



T R A I N   A N D   T E S T

[CS224N] Lecture 8 -Translation, Seq2Seq, Attention

# INDEX

## 1. Machine Translation

[기계 번역]

- Brief History of Machine Translation

## 2. Neural Machine Translation

## 3. Evaluation

## 4. Attention method

+ 부록

# 1. Machine Translation

# Machine Translation의 간단한 역사

## Machine Translation 이란?

- Source language 의 문장을 Target language 의 문장으로 번역하는 것

프랑스어 x: *L'homme est né libre, et partout il est dans les fers*



영어 y: *Man is born free, but everywhere he is in chains*



한글 y: 사람은 자유롭게 태어났지만 어디에서나 사슬에 묶여 있습니다.

# Machine Translation의 간단한 역사

[1950년대] “냉전 당시 러시아어의 영어 번역”

- 간단한 매핑 작업 (러시아어 단어와 영어 단어를 단순 매핑)

[1990년대] “통계적 기계 번역”  
(Statistical Machine Translation)

- 베이즈 공식을 이용해, 문제를 분리하여 해결

[2010년대] “신경망 기반 기계 번역”  
(Neural Machine Translation)

- Single neural network를 활용한 기계 번역

# Machine Translation의 간단한 역사

[1990년대] “통계적 기계 번역”  
(Statistical Machine Translation)

[2010년대] “신경망 기반 기계 번역”  
(Neural Machine Translation)

- **Motivation** ✓ 통계적 모델을 활용해, 번역을 하는 것을 목표로 한다.

[예] 프랑스어 문장  $x$  가 주어졌을 때, 가장 “좋은” 영어 문장  $y$ 를 찾는다.

베이즈 정리 

$$\begin{aligned} & \operatorname{argmax}_y P(y|x) \\ &= \operatorname{argmax}_y P(x|y)P(y) \end{aligned}$$

: 두 가지 문제로 정의할 수 있다.

[Translation Model] [Language Model]

Fidelity

Fluency

- **Details [Language Model]**      프랑스어 문장 x 가 주어졌을 때, 가장 “좋은” 영어 문장 y를 찾는다.

$$P(y)$$

- ✓ 우리가 지난번 시간에 계속 공부한 LM
- ✓ Target language 의 데이터가 필요하다.
- ✓ 문장의 확률을 예측하는 모델

- **Details [Translation Model]** 프랑스어 문장  $x$  가 주어졌을 때, 가장 “좋은” 영어 문장  $y$ 를 찾는다.

$$P(x|y)$$

- ✓ 대량의 Parallel data [병렬 말뭉치]이 필요하다. (쌍으로 이루어진 영어-프랑스어 문장)
- ✓ 학습 방식:

$$P(x|y) = P(x, a|y) \text{ 으로 나누어 계산한다.}$$

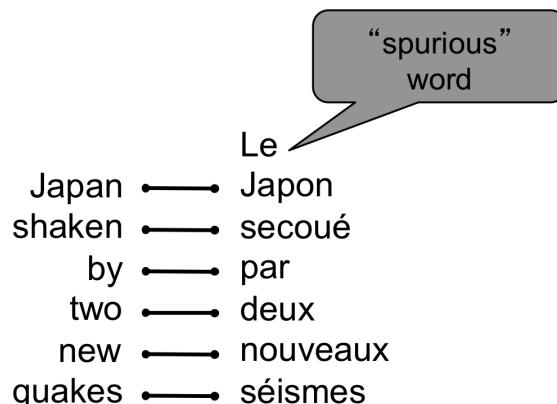
- ✓  $a$  = alignment이다. (word level correspondence between French & English)  
: 즉, 대응되는 단어 관계이다.

- Results [Translation Model] 프랑스어 문장 x 가 주어졌을 때, 가장 “좋은” 영어 문장 y를 찾는다.

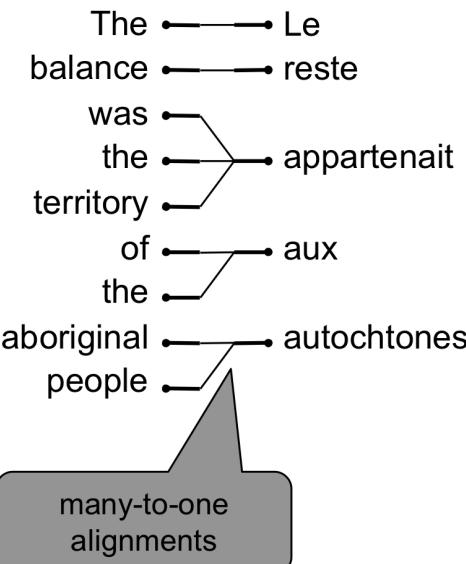
$$P(x|y) = P(x, a|y)$$

a: [allignment] 대응 관계

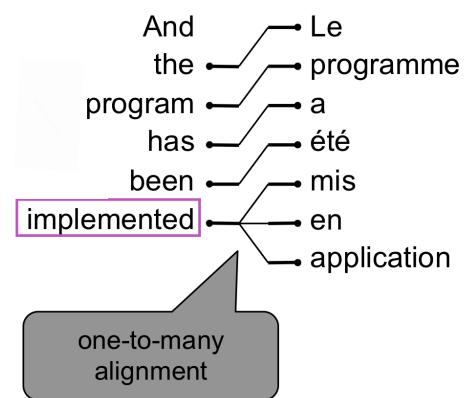
✓ 대응 되는 말이 없음



✓ Many-to-one



✓ one-to-many

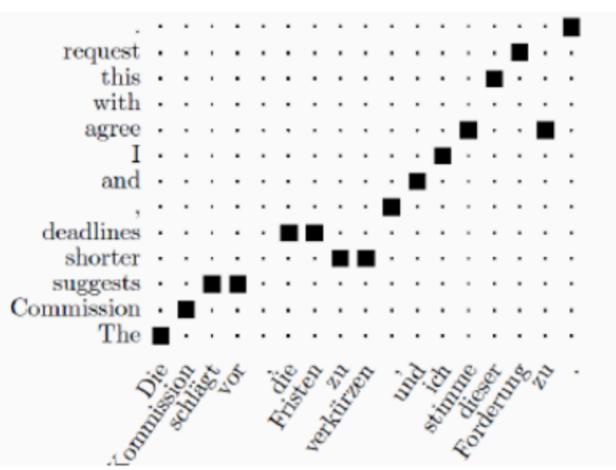


✓ 이외에도 다양한 관계가 존재한다

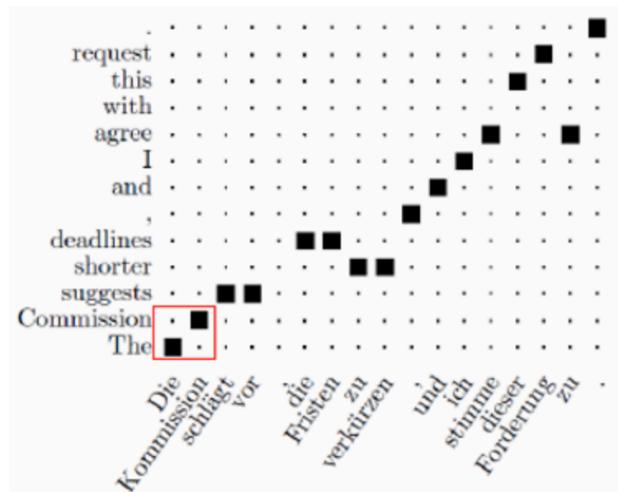
- **Results [Translation Model]** 프랑스어 문장 x 가 주어졌을 때, 가장 “좋은” 영어 문장 y를 찾는다.

$$P(x|y) = P(x, a|y)$$

- ✓ 병렬의 말뭉치를 통해, alignment를 추출한다.
- ✓ 이 alignment로 word/phrase base smt를 이해할 수 있다.

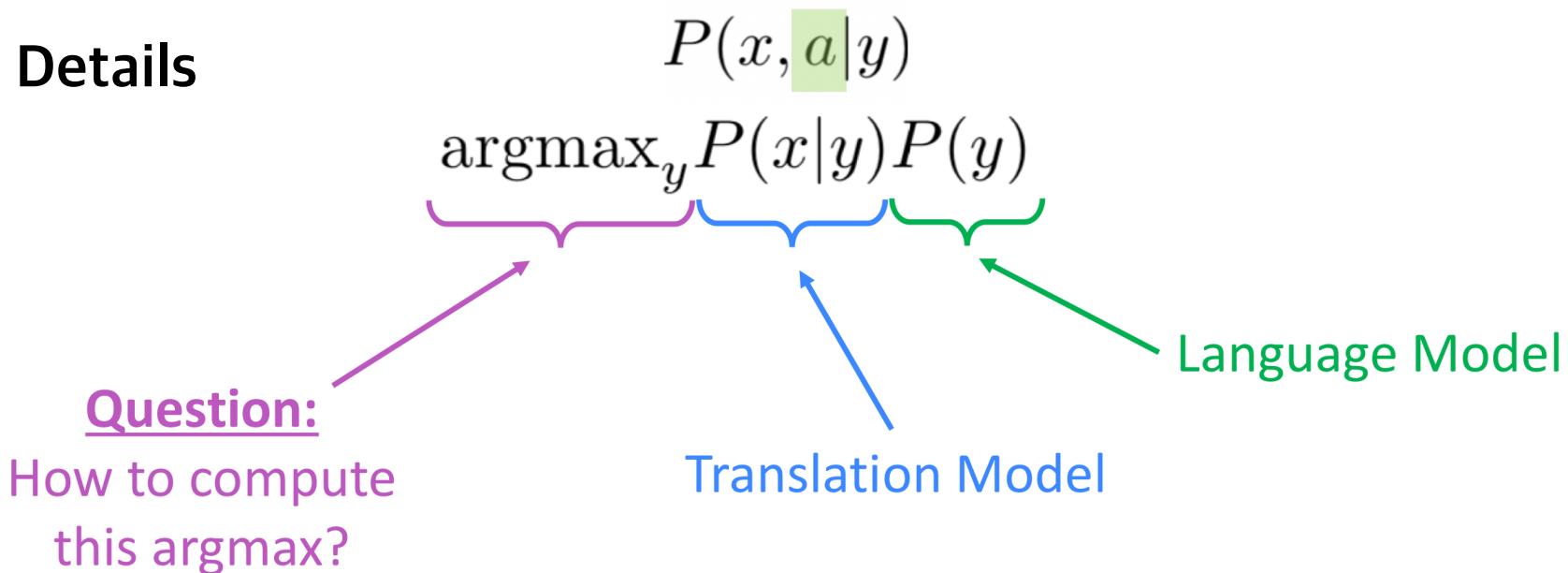


Word Base smt



Phrase Base smt

- Details



- ✓ 모든  $y$  를 계산할 수 없음
- ✓ 그래서, heuristic search algorithm 을 이용하여 낮은 확률의 문장을 제거하며 best 를 찾는다.
- ✓ Decoding process

- Decoding Process

프랑스어 문장  $x$  가 주어졌을 때, 가장 “좋은” 영어 문장  $y$ 를 찾는다.

