



Week 10. Machine Translation, SeqtoSeq, Attention

발표자: 김소민, 황채원

목차

#01 Machine Translation

#02 Statistical Machine Translation (SMT)

#03 Neural Machine Translation (NMT)

#04 Attention











Machine Translation



Machine Translation

컴퓨터 프로그램이 하나의 언어 (원시문서, source text) 로 구성된 문서를 분석하여 다른 언어 (대상문서, target text) 로 구성된 문서로 만드는 번역의 한 형태

한국어 ▾	↔	영어 ▾
나는 배가 고프다 ×		I am hungry. 아이 엠 헝그리.
9 / 5000		번역 수정 번역 평가
  		   

Machine Translation - 발전과정

1950s : Early Machine Translation

- 냉전시대의 영향을 받아 발전 (Russian -> English)
- Rule Based (Bilingual dictionary 사용)

1990s -2010s : Statistical Machine Translation (SMT)

- Data를 통해 확률적 모델 만들기 -> Bayes Rule 사용
- 서로 다른 두 언어의 Parallel 데이터로부터 모델 구축

2014s - : Neural Machine Translation (NMT)

- 단일 신경망을 사용함
- 두개의 RNN을 포함한 Sequence to Sequence 구조

SMT



SMT

핵심: 주어진 **parallel 데이터**로부터 **확률모델**을 구축하기

EX) 불어 (F) -> 영어 (E) 번역

$$\operatorname{argmax}_E P(E|F)$$

Bayes Rule ↓

$$\operatorname{argmax}_E P(F|E)P(E)$$

Translation Model

- 단어와 구가 어떻게 번역되어야 할지를 모델링
-> 번역 정확도
- Parallel Data로부터 학습

Language Model

- 좋은 문장을 생성하도록 모델링
- Monolingual Data로부터 학습

Parallel Data

Translation Model의 학습은?

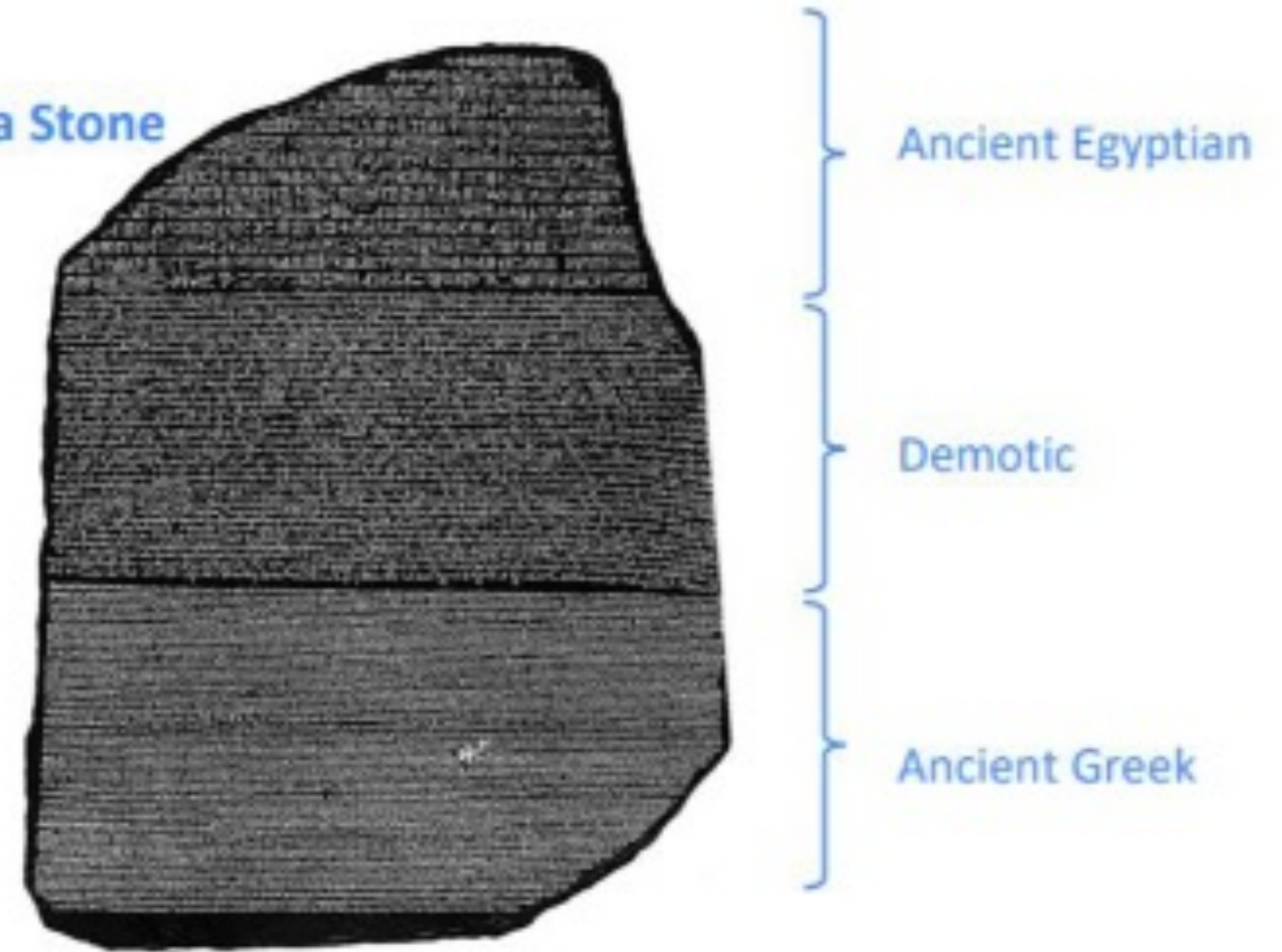
-> 대량의 Parallel Data를 통해!

Parallel Data

-> 번역이 된 문장 쌍

-> ex - French & English sentences

The Rosetta Stone



Alignment

Parallel Data로부터 Translation Model을 어떻게 학습시킬까?

-> Alignment을 통해 학습

Alignment

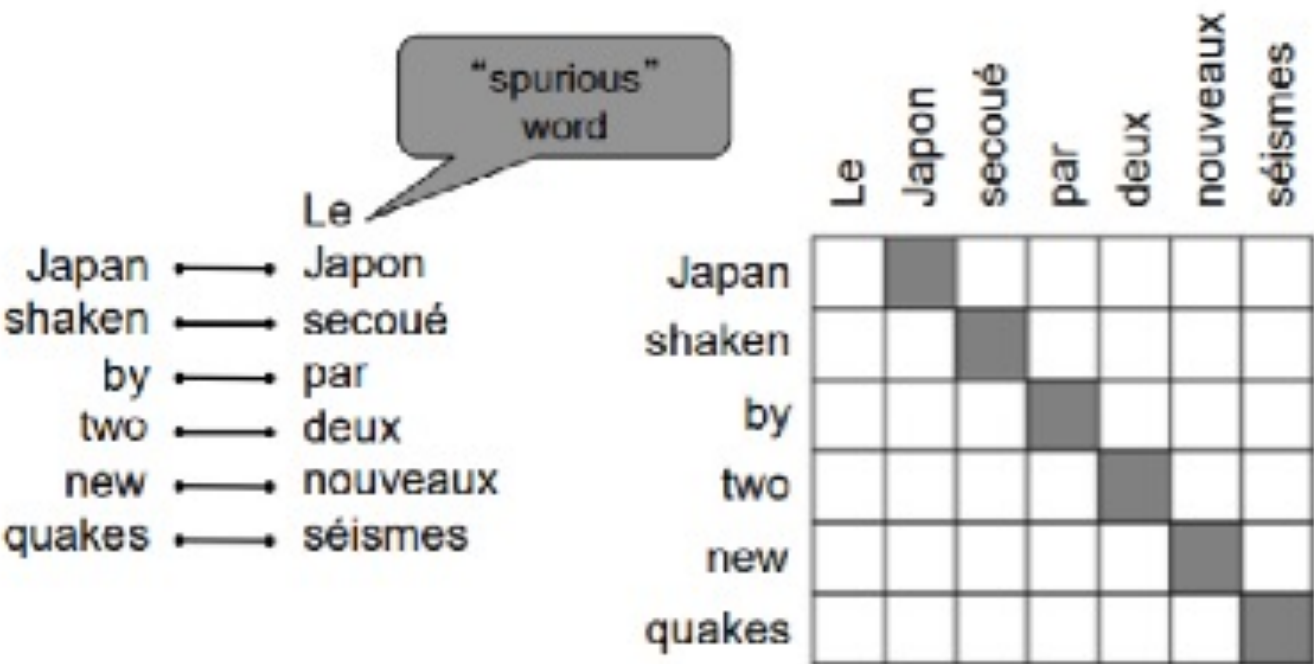
-> Source - Target 언어 문장 Pair에서 일치하는 단어 또는 구를 찾아서 짝을 지어주는 작업

