Sequence to Sequence

Contents

1. Background

I. DNN

II. SMT

III. NMT

2. Sequence to Sequence

I. Encoder - Decoder 구조

II. The model

III. Training

IV. Inference

3. Experimental results & Conclusion

TRAIN AND TEST

Background

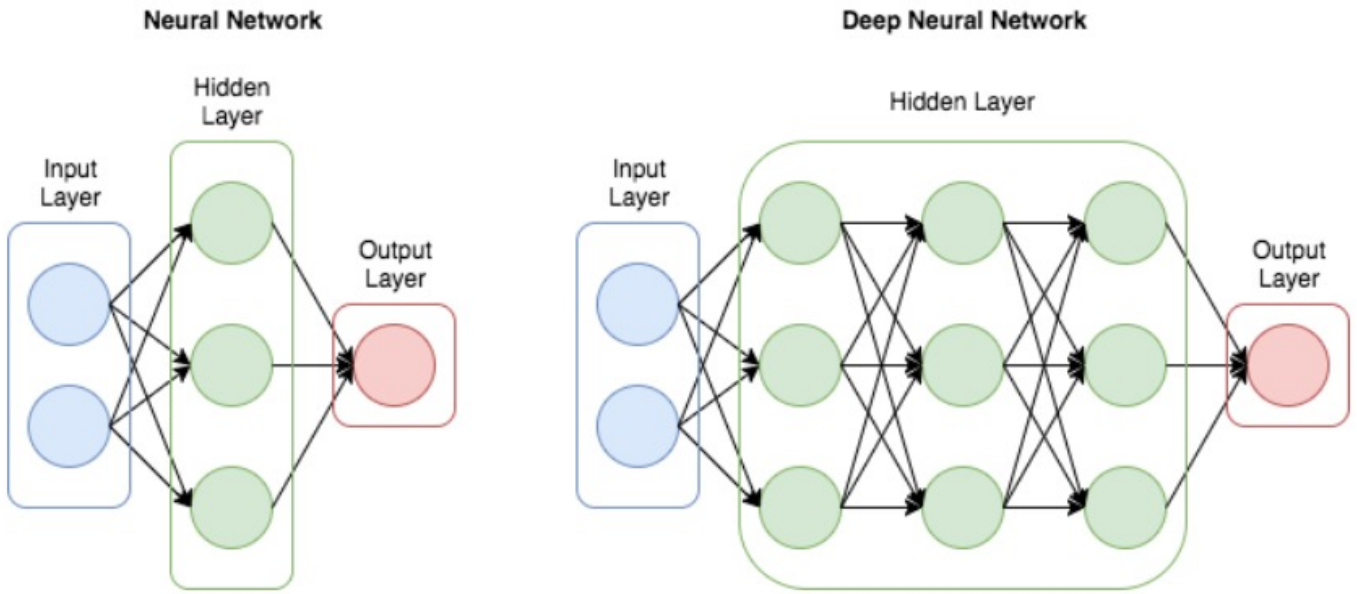
DNN

DNN (Deep Neural Networks)

: 사람의 신경망을 모방해 만든 알고리즘인 ANN의 발전형태

: Hidden Layer의 개수를 ANN보다 추가하여 더 나은 성능을 냄

: 음성인식, 시각적 물체 인식과 같은 어려운 문제들에 대한 좋은 성능을 내는 것으로 알려짐



TRAIN AND TEST

DNN

DNN (Deep Neural Networks)

: 사람의 신경망을 모방해 만든 알고리즘인 ANN의 발전형태

: Hidden Layer의 개수를 ANN보다 추가하여 더 나은 성능을 냄

: 음성인식, 시각적 물체 인식과 같은 어려운 문제들에 대한 좋은 성능을 내는 것으로 알려짐

장점

Flexibility & Power

- 입력 변수간 비선형적 조합이 가능
- 연속형, 범주형 모두 사용 가능

단점

input과 output의 차원이 고정

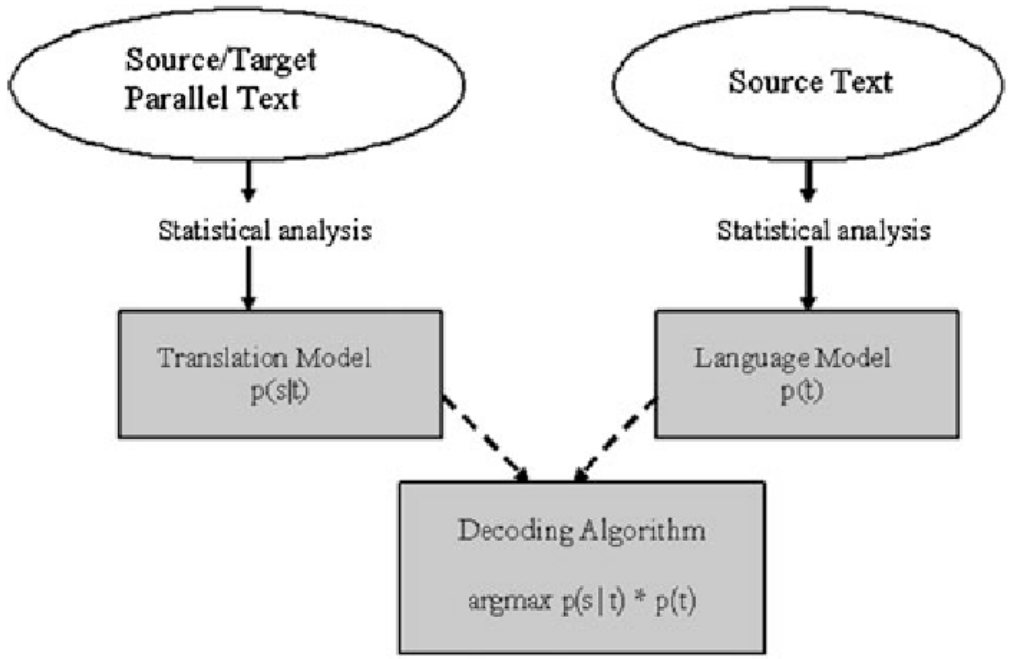
- 음성인식과 MT 등 시퀀셜한 데이터의 경우 적용이 어려움

SMT

SMT (Statistical Machine Translation)

: 대용량 코퍼스에서 학습된 통계정보를 활용하여 기계번역 수행

: 번역모델과 언어모델로 나누어 번역 수행



TRAIN AND TEST

SMT

SMT (Statistical Machine Translation)

: 대용량 코퍼스에서 학습된 통계정보를 활용하여 기계번역 수행

: 번역모델과 언어모델로 나누어 번역 수행

$$argmax_y P(y|x)$$

Bayes rule

$$argmax_y P(x|y)P(y)$$

- 한국어를 영어로 바꾸는 task면 x는 한국어 y는 영어
- 한국어 문장 x가 given시 가장 적절한 영어 문장 y를 찾는 것

- 한국어 -> 영어로 번역하는 단일모델을 만들지 못함
- 베이즈 정리를 이용하여 모델을 변형

x : source 문장
y : target 문장
P(y|x) : 우리가 찾고자 하는 번역 모델

TRAIN AND TEST

SMT

SMT (Statistical Machine Translation)

: 대용량 코퍼스에서 학습된 통계정보를 활용하여 기계번역 수행

: 번역모델과 언어모델로 나누어 번역 수행

$$argmax_y P(x|y)P(y)$$

P(y) : 언어 모델

- 단일 언어 데이터를 이용해 확률 계산
- 단일 언어에서 다음에 올 단어의 확률의 예측하는 모델
- eg) 영어문장 자체가 얼마나 자연스러운지 확률분포로 표현

P(x|y) : 번역 모델

- 두 언어 모두를 이용한 데이터를 이용
- 두 언어의 대응 관계에 대한 모델
- eg) 영어 문장이 주어졌을 때 한국어 문장의 확률분포를 표현

x : source 문장
y : target 문장
P(y|x) : 우리가 찾고자 하는 번역 모델

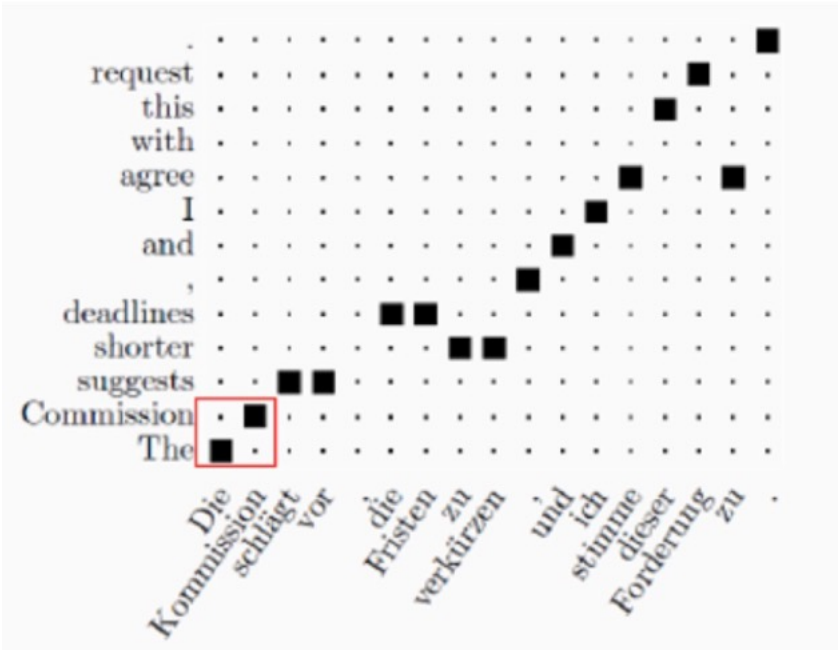
TRAIN AND TEST

SMT

SMT (Statistical Machine Translation)

: 대용량 코퍼스에서 학습된 통계정보를 활용하여 기계번역 수행

: 번역모델과 언어모델로 나누어 번역 수행



Alignment

- Bilingual Corpus를 위해 필요
- 두 언어 사이의 병렬 코퍼스를 이용하여 언어 간의 대응관계 파악
- 단점) feature engineering이 중요