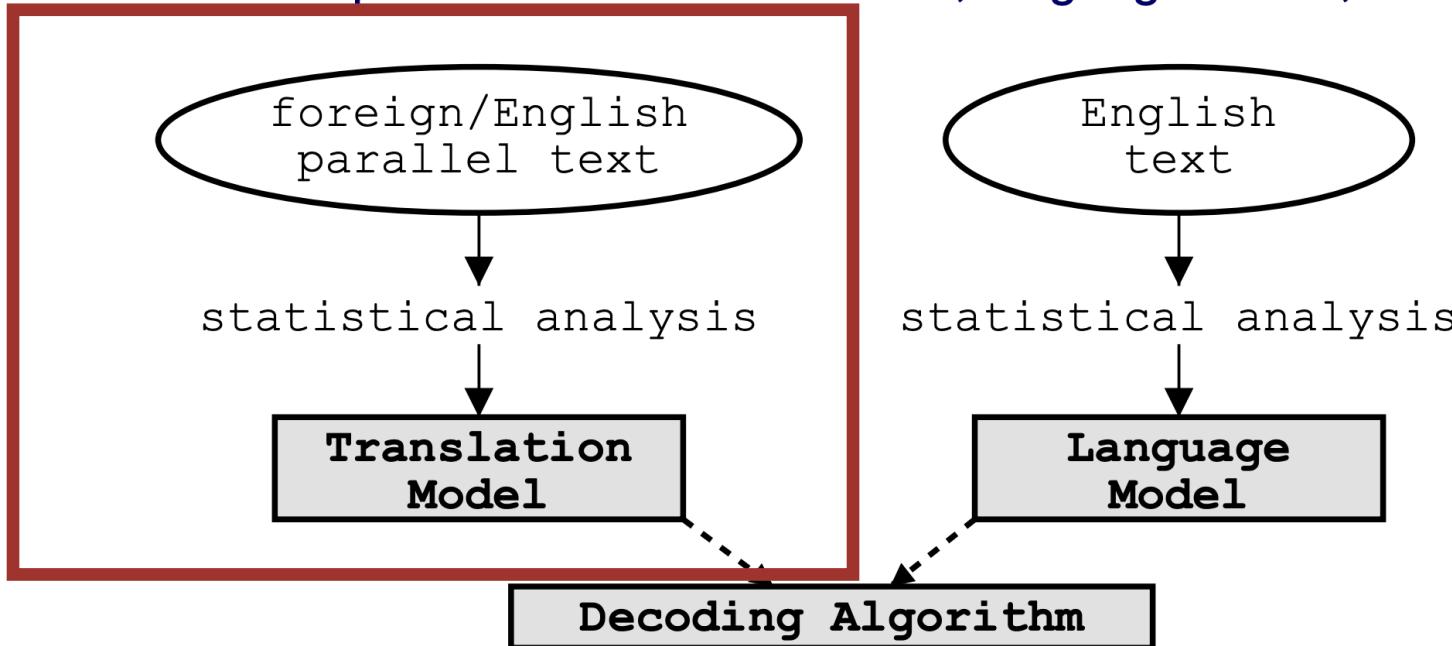
Task of decoding: find the translation  $e_{best}$  with highest probability

[Translation Model]  $e_{best} = \operatorname{argmax}_y P(x|y)$

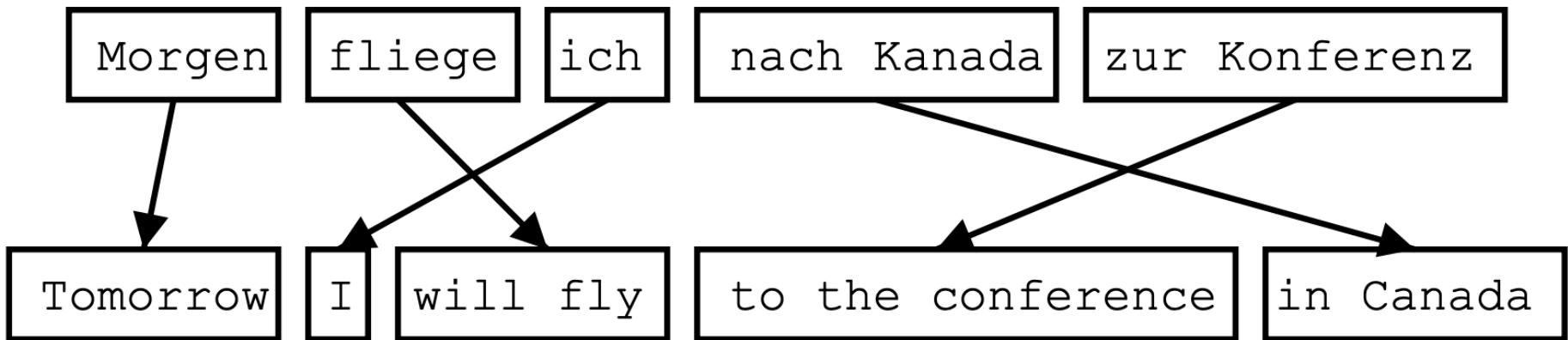
- ✓ 과정1 : Translation Option 살피기
- ✓ 과정2: Precompute Translation Options (alignment table 계산)
- ✓ 과정3: hypothesis expansion
- ✓ 과정4: Find Best option

- Example

- Components: Translation model, language model, decoder



- **Example** 독일어 문장 x 가 주어졌을 때, 가장 “좋은” 영어 문장 y를 찾는다.



- **Example** 독일어 문장  $x$  가 주어졌을 때, 가장 “좋은” 영어 문장  $y$ 를 찾는다.

[Translation Model] : alignments 의 확률 계산

- Phrase Translations for “den Vorschlag”:

English	$\phi(e f)$	English	$\phi(e f)$
the proposal	0.6227	the suggestions	0.0114
's proposal	0.1068	the proposed	0.0114
a proposal	0.0341	the motion	0.0091
the idea	0.0250	the idea of	0.0091
this proposal	0.0227	the proposal ,	0.0068
proposal	0.0205	its proposal	0.0068
of the proposal	0.0159	it	0.0068
the proposals	0.0159	...	...

$$P(x, a|y)$$

$$\operatorname{argmax}_y P(x|y)P(y)$$

- **Example** 독일어 문장  $x$  가 주어졌을 때, 가장 “좋은” 영어 문장  $y$ 를 찾는다.

$$P(x, a|y)$$

$$\operatorname{argmax}_y P(x|y)P(y)$$

## Translation Option

Maria	no	dio	una	bofetada	a	la	bruja	verde
Mary	<u>not</u>	<u>give</u>	<u>a</u>	<u>slap</u>	<u>to</u>	<u>the</u>	<u>witch</u>	<u>green</u>
	<u>did not</u>			<u>a slap</u>	<u>by</u>		<u>green</u>	<u>witch</u>
	<u>no</u>		<u>slap</u>		<u>to the</u>			
	<u>did not give</u>				<u>to</u>			
					<u>the</u>			
				<u>slap</u>		<u>the witch</u>		

## 2

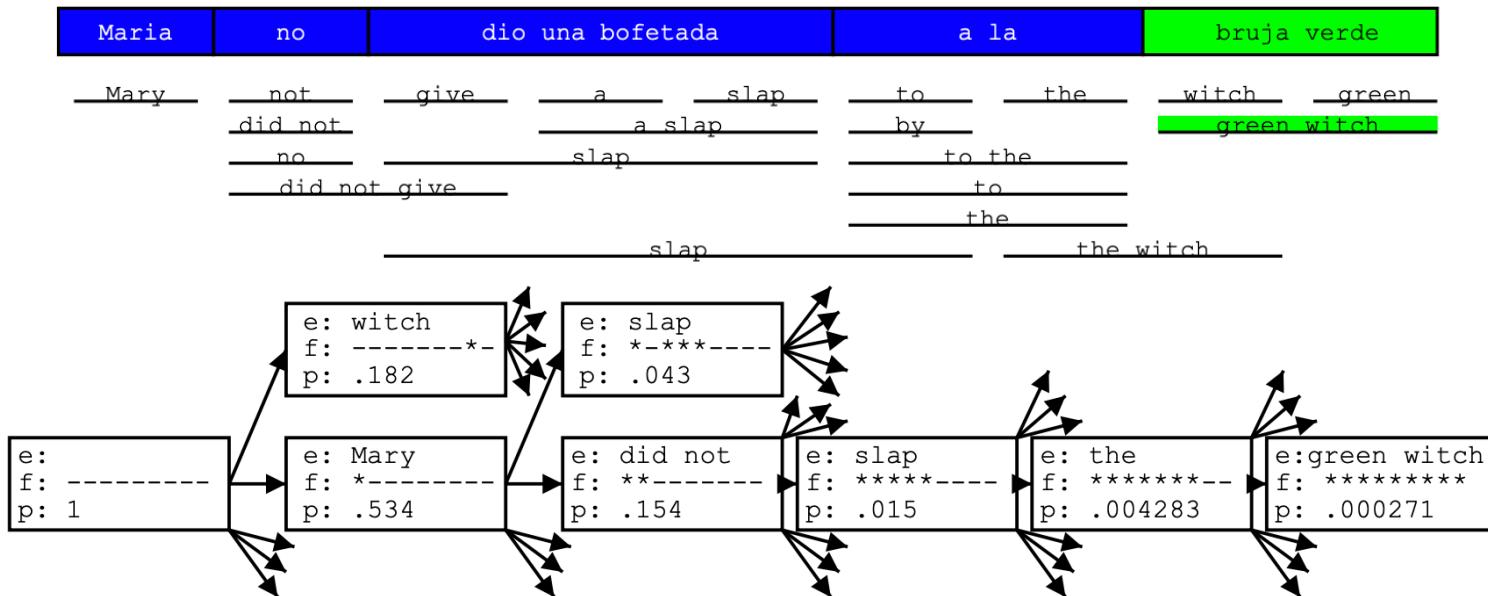
## Statistical Machine Translation

- **Example** 독일어 문장  $x$  가 주어졌을 때, 가장 “좋은” 영어 문장  $y$ 를 찾는다.

$$P(x, a|y)$$

$$\operatorname{argmax}_y P(x|y)P(y)$$

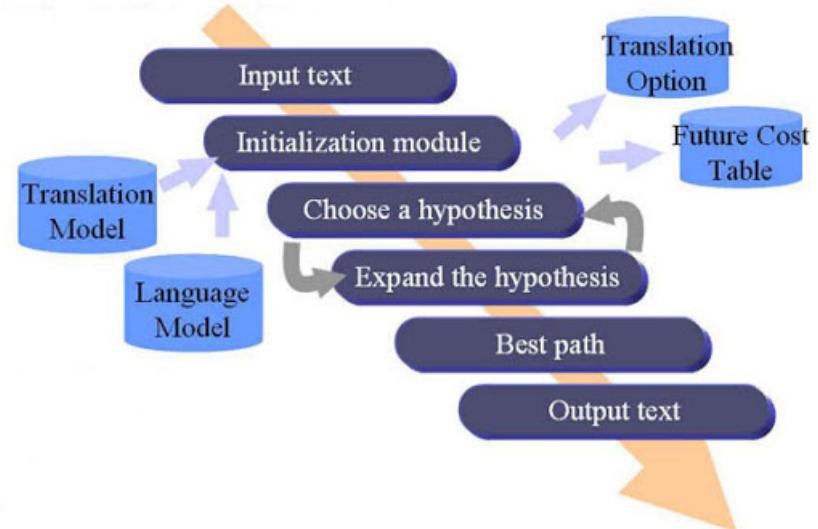
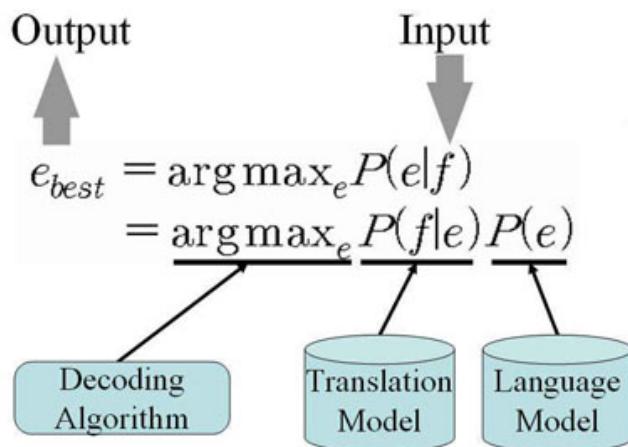
## Hypotheses Expansion



다양한 Hypotheses 탐색 후 가장 높은 스코어의 hypothesis 선정

- 단점

- ✓ 엄청 복잡한 시스템이다… 이해하기 힘듦…
- ✓ 서브 문제로 분할되며, 많은 feature engineering 이 필요하다.
- ✓ Store, maintain 에 많은 노력이 필요하다.



