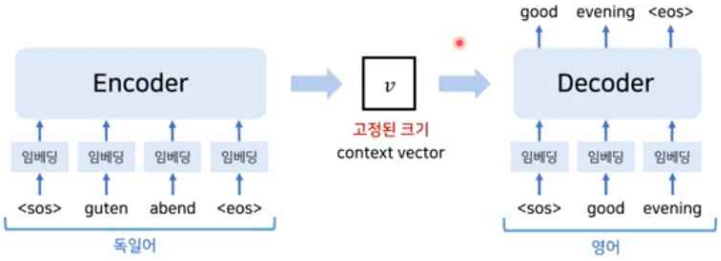
# [ Sequence to Sequence Learning with Neural Networks 개념 정리 ]




LSTM을 활용한 효율적인 기계번역 아키텍처
인코더와 디코더로 구성, 하나의 문장 시퀀스에서 또 다른 문장 시퀀스로 처리하는 것
고정된 크기의 벡터에 문장의 내용을 모두 담아서 문맥 정보를 저장하는 보틀넥으로 사용
BottleNeck, 다 담지 못 한다면 한계점으로 작용

## 1. 사전 개념 - 언어 모델 (Language Model)

언어 모델이란 문장(시퀀스)에 확률을 부여하는 모델을 의미
특정 상황에서 적절한 문장이나 단어를 예측할 수 있다.
즉, P(난 널 사랑해 | I love you) > P(난 널 싫어해 | I love you), 확률로 가장 적절한 언어를 판단해내는 것

하나의 문장은 (W) 여러 개의 단어 (w)로 구성된다
P(W) = P(w1, w2, w3, w4... wn), P(친구와 친하게 지낸다) = P(친구와, 친하게, 지낸다)

결합확률분포 Joint probability

- 연쇄 법칙 (Chain Rule)
  - $P(w_1, w_2, w_3, \ldots, w_n) = P(w_1) * P(w_2|w_1) * P(w_3|w_1, w_2), \ldots, P(w_n|w_1, w_2, \ldots, w_{n-1})$

$$= \prod_{i=1}^{n} P(w_i|w_1, \ldots, w_{i-1})$$

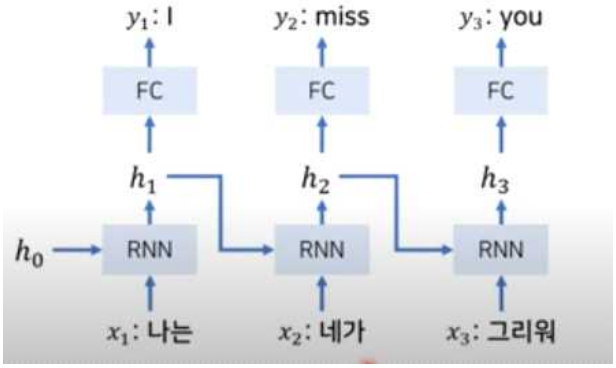
- $P(\text{친구와 친하게 지낸다}) = P(\text{친구와}) * P(\text{친하게}|\text{친구와}) * P(\text{지낸다}|\text{친구와 친하게})$

**-딥러닝 이전의 전통적인 통계적 언어 모델**
카운트 기반의 접근을 사용, 전체 corpus에서의 개수를 이용해서 확률 계산

- $P(\text{지낸다}|\text{친구와 친하게}) = \frac{count(\text{친구와 친하게 지낸다})}{count(\text{친구와 친하게})}$

사전에 정말 큰 corpus를 가지고 있어야하며, 매우 방대한 양의 데이터가 필요
"친구와 친하게" 하는 시퀀스 자체가 학습데이터에 존재하지 않으면?
긴 문장의 처리는 어떻게?
→이와 같은 한계를 극복하기 위해 N-gram 언어 모델이 사용
→일부 인접한 단어만 고려하는 아이디어

**-전통적인 RNN 기반의 번역 과정**




- $h_t = sigmoid(W^{hx}x_t + W^{hh}h_{t-1})$
- $y_t = W^{yh}h_t$

전통적인 RNN 기반의 기계 번역은 입력과 출력의 크기가 같다고 가정
입력 : (x1, x2, … xi) 이라면 출력값 또한 (y1, y2, … yi) 의 동일한 scale을 가짐