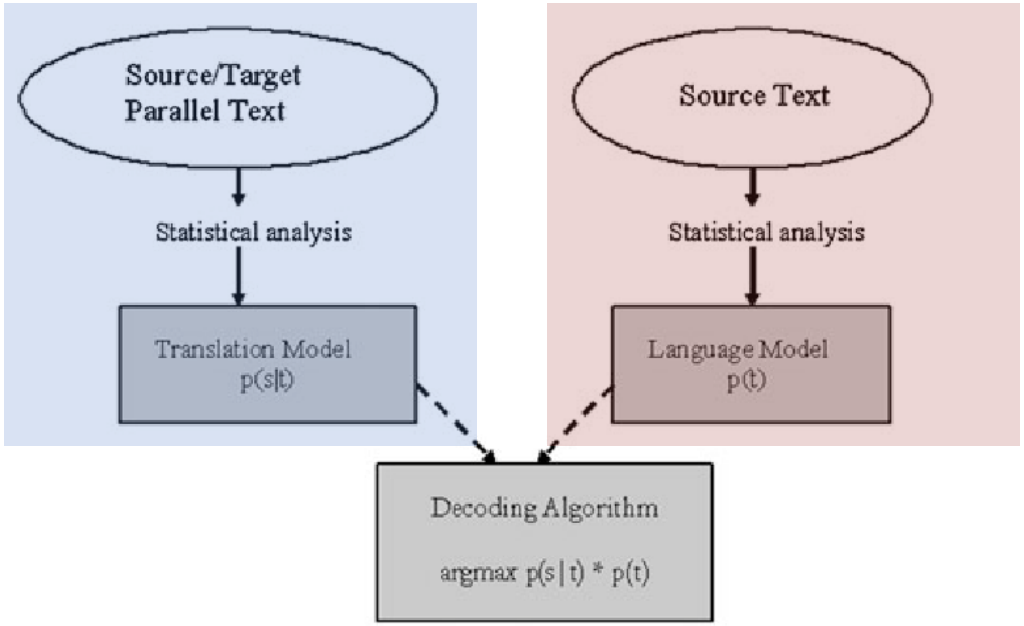
TRAIN AND TEST

SMT

SMT (Statistical Machine Translation)

: 대용량 코퍼스에서 학습된 통계정보를 활용하여 기계번역 수행

: 번역모델과 언어모델로 나누어 번역 수행



언어모델, 번역모델을 각각 만든 후 합쳐서 번역을 수행

↓

결합모델이기 때문에

최적해를 찾기가 어려움

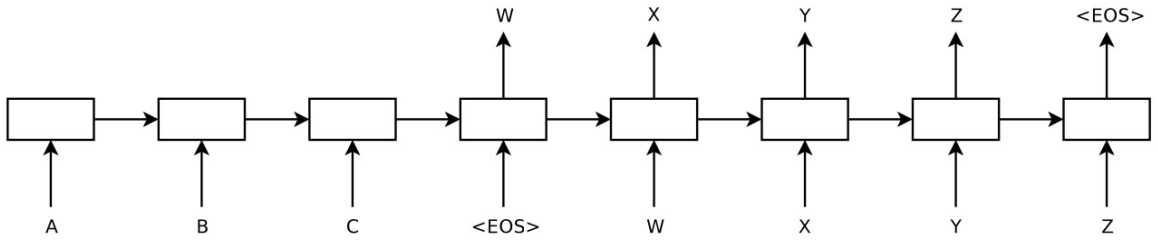
TRAIN AND TEST

NMT

NMT (Neural Machine Translation)

: source 문장에서 target 문장의 다음 단어가 나타날 확률을 single neural network로 예측하는 방법

: Encoder - Decoder 구조로 구성



- 두 언어의 paralell corpus 만 존재하면 학습 가능
- 영어 문장이 given일 때 한국어 문장에서 다음 단어 예측하는 모델 building 가능

↓

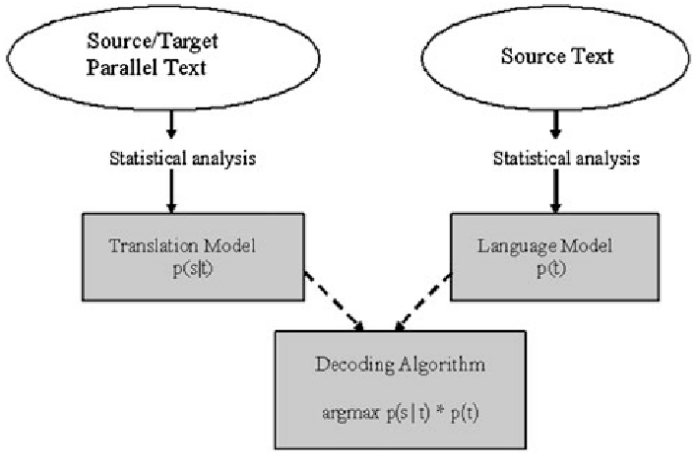
SMT보다 간단함

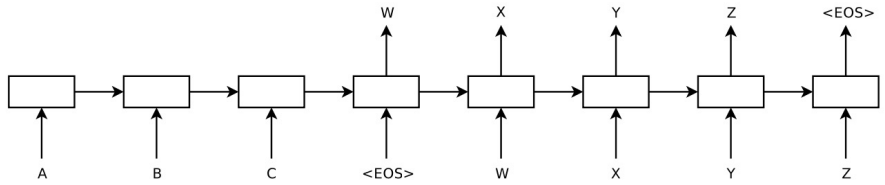
일반적인 machine translation 모델

TRAIN AND TEST

SMT vs. NMT

SMT vs. NMT





- 표현가능 문맥 범위 : SMT) phase 이상 표현 불가 / NMT) 문장 전체 정보
- 모델 최적화 방법 : SMT) 두 모델을 독립적으로 최적화 / NMT) global optimization