# 3

# Neural Machine Translation

[1990년대] “통계적 기계 번역”  
(Statistical Machine Translation)

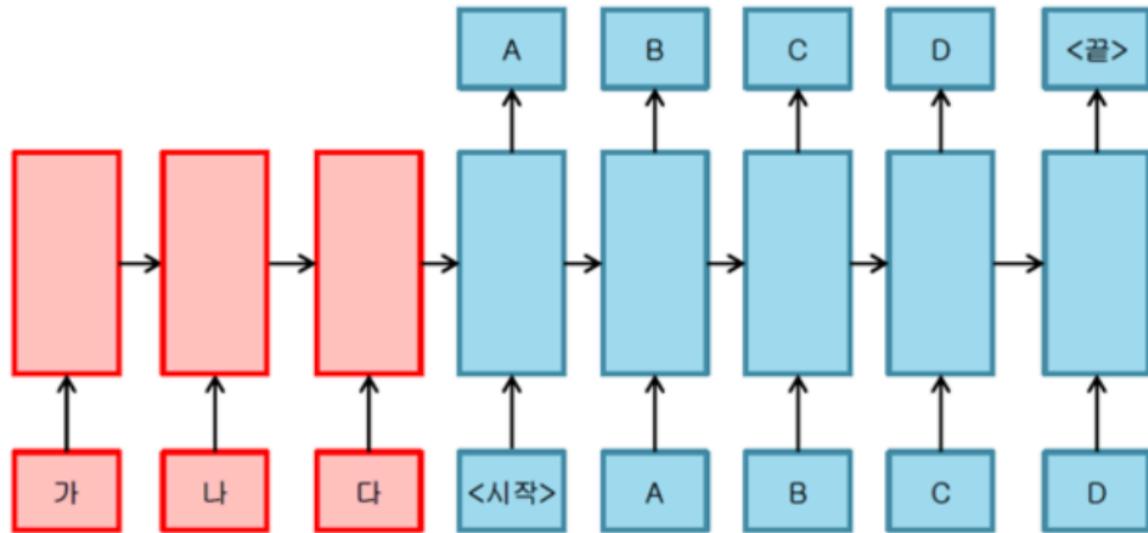
[2010년대] “신경망 기반 기계 번역”  
(Neural Machine Translation)



## 2. Neural Machine Translation

- 소개

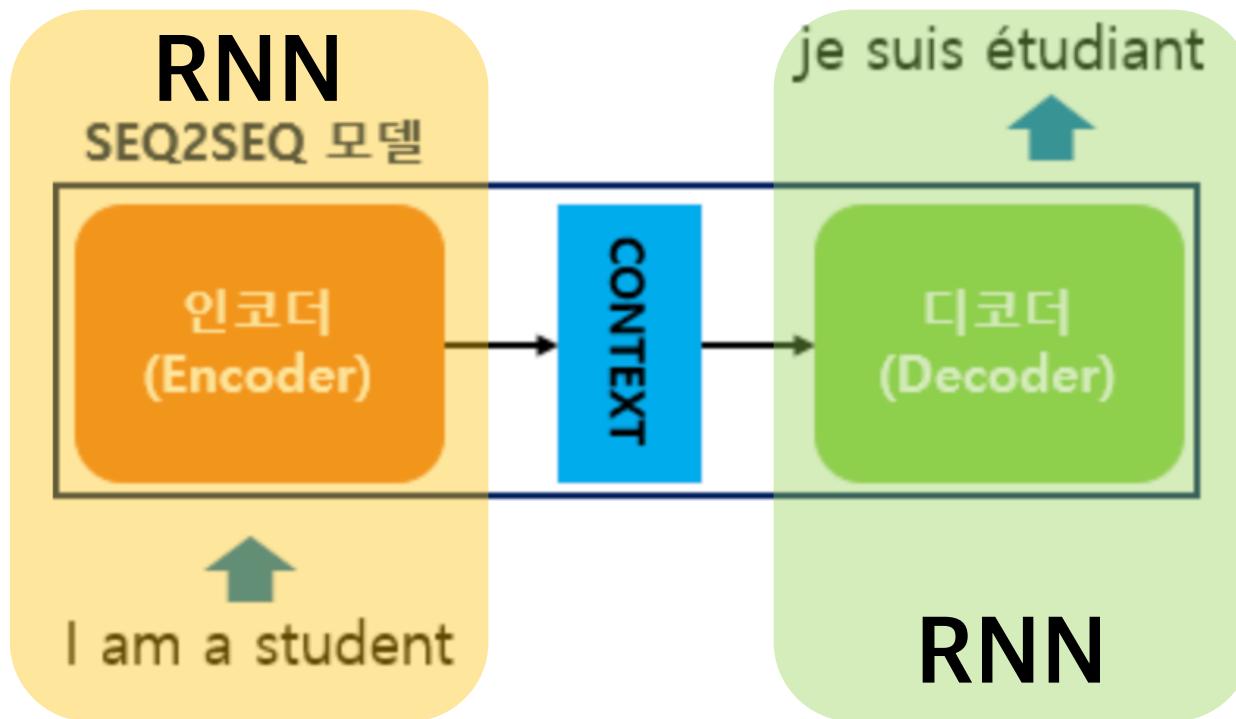
- ✓ Machine Translation을 Single Neural Network으로 할 수 있다!!!!
- ✓ 이 network의 구조는 sequence-to-sequence 라고 불린다.
- ✓ RNN 두개를 포함하는 구조이다 [RNN Encoder-Decoder 구조]