$$P(F|E) \rightarrow \sum_a P(F, a|E)$$

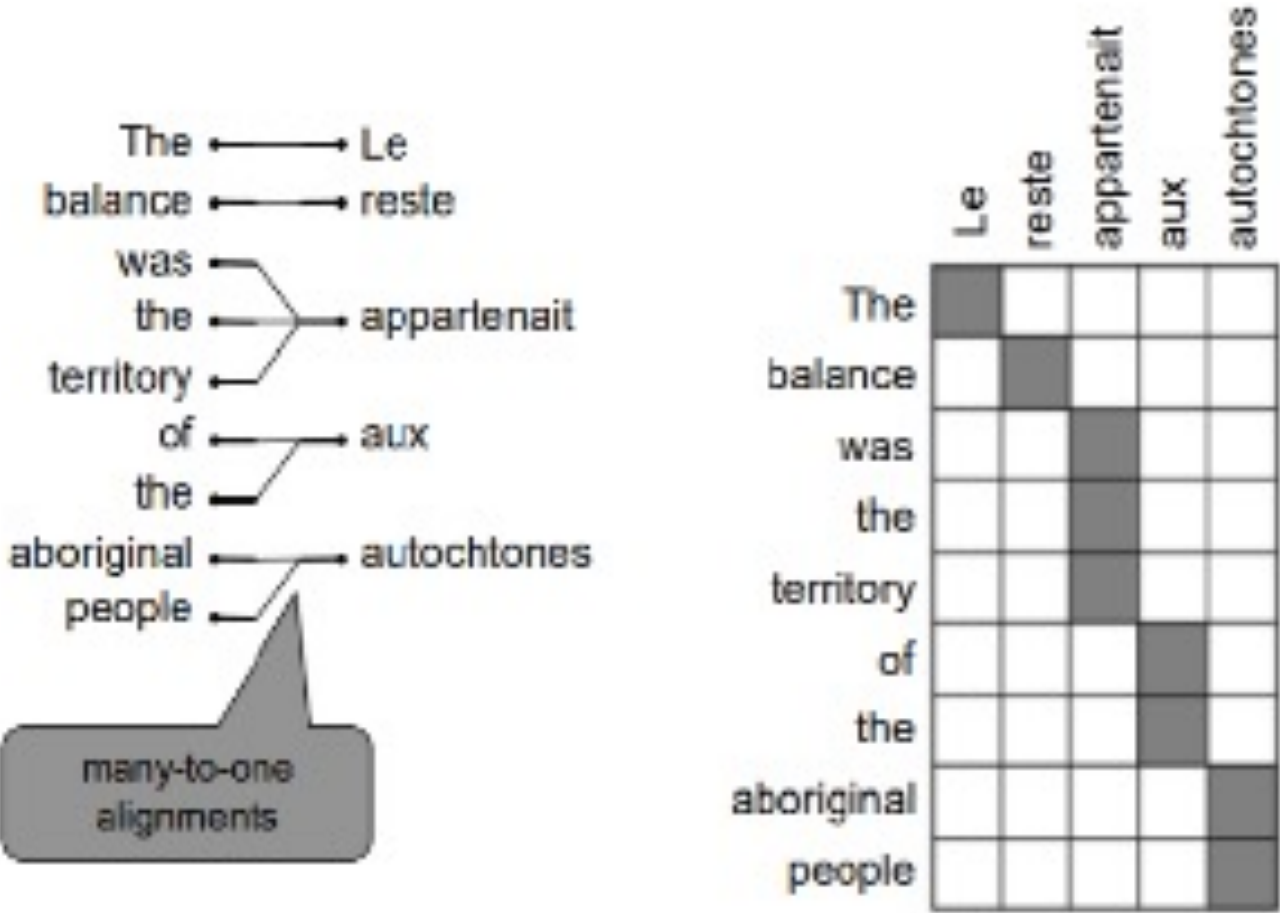
기본적으로 best A와 E를 EM-Algorithm을 통해 찾는 방식으로 학습

Alignment

a. No Counterpart

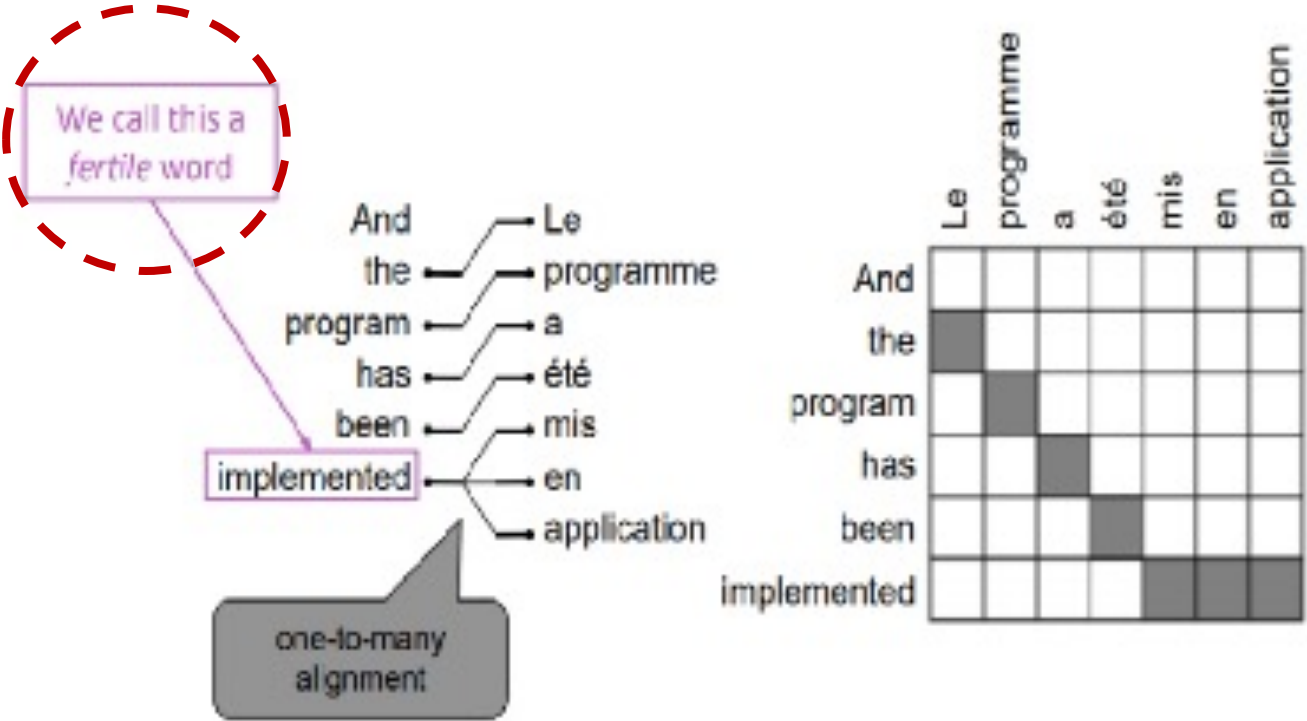


b. Many to One



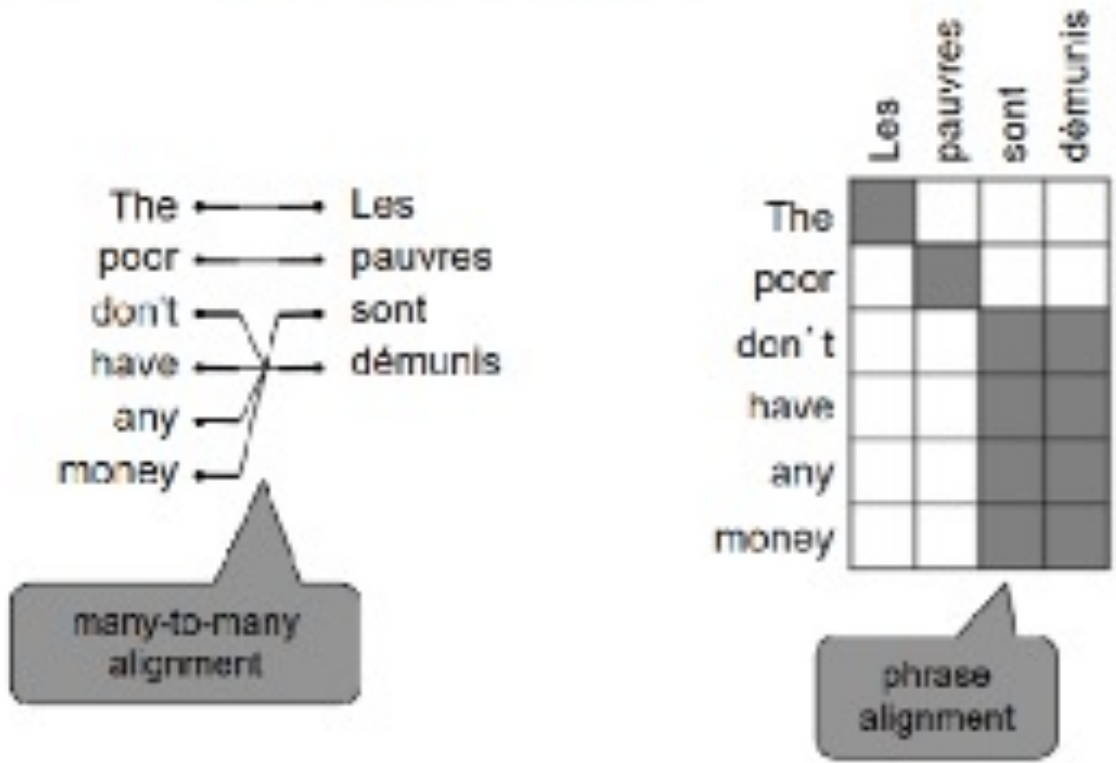
Alignment

c. One to Many



d. Many to Many

Alignment can be *many-to-many* (phrase-level)