TRAIN AND TEST

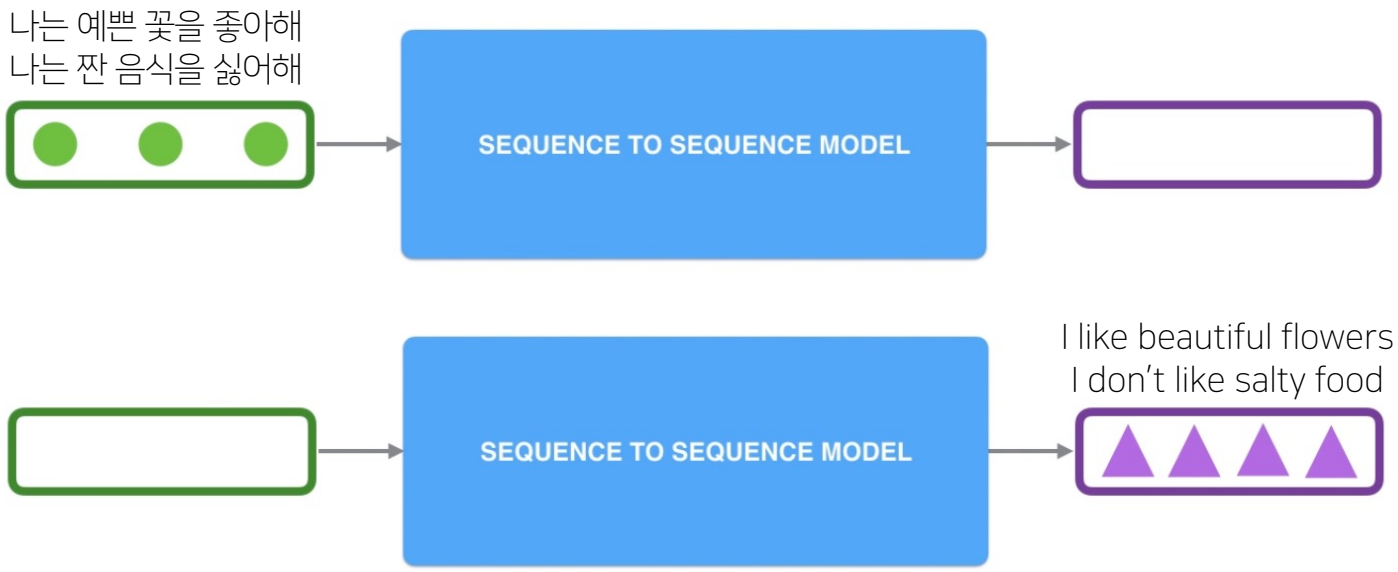
Sequence to Sequence

Sequence to Sequence

Seq2Seq

: 시퀀스를 입력으로 받아서 처리한 후 시퀀스를 출력으로 내보내는 모델

: Encoder - Decoder 구조로 구성



입력 시퀀스의 아이템의 개수와 출력 시퀀스의 아이템 개수가 다를 수 있다

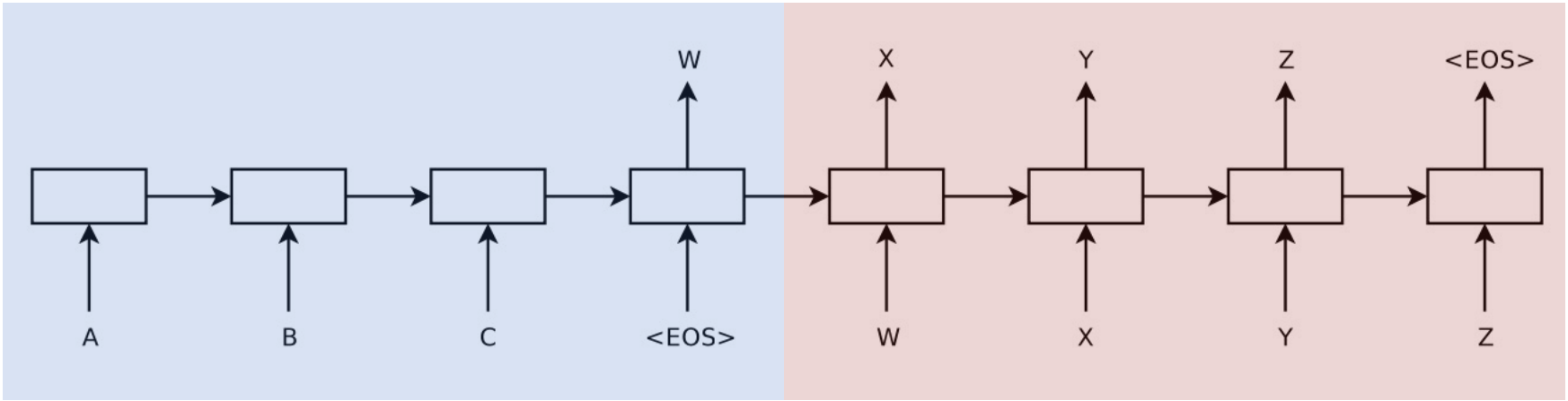
TRAIN AND TEST

Sequence to Sequence

Seq2Seq

: 시퀀스를 입력으로 받아서 처리한 후 시퀀스를 출력으로 내보내는 모델

: Encoder - Decoder 구조로 구성



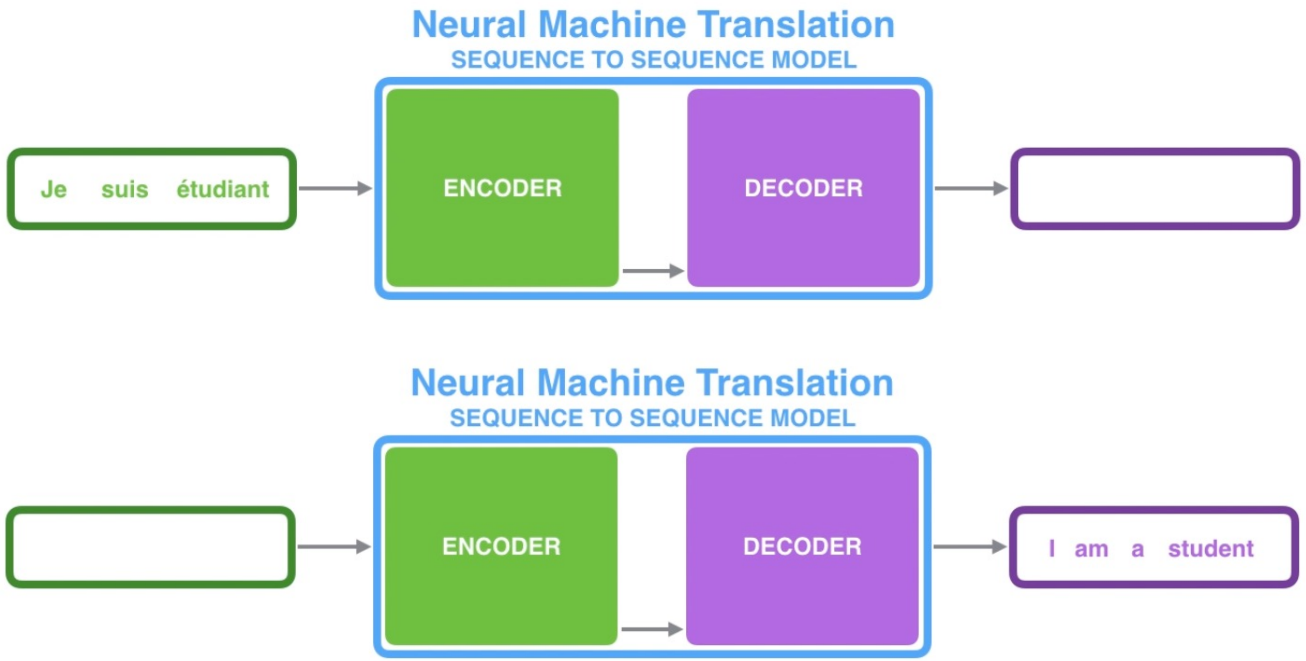
TRAIN AND TEST

Encoder - Decoder 구조

Encoder - Decoder 구조

: 인코더) input sequence의 각 아이템을 가공 후 각 정보를 합쳐서 context vector 생성

: 디코더) 인코더의 context vector를 넘겨 받아서 대응되는 아이템을 시퀀스로 출력

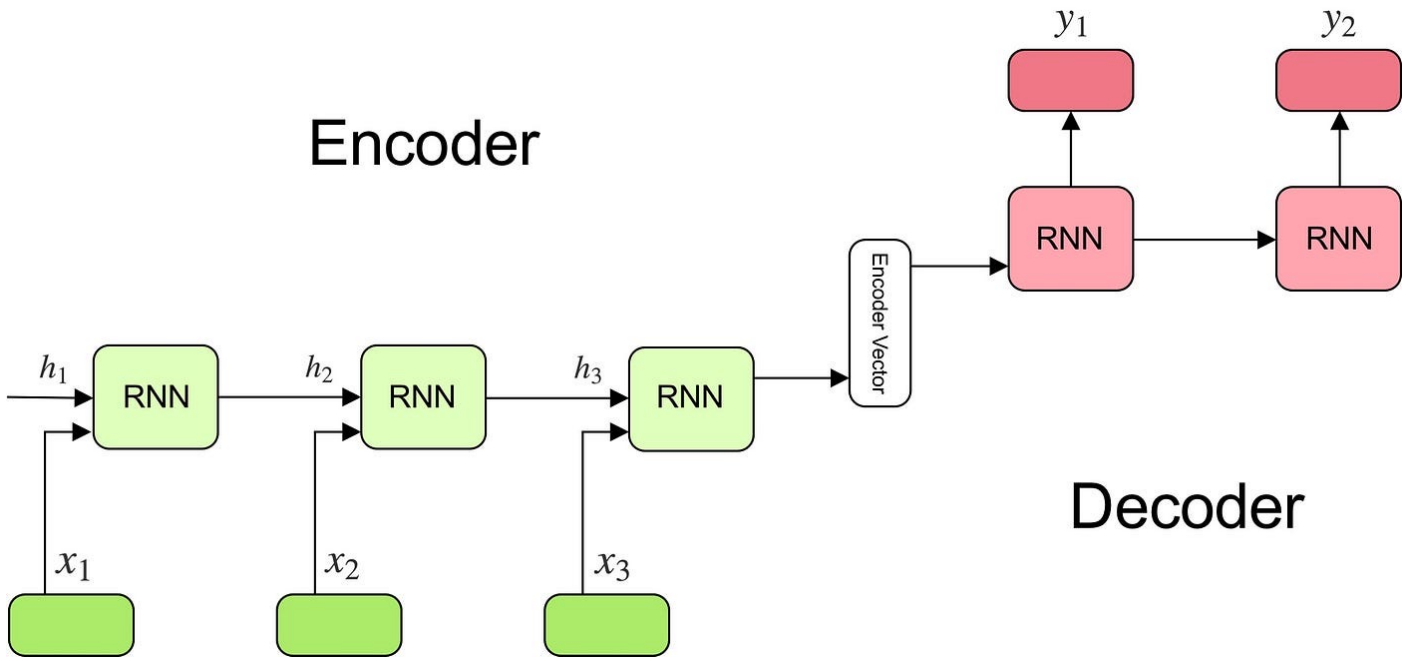


TRAIN AND TEST

RNN based Encoder - Decoder 구조

RNN based Encoder - Decoder

: Encoder - Decoder 구조에서 RNN을 활용



TRAIN AND TEST

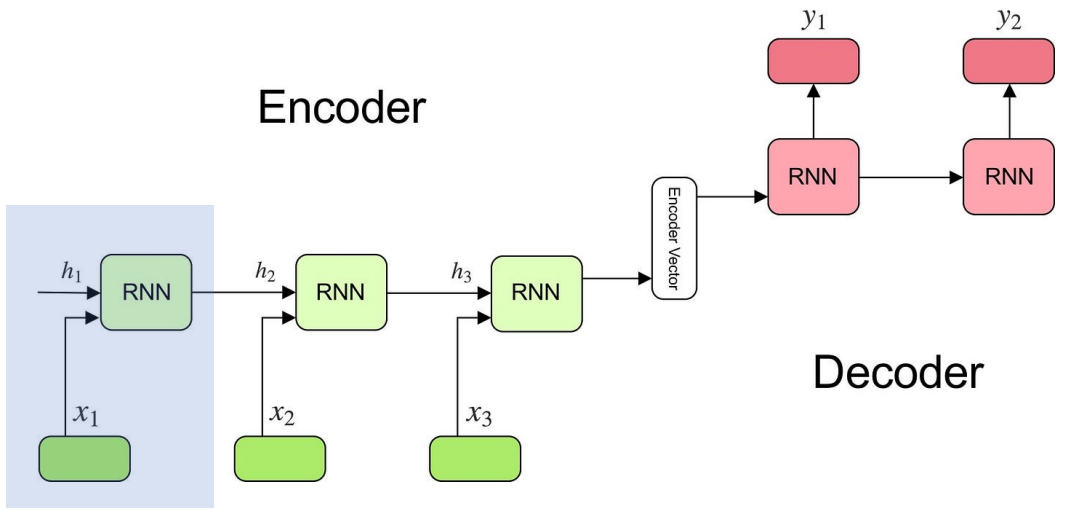
RNN based Encoder - Decoder 구조

RNN based Encoder - Decoder

: Encoder - Decoder 구조에서 RNN을 활용

$$h_t = f\left(x_t, h_{t-1}\right)$$

$$c = q\left(\{h_1, \cdots, h_{T_x}\}\right)$$



- 1. source sentence를 토큰 단위로 sequential하게 받음
- 2. 각 입력 토큰: 이전 시점의 hidden state vector와 함께 들어감
- 3. RNN셀에서 처리하여 hidden state를 update

TRAIN AND TEST

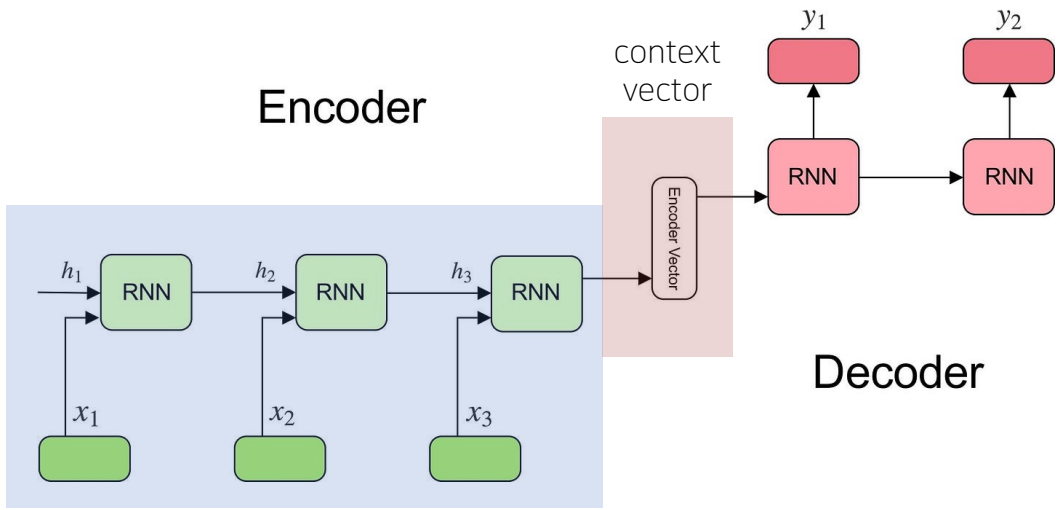
RNN based Encoder - Decoder 구조

RNN based Encoder - Decoder

: Encoder - Decoder 구조에서 RNN을 활용

$$h_t = f\left(x_t, h_{t-1}\right)$$

$$c = q\left(\{h_1, \cdots, h_{T_x}\}\right)$$



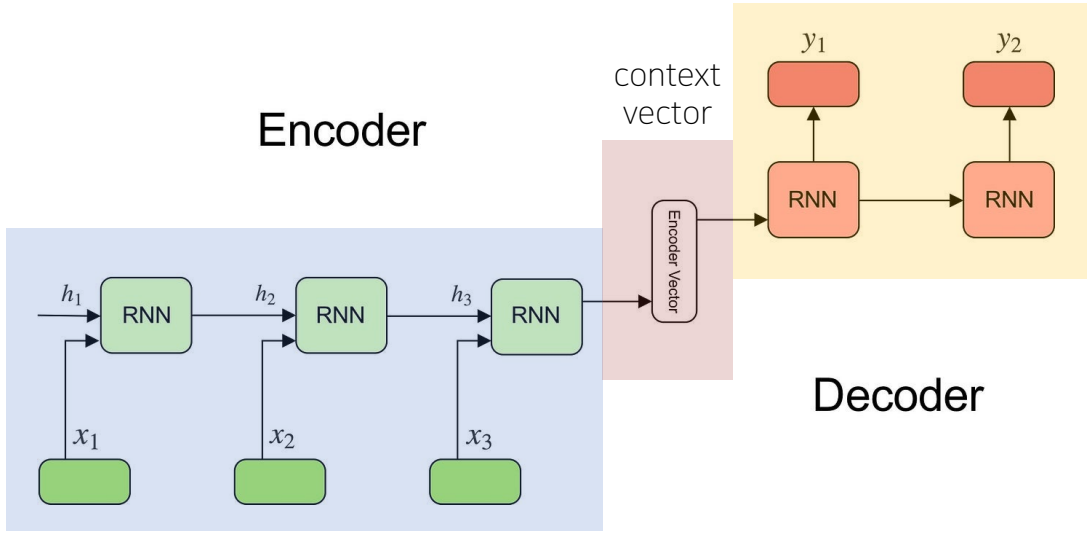
- 모든 input의 토큰이 다 들어왔을 때 최종적으로 생성된 hidden state를 context vector로 지정
- context vector : 인코더 문장의 정보가 축약된 벡터

