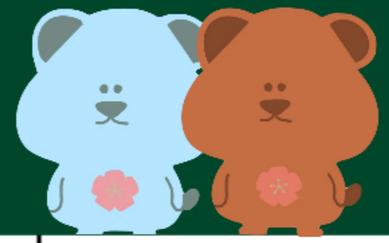
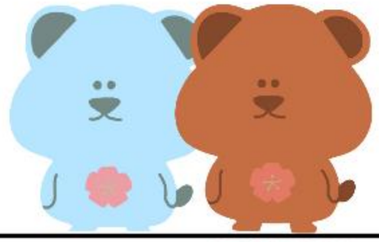


Lecture 7 - Translation, Seq2Seq, Attention

권재선 송민경

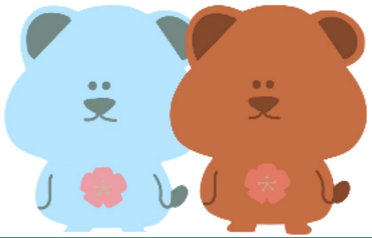
EWHA EURON

목차



EWHA EURON

01. Machine translation



EWHA EURON

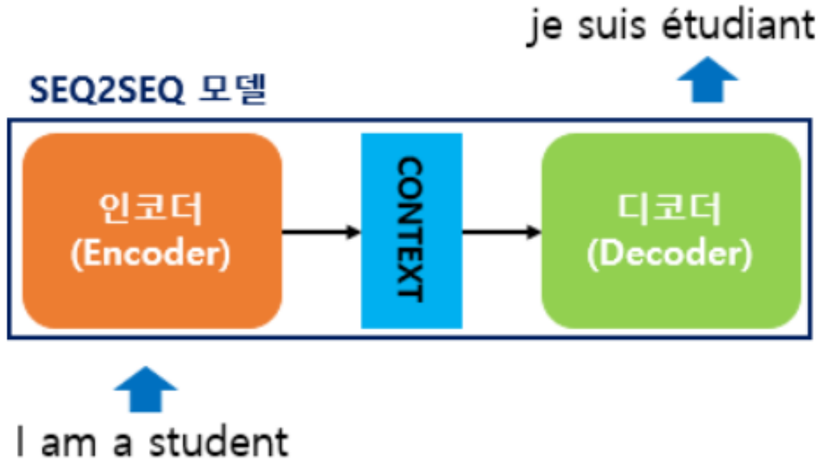
#01 Machine translation

a. 신경망 구조 : Sequence to Sequence

입력된 시퀀스로부터 다른 도메인의 시퀀스를 출력하는 다양한 분야에서 사용되는 모델

e.g. 챗봇(Chatbot)과 기계 번역(Machine Translation)

입력 시퀀스와 출력 시퀀스를 각각 질문과 대답으로 구성 -> 챗봇
입력 시퀀스와 출력 시퀀스를 각각 입력 문장과 번역 문장으로 만들면 -> 번역기
그 외에도 내용 요약(Text Summarization), STT(Speech to Text) 등에서 쓰임.



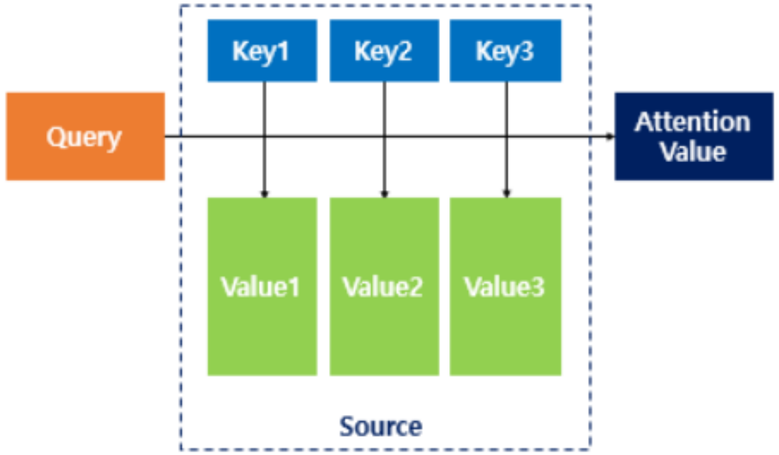
#01 Machine translation

a. 신경망 기법 : attention

기계 번역 분야에서 입력 문장이 길면 번역 품질이 떨어지는 현상 발생

-> 입력 시퀀스가 길어지면 출력 시퀀스의 정확도가 떨어지는 것을
보정해주기 위한 등장한 기법인 어텐션(attention)

어텐션의 기본 아이디어: 디코더에서 출력 단어를 예측하는 매 시점(time step)마다,
인코더에서의 전체 입력 문장을 다시 한 번 참고
단, 전체 입력 문장을 전부 다 동일한 비율로 참고하는 것이 아니라, 해당 시점에서 예측해야 할 단어와
연관이 있는 입력 단어 부분을 좀 더 집중(attention)해서 보게 됨.



#01 Machine translation

a. NLP task : 기계번역

NLP의 다양한 task : 텍스트 분류, 기계번역, 텍스트 요약, 등

->규칙 기반 기계 번역(Rule-Based Machine Translation)부터, 통계 기반 기계 번역(Statistical Machine Translation), 신경망 기계 번역(Neural Machine Translation)까지 발전을 이루어 왔고 뛰어난 성능을 보여주고 있음.

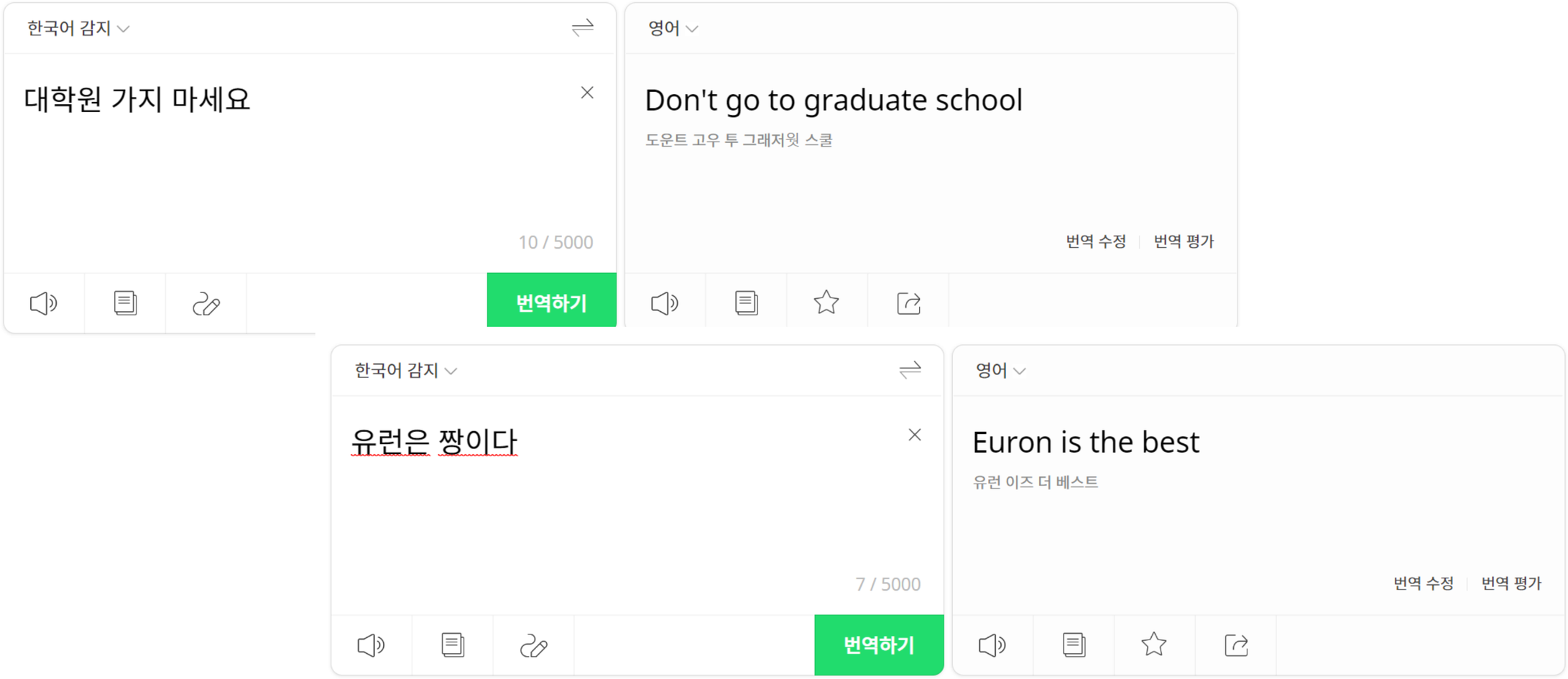
규칙 기반 기계 번역(Rule-Based Machine Translation)은 Source Text와 Target Text의 문법 규칙을 기반으로 번역하는 방법. Source Text의 형태소 분석, 구문 분석, 의미 분석, 화용 분석을 거쳐 이에 역순으로 Target Text를 생성. 언어학적 지식이 상당수 필요하다는 것이 가장 큰 단점