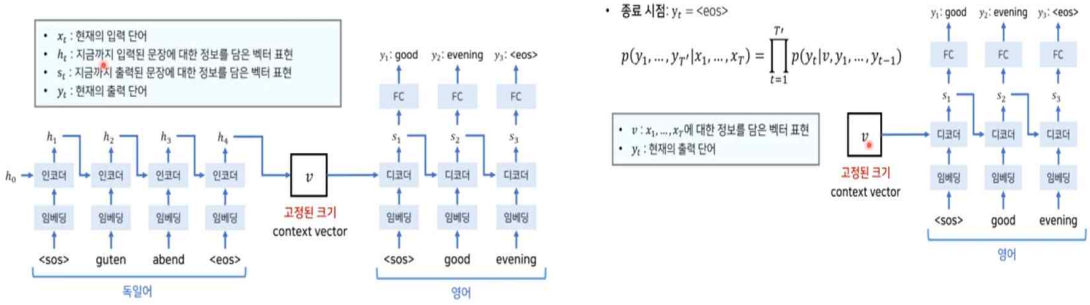
은닉층의 각 값은 이전까지 입력되었던 데이터에 대한 전반적인 문맥 정보를 담고 있음
h0은 대부분 초기화된 값을 사용
h1은 h0의 값과 새로 들어온 데이터 x1의 값을 적절하게 조합하여 새로운 값을 만들어낸 것
h2는 h1의 값과 새로 들어온 데이터 x2의 값을 적절하게 조합하여 새로운 값을 만들어낸 것
이러한 각각의 은닉층 값 hi는 linear한 계층을 거치게 되어(시그모이드 함수 없이) 하나의 출력값인 yi를 만들어내게 되는 것
(CNN과 MLP와 비슷, 마지막 출력 레이어에는 activation 함수를 적용 안 함)

한계점
:입력층과 출력층의 스케일이 동일하다고 가정, 현실적인 번역 문제에서 적용하기 쉽지 않음
:문법적인 어순이 다른 경우 정확한 번역이 불가능 (“그리워”의 위치는 3, miss의 위치는 2)
ㄴSeq2Seq에서 제안하는 방법 : 하나하나 RNN에서 단계단계 해석하는 것이 아니라, 인코더를 통해서 모오오오오오든 문맥 정보를 담는 하나의 context 벡터를 뽑은 뒤에 디코딩을 진행하는 방식을 제안

**- RNN 기반 Sequence to Sequence 개요**
하나하나 RNN에서 단계단계 해석하는 것이 아니라, 인코더를 통해서 모오오오오오든 문맥 정보를 담는 하나의 context 벡터를 뽑는 과정을 거침
그 후에 문맥 벡터로부터 디코더가 번역 결과를 추론하는 방식을 제안
나아가 이 논문에서는 LSTM을 이용해 문맥 벡터를 추출하도록 하여 성능을 향상시킨다
(인코더의 마지막 hidden state만을 context 벡터로 사용, 즉 위 그림의 h3을 context 벡터로 채택한다는 의미, 별다른 합계 계산 x)
**인코더와 디코더는 서로 다른 파라미터 가중치를 가진다