Decoding for SMT

어떤 문장이 가장 좋을까? -> "E" 찾기 = Decoding

$$\operatorname{argmax}_E P(F|E)P(E)$$

- 가능한 E에 대해서 확률을 계산하는 것 : 비용이 너무 큼
- Beam Search과 같은 heuristic search algorithm 사용
+ 너무 낮은 확률을 가진 것들은 제외함

SMT 한계점

- Extremely **Complex**
 - 많은 숫자의 Subcomponents (각각 다르게 학습되므로 복잡도 ↑)
 - Feature Engineering이 많이 수행되었어야 함
 - 언어 현상들을 표현할 수 있는 여러 feature들을 디자인
 - Compile 및 유지하는데에도 많은 비용 소요
 - human effort이 많이 필요함

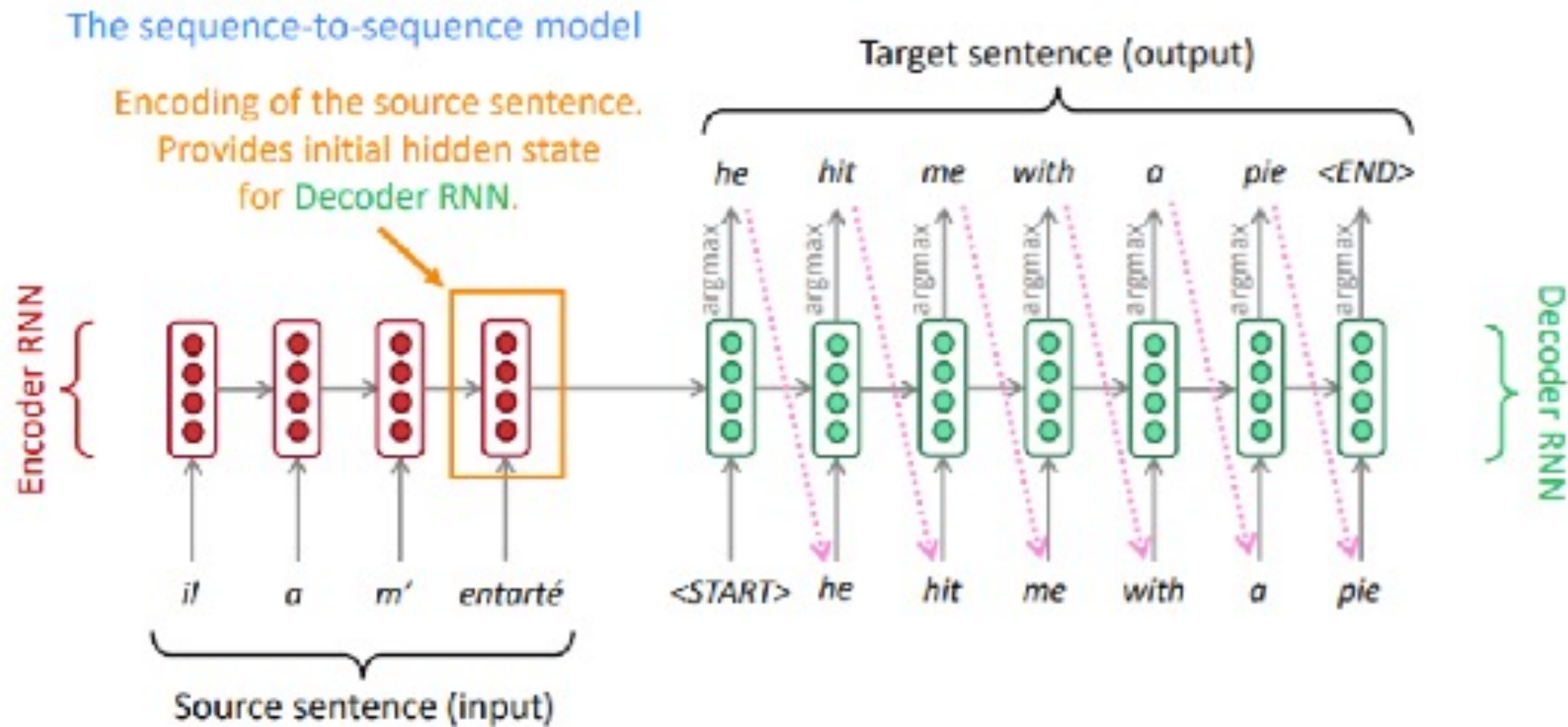
NMT



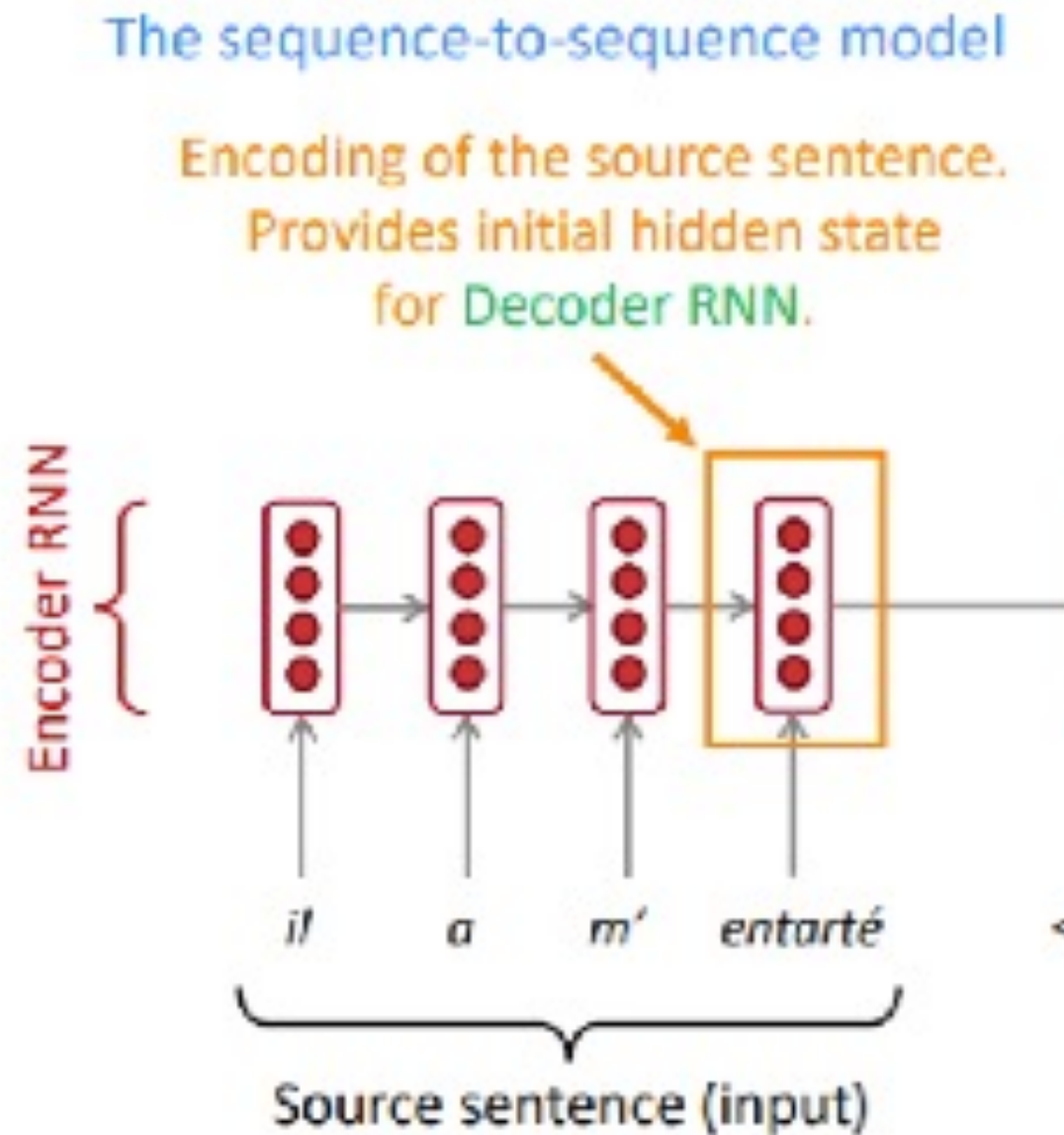
NMT

- 하나의 Neural Net 모델로 Machine Translation 수행
→ 여러 개의 Subcomponent로 분리하여 구성하지 않아도 되는 장점
- Neural Net Architecture = Sequence to Sequence 모델
→ 두개의 RNN (Encoder & Decoder)을 이어 붙인 모델
- 기존의 SMT보다 높은 성능 보임
- 현재는 대부분 NMT 기반의 번역기를 사용