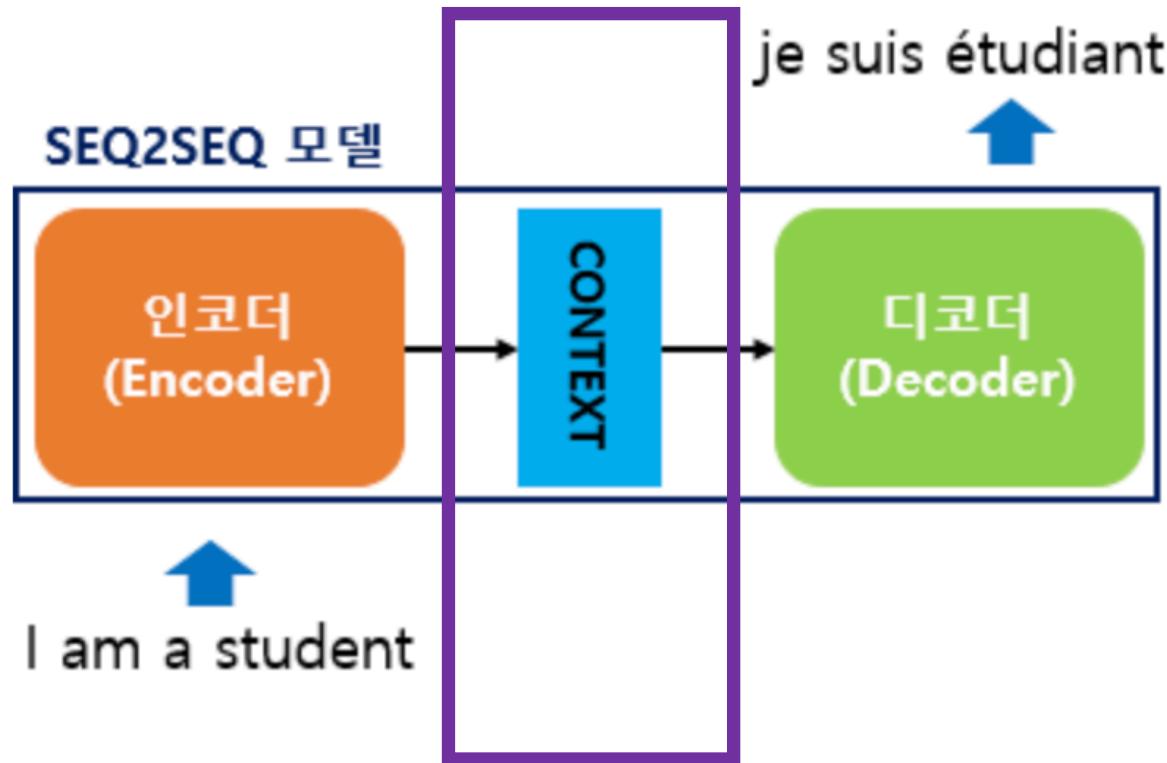
# 2

## NMT Model Explanation

- 모델 구성 [0] 그림 전체적인 구성



- 모델 구성 [0] 그림 전체적인 구성

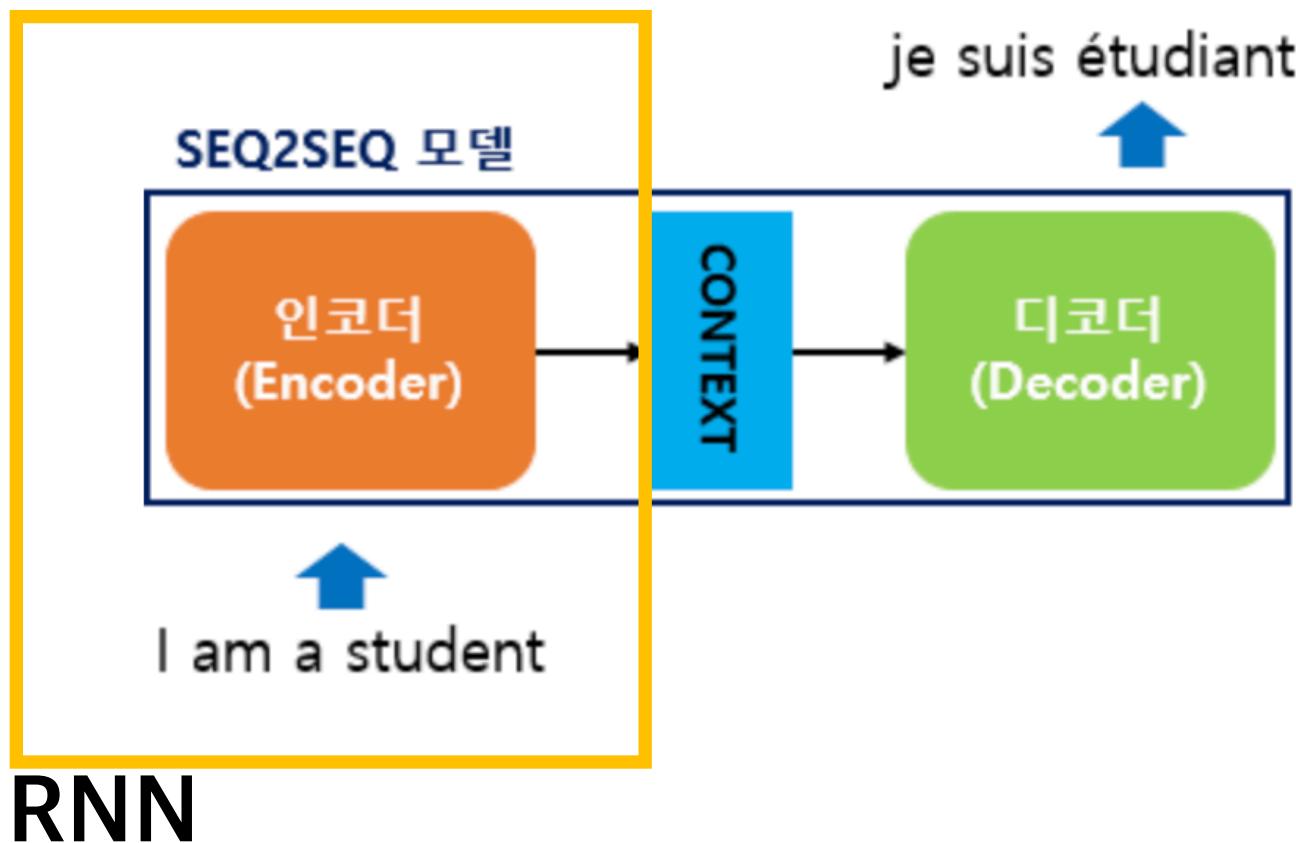


- 모델 구성 [1] Context Vector

- ✓ 인코더에서 받은 정보를 하나의 벡터로 압축하는 단계
- ✓ 인코더 정보를 디코더로 전송하는 단계



- 모델 구성 [0] 그림 전체적인 구성



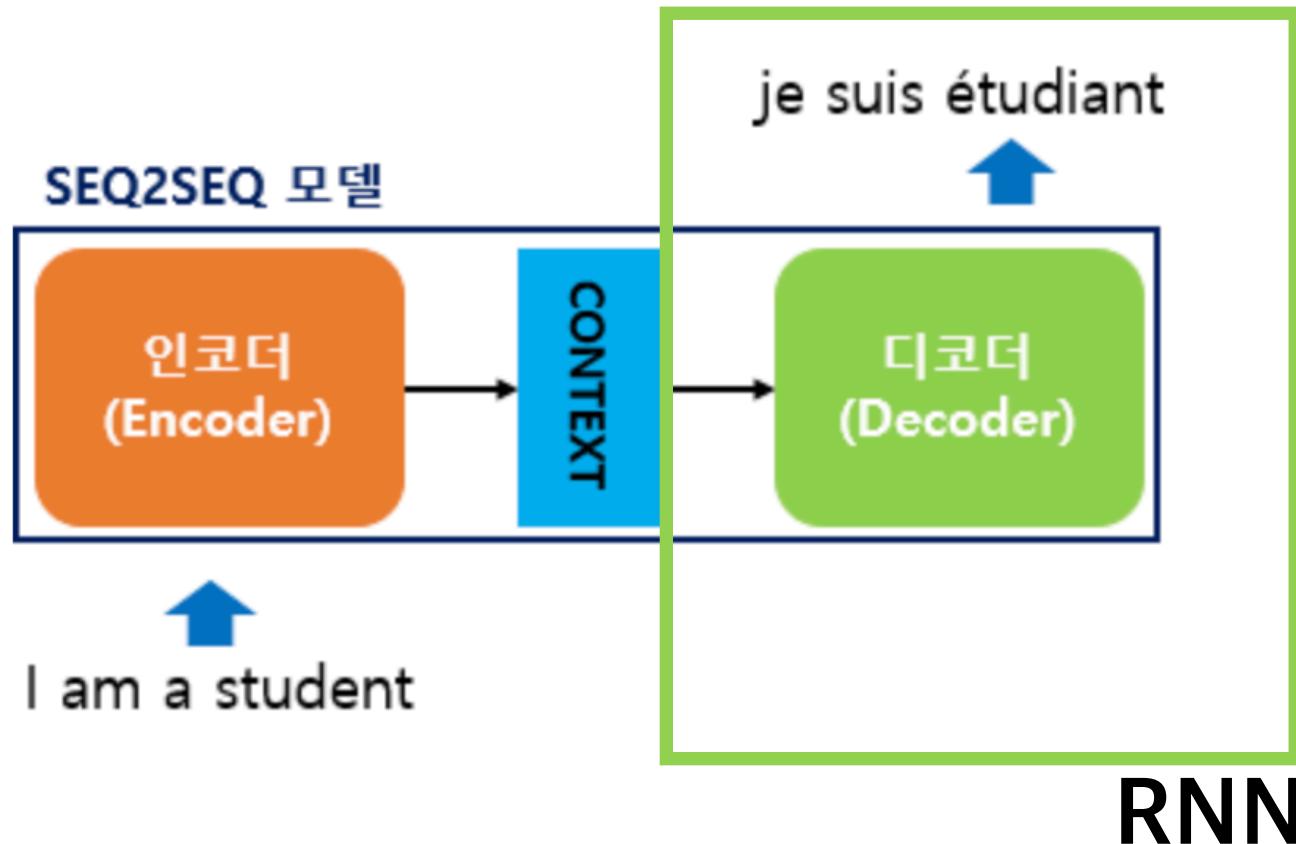
- 모델 구성 [2] Encoder

- ✓ 입력 문장을 받는 RNN 셀 (RNN이라고해서 RNN만이 아닌, GRU, LSTM 등 다양하게 사용)
- ✓ [입력] 단어 토큰화를 통한 단어 토큰이 RNN 셀의 각 시점의 입력
- ✓ 모든 단어를 입력 받은 뒤 인코더 RNN 셀의 마지막 시점의 은닉 상태를 컨텍스트 벡터로 넘김
- ✓ 이 컨텍스트 벡터는 디코더 RNN 셀의 첫번째 은닉 상태로 사용

# 2

## NMT Model Explanation

- 모델 구성 [0] 그림 전체적인 구성



- 모델 구성 [3] Decoder

- ✓ RNN-LM이다.
- ✓ 초기 입력으로 문장의 시작을 의미하는 토큰 <sos>/<start> 가 들어감.
- ✓ 디코더는 <sos>/<start>가 입력되면, 다음에 등장할 확률이 높은 단어를 예측
- ✓ 예측된 단어를 다음 시점의 RNN 셀에 보낸다. (반복)
- ✓ 문장의 끝을 의미하는 심볼인 <eos>/<end> 가 다음 단어로 예측될 때까지 반복

- 모델 구성 [3] Decoder

- ✓ RNN-LM이다.
- ✓ 초기 입력으로 문장의 시작을 의미하는 토큰 <sos>/<start> 가 들어감.

✓ 디코더는 <sos>/<start>가 입력되었을 때 오해하지 않고 다음 단어를 예측

## TEST STAGE

- ✓ 문장의 끝을 의미하는 심볼인 <eos>/<end> 가 다음 단어로 예측될 때까지 반복

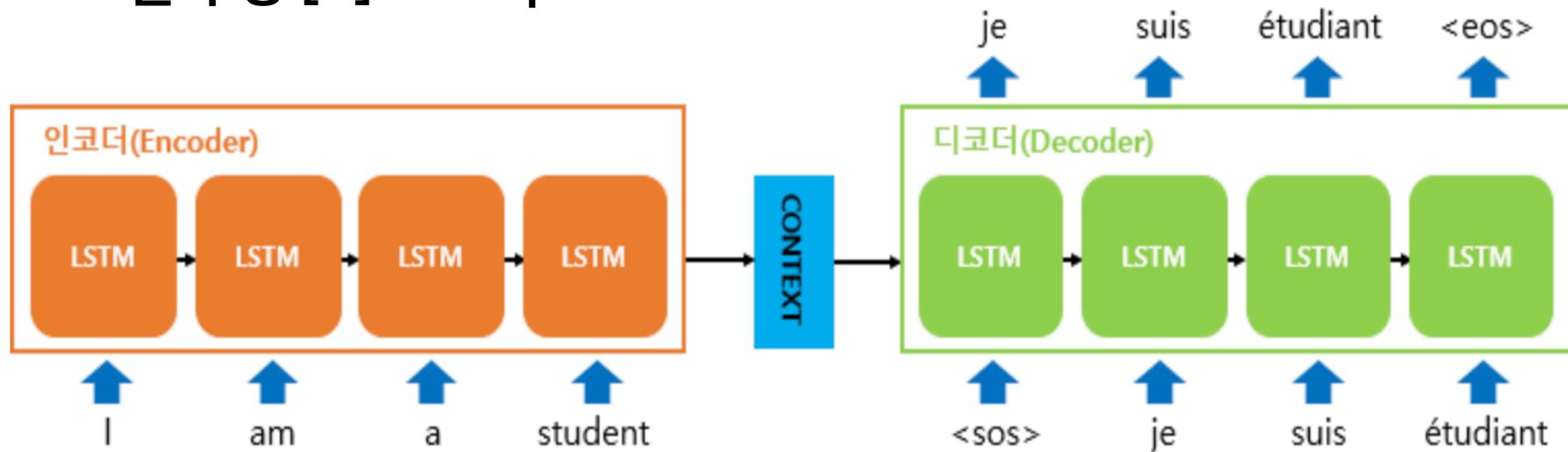
## TRAIN STAGE

- ✓ 정답을 알려주면서 학습을 진행
- ✓ Ex) 컨텍스트 벡터와 실제 정답을 <sos> je suis étudiant를 입력 받아,  
je suis étudiant <eos>가 나와야 된다고 정답을 알려주면서 훈련 진행 [교사 강요]

## 2

## NMT Model Explanation

- 모델 구성 [4] Example

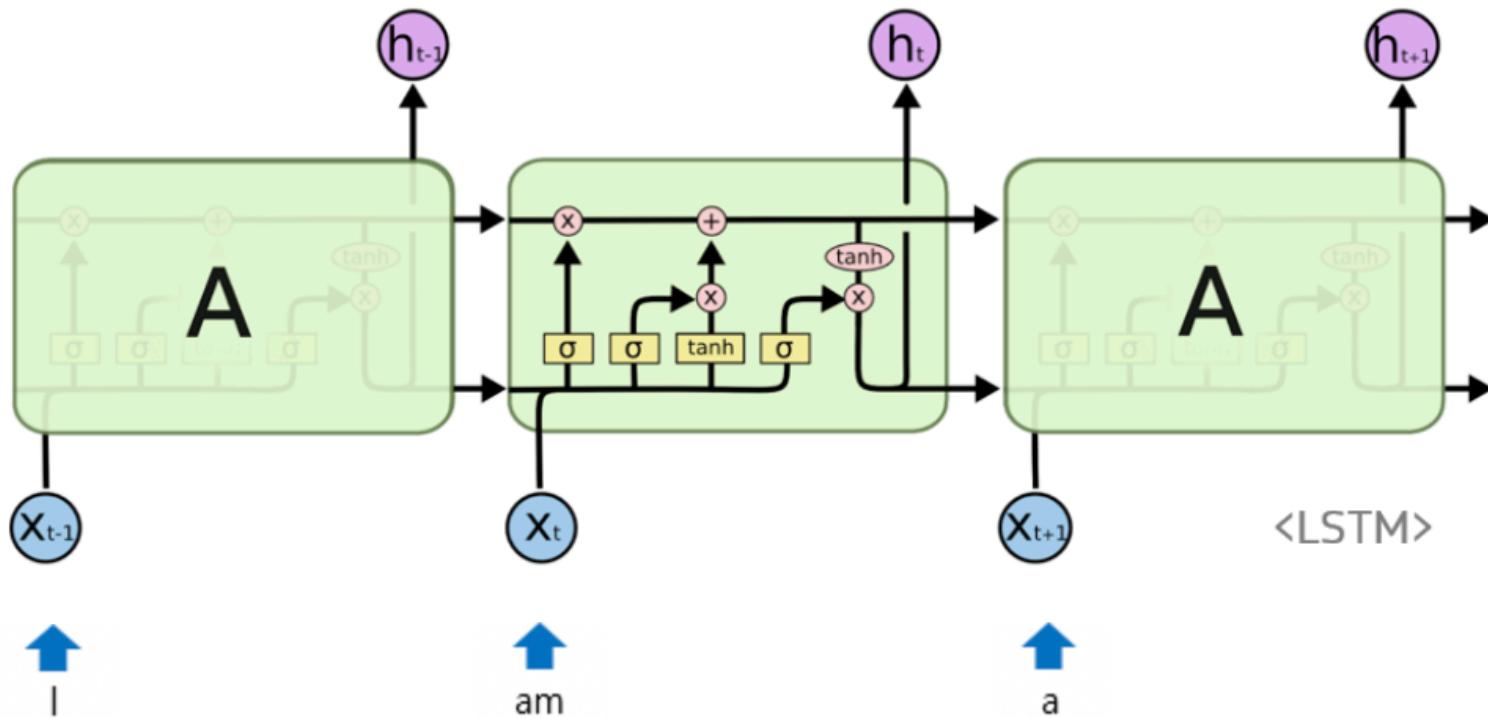


### [1] Word Embedding 진행

-	<table border="1" style="border-collapse: collapse; width: 100px;"> <tr><td>0.157</td></tr> <tr><td>-0.25</td></tr> <tr><td>0.478</td></tr> <tr><td>-0.78</td></tr> </table>	0.157	-0.25	0.478	-0.78
0.157					
-0.25					
0.478					
-0.78					
am	<table border="1" style="border-collapse: collapse; width: 100px;"> <tr><td>0.78</td></tr> <tr><td>0.29</td></tr> <tr><td>-0.96</td></tr> <tr><td>0.52</td></tr> </table>	0.78	0.29	-0.96	0.52
0.78					
0.29					
-0.96					
0.52					
a	<table border="1" style="border-collapse: collapse; width: 100px;"> <tr><td>0.75</td></tr> <tr><td>-0.81</td></tr> <tr><td>0.96</td></tr> <tr><td>0.12</td></tr> </table>	0.75	-0.81	0.96	0.12
0.75					
-0.81					
0.96					
0.12					
student	<table border="1" style="border-collapse: collapse; width: 100px;"> <tr><td>0.88</td></tr> <tr><td>-0.17</td></tr> <tr><td>0.29</td></tr> <tr><td>0.48</td></tr> </table>	0.88	-0.17	0.29	0.48
0.88					
-0.17					
0.29					
0.48					