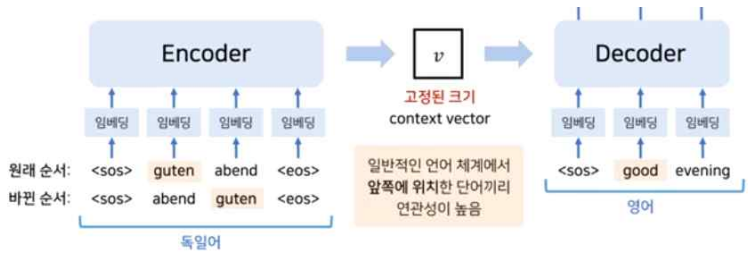
긴 문장을 다룰 수 있는 능력이 향상됨

## 2. 사전 개념 - RNN 기반의 Sequence to Sequence




(1) 독일어 문장이 들어옴, 딥러닝 기반에서는 임베딩을 붙이는 경우가 많음
등장 빈도수가 높은 단어가 20000개 정도라고 한다면 , 입력 단어를 20000차원의 원핫인코딩으로 나타내면 너무 크기가 큼, 따라서 1000차원 정도로 작은 데이터로 표현하는 layer
(2) 인코딩 레이어를 거친 RNN을 거침
(3) 마지막 은닉층 값이 context 벡터로 사용됨, 그리고 이것이 디코더의 입력값이 됨
(4) 디코더 부분도 마찬가지로 입력을 임베딩을 통해 차원 축소
(5) 초기 값이 context 벡터 값, 디코더를 거쳐 hidden state 값을 만들어내고, 이 값들이 각각 간단한 linear layer(FC layer)을 거쳐서 결과값이 나오게 함
(6) 종료 시점은 결과값 yi가 <eos> 가 나왔을 때

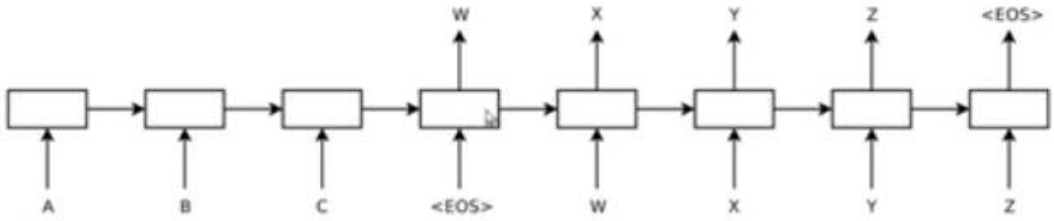
*디코더와 인코더는 서로 다른 언어를 사용하기 때문에 입출력 차원이 서로 다를 수 있음
*인코더 구현 시 <sos> <eos> 토큰을 추가
*context vector은 한정된 크기이므로, 훈련 시 짧은 길이의 텍스트 데이터만 받다가 실제 test 시 긴 문장이 들어온다면 정보 성능이 떨어질 수 있게 됨




** 기본적인 RNN 대신에 LSTM을 활용하였을 때 더 높은 정확도를 보인다
** 실제 학습 및 테스트 과정에서 입력 문장의 순서를 거꾸로 하였을 때 더 높은 정확도를 보이며, 여기서 출력 문장의 순서는 바꾸지 않음 (일반적인 언어 체계에서 앞쪽에 위치한 단어끼리의 연관성이 높다는 언어적 특징에서 기인한 결과, 주어가 가장 먼저 등장하는 것이 예시) - 입력 순서가 앞쪽일수록 vanishing 현상이 발생하여 중요도가 낮아지는 현상을 방지

## 3. 논문리뷰 - Sequence to Sequence Learning with Neural Networks

(1) Introduction
input 문장의 전부를 하나의 context 벡터로 변환한 후 처리하는 방법 활용




A, B, C가 입력으로 들어오게 되고 EOS를 만나게 된다면 hidden state 마지막 값을 context 벡터로 사용하여 디코더 파트에서 같이 받아서 번역 결과를 반환
실제 구현에서는 문장이 'ABC"라면 뒤집혀진 순서로 "CBA"로 입력을 진행 -> 더 성능이 굿
앙상블 기법을 사용해서 더 성능을 높임
나아가 LSTM을 사용하였을 때 긴 문장에 대해서도 대처가 아주 좋아짐

(2) The Model
RNN 모델의 수학적인 표현은 다음과 같을 수 있음
input은 (x1, x2, ... , xr), 출력값은 (y1, y2, ... , yr) 일 때

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

현재 hidden state 값을 계산할 때에는 현재 입력 데이터의 값에 x를 위한 가중치를 곱한 값과, 이전 hidden state 값에 은닉층을 위한 가중치를 곱한 값을 더한 다음 activation 함수를 취해준 값을 사용한다
그리고 현재 hidden state는 linear한 층을 지나며 결과값으로 도출되게 되는 것