통계 기반 기계 번역(Statistical Machine Translation)은 사전 데이터를 바탕으로 Word Alignment를 진행해 얻을 수 있는 대역어 테이블에서 단어 혹은 구 단위로 묶어 번역 모델을 생성하는 과정을 거쳐 Source Text와 가장 비슷하다고 예측한 Source Text를 생성. 어순이 다른 언어 간 번역이 이루어질 때 성능이 부족하다는 것이 가장 큰 단점

신경망 기계 번역(Neural Machine Translation)은 데이터를 통해 학습된다는 점에서 통계 기반 기계 번역과 유사하다고 볼 수 있으나, 통계 기반 기계 번역이 사용하는 데이터보다 더욱 방대한 데이터를 통해 딥러닝한다는 점이 신경망 기계 번역의 특징. 데이터의 질과 양에 따라 언어별 번역 정확도가 다르다는 것이 단점.(영어와 한국어/태국어)

#01 Machine translation

- Machine translation? 컴퓨터 프로그램이 하나의 언어(source text)로 구성된 문서를 분석하여 다른 언어(대상 문서, target text)로 구성된 문서로 만드는 하나의 번역 형태



#01 Machine translation

발전 과정

1950s : Early Machine Translation
- 냉전시대 영향 (주로 Russian -> English)
- Rule Based (Bilingual dictionary 사용)

1990s -2010s : Statistical Machine Translation (SMT)
- Data를 통해 만든 확률적 모델 -> Bayes Rule 사용
- 서로 다른 두 언어의 Parallel 데이터로부터 모델 구축

2014s - : Neural Machine Translation (NMT)
- 단일 신경망을 사용
- 두개의 RNN을 포함한 Sequence to Sequence 구조

#01 Machine translation

#1 Machine translation

c. Statistical Machine Translation : Translation model + language model → 베이즈룰 적용

주어진 parallel 데이터로부터 확률모델을 구축하는 것이 목적!

EX) 불어 (F) -> 영어 (E) 번역

$$argmax_E P(E|F)$$

Bayes Rule ⇩

$$argmax_E P(F|E)P(E)$$

Translation Model

- 단어와 구가 어떻게 번역되어야 할지를 모델링
-> 번역 정확도

- Parallel Data로부터 학습

Language Model

- 좋은 문장을 생성하도록 모델링
- Monolingual Data로부터 학습

#01 Machine translation

#1 Machine translation

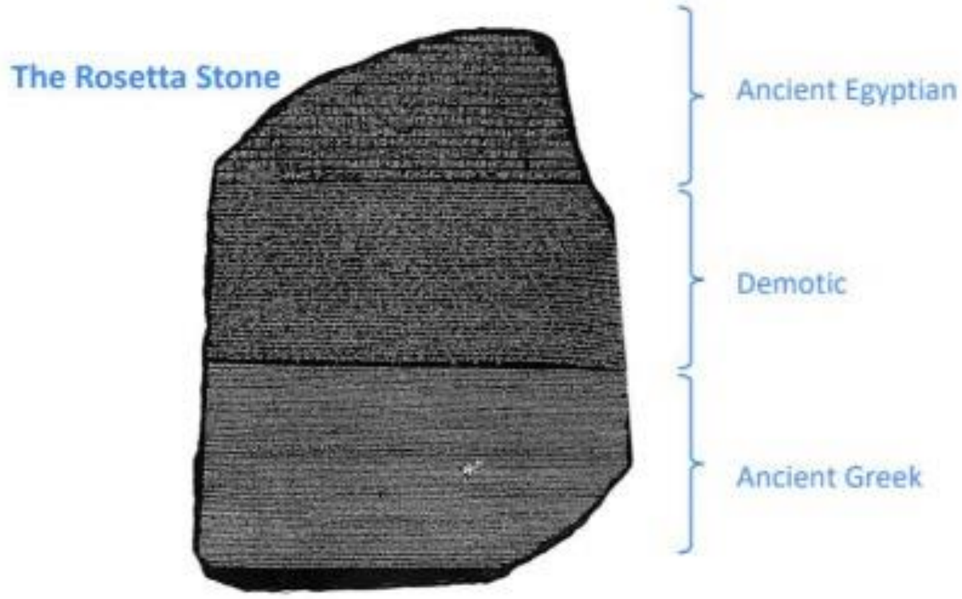
c. Statistical Machine Translation : Translation model + language model → 베이즈룰 적용

c-1. parallel data

번역 모델(translation model)의 학습은 대량의 parallel data를 통해 이루어짐

번역이 된 문장 쌍

e.g. 유런은 짱이야(한국어) & Euron is the best(영어)



#01 Machine translation

#1 Machine translation

c. Statistical Machine Translation : Translation model + language model → 베이즈룰 적용

c-1. alignment

parallel data -> Alignment

Source – Target 언어 문장 Pair에서 일치하는 단어 또는 구를 찾아서 짝을 지어주는 작업(Alignment)을 통해 Parallel로 부터 Translation모델을 학습시킴.

$$P(F|E) \rightarrow \sum_{a} P(F, a|\, E)$$

기본적으로 best A와 E를 EM-Algorithm을 통해 찾는 방식으로 학습

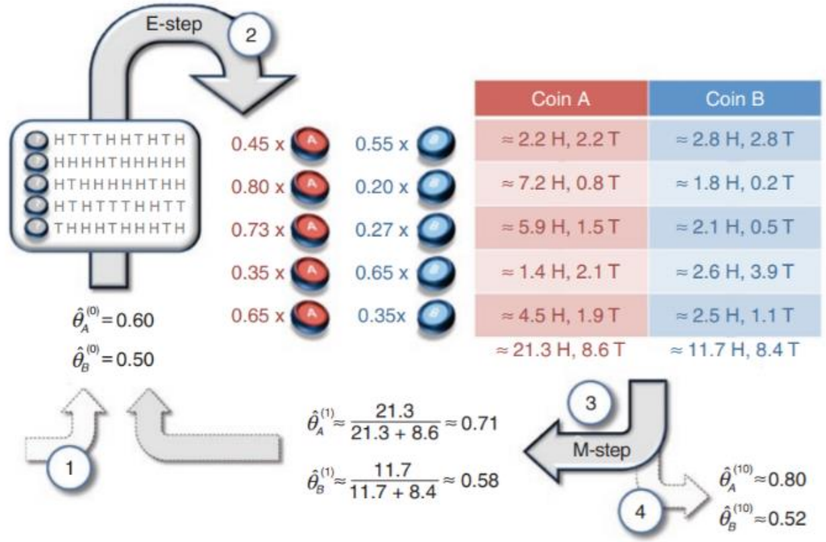
#01 Machine translation

#1 Machine translation

c. Statistical Machine Translation : Translation model + language model → 베이즈룰 적용

c-1. EM 알고리즘

EM 알고리즘 (기댓값 최대화 알고리즘(expectation-maximization algorithm, EM algorithm))
모수에 관한 추정 값으로 로그 가능도(log likelihood)의 기댓값을 계산하는 기댓값 (E) 단계와
이 기댓값을 최대화하는 모수 추정값들을 구하는 최대화 (M) 단계를 번갈아가면서 적용.
이 두 단계(2-step)를 번갈아 가며 최적화 값을 찾아가는 알고리즘



#01 Machine translation

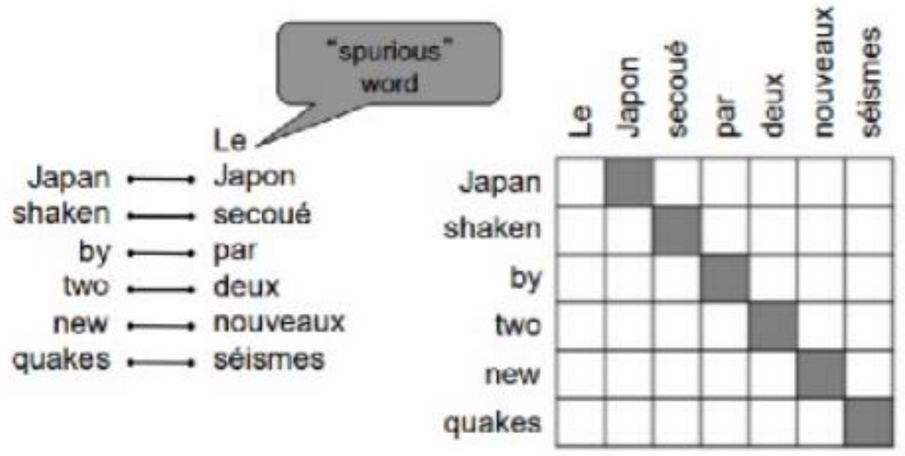
#1 Machine translation

c. Statistical Machine Translation : Translation model + language model → 베이즈룰 적용