TRAIN AND TEST

RNN based Encoder - Decoder 구조

RNN based Encoder - Decoder

: Encoder - Decoder 구조에서 RNN을 활용



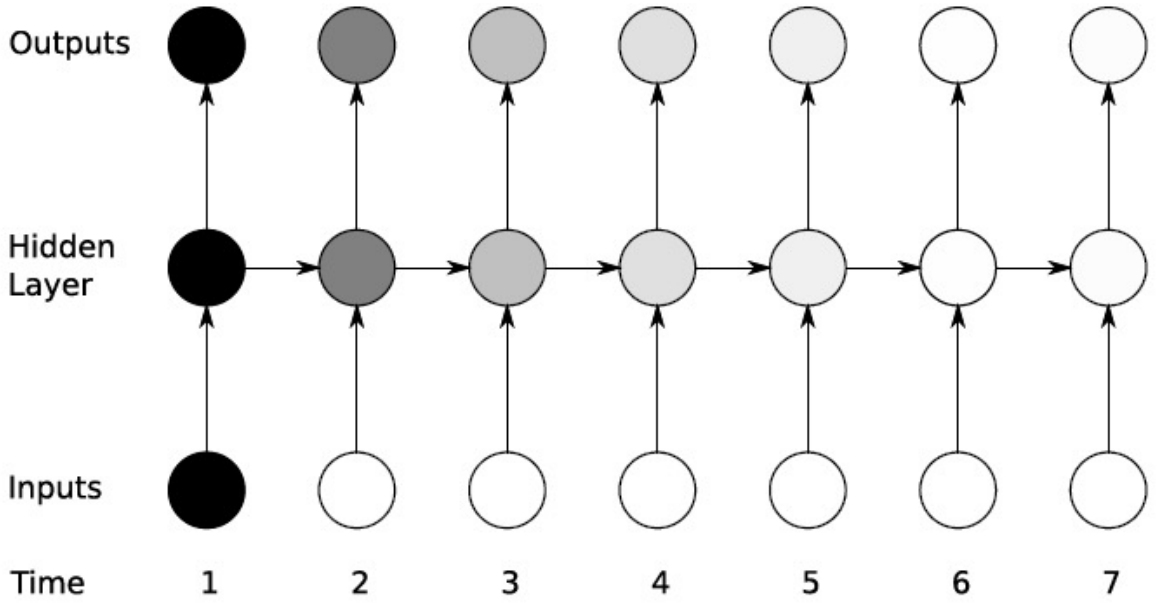
- context vector와 이전 시점의 hidden state vector, 이전 시점의 예측 토큰으로 t시점의 hidden state vector로 update
- t시점의 hidden state vector와 context vector, 이전 시점의 예측 토큰으로 다음에 올 토큰을 확률적으로 예측

TRAIN AND TEST

RNN based Encoder - Decoder 구조

RNN based Encoder - Decoder

: Encoder - Decoder 구조에서 RNN을 활용



- context vector : 인코더의 가장 마지막 hidden state vector → 마지막 item에 더 큰 영향, 앞 쪽 item에는 적은 영향 받음
- LSTM : 길이가 긴 문장에 대한 장기 의존성 문제를 해결 → 본 논문에서 RNN대신 cell로 사용

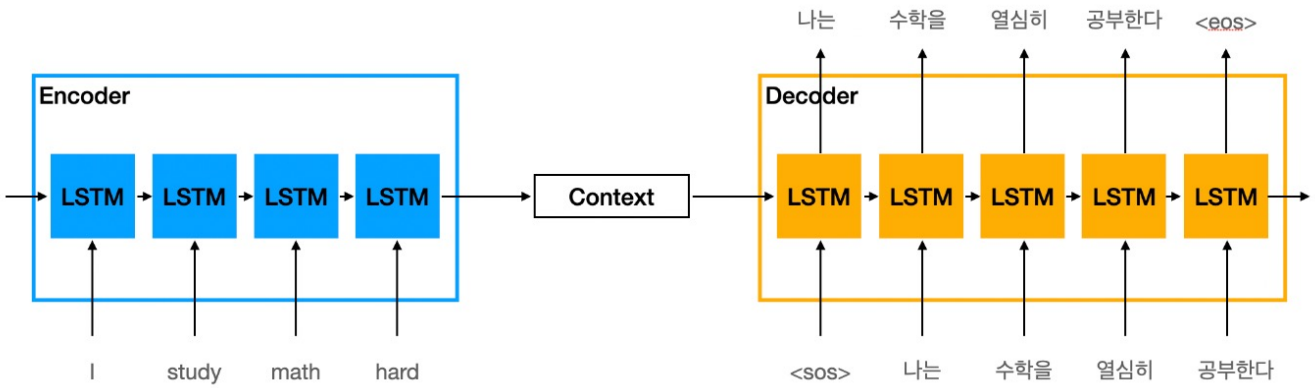
TRAIN AND TEST

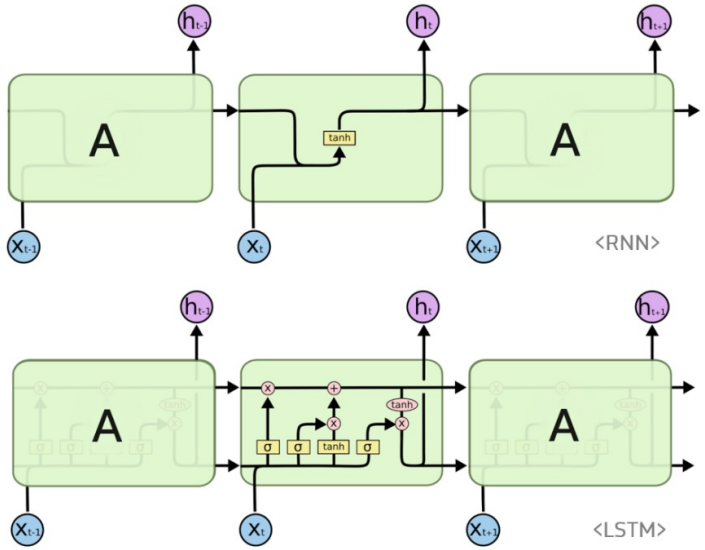
The model

LSTM

: RNN 대신 LSTM을 이용한 Encoder - Decoder 구조 이용

: 길이가 긴 문장에 대한 정보를 잃지 않게 하기 위함





- cell state와 hidden state로 구성
- RNN에 cell state을 추가 : 길이가 긴 문장에서의 문제를 해결
- cell state : 정보를 장기적으로 유지, 전달 / hidden state : 현 시점에서 입력에 대한 모델의 출력

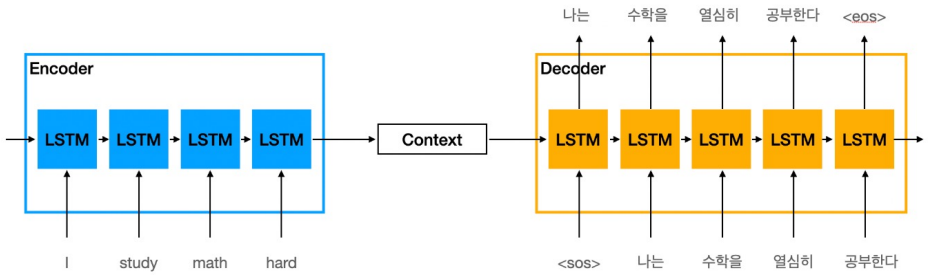
TRAIN AND TEST

The model

LSTM

: RNN 대신 LSTM을 이용한 Encoder - Decoder 구조 이용

: 길이가 긴 문장에 대한 정보를 잃지 않게 하기 위함



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

P(I, study, math, hard | 나는, 수학을, 열심히, 공부한다)

= P(I| context , <SOS>) x P(study | context , <SOS>, I)

x P(math | context , <SOS> , I, study) x P(hard | context , <SOS> , I, study, math, hard)

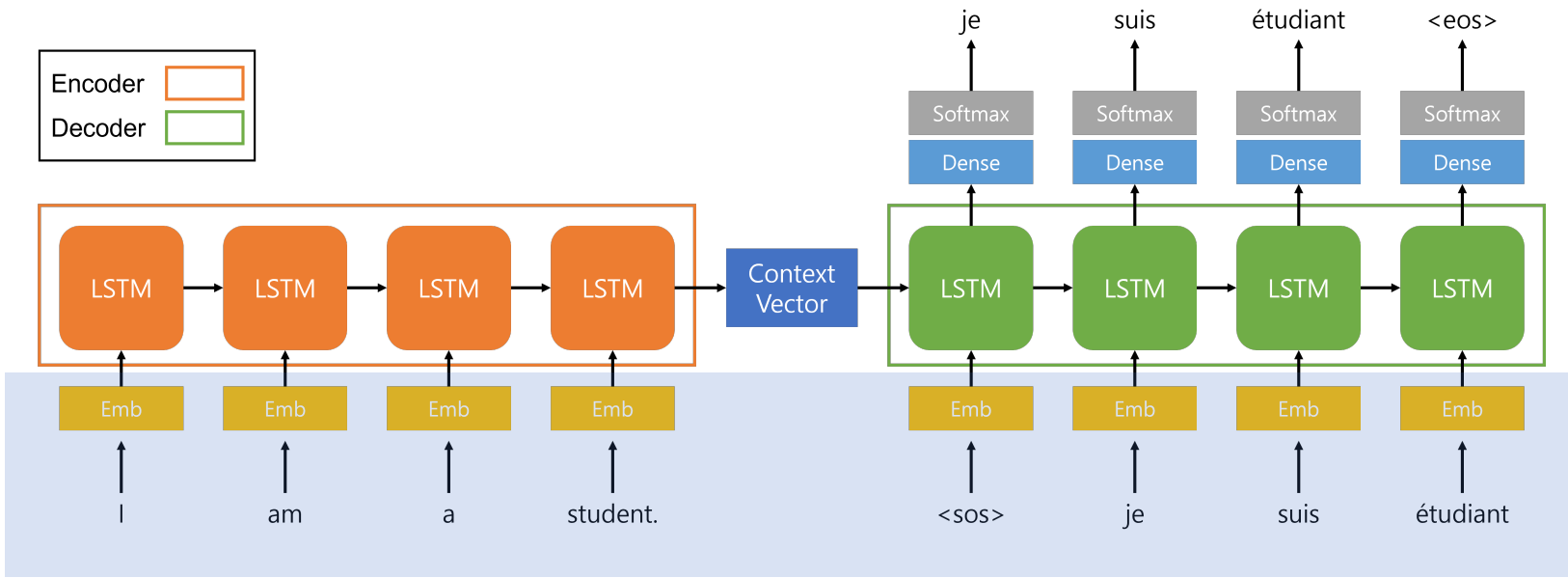
v : hidden state of input sequence
y : target 문장
P(y|x) : 우리가 찾고자 하는 번역 모델