Sequence to Sequence 모델



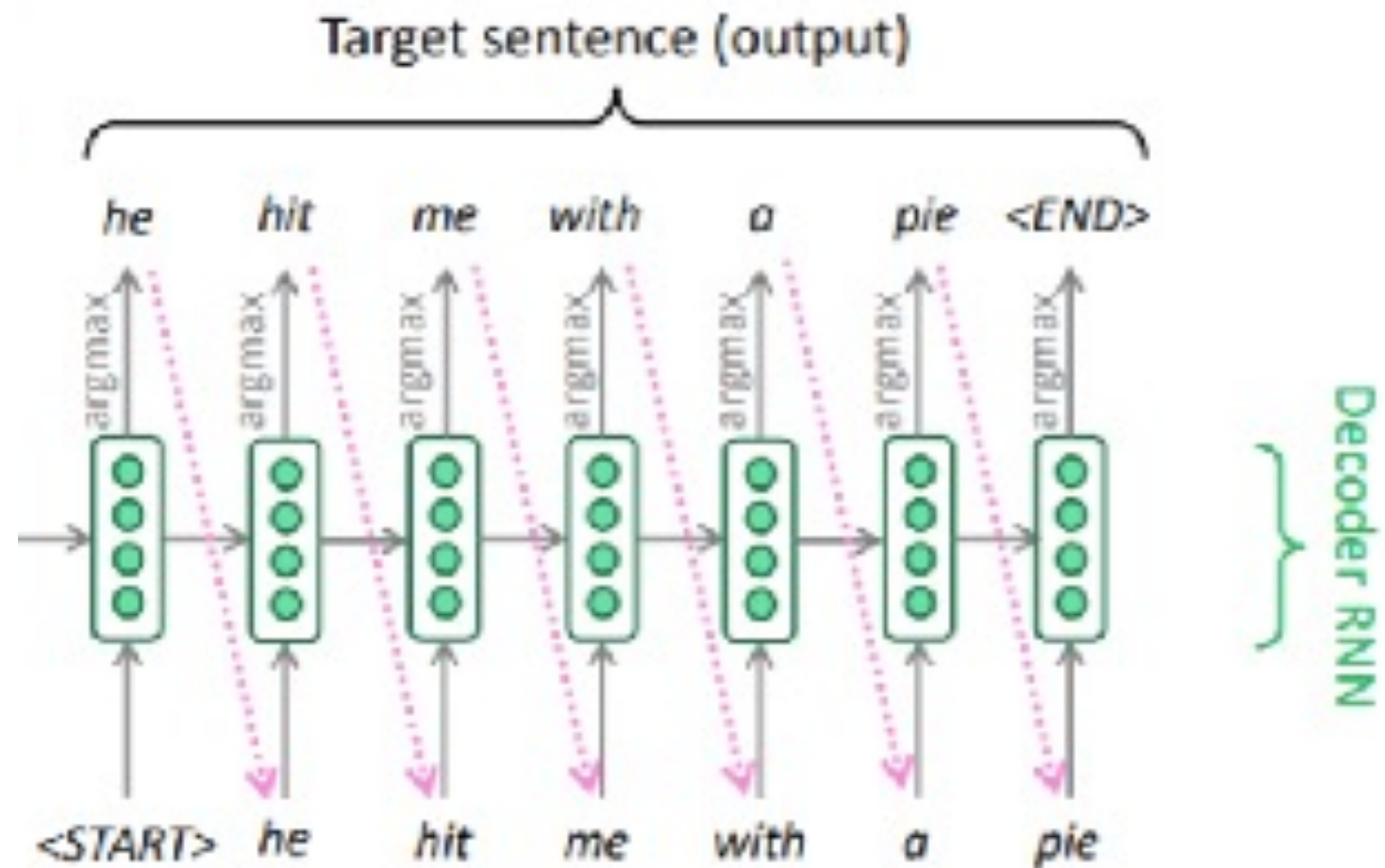
Encoder



Encoder

- Source Sentence의 context 정보 요약
- Last Hidden State를 Decoder로 전달하여 Decoder의 initial state으로 사용하게끔 함

Decoder



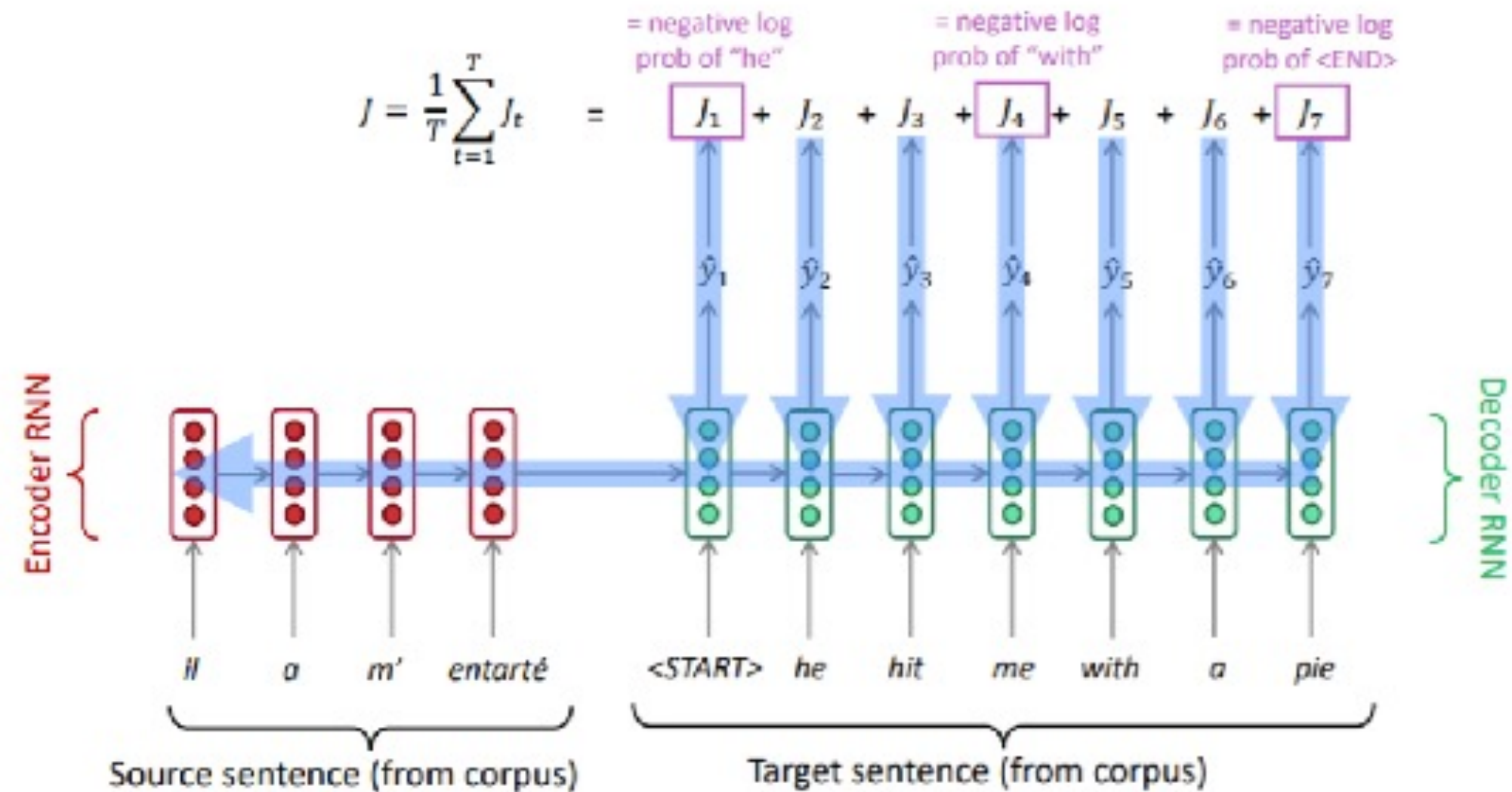
→ 테스트 시 Decoder 의 동작 설명

Decoder

- Encoder로 받은 정보를 기반으로 target sentence 생성
 - Conditional Language Model의 한 종류
 - $P(y|x)$ 를 직접적으로 계산함
- SMT보다 훨씬 간단