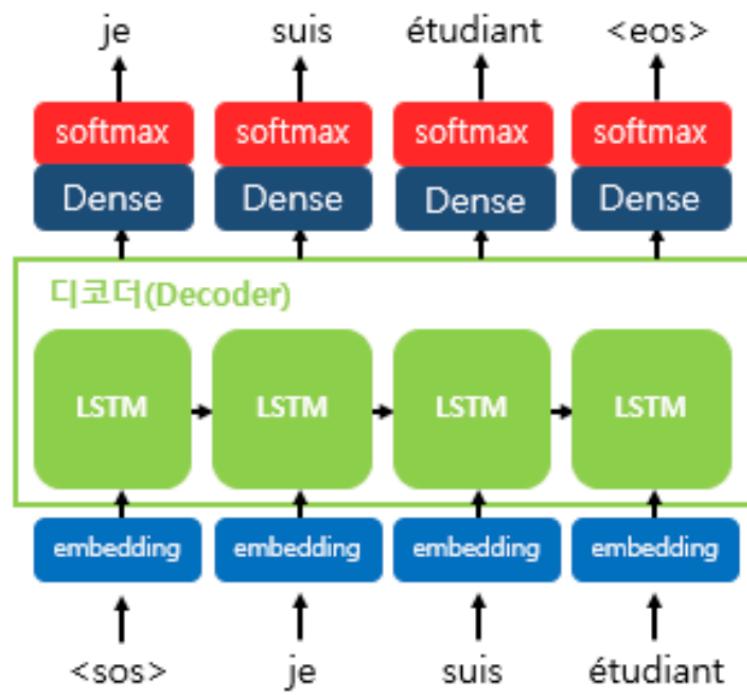
- 모델 구성 [4] Example

## [2] Encoder



- 모델 구성 [4] Example

## [4] Decoder



- ✓ RNN 셀의 은닉 상태인 **컨텍스트 벡터**를 첫번째 hidden state으로 사용
- ✓ 첫번째 RNN 셀은 이 hidden state과, 현재 t에서의 입력 값인 **<sos>**로부터, 다음에 등장할 단어를 예측
- ✓ 각 시점(time step)의 RNN 셀에서 출력 벡터가 나오면, 해당 벡터는 **소프트맥스** 함수를 통해 출력 시퀀스의 각 단어별 확률값을 **반환**하고, 출력 단어를 결정

- 장점

- ✓ SMT와 달리, sub problem으로 나누지 않는다. [direct calculation]

NMT directly calculates  $P(y|x)$ :

$$P(y|x) = P(y_1|x) P(y_2|y_1, x) P(y_3|y_1, y_2, x) \dots P(y_T|y_1, \dots, y_{T-1}, x)$$

SMT :  $\operatorname{argmax}_y P(x|y)P(y)$

- 훈련 방식

- ✓ Big Parallel corpus로 훈련한다. [병렬 말뭉치]

- **Training Process (1) Encoder**

### Encoder

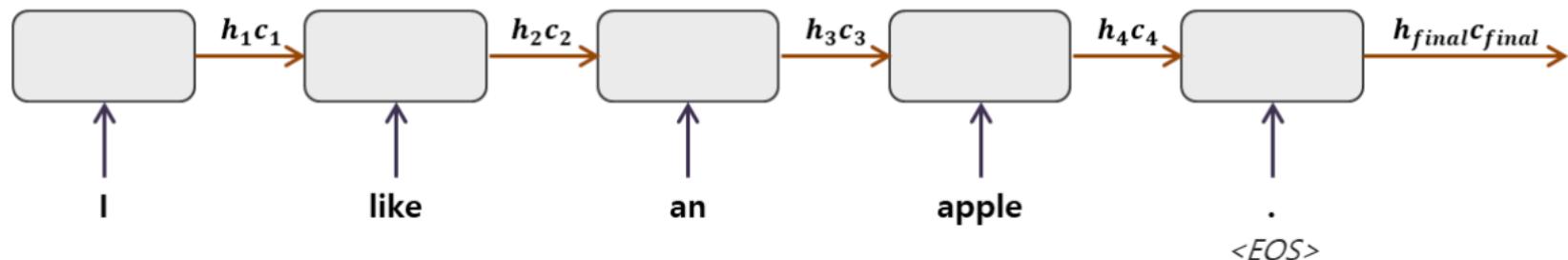


Figure 2: Seq2seq Part of Encoder

- ✓ 순차적으로 RNN에 입력을 받는다. [LSTM: Cell, hidden state 반환, GRU: hidden state만 반환 ]
- ✓  $H_{final}, C_{final}$ 을 반환한다. (LSTM)
- ✓ EOS를 통해, 문장의 종료를 나타낸다. (Padding 을 통해, 문장의 길이를 맞춘다.)

- Training Process (2) Decoder

Teacher Forcing

Decoder

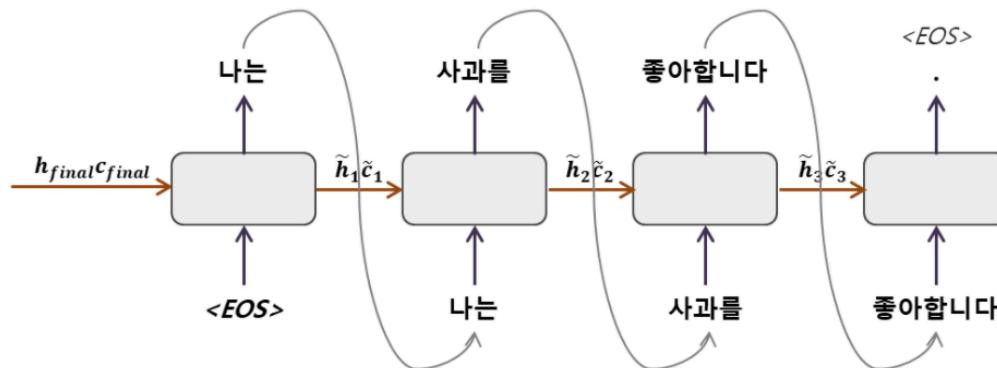


Figure 3: Seq2seq Part of Decoder

- ✓ 이전 encoder에서 반환된  $H_{final}, C_{final}$ 을 초기 입력 값으로 받는다.
- ✓ Train을 할 때에는, 정답 레이블을 알기에,  $t+1$  시점에서 정답을 이용한다.

- Train vs Test

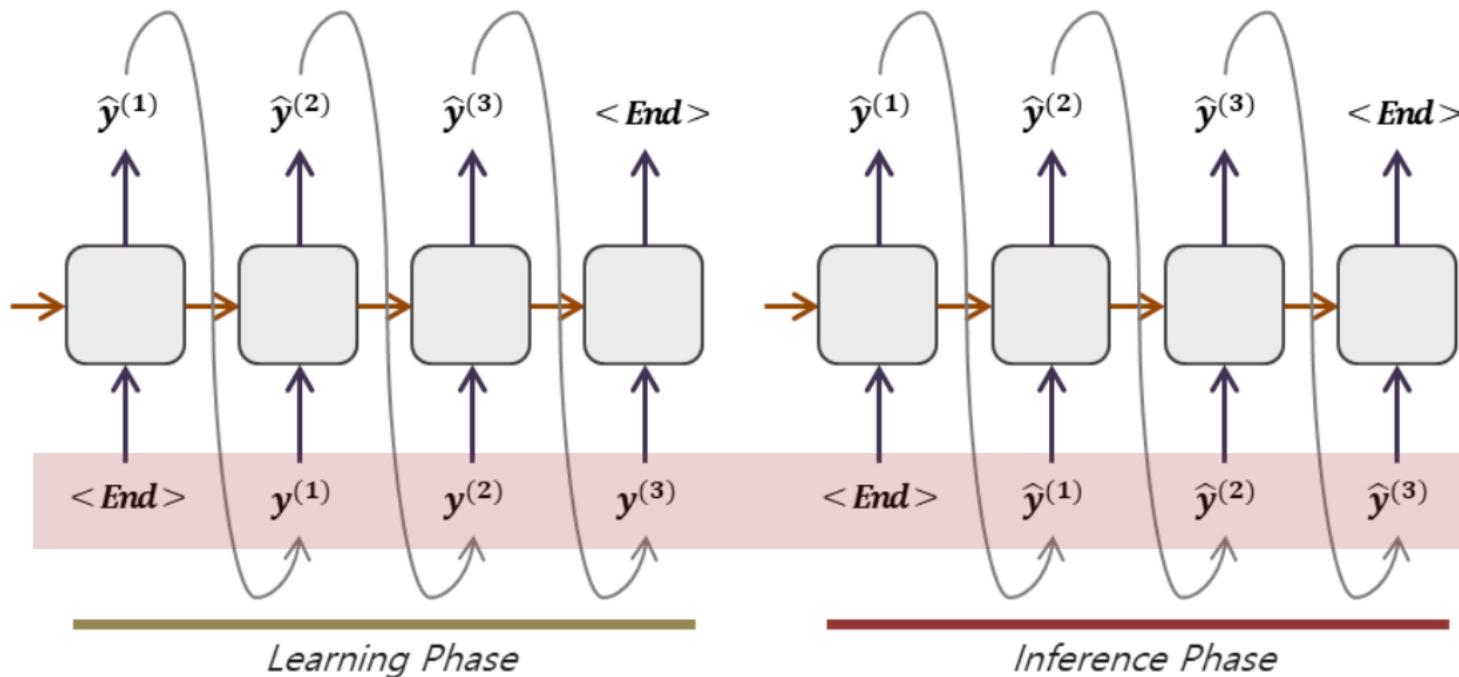


Figure 5: Decoder 작동원리

- 장점

- ✓ 성능이 좋음
- ✓ End-to-end로 최적화된다 (문제가 분리되지 않고 하나의 문제로 해결)
- ✓ Feature engineerin이 불필요하다

- 단점

- ✓ 해석이 불가능하다
- ✓ Control하기 어렵다.