c-1. Alignment 4가지 경우 예시

Alignment의 4가지 종류

1. No Counterpart



#01 Machine translation

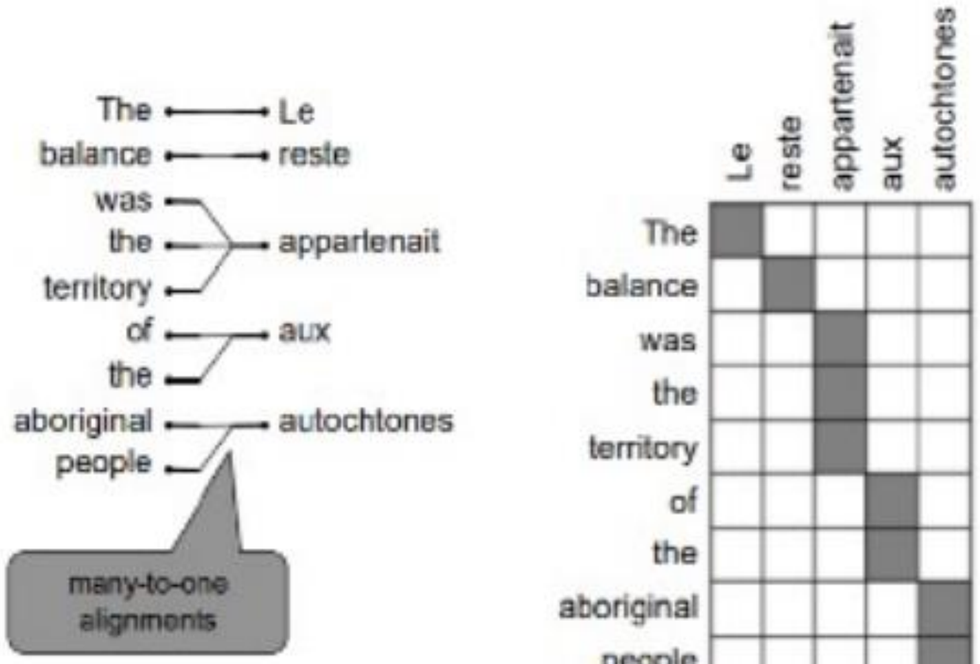
#1 Machine translation

c. Statistical Machine Translation : Translation model + language model → 베이즈룰 적용

c-1. Alignment 4가지 경우 예시

Alignment의 4가지 종류

2. Many to One



#01 Machine translation

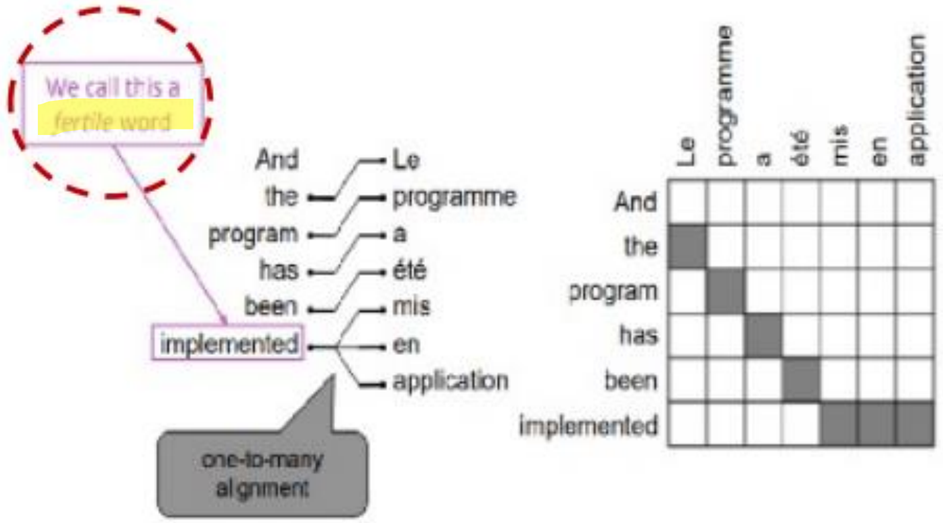
#1 Machine translation

c. Statistical Machine Translation : Translation model + language model → 베이즈룰 적용

c-1. Alignment 4가지 경우 예시 (One-to-many 에서 fertile 의 의미)

Alignment의 4가지 종류

3. One to Many



#01 Machine translation

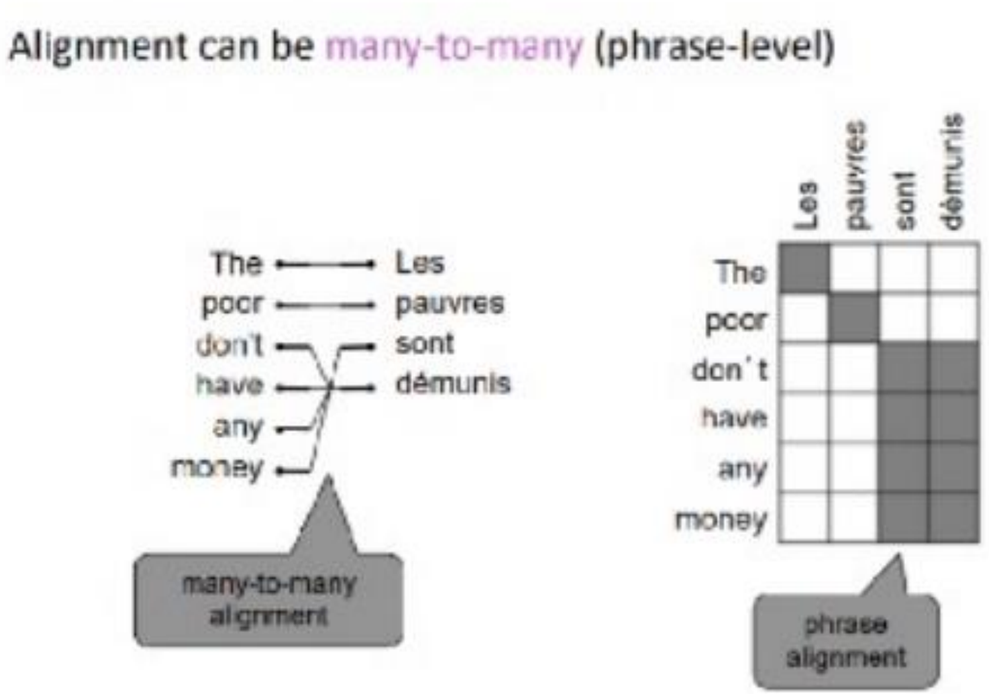
#1 Machine translation

c. Statistical Machine Translation : Translation model + language model → 베이즈룰 적용

c-1. Alignment 4가지 경우 예시

Alignment의 4가지 종류

4. Many to Many



#01 Machine translation

#1 Machine translation

c. Statistical Machine Translation : Translation model + language model → 베이즈룰 적용

c-2. decoding for SMT

Decoding for SMT: 많은 문장 후보들 중에서 어떤 문장이 가장 좋을지 찾는 것

$$argmax_E P(F|E)P(E)$$

1. 무차별 대입: 가능한 E에 대해 모든 확률 계산 -> 비용 너무 큼
2. Heuritic 알고리즘: Beam Search와 같이 너무 낮은 확률을 갖는 것들은 제외

#01 Machine translation

#1 Machine translation

c. Statistical Machine Translation : Translation model + language model → 베이즈룰 적용

c-3. SMT 의 단점 → 이를 극복한 NMT의 등장

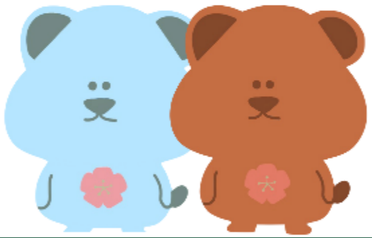
-1. 너무 복잡: 많은 숫자의 subcomponents가 각각 다르게 학습되므로 복잡도 높음(그래도 좋은 성능)
-2. Feature engineering이 많이 수행되어야 함 -> 언어 현상들을 표현할 수 있는 여러 feature들을 디자인
-3. Compile 및 유지하는 데에도 많은 비용 소요
-4. 사람의 손(human effort)을 많이 필요로 함.

→NMT

-1. 하나의 Neural Net 모델로 Machine Translation 수행
→ 여러 개의 Subcomponent로 분리하여 구성하지 않아도 되는 장점
-2. 기존의 SMT보다 높은 성능 보임