Seq2Seq Training

- Seq2Seq 모델은 하나의 시스템으로 최적화 됨
- **End-to-End로 backpropagation이 이뤄짐**



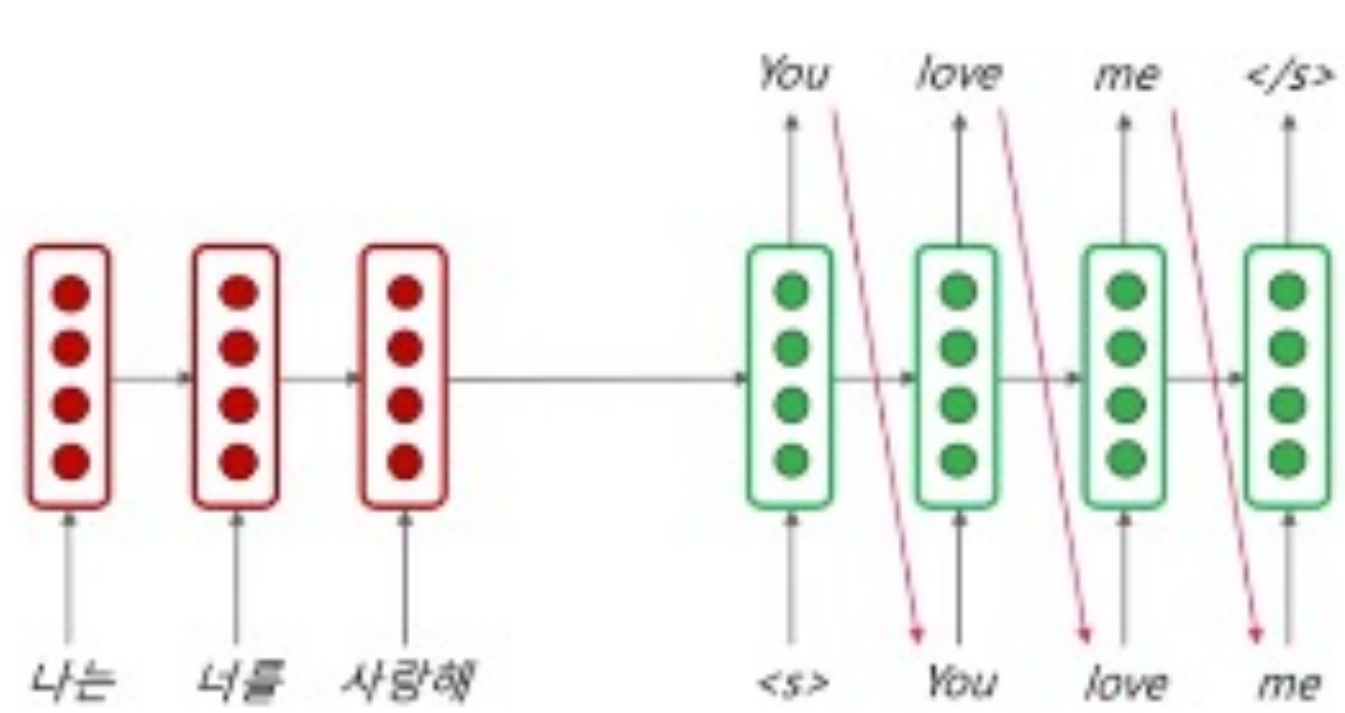
Seq2Seq - Training

Teacher Forcing

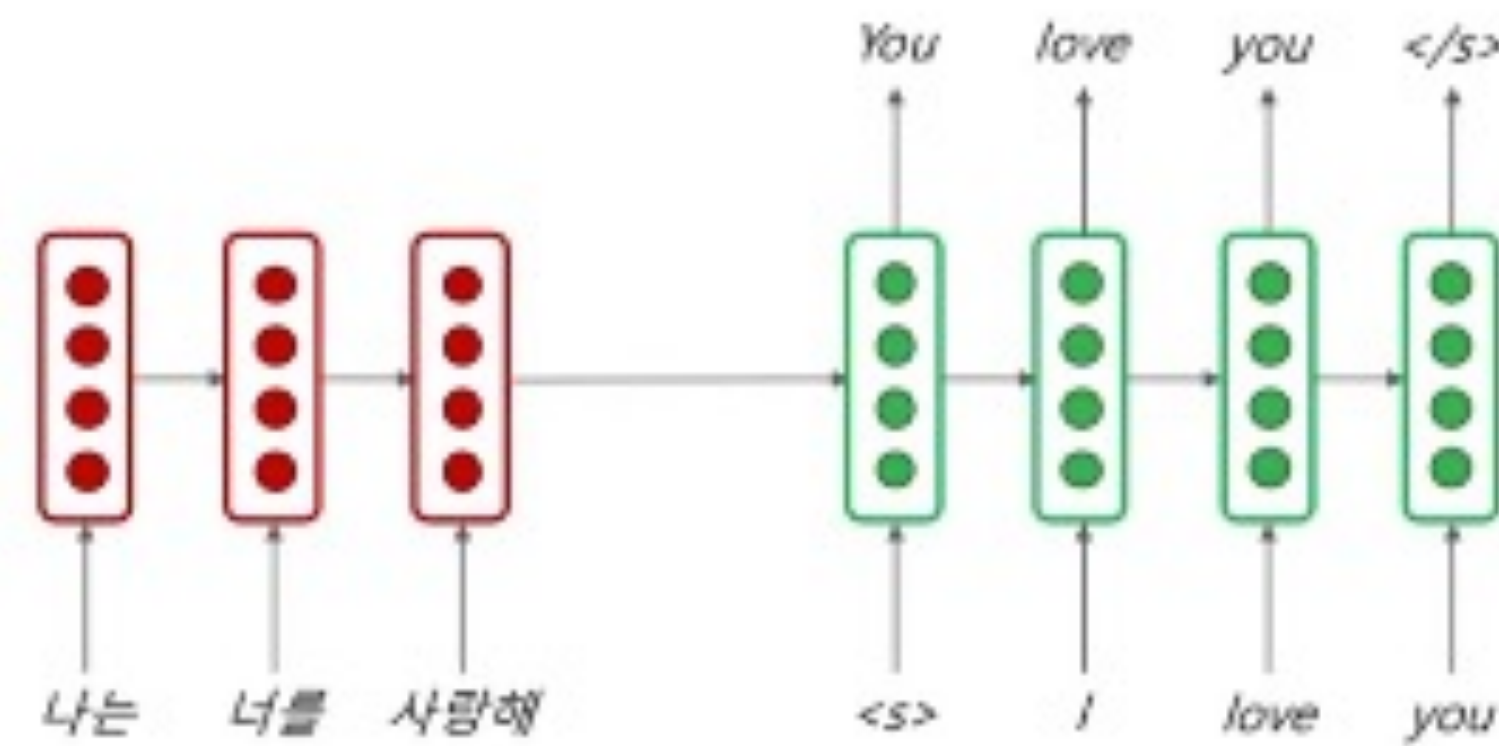
- 모델 학습 시 Decoder에서 실제 정답 데이터를 사용할 비율을 설정하는 하이퍼파라미터
- 초기에 잘못 생성한 단어로 인해 계속해서 잘못된 문장이 생성되어 학습이 잘 안되는 것을 방지
- 높게 설정할수록 빠른 학습이 가능, 하지만 학습데이터에 Overfit될 수도 있기 때문에, 적절히 설정
- Inference 단계에서는 ratio를 0으로 설정해, 모델이 생성한 단어만으로 다음 단어 추론

Seq2Seq - Training

Teacher Forcing



Teacher Forcing 사용 안 한 경우



Teacher Forcing 사용한 경우

Sequence-to-sequence is versatile!

Seq2seq은 기계 번역 뿐만 아니라 다른 면에서도 유용합니다.

1. Summarization : long text -> short text
2. Dialogue : previous utterances -> next utterances
3. Parsing : input text -> output parse as sequence
4. Code generation : natural language -> python code

Sequence-to-sequence

NMT의 세가지 방식

1. Exhaustive Search : 가능한 모든 경우의 문장에 대한 확률을 계산하는 방식
2. Greedy Decoding : 각 단계에서 가장 확률이 높은 단어만을 선택하는 방식
3. Beam Search Decoding : 각 단계에서 k개의 단어를 선택하는 방식

Exhaustive Search

$$\begin{aligned} P(y|x) &= P(y_1|x) P(y_2|y_1, x) P(y_3|y_1, y_2, x) \dots, P(y_T|y_1, \dots, y_{T-1}, x) \\ &= \prod_{t=1}^T P(y_t|y_1, \dots, y_{t-1}, x) \end{aligned}$$

가능한 모든 경우의 문장에 대한 확률을 계산하는 방식

-> 계산량이 많이 비효율적!