TRAIN AND TEST

The model

Embedding

: Source/Target 문장을 임베딩 → 각각 Encoder/Decoder의 input으로 활용

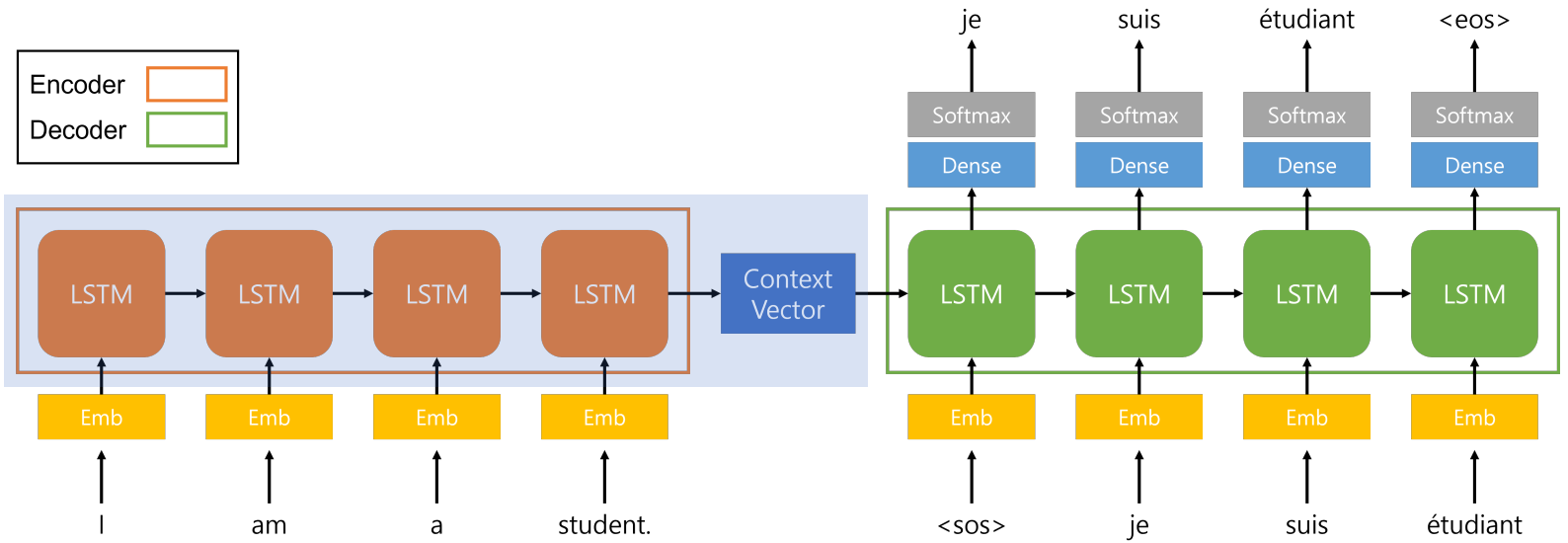


The model

Encoder

: 임베딩된 단어와 이전시점의 hidden state를 LSTM으로 연산 -> hidden state update

: 마지막 hidden state = context vector



TRAIN AND TEST

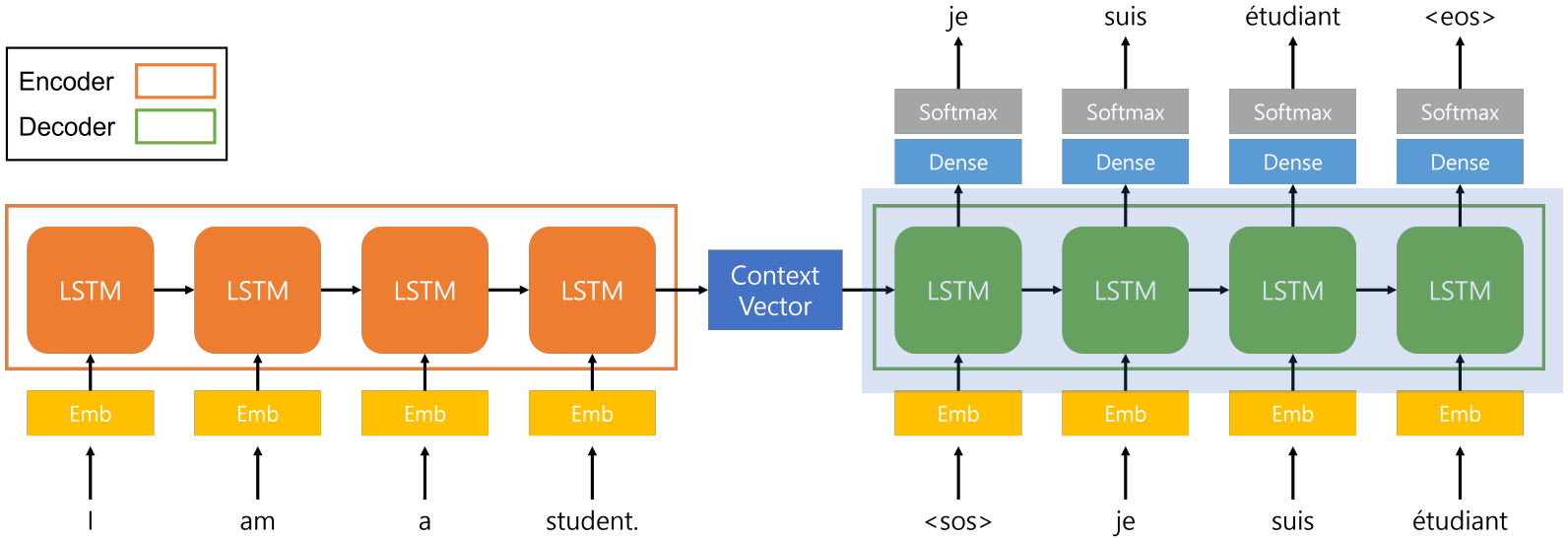
The model

Decoder

: 임베딩된 target sentence의 단어와 이전 시점의 hidden state를 LSTM으로 연산

-> hidden state update & 다음에 나올 확률이 높은 단어 예측

: training 단계에서만 target sentence 임베딩 필요

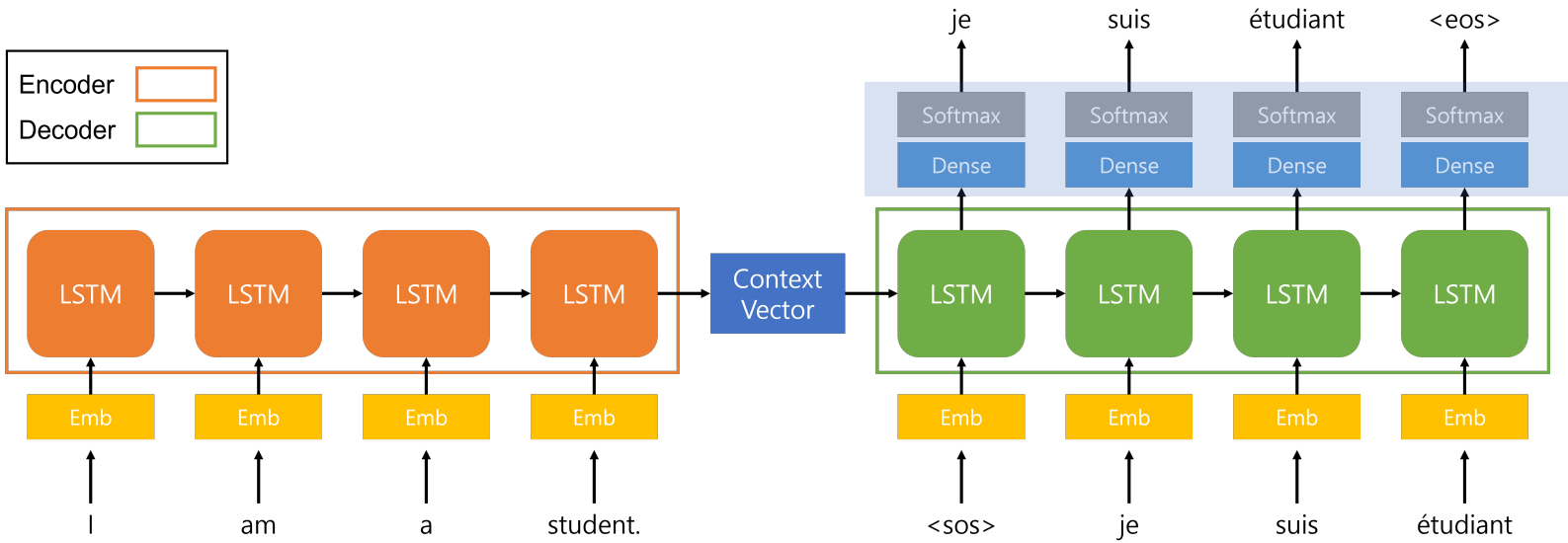


The model

Dense & Softmax Layer

: Decoder에서 LSTM cell의 output을 dense layer에 태우고 softmax에 태워 각 단어가 나올 확률을 계산

: 각 토큰이 나올 확률값이 output, output vector에서 가장 확률값이 큰 토큰이 최종 output으로 나옴

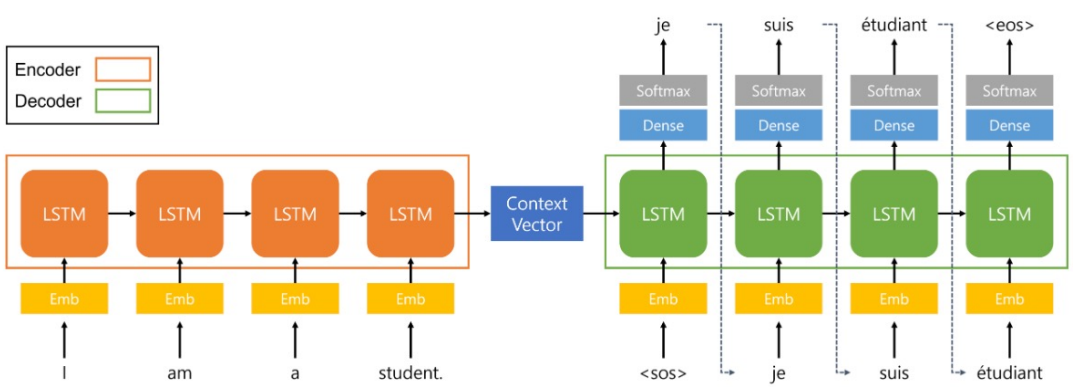


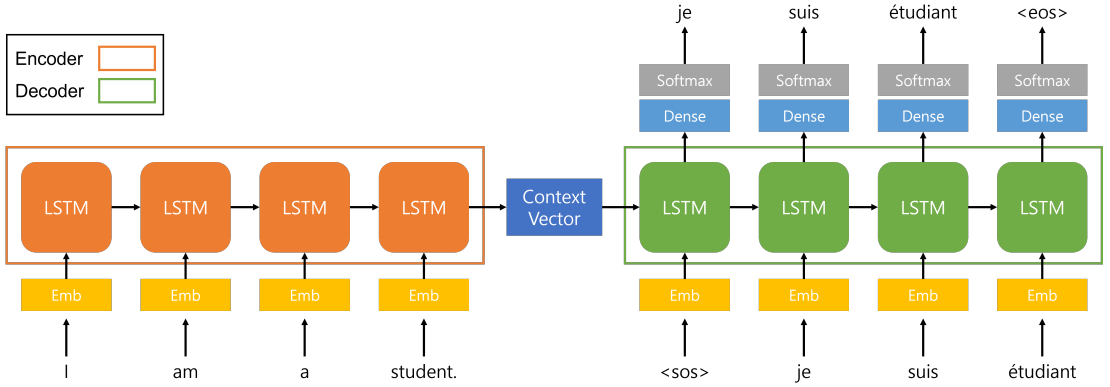
Training

Teacher Forcing

: target source (Ground Truth) 를 디코더의 다음 입력으로 넣어주는 방법

: 이전 시점의 decoder의 출력 단어를 다시 decoder의 입력값으로 넣으면 학습이 잘 진행되지 않기 때문에 사용





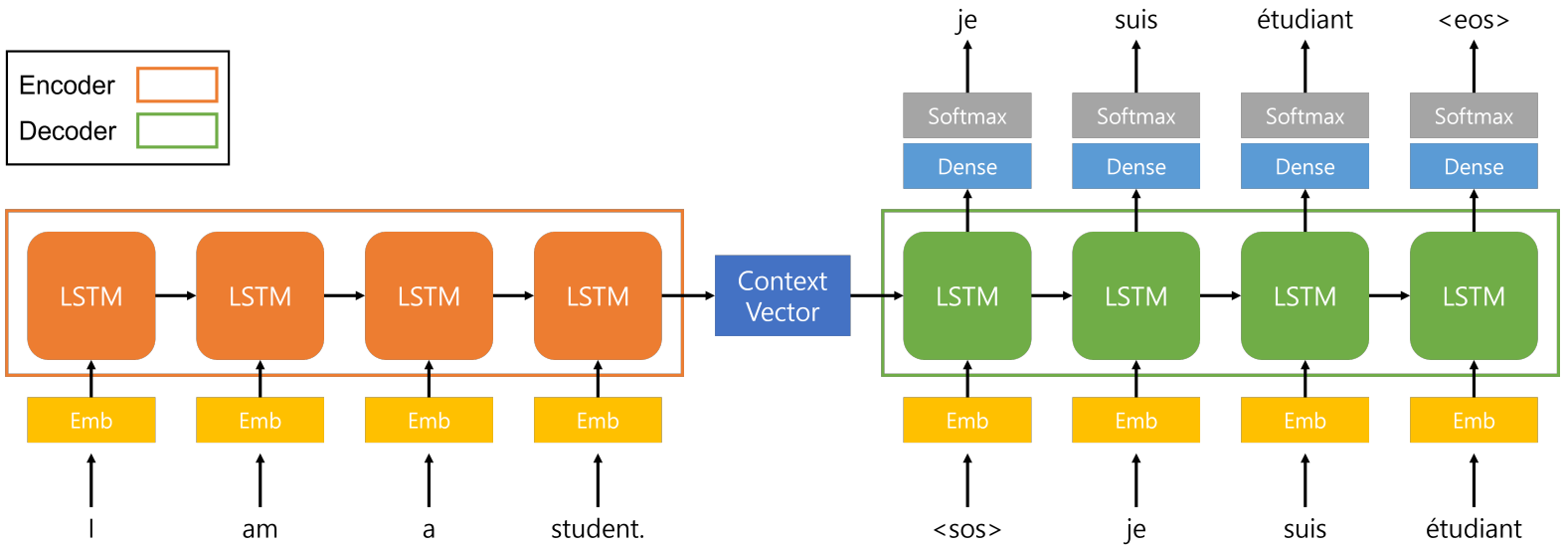
TRAIN AND TEST

Training

Reversing the Source Sentences