→현재는 대부분 NMT 기반의 번역기를 사용

02. Sequence to Sequence

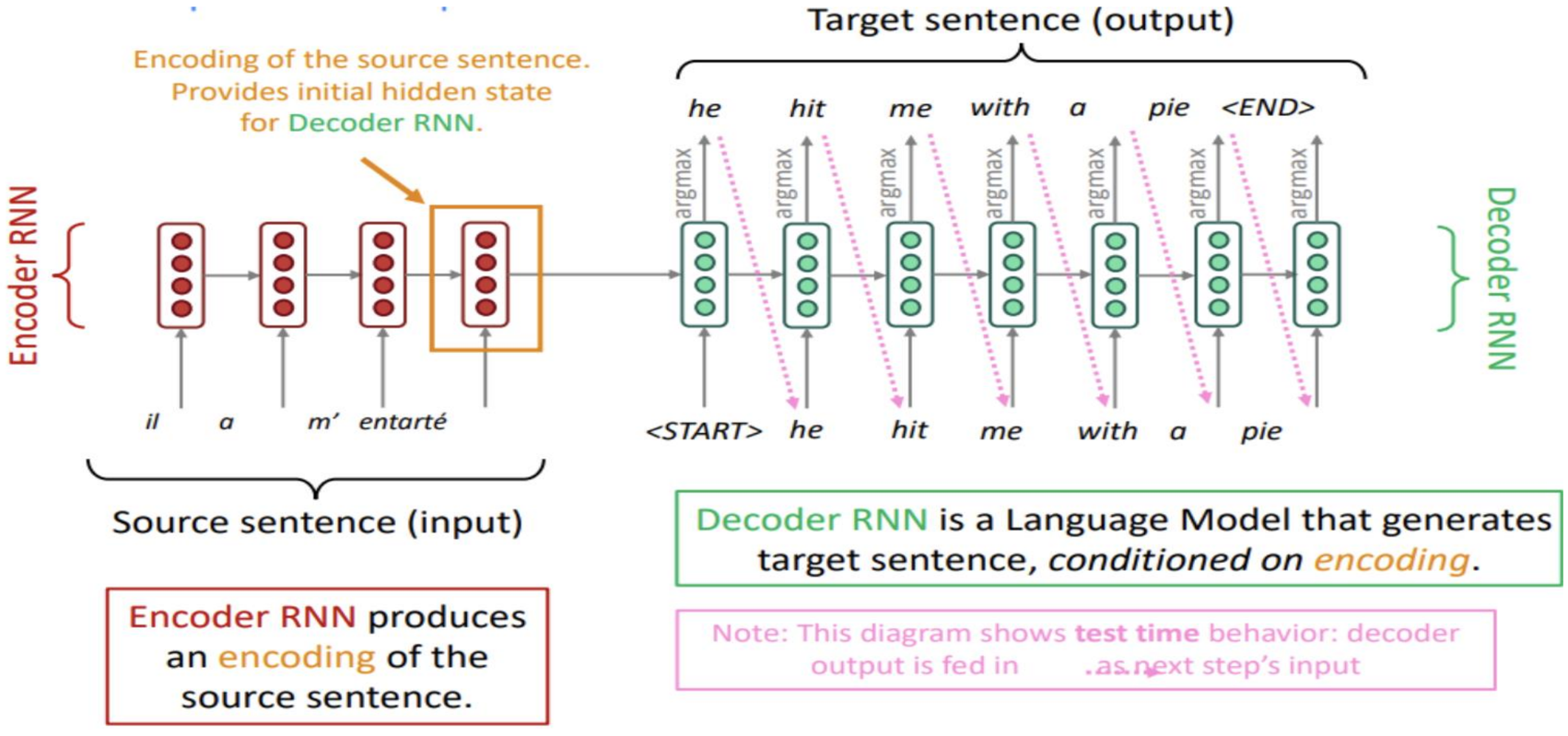


EWHA EURON

2. Sequence-to-sequence

#1 What is Neural Machine Translation(NMT)?

• NMT는 단일 end-to-end 신경망을 이용하여 기계 번역을 수행하는 방법으로, 신경망 아키텍처는 sequence-to-sequence 모델이라 불리고, 2개의 RNN으로 구성된다.



1) source 문장을 Encoder RNN에 넣어 생성된 hidden state를 Decoder RNN에 입력으로 넣음
2) Decoder RNN의 경우 시작토큰 <START>와 함께 두 개의 입력을 받아서 가장 단어를 추론
3) 그 단어를 또 다음 입력으로 넣어주어 문장을 생성

2. Sequence-to-sequence

The sequence-to-sequence model

* Use-Case
1. Summarization (긴 텍스트 → 짧은 텍스트)
2. Dialogue (이전 대화 → 다음 대화)
3. Parsing (입력 텍스트 → parsing 방법)
4. Code 생성 (자연어 → code)
5. Time series
6. Voice generation

* NMT : decoder에서 target 문장의 다음 단어 예측, source 문장에 조건부 이므로 conditional probability 직접 계산 (w/0 Bayes Rule)

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

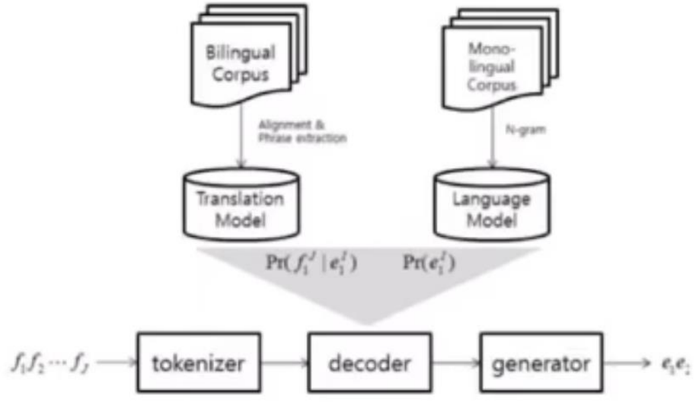
Probability of next target word, given target words so far and source sentence *x*

*End-to-end 학습이 가능
* NMT 학습을 위해 필요한 것? parallel corpus가 필요

EWHA EURON

2. Sequence-to-sequence

Statistical Machine Translation (SMT)

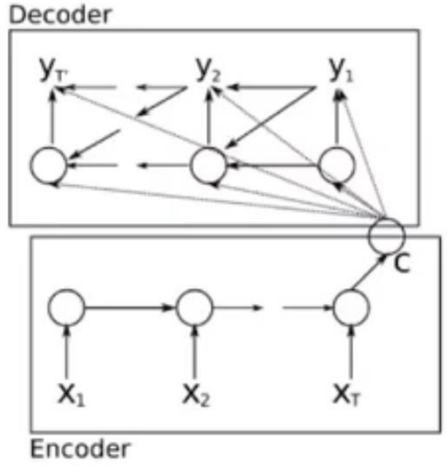


$$\text{argmax}_y P(x|y)P(y)$$

- 대용량 corpus 로부터 학습된 통계정보 활용
- 번역모델과 언어모델로 각각 나누어서 번역 수행
- 한계점:

  feature engineering 의 중요도 매우 높음 & complex

Neural Machine Translation(NMT)



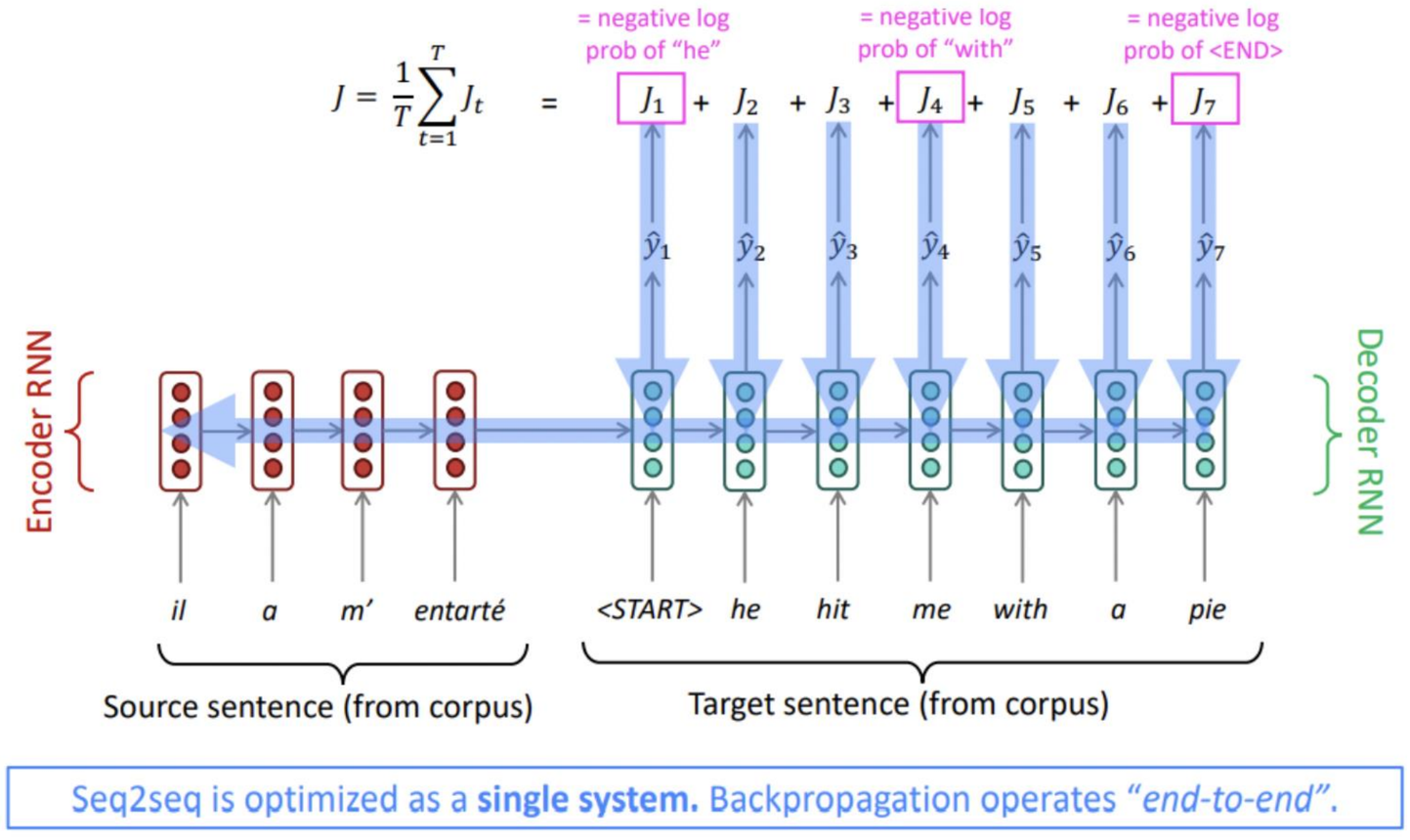
$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots P(y_T|y_1,\ldots,y_{T-1},x)$$