Greedy Decoding

각 단계에서 가장 확률이 높은 하나의 단어만을 선택하고 나머지 단어는 제거한다

[Greedy Decoding 의 문제점]

Greedy decoding has no way to undo decisions!

- Input: *il a m'entarté* (*he hit me with a pie*)
- → *he* _____
- → *he hit* _____
- → *he hit a* _____ (*whoops! no going back now...*)

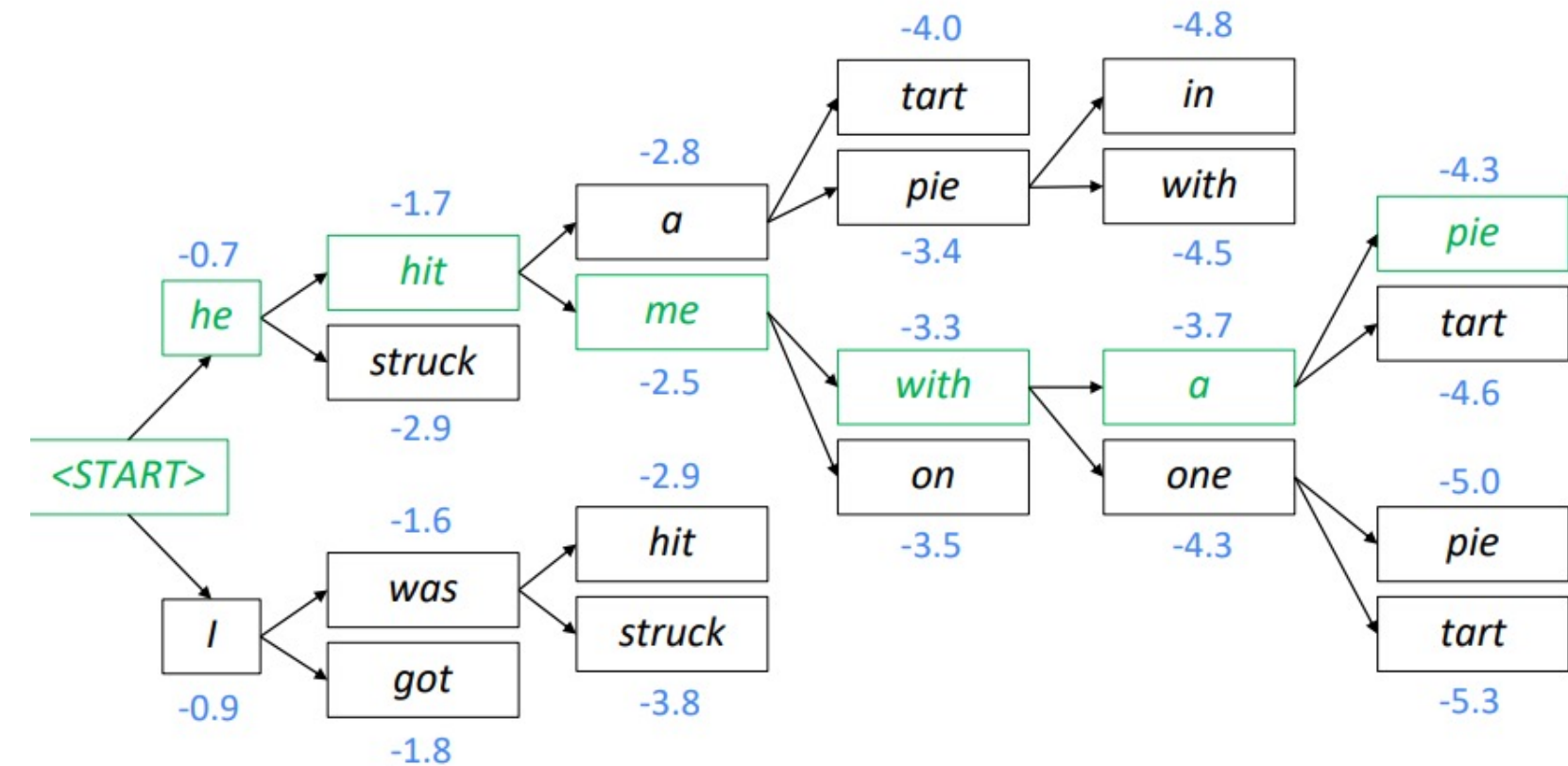
Beam Search Decoding

각 단계에서 확률이 높은 k개의 단어(k = beam size)를 선택한다

- 각 확률은 log가 취해진 확률이므로 음수의 값을 가지며, 0에 가까울수록 확률이 높은 것.
- 각 단계마다 확률이 높은 k개의 단어만 남기고 나머지는 제거한다.
- Greedy Decoding과 마찬가지로 optimal solution을 보장하지는 않는다.
- Exhaustive search 보다는 효율적이며, Greedy Decoding이 찾아내지 못하는 솔루션을 찾아낼 수 있다.

Beam search decoding: example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



Advantages/Disadvantages of NMT

[Advantage]

- SMT에 비해 좋은 성능 - fluent, 맥락에 더 잘 부합하며, 구문의 번역도 우수함
- Subcomponents로 쪼개졌던 SMT와는 달리 single neural network 단일모델로 최적화
- 인간의 엔지니어링을 덜 필요로 한다

[Disadvantage]

- 해석이 어렵다 (hard to debug)
- Control이 어렵다

BELU

- BELU(BiLingual Evaluation Understudy)
- Machine-written translation vs. human-written translation
- N-gram precision 기반으로 한 similarity score을 계산한다
- 유용하지만 불완전하다

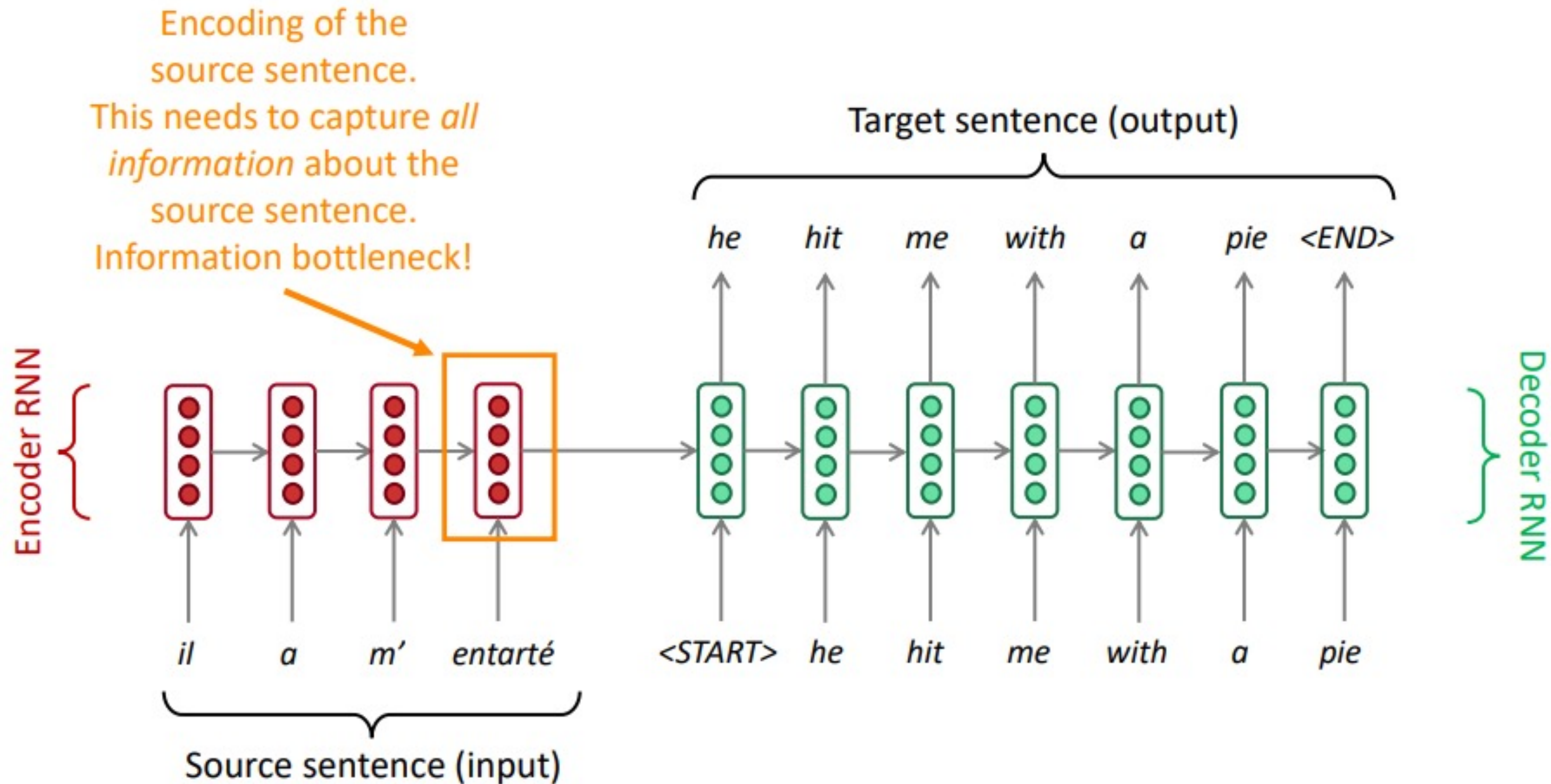
Difficulties

- Out-of-vocabulary words
- Domain mismatch between train and test data
- Maintaining context over longer text
- Low-resource language pairs
- Gender/race bias