: source sentence의 순서를 거꾸로 사용하는 방법

: 성능향상의 이유) source sentence의 초기 단어와 target sentence의 초기 단어 사이의 거리가 가까워지기 때문이라고 추측



⇒ student a am I

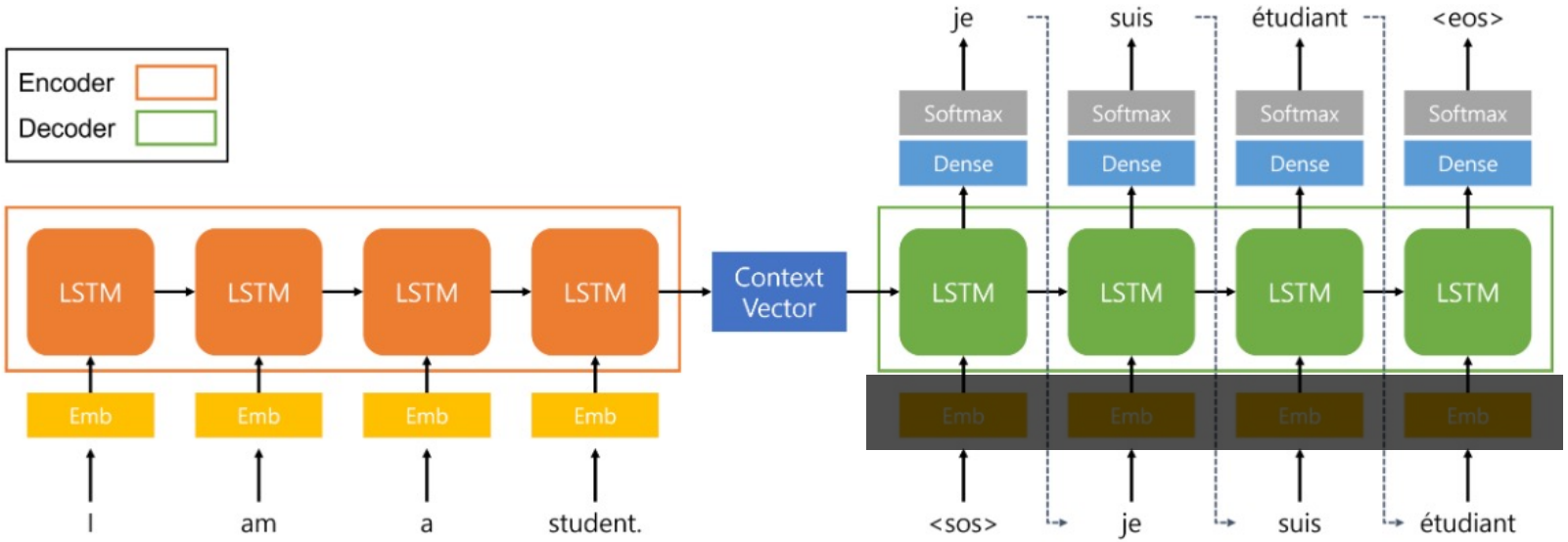
Inference

Inference

: Decoder가 예측한 값을 다음 input으로 활용하여 번역

: Decoder 단계에서의 정답 번역문이 없기에 embedding도 필요하지 않음

: SOS 토큰을 초기 Input으로 받아서 예측 진행



TRAIN AND TEST

Inference

Inference

: Decoder가 예측한 값을 다음 input으로 활용하여 번역

: Decoder 단계에서의 정답 번역문이 없기에 embedding도 필요하지 않음

: SOS 토큰을 초기 Input으로 받아서 예측 진행

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

- training 과정 서의 목표 = source sentence가 주어졌을 때 올바른 번역 T가 나올 log prob. 를 maximize 하는 것

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

- training 이 끝나면 가장 그럴듯한 번역을 생성

-> simple left-to-right beam search decoder를 사용

TRAIN AND TEST

Inference

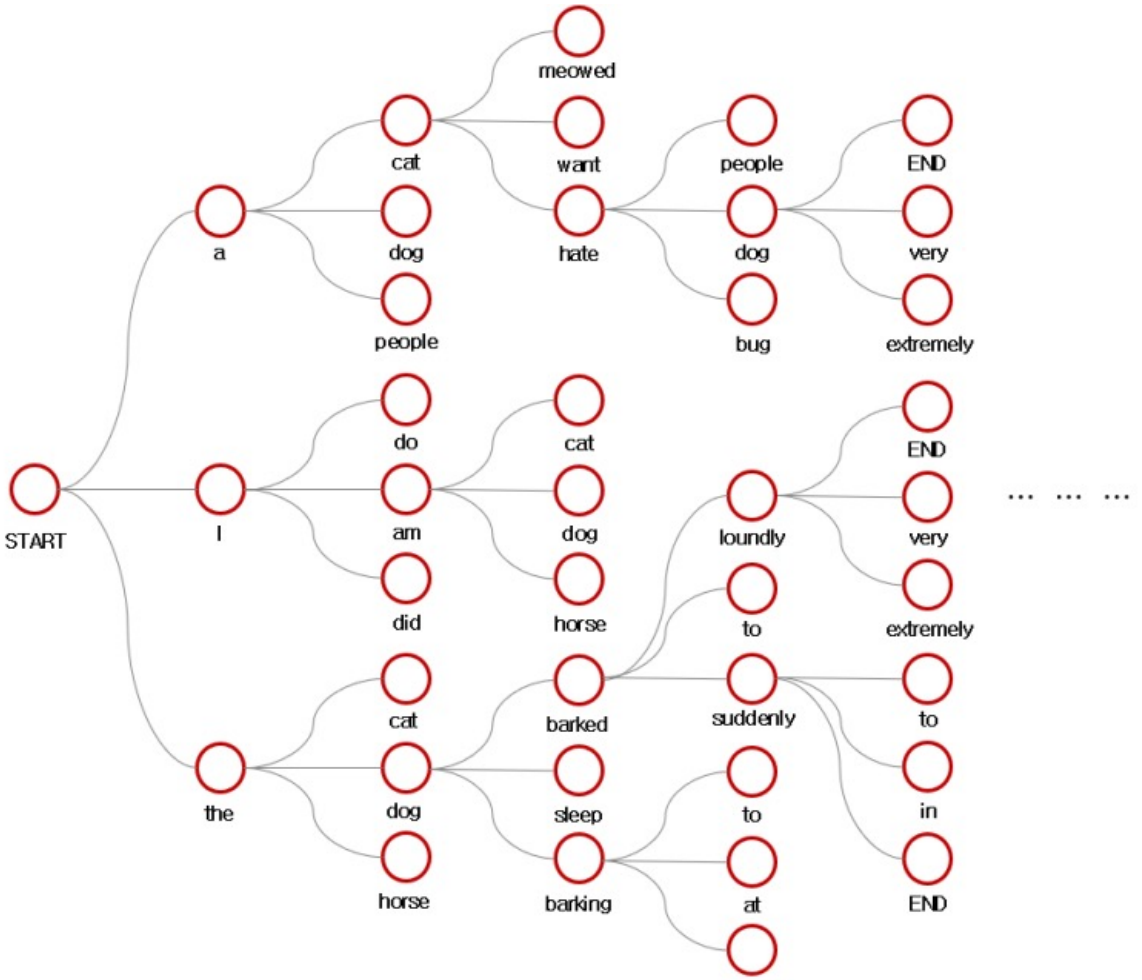
Beam Search

: 각 step에서 탐색할 영역 k개를 가장 확률이 높은 것들로 유지

: k가 클수록 넓은 영역을 탐색 -> 더 좋은 target sequence 생성

: 길이로 normalize -> 문장이 길수록 score 낮아지는 것 방지

$$score(y_1, \dots, y_t) = \frac{1}{t} \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$$