### 3. Evaluation

## • 소개

- ✓ Bilingual Evaluation Understudy
- ✓ Machine-written-translation vs human-written-translation 의 유사도 점수 측정
  - ✓ 조건1: n-gram precision
  - ✓ 조건2: 짧은 문장에 대한 Penelty(규제)
    - n-gram을 통한 순서쌍들이 얼마나 겹치는지 측정(precision)
    - 문장길이에 대한 과적합 보정 (Brevity Penalty)
    - 같은 단어가 연속적으로 나올때 과적합 되는 것을 보정(Clipping)

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

## 4. Attention Method

- seq-to-seq bottleneck problem

### Encoder

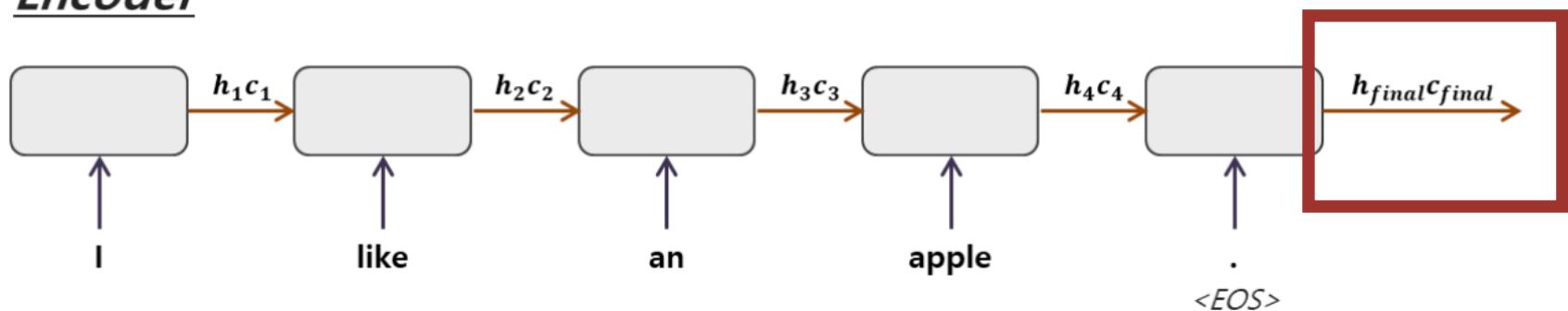
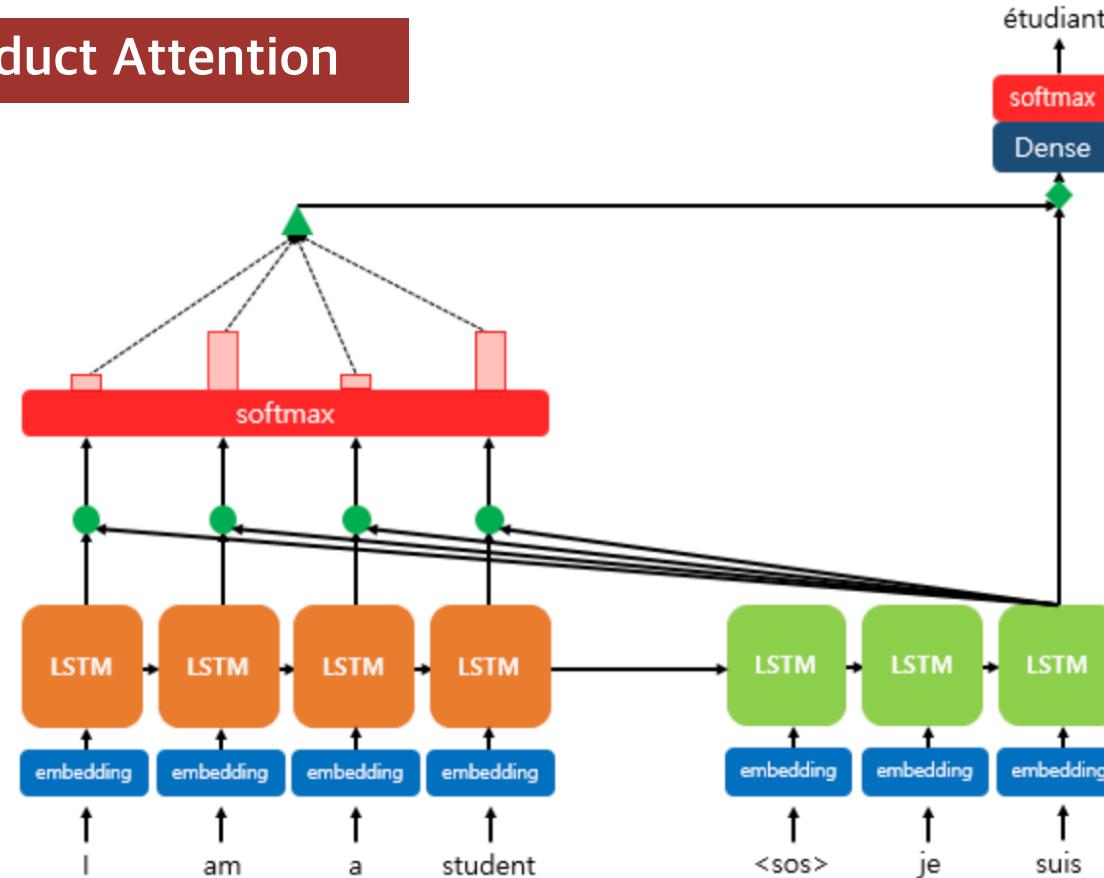


Figure 2: Seq2seq Part of Encoder

- ✓ Encoder에서 전체 문장을 마지막 노드로만 표현한다.
- ✓ 이때, 전체 문장을 반영해야되는 bottleneck problem 발생

- 중심 아이디어 ✓ Encoder - Decoder을 직접적으로 연결한다.

### Dot Product Attention

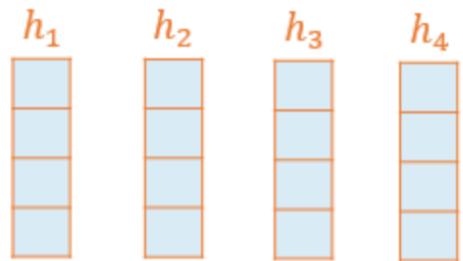


- 중심 아이디어 ✓ Encoder - Decoder을 직접적으로 연결한다.

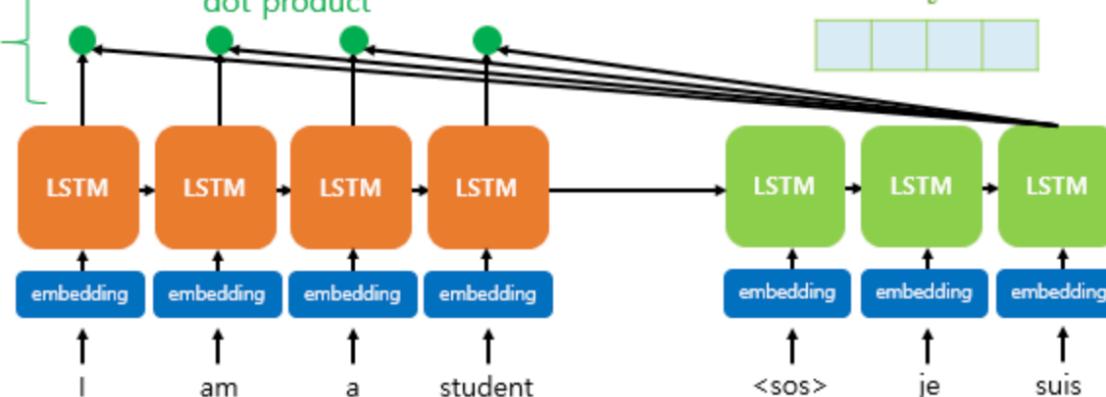
### Dot Product Attention

#### (1) Attention Score 구하기

Encoder의 hidden state



현재 시점의 hidden state

 $s_t^T$ 


- ✓ 원래 Decoder에서 필요한 조건
- ✓ 이전 시점의 hidden state
- ✓ 이전 시점의 출력 단어

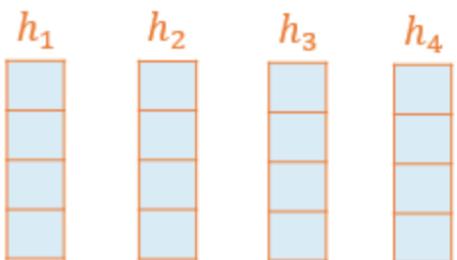
\*Encoder/decoder hidden state 차원 동일

- 중심 아이디어 ✓ Encoder - Decoder을 직접적으로 연결한다.

### Dot Product Attention

#### (1) Attention Score 구하기

Encoder의 hidden state

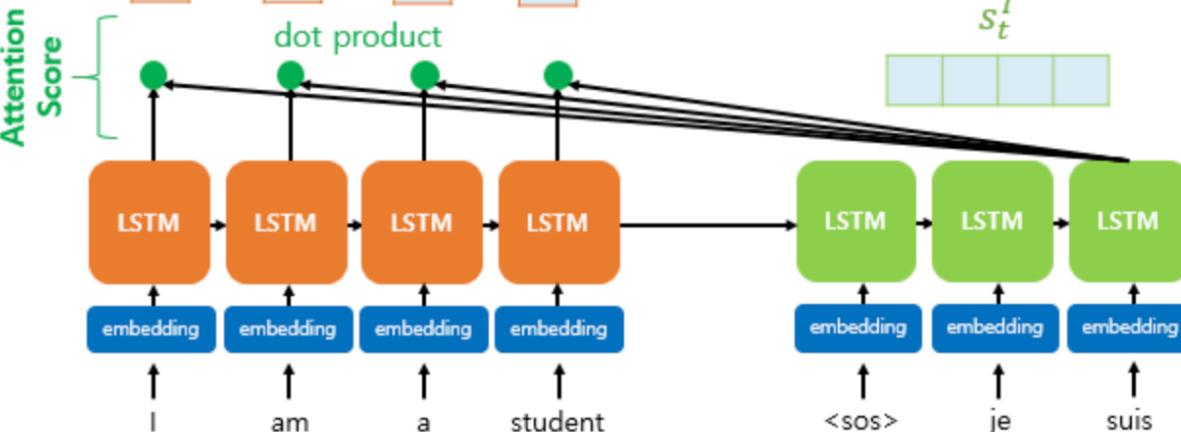


## 새 조건: ATTENTION SCORE

현재 시점의 hidden state

 $s_t^T$ 


✓ 원래 Decoder에서 필요한 조건



✓ 이전 시점의 hidden state

✓ 이전 시점의 출력 단어

\*Encoder/decoder hidden state 차원 동일

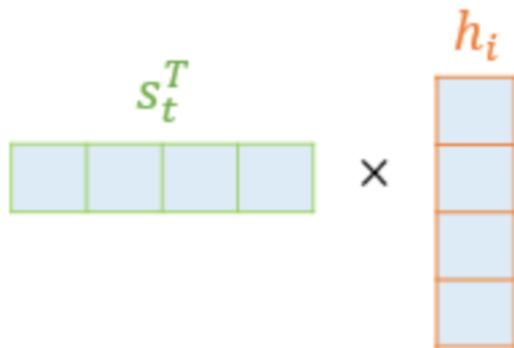
- 중심 아이디어 ✓ Encoder - Decoder을 직접적으로 연결한다.

### Dot Product Attention

#### (1) Attention Score 구하기

어텐션 스코어( $a_t$ )

현재 디코더의 시점  $t$ 에서 단어를 예측하기 위해,  
인코더의 모든 은닉 상태 각각이  
디코더의 현 시점의 은닉 상태  $s_t$ 와 얼마나 유사한지를 판단하는 스코어



$$\text{score}(s_t, h_i) = s_t^T h_i$$

$$e^t = [s_t^T h_1, \dots, s_t^T h_N]$$

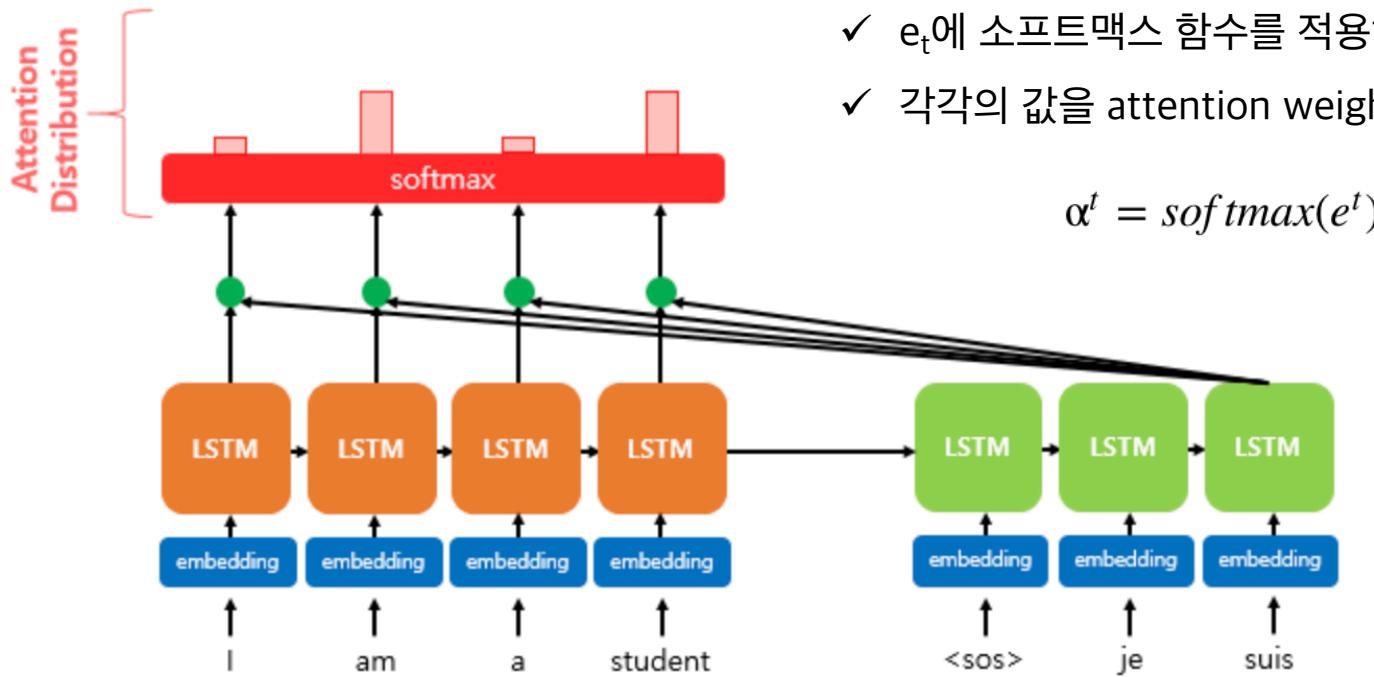
모든 은닉 상태의 어텐션 스코어의 모음값

\*Encoder/decoder hidden state 차원 동일

- 중심 아이디어 ✓ Encoder - Decoder을 직접적으로 연결한다.