- source 문장에서 target 문장의 다음 단어가 나타날 확률을 single neural network 로 예측
- Encoder 와 decoder 의 구조로 이루어짐
- 필요한 데이터: ONLY parallel corpus

EWHA EURON

2. Sequence-to-sequence

#2 Training a Neural Machine Translation system



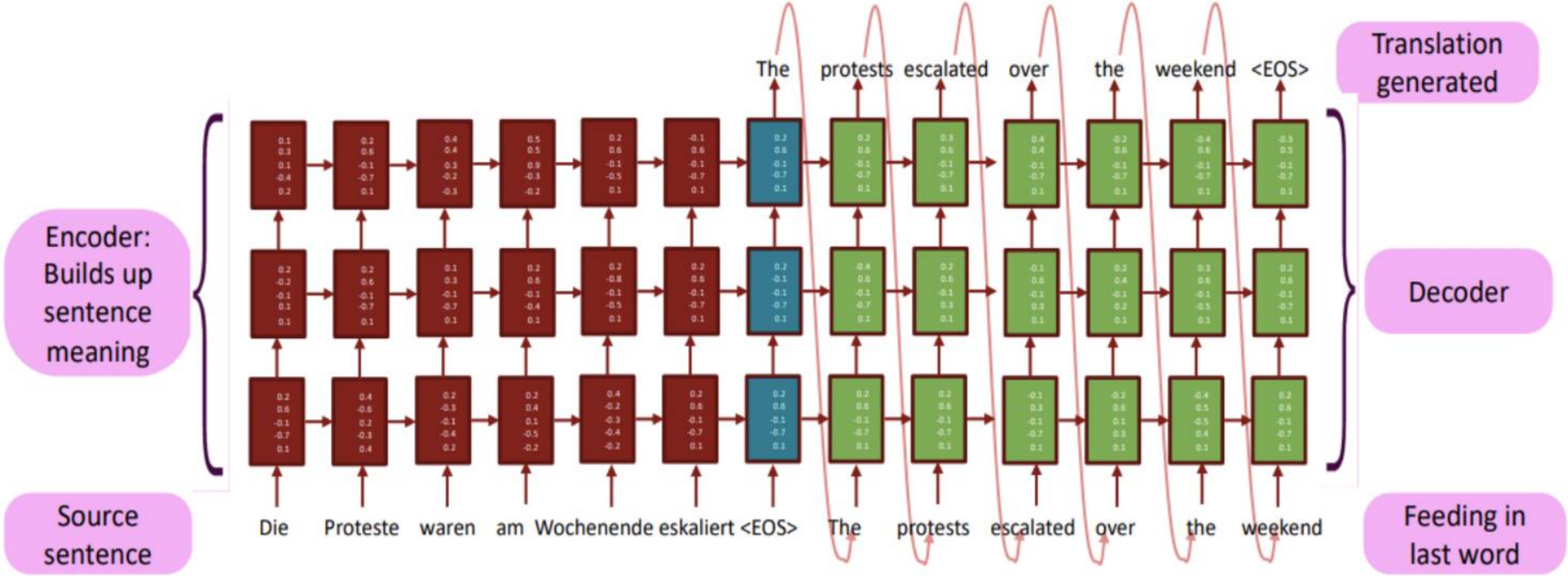
Seq2seq is optimized as a **single system.** Backpropagation operates *"end-to-end"*.

1) 시스템에서 예측한 단어와 실제 단어의 교차 엔트로피를 손실로서 구하고 teacher forcing으로 각 단어들에 대한 손실을 모두 구함

2) 구한 모든 손실의 평균을 통해 역전파를 수행하는데 이 때 역전파는 decoder 뿐아니라 encoder의 매개변수까지 업데이트

2. Sequence-to-sequence

#3 Multi-layer RNNs

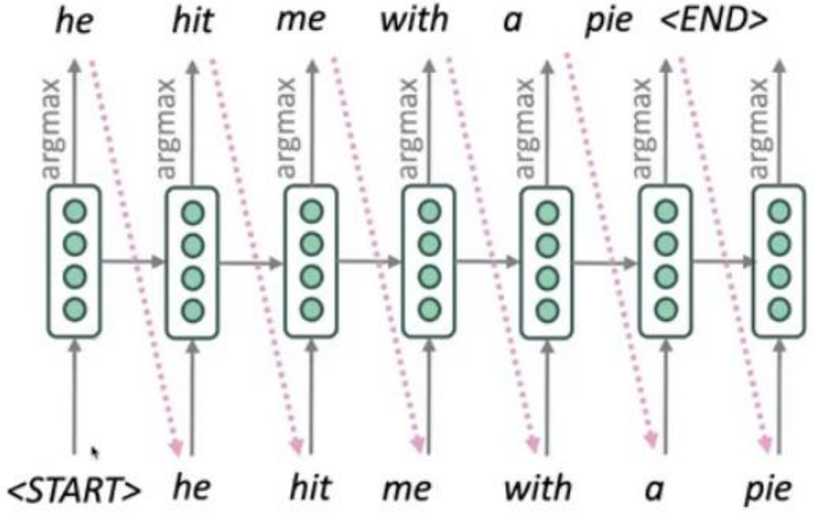


1) Timestep의 진행과 평행하게 layer 추가하여 또 다른 차원에 대해 깊게 적용 가능
2) 복잡한 표현의 계산을 가능하게 하므로 좀더 higer-level feature 학습하도록 함
3) 좋은 성능을 내는 RNN 모델 : 대부분 Multi-layer RNNs 구조 사용
   - Encoder with 2-4 layers, Decoder with 4 layers (2017 Britx et al.의 논문)
   - Transformer-based: 12 or 24 layers

2. Sequence-to-sequence

#4 Greedy decoding

Decoder 에서 target 문장 생성시 : Hidden state로부터 argmax로 가장 확률 높은 단어 뽑음 : Greedy Decoding
(각 단계에서 가장 확률이 높은 것을 취함)



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

Problems with greedy decoding:
한 스탭에서 단어를 잘못 추론했을 경우 되돌아갈 수 없음

2. Sequence-to-sequence

#5 Exhaustive search decoding

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

이상적으로 우리는 위 수식값 (최대 확률 문장으로 가는 단어 y를 구하고 싶음)
⇒ 모든 가능한 나열 계산하기

: decoder의 각 단계 t에서 vocab 크기 V^t 개의 가능한 모든 번역을 수행
복잡도 : O(V^t)로 너무 큰 비용이 듬

2. Sequence-to-sequence

#5 Exhaustive search decoding

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

이상적으로 우리는 위 수식값 (최대 확률 문장으로 가는 단어 y를 구하고 싶음)
⇒ 모든 가능한 나열 계산하기

: decoder의 각 단계 t에서 vocab 크기 V^t 개의 가능한 모든 번역을 수행
복잡도 : O(V^t)로 너무 큰 비용이 듬

2. Sequence-to-sequence

#6 Beam search decoding

핵심 아이디어: 각 step에서 k개의 가장 그럴듯한 부분 번역들을 계속 추적한다면 어떨까? (**hypotheses**)

- k: beam size임(일반적으로 5~10)
- hypotheses y1,...,yty1,...,yt 의 점수는 log 확률을 가짐

$$score(y_1, \ldots, y_t) = logP_{LM}(y_1, \ldots, y_t|x) = \sum_{i=1}^{t} logP_{LM}(y_i|y_1, \ldots, y_{i-1}, x)$$