TRAIN AND TEST

Experimental Results & Conclusion

Experimental Results

BLEU score

: BLEU score로 번역의 질 계산

$$BLEU = min(1, \frac{output\ length(\text{예측 문장})}{reference\ length(\text{실제 문장})})(\prod_{i=1}^{4} precision_i)^{\frac{1}{4}}$$

같은 단어가 연속으로 나올 때 과적합 되는 것을 보정 해준 후
n-gram 을 통해서 순서쌍들이 얼마나 겹치는지 측정

문장 길이에 대한 과적합 보정

https://donghwa-kim.github.io/BLEU.html

Experimental Results

Results

Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

앙상블한 결과가 baseline (SMT) 보다 좋은 성능을 보임

TRAIN AND TEST

Experimental Results

Results

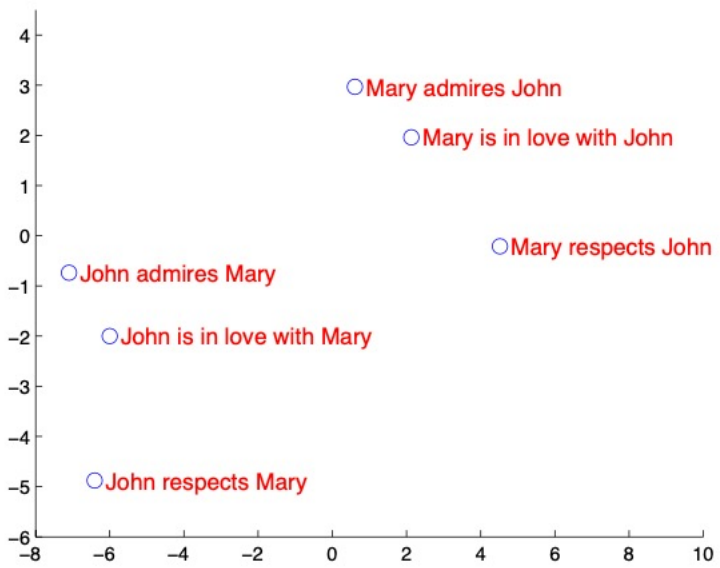
Method	test BLEU score (ntst14)
Baseline System [29]	33.30
Cho et al. [5]	34.54
Best WMT'14 result [9]	**37.0**
Rescoring the baseline 1000-best with a single forward LSTM	35.61
Rescoring the baseline 1000-best with a single reversed LSTM	35.85
Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs	**36.5**
Oracle Rescoring of the Baseline 1000-best lists	~45

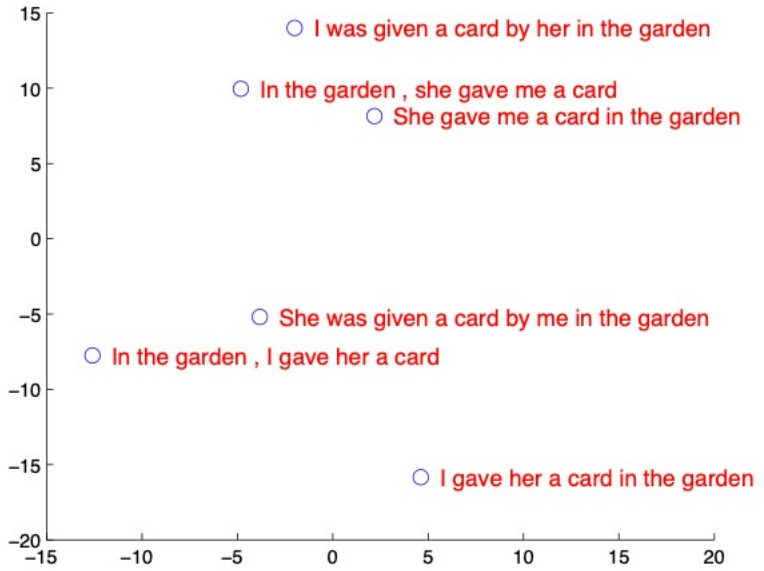
앙상블 & Reversed LSTM을 가지고 했을 때 좋은 성능을 보임.

TRAIN AND TEST

Experimental Results

Results





Encoder에 임베딩 된 결과를 2차원 PCA로 축소해 시각화 한 그래프

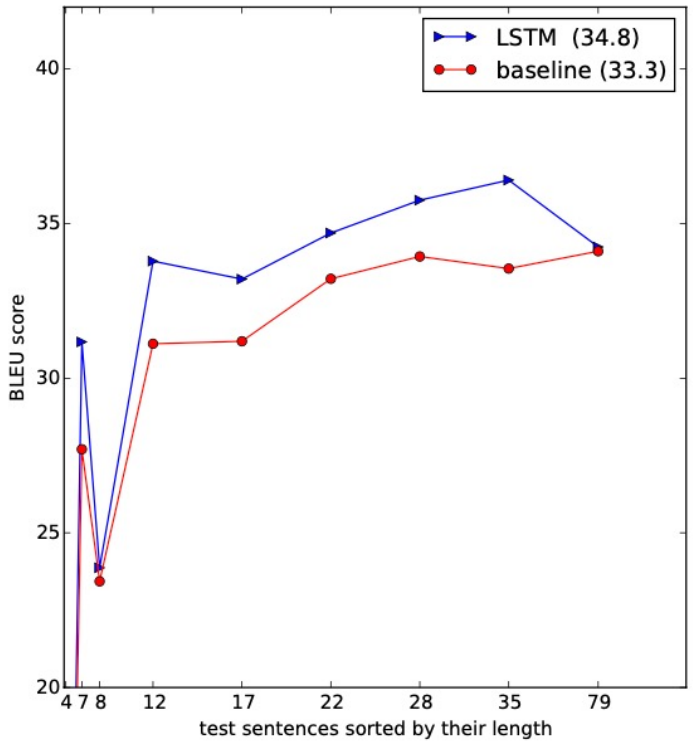
구들이 문장 의미에 따라서 clustered되어 있어 context vector가 잘 만들어졌음을 알 수 있음

단어 순서의 변화에는 민감하나 능동과 수동에는 큰 영향을 받지 않음

TRAIN AND TEST

Experimental Results

Results



35개의 단어로 구성된 문장 전까지를 보면 LSTM이 baseline code보다 degrade된 것을 찾을 수 없음

-> 긴 문장에서도 BLEU score가 향상되는 양상을 확인 가능

TRAIN AND TEST

Conclusion

Main Contribution

1) LSTM Encoder - Decoder 사용 : long time dependency 해결
2) Large Deep LSTM 사용 : 4개의 layer
3) Reversed Order Input Sequence : 긴 문장을 올바르게 번역하는 LSTM 성능

Attention (Bahdanau)

Contents

1. Background

I. Encoder - Decoder 구조의 문제점

II. RNN based Encoder - Decoder

III. Attention

2. The model

I. Decoder

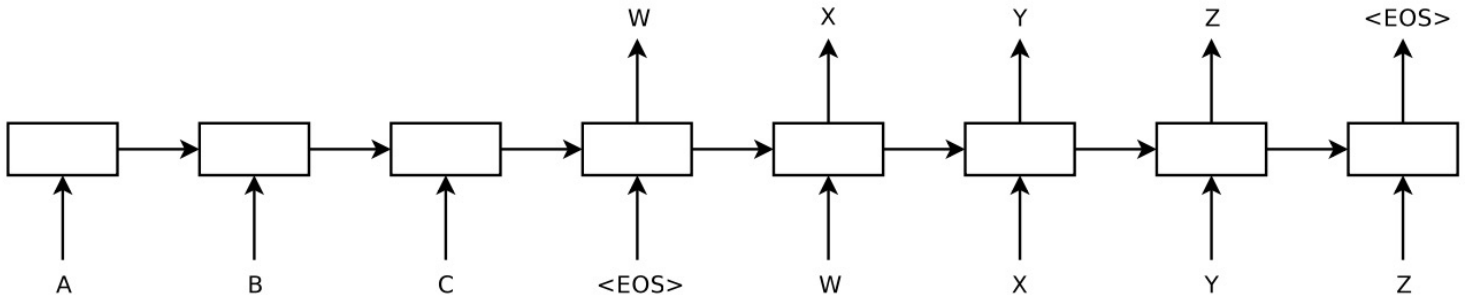
TRAIN AND TEST

Background

Encoder-Decoder 구조의 문제점

: source sentence의 모든 가능한 정보를 **고정된 길이의 vector로 표현** -> 긴 문장을 다루기 어려움, 성능 향상에 방해

: LSTM도 Long term dependency 문제를 완벽히 해결할 수 없음



TRAIN AND TEST

Background

Encoder-Decoder 구조의 문제점

: source sentence의 모든 가능한 정보를 **고정된 길이의 vector로 표현** -> 긴 문장을 다루기 어려움, 성능 향상에 방해

: LSTM도 Long term dependency 문제를 완벽히 해결할 수 없음

Core Idea

- align과 translate를 jointly하게 학습하는 encoder - decoder model
- 모델이 자동적으로 target word와 관련된 source sentence의 부분만 search하게 함
- input 문장을 벡터 시퀀스로 인코딩 -> 디코딩 시 벡터 시퀀스에서 필요한 부분만 attention해 사용