[한계점 1]
이와 같은 RNN 모델의 단점은 입력 값과 출력값의 scale 이 다를 때에는 사용하기 어렵다는 것이다
->입력 시퀀스가 하나의 고정된 크기의 벡터로 바뀔 수 있도록 하는 방법을 제안
-> 인코더 파트를 위한 RNN과 디코더 파트를 위한 RNN을 따로 사용하여 디코더 파트에서 최종적인 출력 시퀀스를 출력하게 된다는 점이 이 논문의 제안임

[한계점 2]
기본적인 RNN은 Longterm dependancy를 처리하기 어려움
-> 하나의 context 벡터로 변환함에 따라 긴 문장을 잘 처리할 수 있는 구조가 되었음

번역 문제를 해당 논문에서 제안하는 방식으로 수식을 표현하면 다음과 같을 수 있음

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

입력과 출력의 크기는 동일하지 않아도 됨
입력에 대한 최종 인코딩 값은 출력의 가장 앞단의 v 벡터 하나로 바뀌게 되고 매번 출력 결과를 recurrent하게 거쳐 최종 출력물을 생성해낼 수 있음
입력 : “a”, “b”, “c”, “<EOS>”
출력 : “w”, “x”, “y”, “z”, “<EOS>”

인코더 파트와 디코더 파트에 사용되는 LSTM의 파라미터는 서로 다름
LSTM은 하나만 사용하는 것이 아니라 네 개의 Layer을 겹쳐서 사용, RNN 모델과 같이 하나만 겹쳐서 사용하는 것이기 때문에 model의 capacity가 올라가게 됨

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

즉, 얘를 네 번 쌓게 되는 것이므로 모델이 더 고도화, 그러나 너무 깊게 쌓게 되면 비용이 증가하긴 하겠지
그리고 문장의 단어 순서를 뒤집어서 입력을 하게 되면 더 성능이 향상되는 결과를 발견할 수 있었음

(3) Experiment
[1] Dataset : WMT’14 영어 -> 불어

[2] Decoding and Rescoring
S는 Source input, T는 Target translation 이라면,

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

와 같이 S가 입력되었을 때 타겟값 T가 출력되도록 학습을 진행, log를 붙임으로써 확률값이 높아지는 방향으로 학습을 진행하였음

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

학습이 모두 이루어진 다음, 가장 높은 확률을 가지는 문장을 반환할 수 있도록

beam search 방법을 디코더에 사용 - 그리디하게 항상 가장 높은 확률의 문장을 선택해서 반환하는 것이 아니라 특정 깊이만큼 더 깊게 들어가볼 수 있도록 하여 결과적으로 확률이 더 높은 아이를 반환

[3] Reversing Source Sentence : 입력을 거꾸로 입력하면 성능이 더 좋아짐

[4] Training Details
실제로 LSTM을 4차례 깊게 쌓은 구조로 설계
워드 임베딩은 1000차원으로 만들어서 차원을 줄임
LSTM의 파라미터는 -0.08 ~ 0.08 사이의 값을 따르는 uniform distribution을 따르도록 초기화하였음
모멘텀 없이 stochastic 방법을 통해 gradient을 조정하는 방식을 취하였고 batchsize은 128
-배치 크기가 큰 경우 (길이가 긴 문장에 맞추는 경우) 다른 짧은 문장들 또한 이 사이즈를 맞추기 위해 패딩이 들어가야 하는데, 이는 너무 학습 효율의 낭비를 발생시킴, 아주 작은 배치사이즈를 채택하며 학습 속도를 2배로 올림

[8] Model Analysis

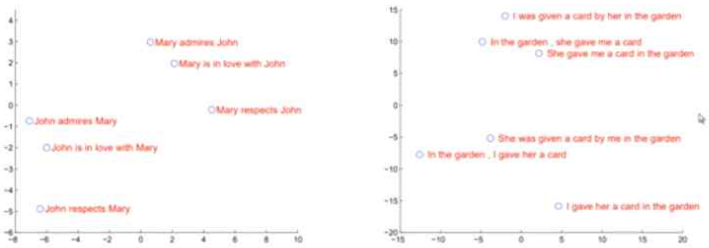



Figure 2: The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words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

LSTM의 hidden state 값을 PCA 사영을 통해 2차원으로 나타낸 결과값에 대한 그래프
비슷한 의미의 문장끼리 잘 클러스터링이 된 것을 확인할 수 있음