### Dot Product Attention

### (2) Attention Distribution 구하기



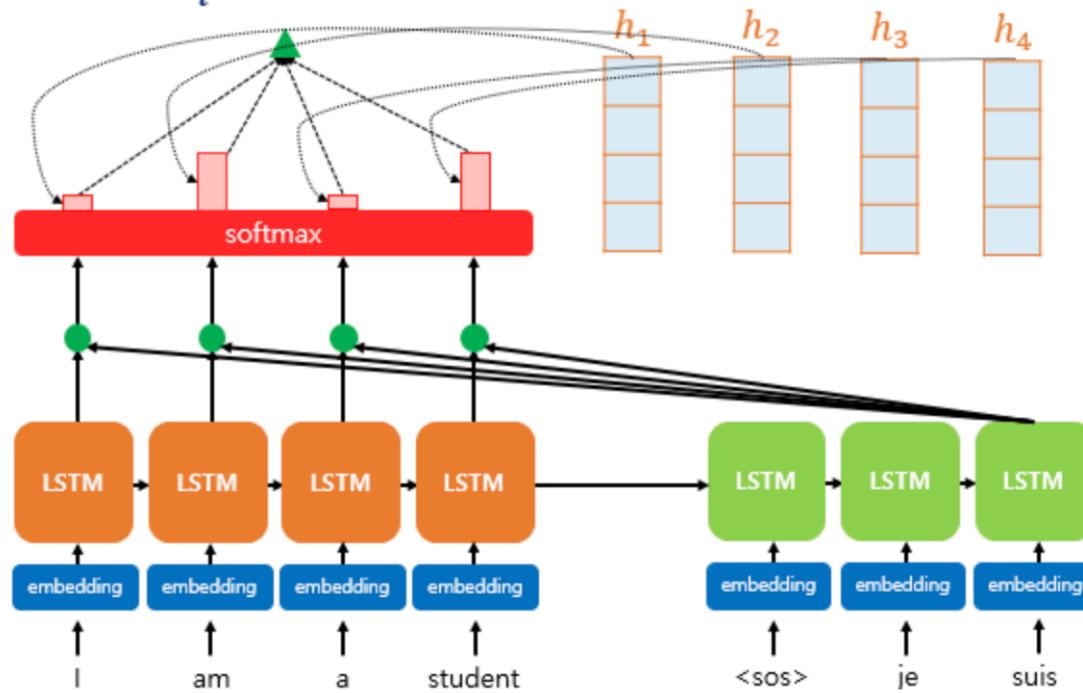
\*Encoder/decoder hidden state 차원 동일

- 중심 아이디어 ✓ Encoder - Decoder을 직접적으로 연결한다.

### Dot Product Attention

### (3) Attention Value 구하기

Attention Value  $a_t$



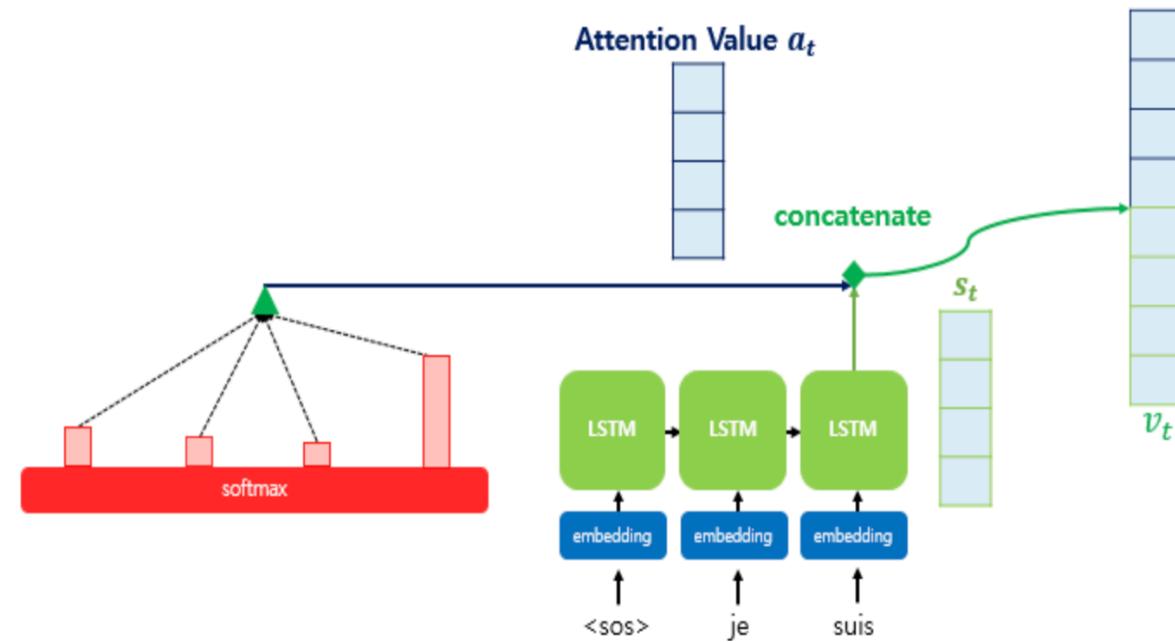
- ✓ 각 인코더의 은닉 상태와 어텐션 가중치값들을 곱하고, 최종적으로 모두 더한다. (weighted sum)

$$a_t = \sum_{i=1}^N \alpha_i^t h_i$$

\*Encoder/decoder hidden state 차원 동일

- 중심 아이디어 ✓ Encoder - Decoder을 직접적으로 연결한다.

## Dot Product Attention (4) Concatenation



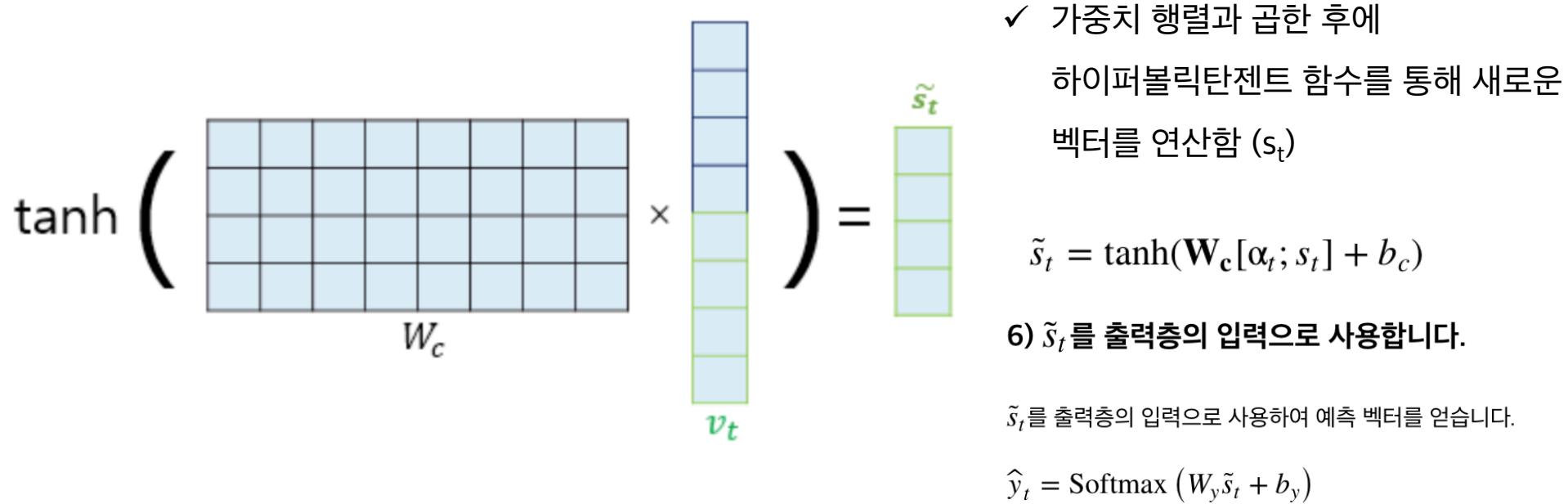
- ✓ 새로 구한  $a_t$ 와  $s_t$ 를 연결한다. ( $v_t$ )
- ✓  $V_t$ 는 인코더의 정보를 잘 활용할 수 있다는 장점이 있다.

\*Encoder/decoder hidden state 차원 동일

- 중심 아이디어 ✓ Encoder - Decoder을 직접적으로 연결한다.