Attention



The bottleneck problem



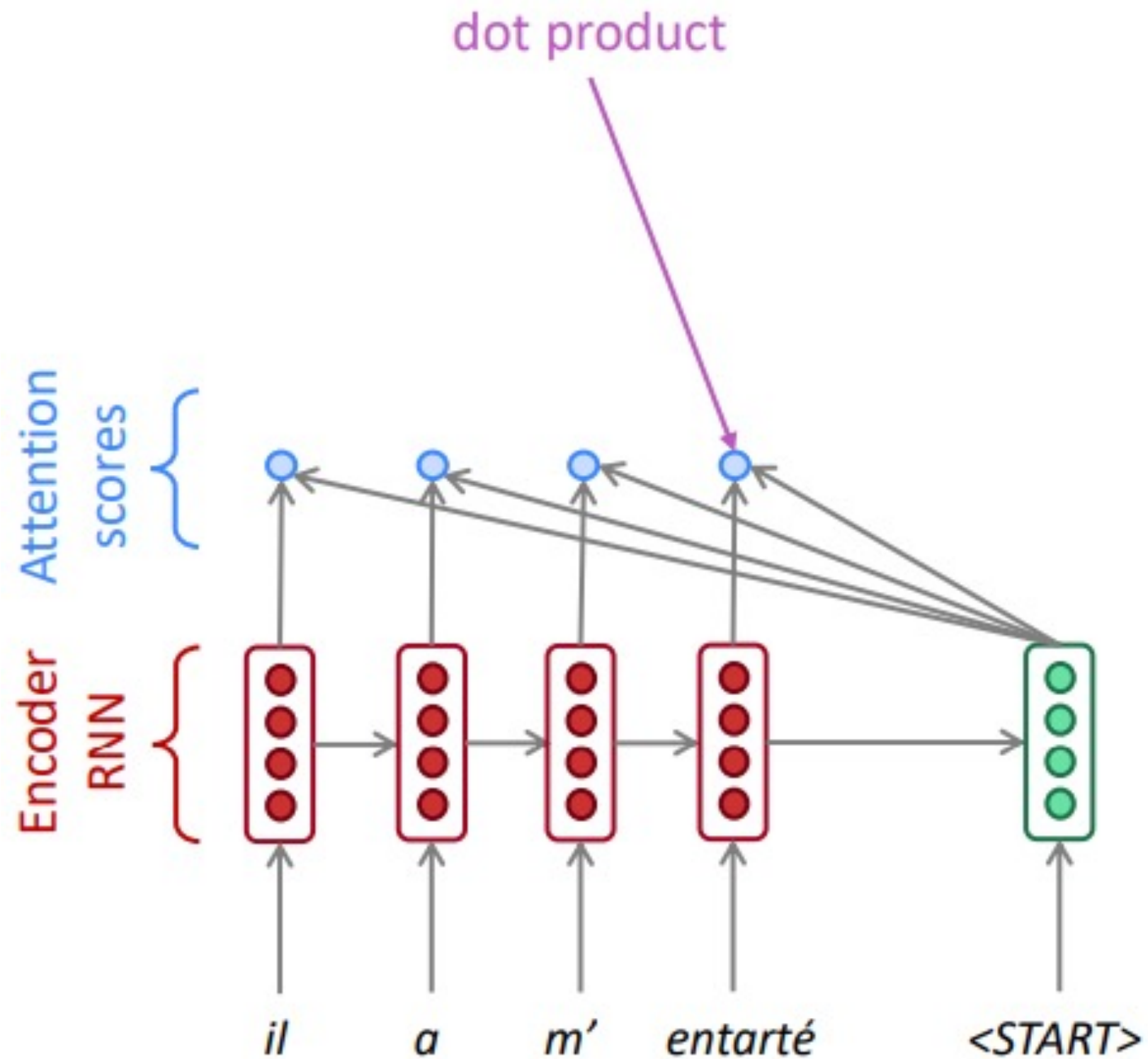
Attention

- Bottleneck problem의 해결
- Core idea : decoder의 각 단계마다 source sentence의 특정 부분에 집중하기 위해 encoder와 직접적으로 연결한다.

Attention

#01

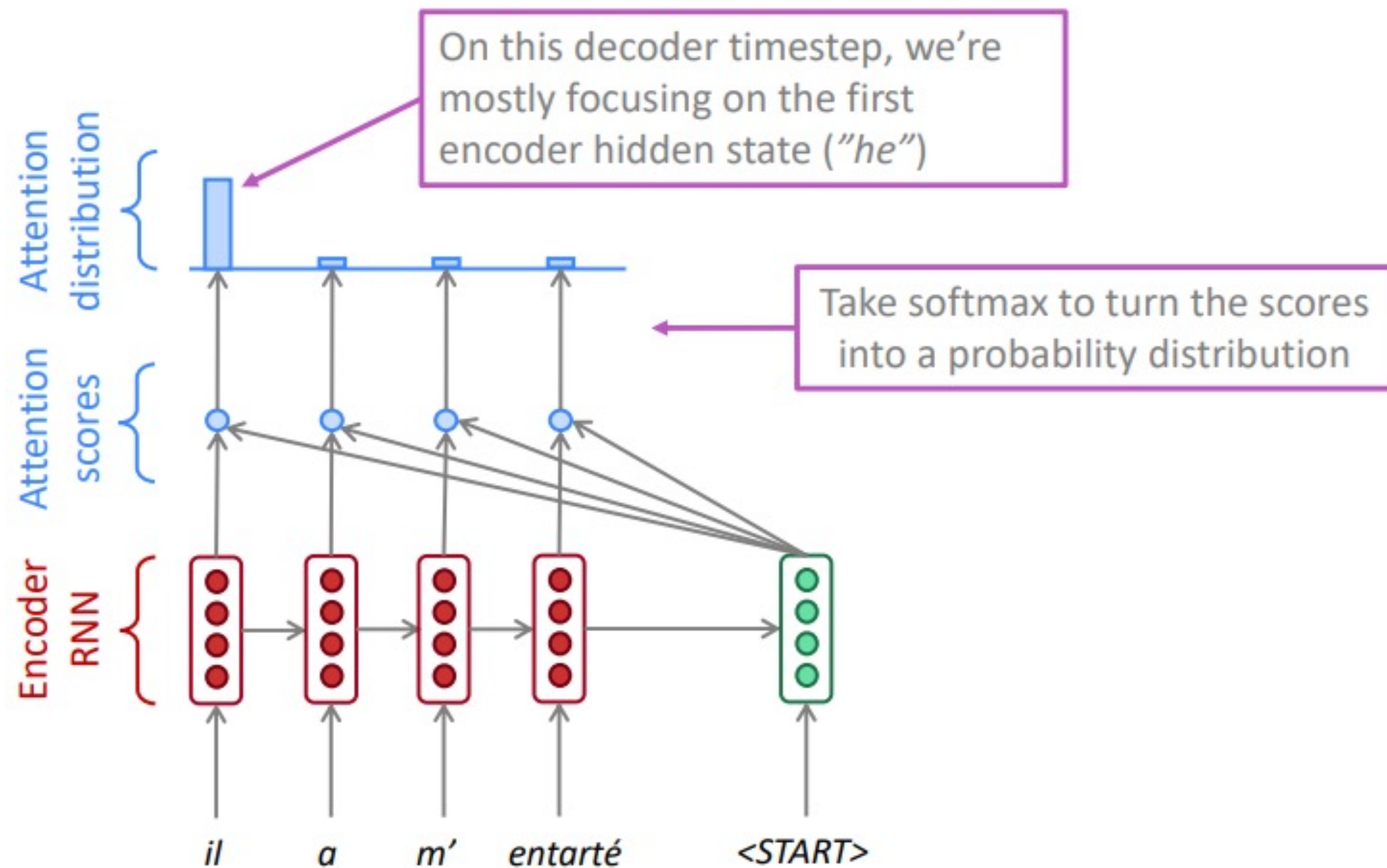
Decoder의 hidden state와
encoder의 hidden state를,
dot product로 attention score계산



