 RNN 셀은 다음 시점에 해당하는 t+1의 RNN 셀의 입력으로 현재 t에서의 hidden state을 input으로 보냅니다. 교사 강요(teacher forcing)는 이와 다르게, **t-1에서의 예측 값을 t시점의 input으로 사용하는 것이 아니라 t-1의 실제값을 t에서의 input으로 넣어줍니다. 코드를 자세히 살펴보면 train에서 항상 teacher forcing을 사용하는 것이 아니라 확률적으로 teacher forcing을 시행합니다.**

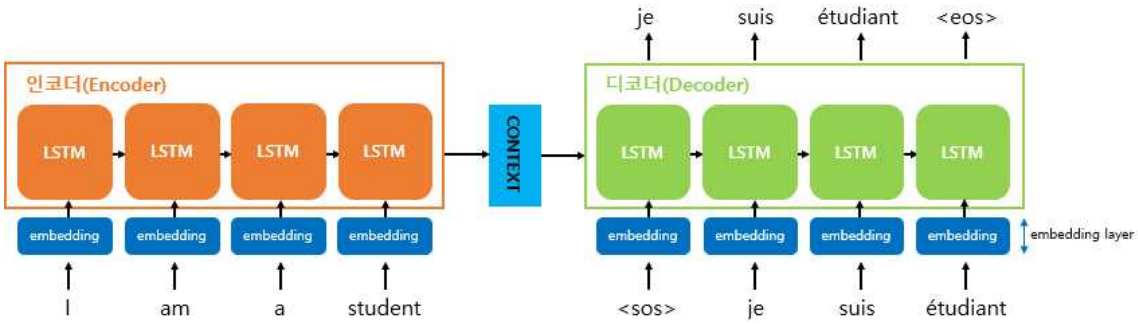
교사 강요를 하는 이유는 t-1의 예측값이 실제값과 다를 수 있기 때문입니다. 정확한 데이터로 학습하기 위해 예측값을 다음 시점으로 넘기는 것이 아니라 실제값을 매번 입력값으로 사용해야 합니다. 다시 말하면, 디코더의 Train 단계에서는 교사 강요 방식으로 학습하지만 테스트 단계에서는 일반적인 RNN 방식으로 예측합니다. 즉, 테스트 단계에서는 Context Vector를 입력값으로 받아 이미 훈련된 디코더로 다음 단어를 예측하고, 그 단어를 다시 다음 시점의 입력값으로 넣어줍니다. 이렇게 반복하여 최종 예측 문장을 생성하는 것입니다.

디코더의 학습 단계에서는 필요한 데이터가 Context Vector와 <sos>, je, suis, étudiant입니다. 하지만 테스트 단계에서는 Context Vector와 <sos>만 필요합니다. 학습 단계에서는 교사 강요를 하기 위해 <sos>뿐만 아니라 je, suis, étudiant 모두가 필요한 것입니다. 하지만 테스트 단계에서는 Context Vector와 <sos>만으로 첫 단어를 예측하고, 그 단어를 다음 시점의 입력으로 넣습니다.

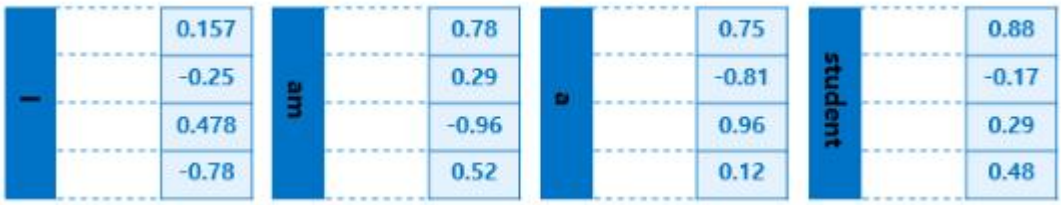
4. Word Embedding

기계는 텍스트보다 숫자를 잘 처리합니다. 자연어 처리에서 텍스트를 벡터(input)로 바꾸는 방법으로 주로 워드 임베딩(Word Embedding)이 사용됩니다. 즉, seq2seq에서 사용되는 모든 단어들은 임베딩 벡터로 변환 후 입력으로 사용됩니다. <그림 3>에서 Word Embedding을 추가한 것이 <그림 4>입니다.

I, am, a, student라는 문자도 모두 숫자로 즉, 벡터로 표현해야 합니다. <그림 5>는 I, am, a, student에 대해 워드 임베딩을 한 결과인 벡터를 나타냅니다. <그림 5>는 각 단어에 대해 4차원으로 표현했지만, 실제는 몇 백 차원으로도 표현한다고 합니다.
(0 혹은 1로 표현하는게 아니라 단어의 특성에 맞게 벡터로 처리되는 것 같습니다. am과 a가 약간의 유사성을 가져서 그렇게 추측해봅니다...!)

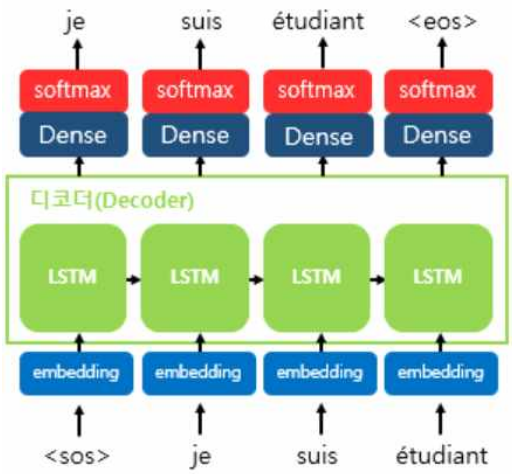


<그림 4> Word Embedding을 추가한 seq2seq 구조



<그림 5> Word Embedding

5. 디코더의 출력층



<그림 6> 디코더의 출력층

디코더에서 중점적으로 봐야 할 부분은 Softmax입니다. 디코더 RNN 셀의 출력으로 다양한 단어에 대한 벡터 값이 나올 것입니다. 그중 확률이 가장 높은 단어를 선택하기 위해 softmax를 취해줍니다. 이를 통해 최종 예측 단어를 생성합니다.

6. seq2seq 모델의 한계

1. 하나의 고정된 크기의 Context Vector에 대한 모든 정보를 압축하려고 하니 정보 손실이 발생
   - 고정된 벡터의 길이가 아닌 입력 문장의 길이에 맞추는 방법으로 해결
     → <그림 7>처럼 각 time step마다 hidden state를 만들어 stack을 쌓는 것으로 해결
2. 입력 sequence가 길어질수록 초기 입력 sequence의 정보를 잃는 Gradient Vanishing 문제 발생

<그림 7> 고정된 크기의 Context vector 문제 해결

7. 정리

seq2seq는 인코더와 디코더로 구성되어 있으며, 인코더는 입력 문장의 정보를 압축하는 기능을 합니다. 인코더에 의해서 압축된 정보는 Context 벡터라는 형식으로 디코더에 전달됩니다. 디코더는 훈련 단계에서는 교사 방식으로 훈련되며, 테스트 단계에서는 인코더가 전달해준 Context 벡터와 <sos>를 입력값으로 하여 단어를 예측하는 것을 반복하며 문장을 생성합니다.

8. 참고

· hidden state가 만들어지는 과정

https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

9. 코드

코드는 따로 첨부하겠습니다!