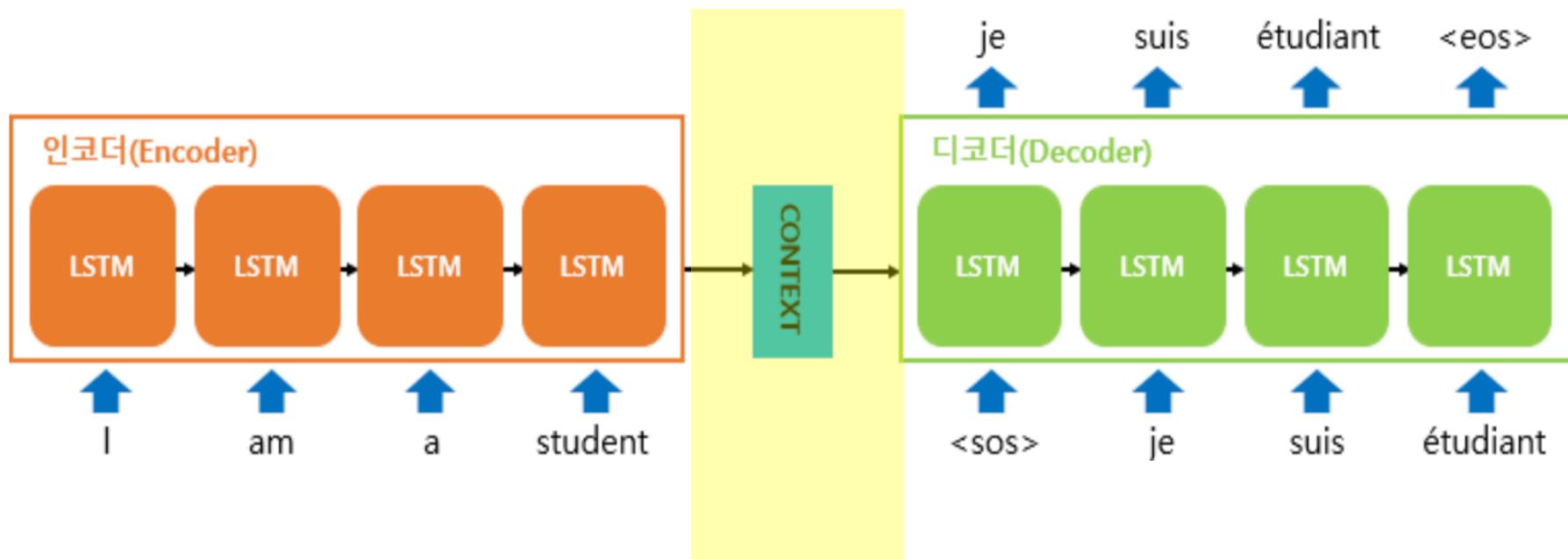
### Dot Product Attention

#### (5) 새로운 state 계산

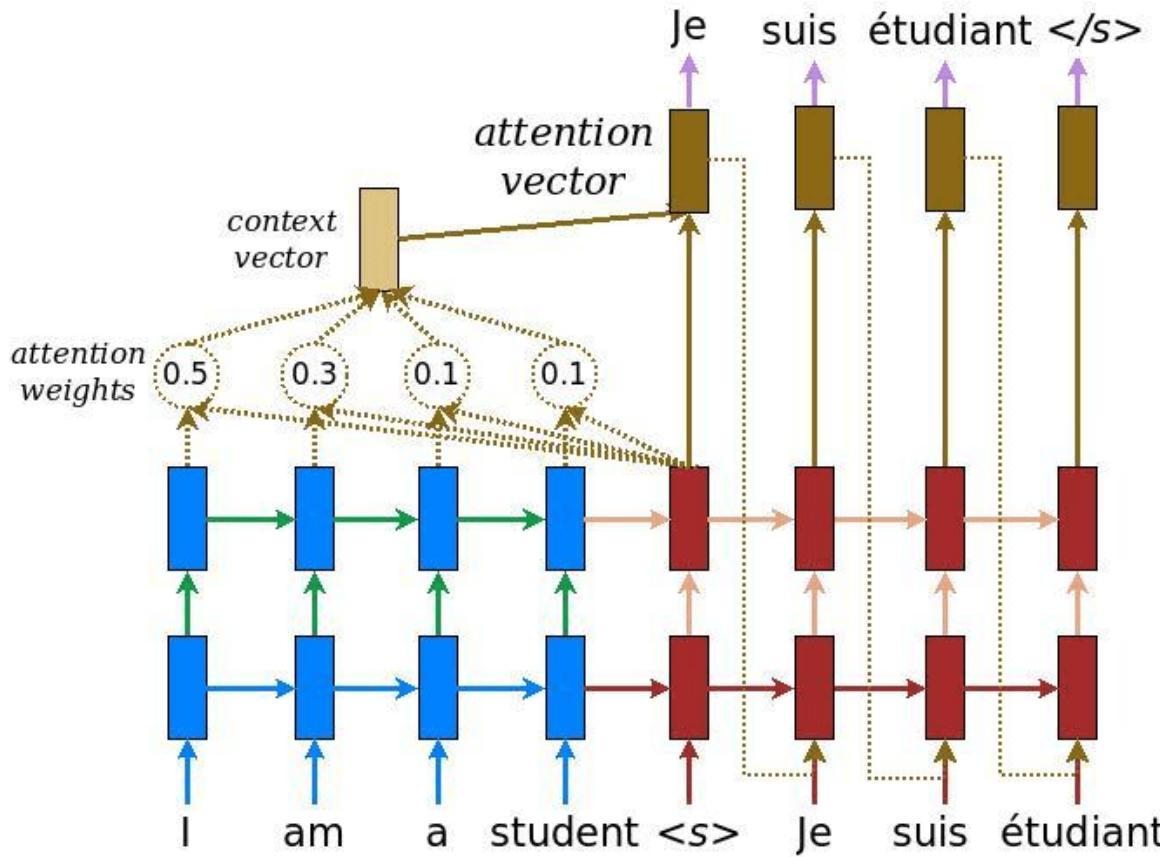


\*Encoder/decoder hidden state 차원 동일

## 기존 neural machine translation



## Attention Mechanism



Encoder hiddenstate :  $h_1 \dots h_n$   
 Decoder hidden state:  $s_1 \dots s_t$

$$e^t = [s_t^T h_1, \dots, s_t^T h_N]$$

$$\alpha^t = \text{softmax}(e^t)$$

$$a_t = \sum_{i=1}^N \alpha_i^t h_i$$

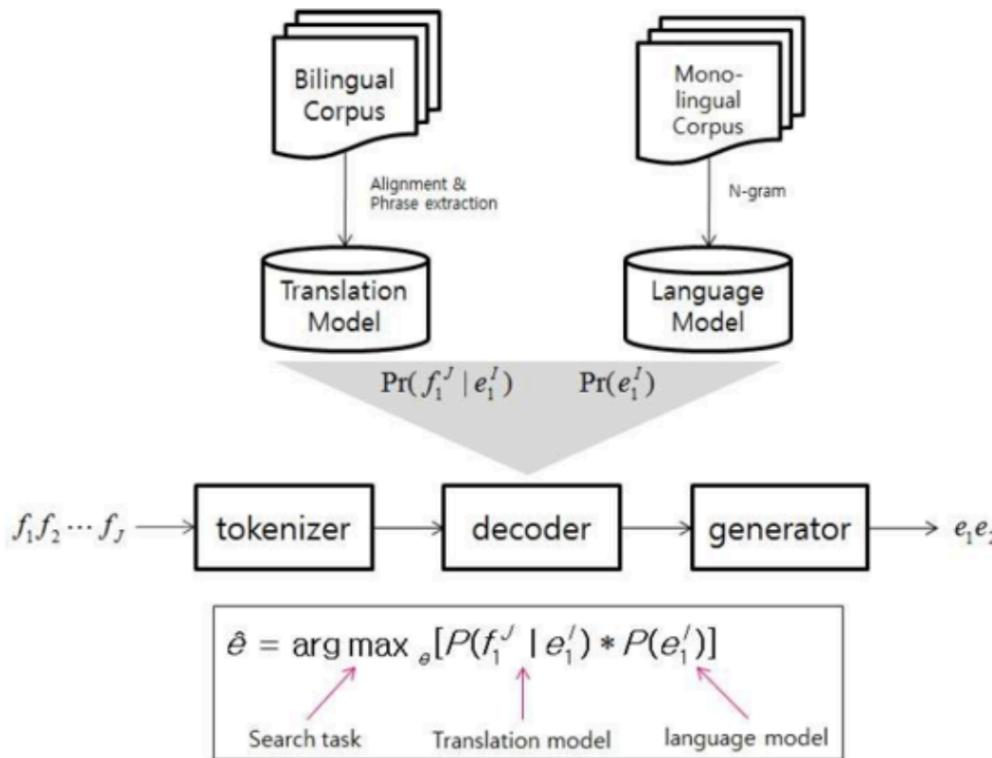
$$\tilde{s}_t = \tanh(\mathbf{W}_c[\alpha_t; s_t] + b_c)$$

$$\hat{y}_t = \text{Softmax}(W_y \tilde{s}_t + b_y)$$

+ 부록

- **SMT 이해하기** 두가지 요소: Translation model/ Language Model

- 구조



- **SMT 이해하기** 두가지 요소: Translation model/ Language Model 의 계산

- **Translation Model**

- phrase translation probability  $p(\text{Mary} | \text{Maria})$
  - reordering costs
  - phrase/word count costs
  - ...

- **Language Model**

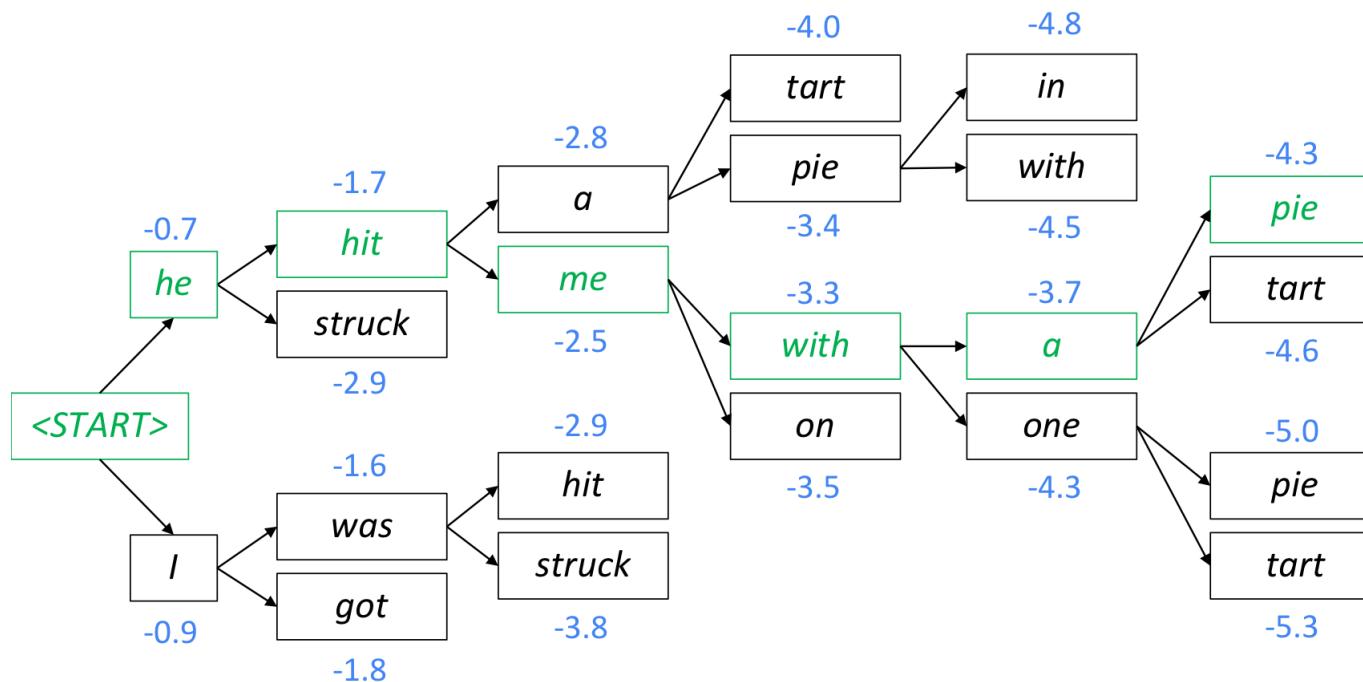
- uses trigrams:
  - $p(\text{Mary did not}) = p(\text{Mary} | <\text{s}>) * p(\text{did} | \text{Mary}, <\text{s}>) * p(\text{not} | \text{Mary did})$

- **decoding**

- ✓ Greedy Decoding : 가장 확률 높은 단어 다음 스텝 사용
- ✓ Exhaustive Search Decoding : 모든 단어를 기억, 실행
- ✓ Beam Search Decoding : k 개의 그럴듯한 가설만을 진행

- **decoding**

- ✓ Beam Search Decoding : k 개의 그럴듯한 가설만을 진행



- Example

- 1-gram precision:

$$\frac{\text{일치하는 } 1\text{-}gram \text{의 수(예측된 sentence 중에서)}}{\text{모든 } 1\text{-}gram \text{쌍 (예측된 sentence 중에서)}} = \frac{10}{14}$$

- 2-gram precision:

$$\frac{\text{일치하는 } 2\text{-}gram \text{의 수(예측된 sentence 중에서)}}{\text{모든 } 2\text{-}gram \text{쌍 (예측된 sentence 중에서)}} = \frac{5}{13}$$

- 3-gram precision:

$$\frac{\text{일치하는 } 3\text{-}gram \text{의 수(예측된 sentence 중에서)}}{\text{모든 } 3\text{-}gram \text{쌍 (예측된 sentence 중에서)}} = \frac{2}{12}$$

- 4-gram precision:

$$\frac{\text{일치하는 } 4\text{-}gram \text{의 수(예측된 sentence 중에서)}}{\text{모든 } 4\text{-}gram \text{쌍 (예측된 sentence 중에서)}} = \frac{1}{11}$$

$$\left(\prod_{i=1}^4 precision_i\right)^{\frac{1}{4}} = \left(\frac{10}{14} \times \frac{5}{13} \times \frac{2}{12} \times \frac{1}{11}\right)^{\frac{1}{4}}$$



T R A I N   A N D   T E S T