Background

RNN based Encoder - Decoder

: Encoder - Decoder 구조에서 RNN을 활용

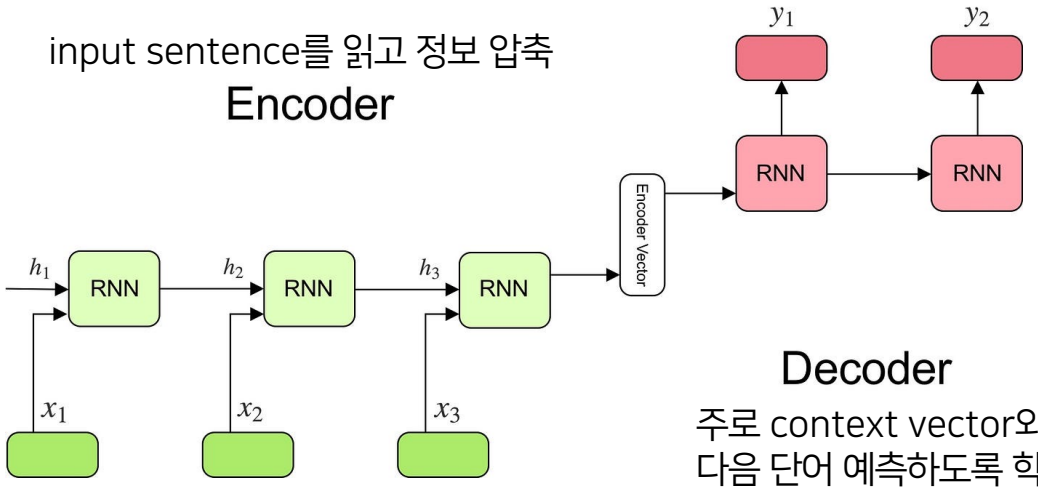
$$h_t = f\left(x_t, h_{t-1}\right)$$

$$c = q\left(\{h_1, \cdots, h_{T_x}\}\right)$$

$$q\left(\{h_1, \cdots, h_T\}\right) = h_T$$

input sentence를 읽고 정보 압축

Encoder



$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c)$$

Decoder

주로 context vector와 모든 이전에 예측된 단어를 이용해
다음 단어 예측하도록 학습

ht : t에서의 hidden state
c : hidden state에서 만들어진 벡터

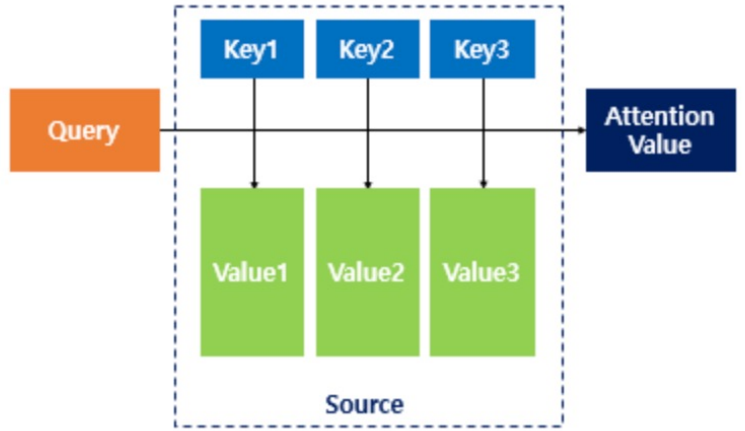
TRAIN AND TEST

Background

Attention

1. 주어진 쿼리에 대해 모든 키와의 유사도를 구함
2. 유사도를 키와 맵핑된 각각의 value에 반영
3. 유사도가 반영된 value를 더해서 return

= Attention Value



$$Attention(Q, K, V) = Attention\ Value$$

Q (쿼리) : **t - 1 시점의** decoder cell에서의 hidden state

K (키) : 모든 시점의 encoder cell의 hidden states

V (밸류) : 모든 시점의 encoder cell의 hidden states

ht : t에서의 hidden state
c : hidden state에서 만들어진 벡터

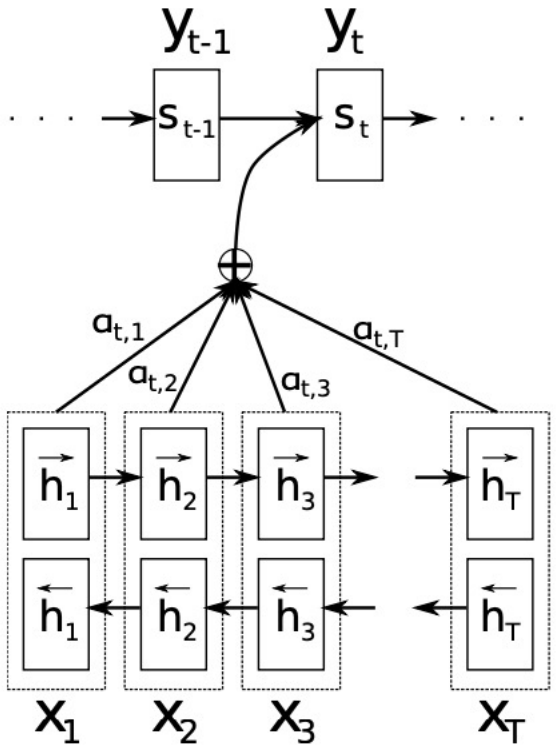
The model : Decoder

Decoder

: attention 메커니즘이 활용되는 부분

: source 문장에서 어느 부분에 집중을 해야할지 결정함

: source 문장이 모든 정보를 고정된 길이의 벡터로 encoding을 하지 않아도 되게 도와줌



si : time I 에서의 decoder의 hidden state

The model : Decoder

Decoder

i=t로 바꾸어 표현
현 시점 t를 구하기 위한 alignment 모델

$$e_{t-1,j} = a(s_{t-1}, h_j)$$

$$a(s_{t-1}, H) = W_a^T tanh(W_b s_{t-1} + W_c H)$$

e : Attention score

: 기존의 모델과 다르게 하나의 변수로 alignment을 제시

: input의 j번째 위치와 output의 t-1번째 위치의 match정도를 scoring

: s_{t-1}와 h_j의 유사정도를 a를 이용해 나타내는 attention score

a : Alignment model = Score Function

: a = fcnn 이용해 기본 모델과 동시에 학습됨

: W = 학습 가능한 가중치 행렬

: 다양한 alignment 모델이 제시될 수 있음.

위의 수식은 본 논문의 모델

TRAIN AND TEST

The model : Decoder

Decoder

$$\alpha_{t-1,j} = \frac{exp(e_{t-1}, j)}{\Sigma_{k=1}^{Tx} exp(e_{t-1}, k)}$$

alpha

: softmax함수로 attention score e를 attention weight로 변경

TRAIN AND TEST

The model : Decoder

Decoder

$$c_t = \Sigma_{j=1}^{Tx} \alpha_{t-1} h_j$$

Context vector

: attention weight와 인코더 hidden state hj의 weighted sum 형태

: **각 hidden state hj에 대해 jth input과 (t-1)th output의 관련성에 대한 내용**

=> 번역이 필요한 output 문장에 대해 input 문장의 어느 부분에 더 집중해서 보아야할지 계산

The model : Decoder

Decoder

$$p(y_t|y_1, \ldots, y_{t-1}, X) = g(y_{t-1}, s_t, c_t)$$

$$s_t = f(s_{t-1}, y_i, c_i), \quad s_t = Decoder(s_{t-1}, y_t, c_t)$$

본 논문의 모델에서 새롭게 정의한 조건부 확률

: source 문장 전체(x)와 이전시점까지의 target 단어들이 주어졌을 때 현 시점 t의 target 단어가 나올 확률

= nonlinear function g에 input으로

1) 이전시점의 target 단어, 2) 현재시점의 hidden state, 3) **타겟 단어 yt에 condition된 context vector ct를 넣은 값**

cf) $p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c),$

si : time I 에서의 decoder의 hidden state

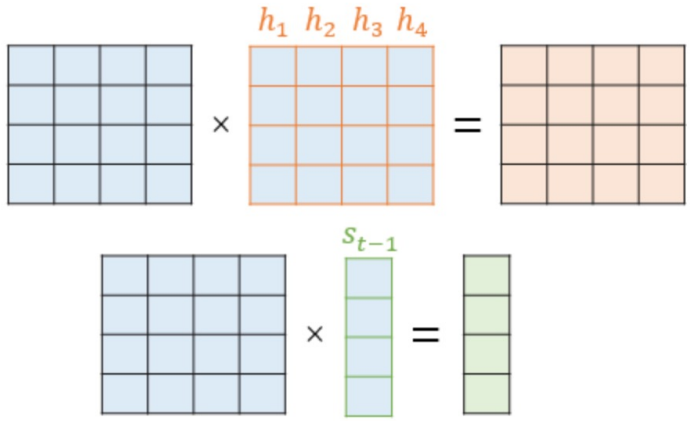
TRAIN AND TEST

The model：Decoder

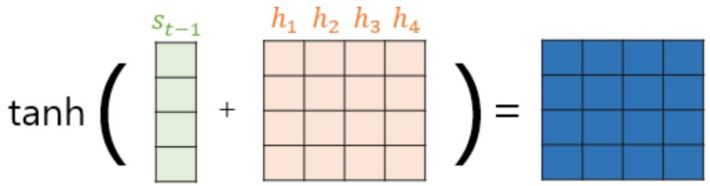
Decoder

$$a(s_{t-1}, H) = W_a^T tanh(W_b s_{t-1} + W_c H)$$

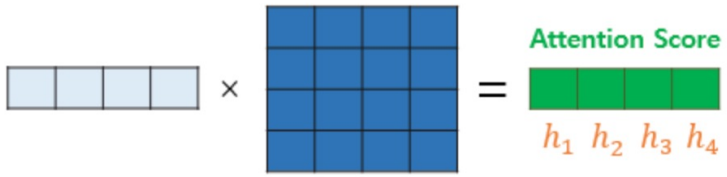
$W_c H$ $W_b s_{t-1}$



$tanh(W_b s_{t-1} + W_c H)$



$W_a^T tanh(W_b s_{t-1} + W_c H)$



TRAIN AND TEST

The model : Decoder

Decoder

$$\alpha_{t-1,j} = \frac{exp(e_{t-1}, j)}{\Sigma_{k=1}^{Tx} exp(e_{t-1}, k)}$$

소프트맥스 함수 통해 어텐션 분포 구함

$$c_t = \Sigma_{j=1}^{Tx} \alpha_{t-1} h_j$$

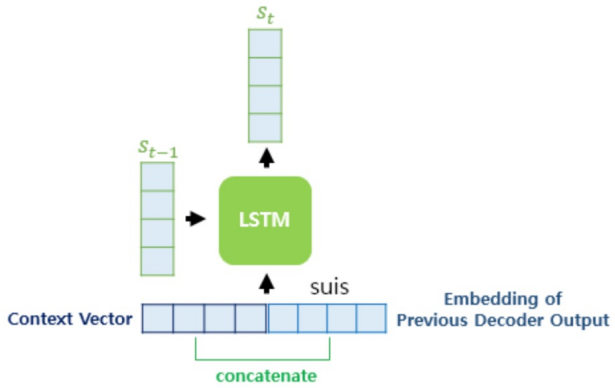
각 인코더의 어텐션 가중치&은닉상태를 가중합 -> 어텐션 값 구함

$$s_t = Decoder(s_{t-1}, y_t, c_t)$$

c와 디코더의 이전 output 연결한 벡터와 이전 시점의 은닉상태를 input -> st 구함

Attention Distn, Attention Weight

가중합 결과 = context vector



st : 출력층으로 전달 -> 현시점의 output 구함

TRAIN AND TEST

참조

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

https://hong-zone17.tistory.com/76

https://heekangpark.github.io/nlp/attention

https://velog.io/@bo-lim/Beam-Search

https://www.youtube.com/watch?v=PipiRRL50p8&t=183s

https://blog.naver.com/sooftware/221809101199

https://youtu.be/0lgWzluKq1k?si=TKLRYvnXjPU8bM2P

https://donghwa-kim.github.io/BLEU.html

https://velog.io/@nkw011/seq-to-seq

https://it-ist.tistory.com/27

https://blog.naver.com/bcj1210/221581535580

https://yjjo.tistory.com/46

https://tigris-data-science.tistory.com/entry/DL-%EC%89%BD%EA%B2%8C-%ED%92%80%EC%96%B4%EC%93%B4-Attention-Mechanism-1-Bahdanau-Attention

https://wikidocs.net/73161

TRAIN AND TEST

TRAIN AND TEST