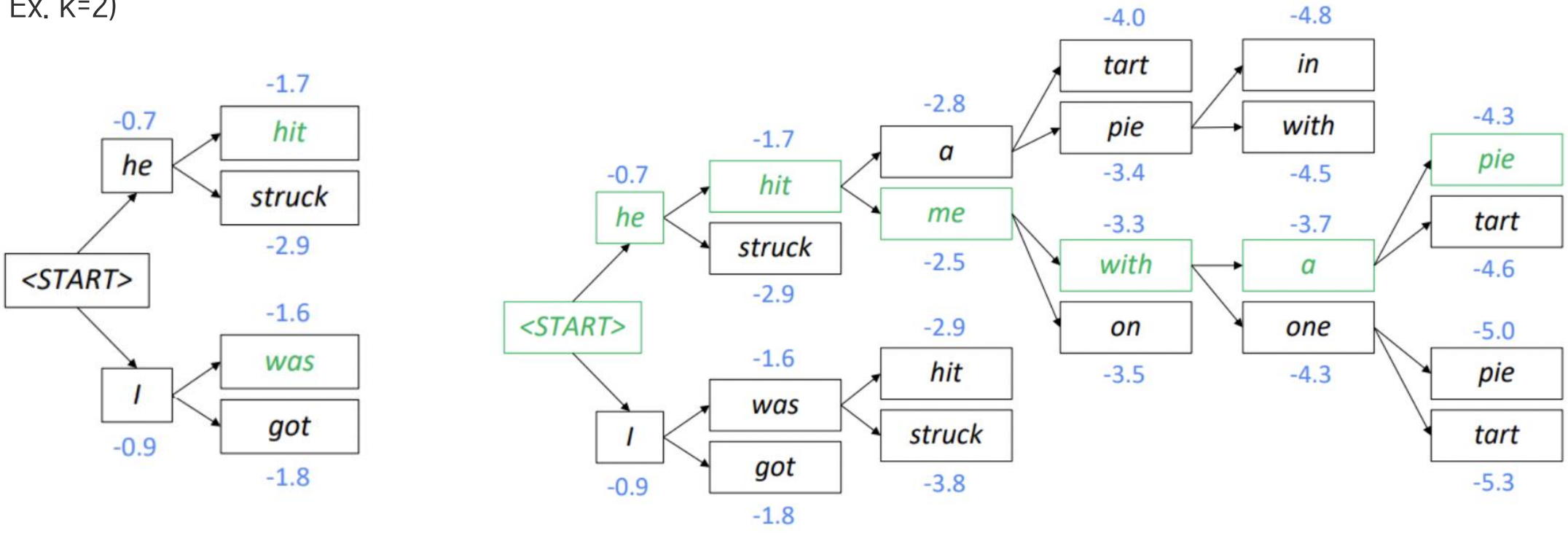
즉 모든 점수는 음수이고, 높을수록 좋음
⇒ 결과적으로 가장 높은 점수 k개를 각 단계에서 계속 추적 !

이 방법은 최적의 해를 찾아주는 방법은 아니지만 exhaustive search 방법보다는 훨씬 효율적임.

2. Sequence-to-sequence

#6 Beam search decoding

Ex. k=2)



<START>로 시작, he 와 I 추론
-> 각각 2개씩 추가로 단어 추론
이 step에서 he hit (-1.7), I was (-1.6)이 점수가 높기 문에 이 두부분 번역 계속 추적

2개의 점수가 높은 문장 계속 추적해 나가면 오른쪽과 같이 he hit me with a pie라는 문장 도출!

2. Sequence-to-sequence

#6 Beam search decoding

중단 기준

greedy decoding에서 코드 중단 기준은 <END> 코튼이 생성될 때
예) <START> he hit me with a pie <END>

- 그러나 beam search decoding에서는 다양한 hypotheses가 다양한 timestep에서 <END> 토큰을 생성할 것
- 그래서 hypotheses가 <END>를 생성하면, 해당 hypothses는 생성이 완료된 것으로 간주
- 해당 hypotheses는 따로 저장해두고 나머지 hypotheses의 탐색을 지속
- 결과적으로 특정 max timestep T에 도달할 때까지 지속하거나 최소 n개의 완성된 hypotheses가 나올 때가지 탐색을 지속

종료

완료된 hypotheses 목록을 가지고 누가 가장 높은 점수를 가지는지 비교

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

- 수식에서 확인할 수 있듯이 긴 문장이 작은 점수를 가질 수 밖에 없음
=> 단어수로 일반화를 시켜준 후 가장 높은 점수의 문장을 선택!

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

2. Sequence-to-sequence

#7 Advantages and Disadvantages of NMT

SMT와 비교 하여..

장점

1) 더 좋은 성능
 - more fluent, Better use of context, Better use of phrase similarities

2) Single neural network가 end-to end로 최적화됨
- Subcomponent 개별로 최적화 시킬 필요 없음

3) 사람의 engineering effort 적게 필요함

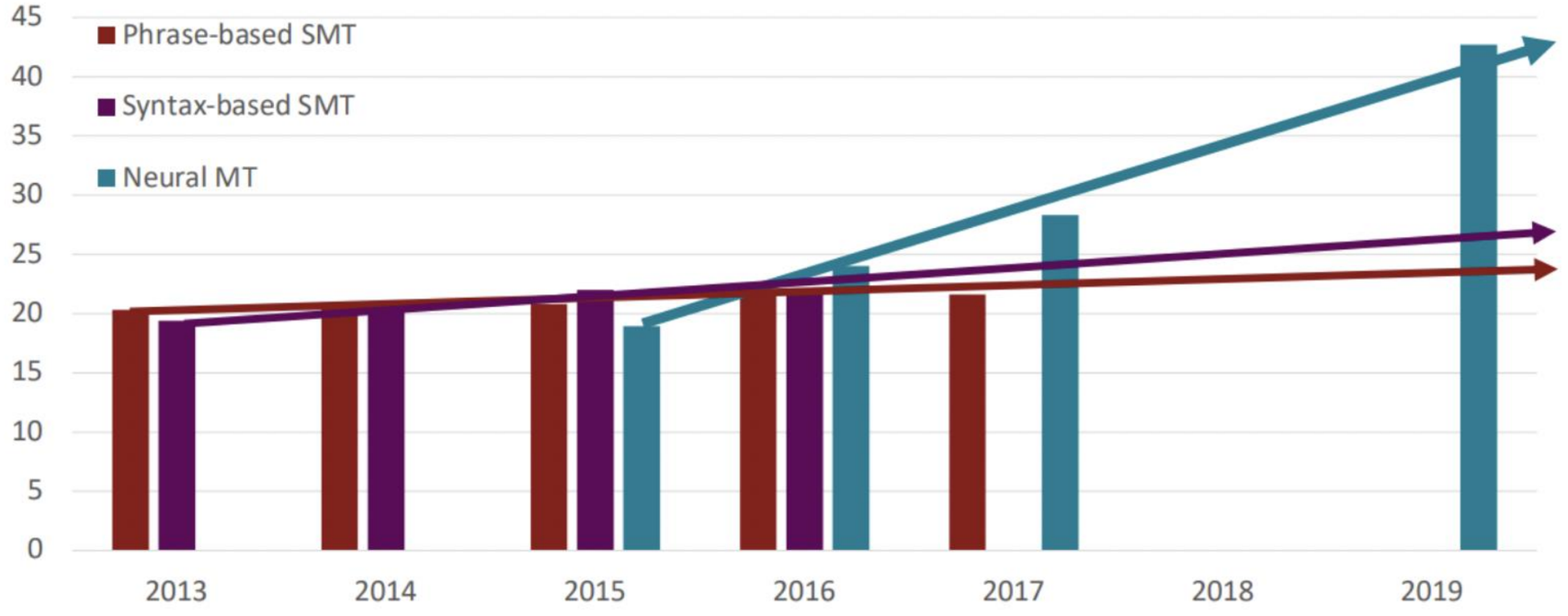
단점

1) Less interpretable : Debug 어려움

2) 조절하기 힘듬
- 번역을 위한 rules 나 guidelines을 특정하기 어려움
- Safety concerns

2. Sequence-to-sequence

#8 MT progress over time



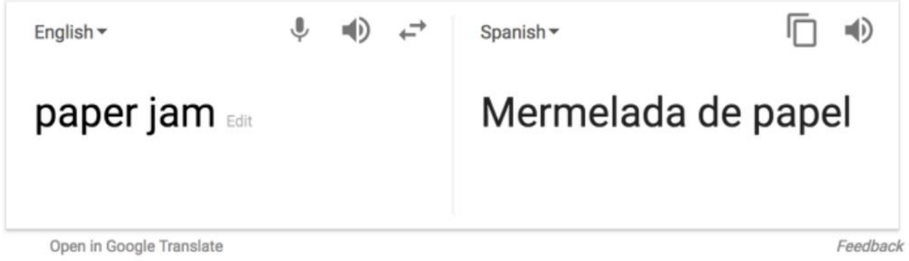
NMT는 2014년 이후 주력 방법으로 자리 잡았으며 2016년 구글 번역도 SMT에서 NMT로 전환
2018년 모든 번역을 하는 기업들을 NMT를 사용

2. Sequence-to-sequence

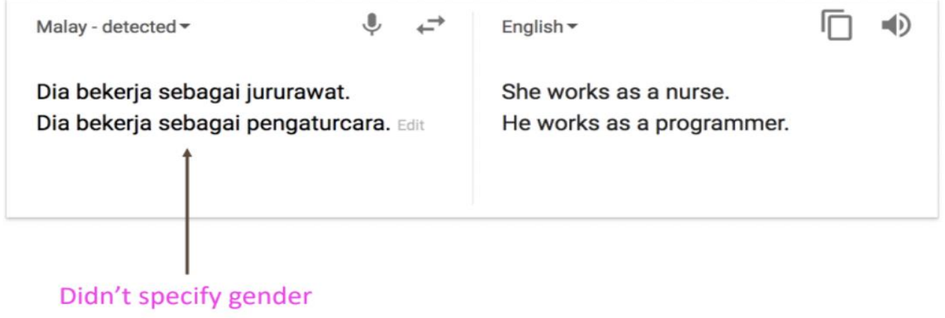
#9 Is Machine Translation solve?