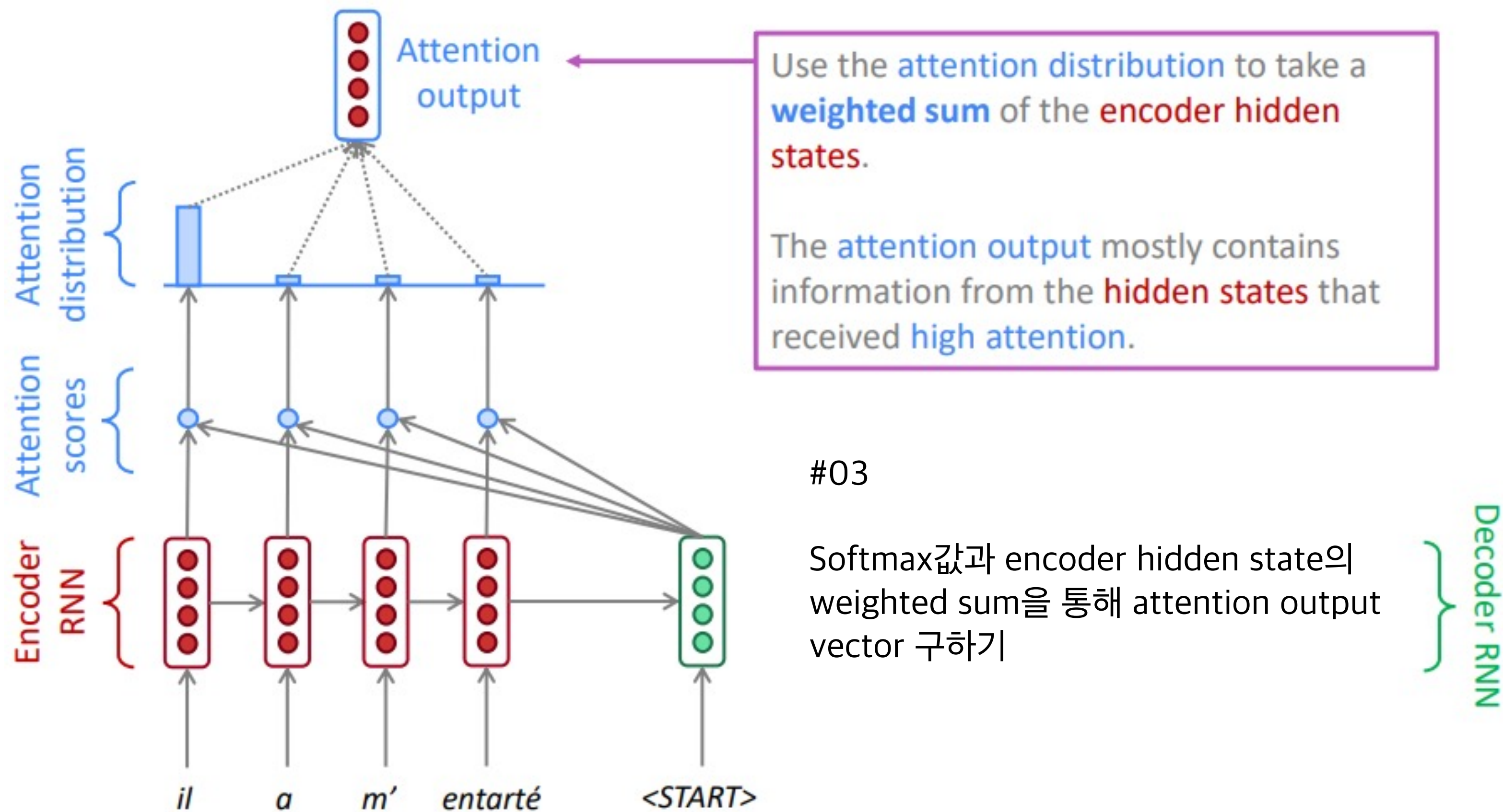
Attention

#02

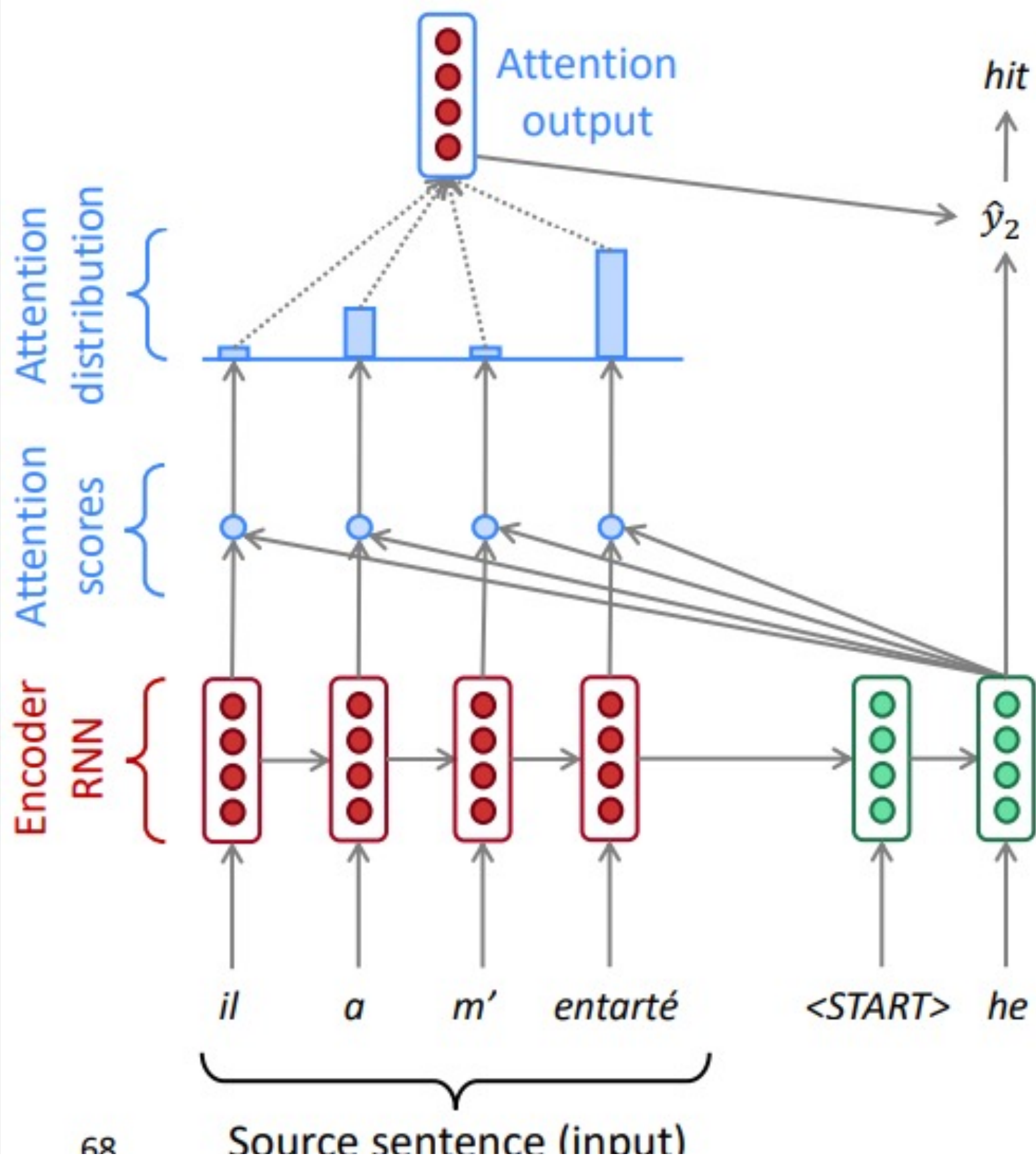
앞선 단계에서 구한 attention scores에
softmax 연산을 취해서
현재 단계에서의 attention score
확률분포를 구한다



Attention



Attention



#04

Attention output vector와 decoder hidden state를 concatenate한 벡터를 이용해 \hat{y}_2 를 구한다

Sometimes we take the **attention output** from the previous step, and also feed it into the decoder (along with the usual decoder input). We do this in Assignment 4.

Attention is great

- NMT 성능이 향상된다
- Bottleneck 문제가 해결된다
- Vanishing gradient problem을 완화한다
- 오류 추적 가능성

THANK YOU