- **Nope!**
- Many difficulties remain:
  - Out-of-vocabulary words
  - Domain mismatch between train and test data
  - Maintaining context over longer text
  - Low-resource language pairs
  - Failures to accurately capture sentence meaning
  - Pronoun (or zero pronoun) resolution errors
  - Morphological agreement errors

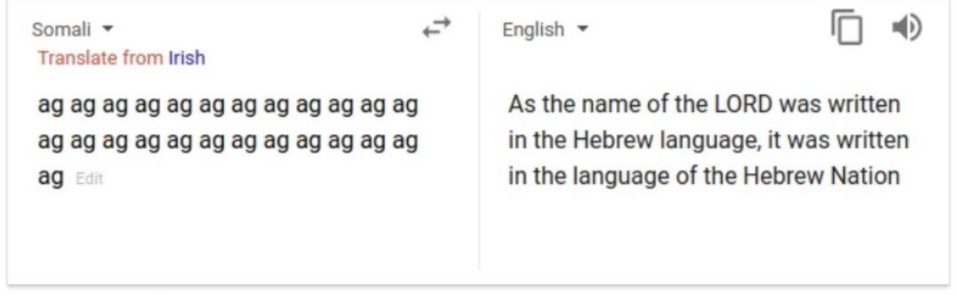
관용표현



Bias 학습



Uninterpretable(해석 불가능 시스템일 때의 이상행동)



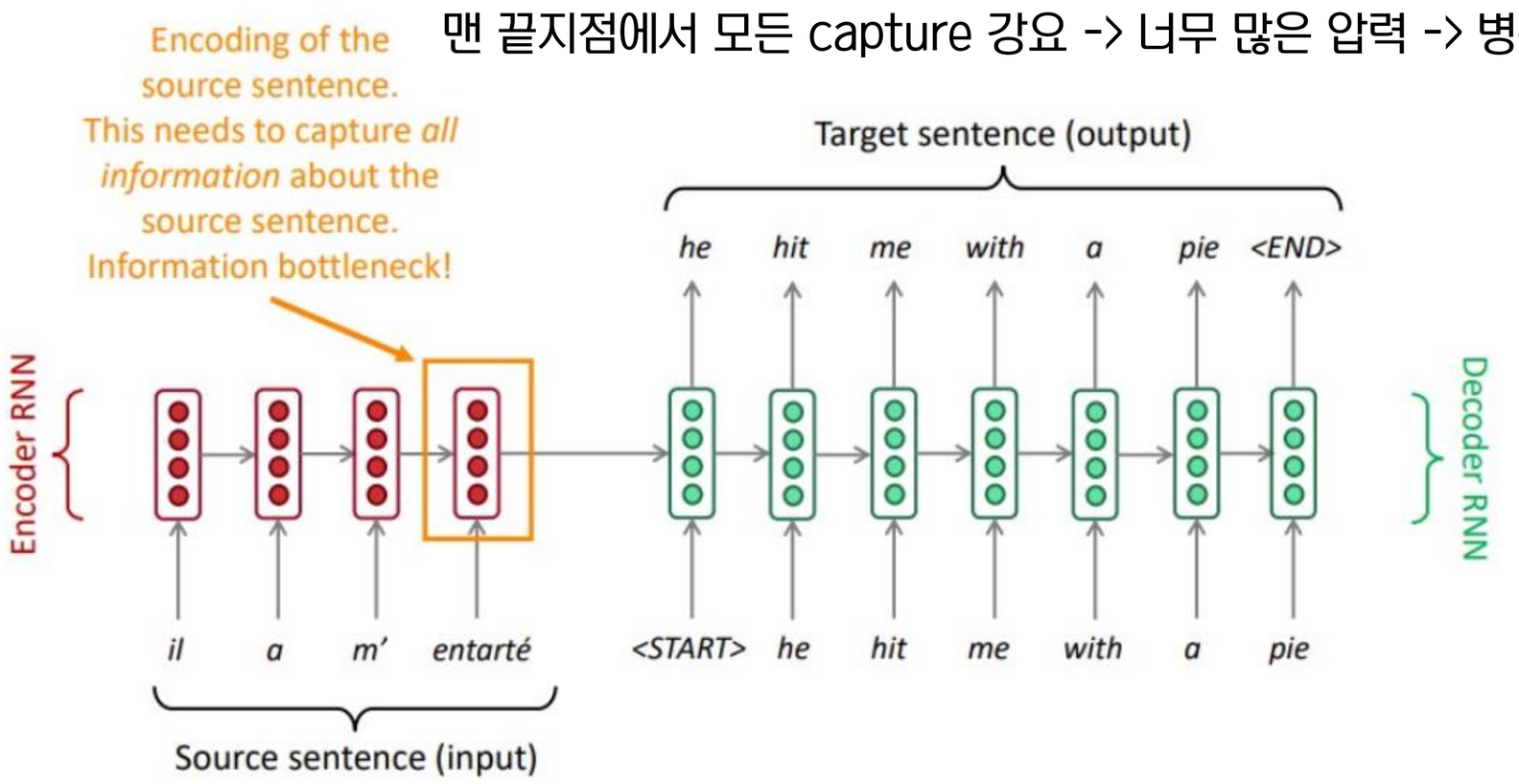
03. Attention

EWHA EURON

#03 Attention

#1 information bottleneck problem

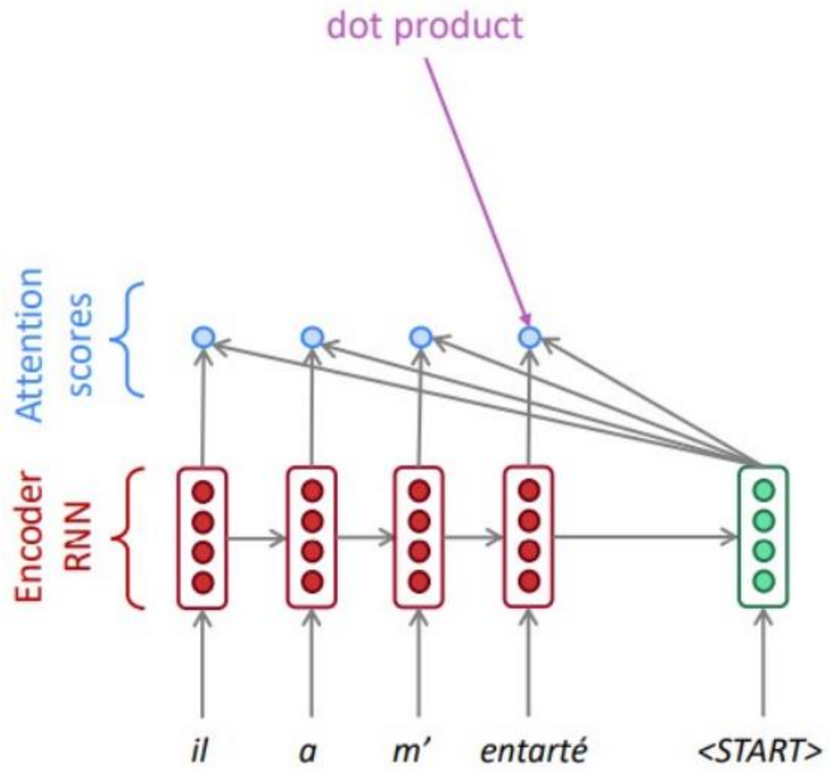
맨 끝지점에서 모든 capture 강요 -> 너무 많은 압력 -> 병목문제



→decoder의 각 단계마다 source sentence의 특정 부분에 집중하기 위해 encoder와 직접적으로 연결

#03 Attention

#2 Attention 개념/과정 : attention score, attention weight, output, concat with attention

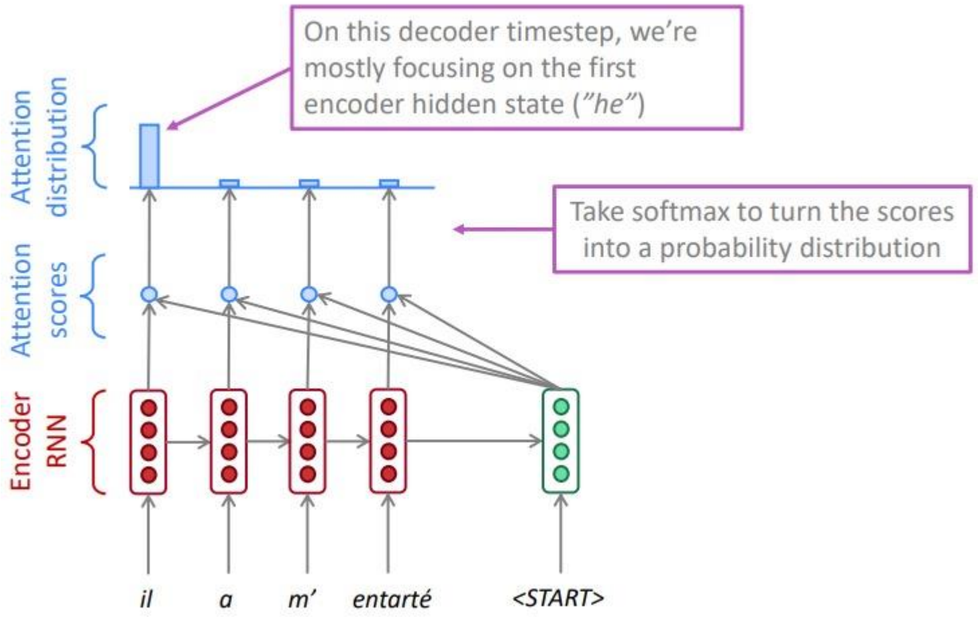


attention score

: Decoder의 hidden state와
encoder의 hidden state를,
dot product로 attention score계산

#03 Attention

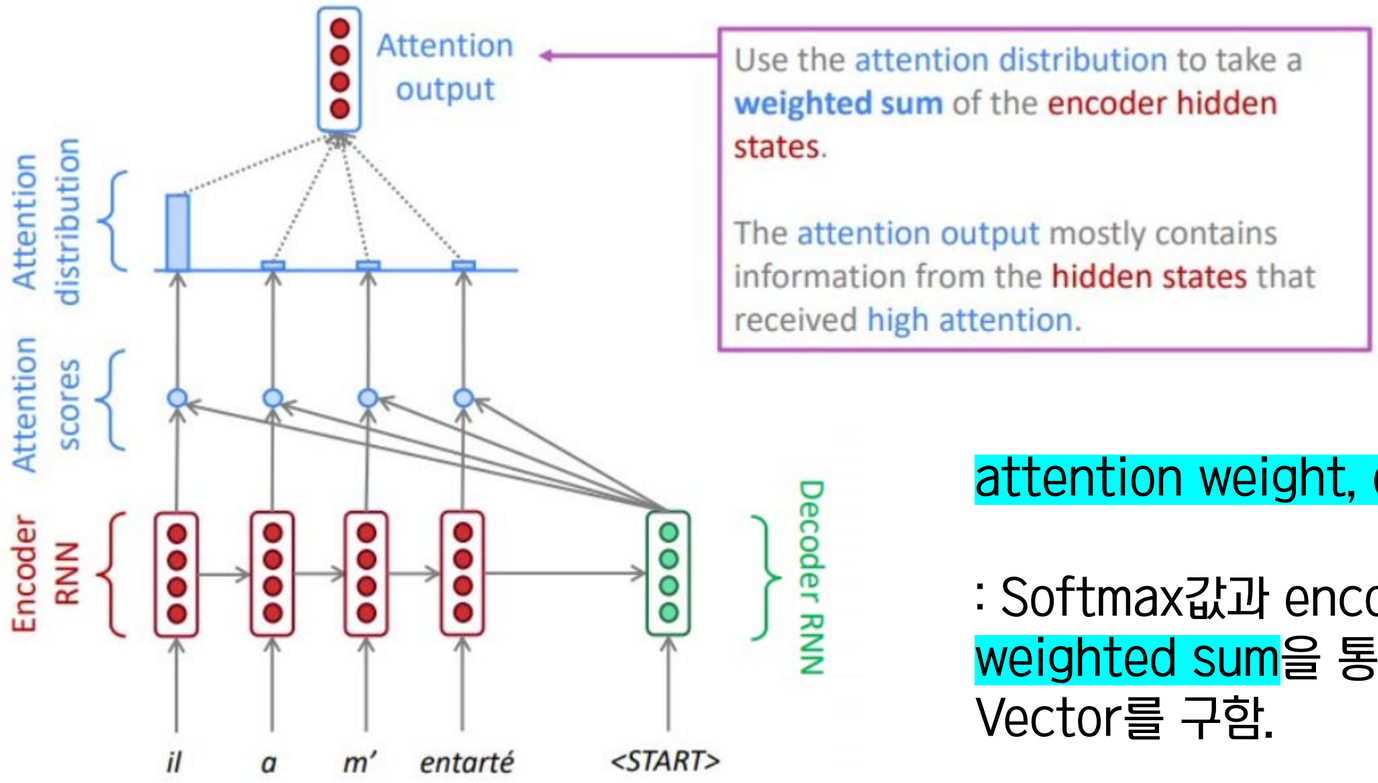
#2 Attention 개념/과정 : attention score, attention weight, output, concat with attention



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

#03 Attention

#2 Attention 개념/과정 : attention score, attention weight, output, concat with attention

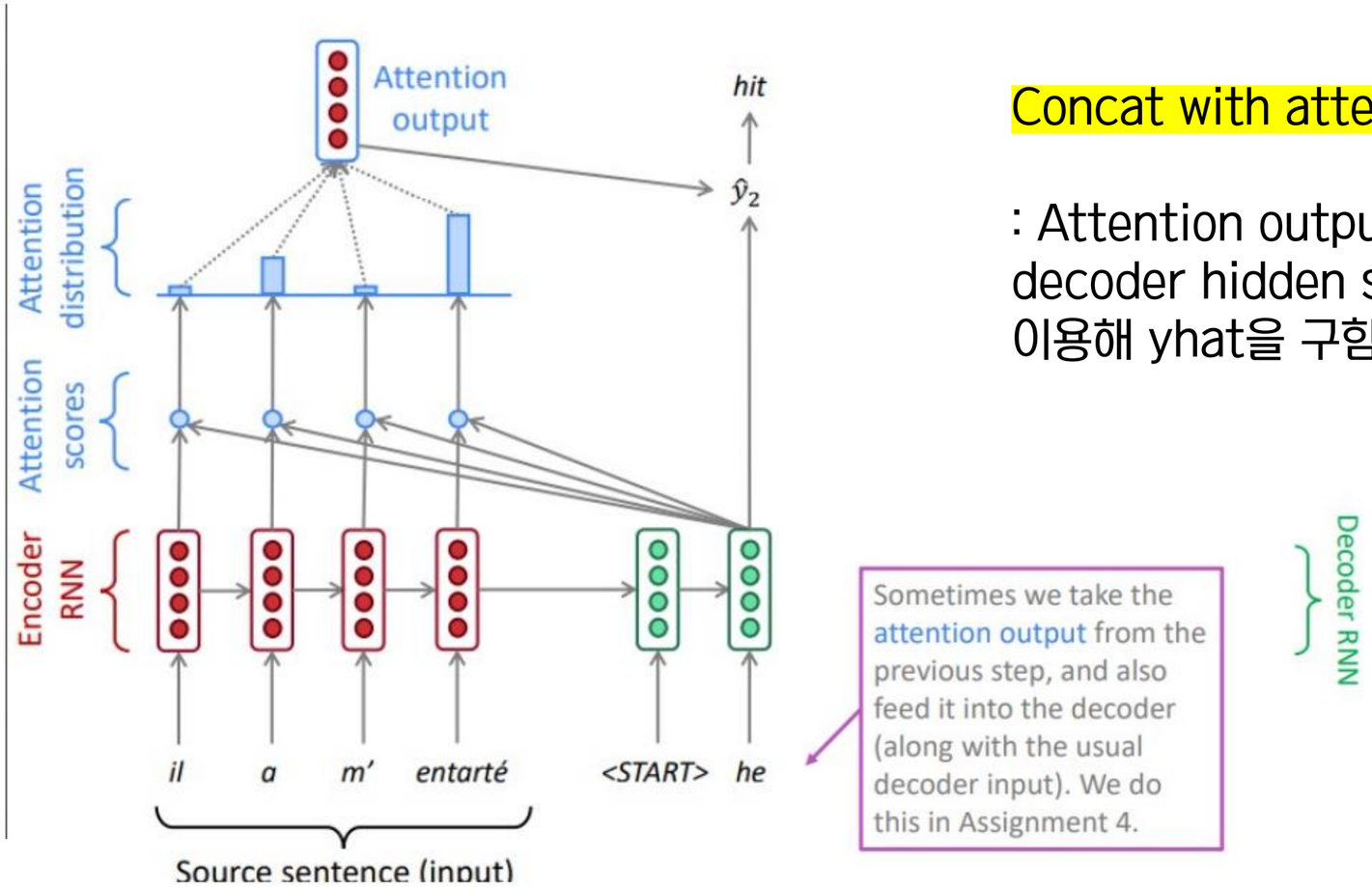


attention weight, output

: Softmax값과 encoder hidden state의 weighted sum을 통해 attention output Vector를 구함.

#03 Attention

#2 Attention 개념/과정 : attention score, attention weight, output, concat with attention



Concat with attention

: Attention output vector와
decoder hidden state를 concatenate한 벡터를
이용해 yhat을 구함

#03 Attention

#3 딥러닝에서 사용하는 일반적인 기술 attention : query vector, value vector

Attention 함수 정리

Attention(Q, K, V) = Attention Value

Attention(Q, K, V) = Attention Value
어텐션 함수는 주어진 '쿼리(Query)'에 대해서 모든 '키(Key)'와의 유사도를 각각 구하고,
이 유사도를 키와 맵핑되어있는 각각의 '값(Value)'에 반영해주어 모두 더해 리턴하는 함수.

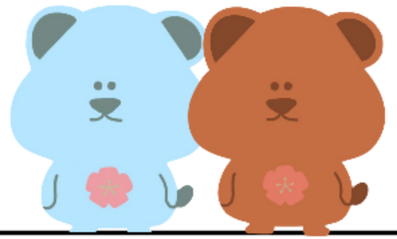
Q = Query : t 시점의 디코더 셀에서의 은닉 상태
K = Keys : 모든 시점의 인코더 셀의 은닉 상태들
V = Values : 모든 시점의 인코더 셀의 은닉 상태들

#03 Attention

#4 Attention is great

- NMT 성능이 향상
- Bottleneck 문제가 해결
- Vanishing gradient problem을 완화
- 오류 추적 가능성

THANK YOU



EWHA EURON