14기 정규세션

ToBig's 14기 고경태

Machine Translation, Sequence-to-sequence and Attention

Contents

Overview

- Introduce a new task: Machine Translation

is a major use-case of

- Introduce a new neural architecture: sequence-to-sequence

is improved by

- Introduce a new neural technique: attention

Machine Translation -> sequence to sequence -> attention!

14기 정규세션

ToBig's 14기 고경태

Machine Translation

Machine Translation

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

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

MT는 한 (소스) 언어의 문장을 (목표) 언어의 문장으로 번역하는 것.

1950`s : Early Machine Translation

Machine Translation research began in the early 1950s.

- Russian → English (motivated by the Cold War!)

1990s-2010s: Statistical Machine Translation

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

Translation Model

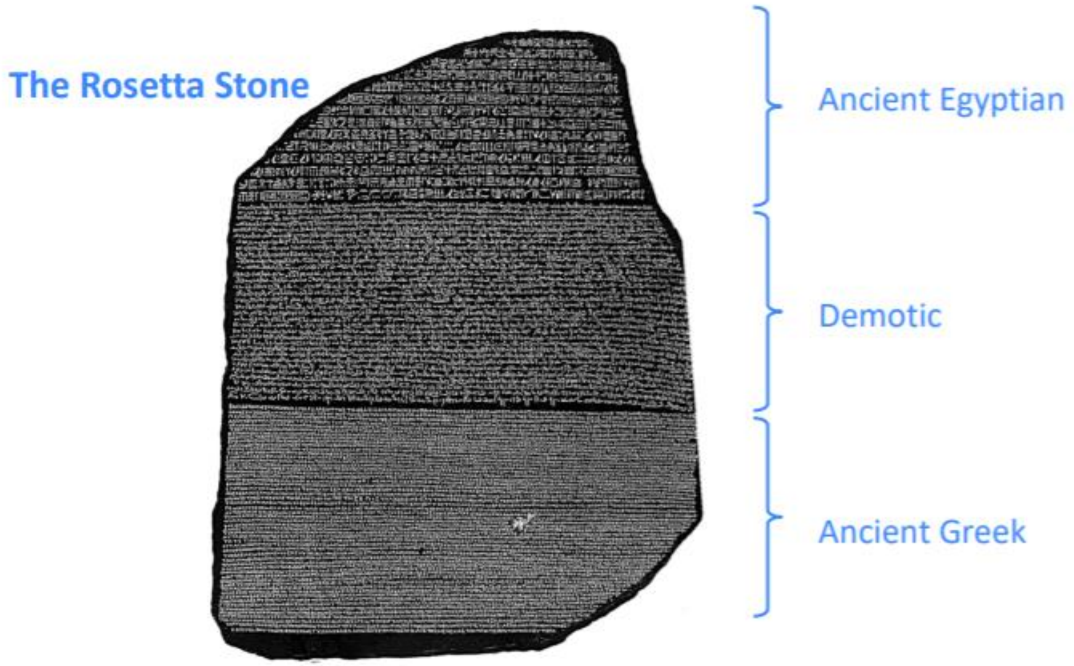
Models how words and phrases should be translated (*fidelity*). Learnt from parallel data.

Language Model

Models how to write good English (*fluency*). Learnt from monolingual data.

1. Translation Model : 작은 단어와 구의 번역
2. Language Model : 좋은 문장, 좋은 구조 도출

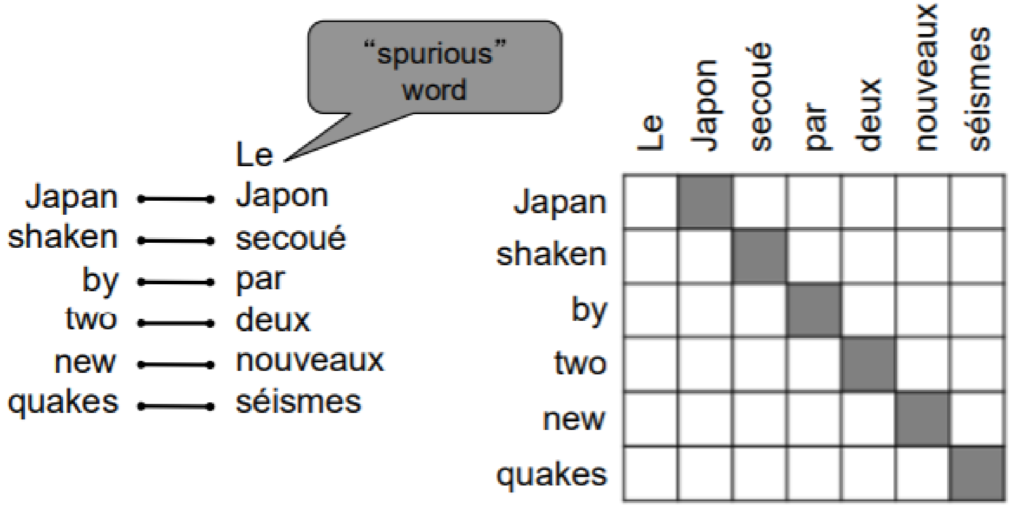
Learning alignment for SMT



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

A is the alignment

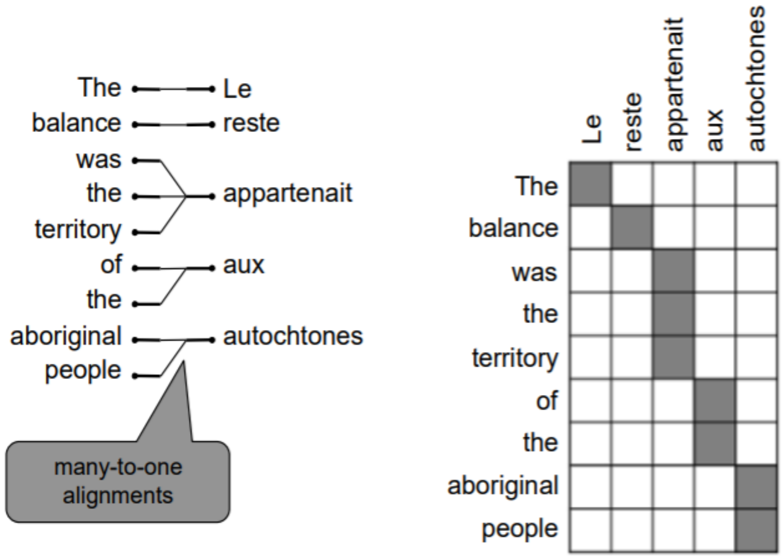
What is alignment?



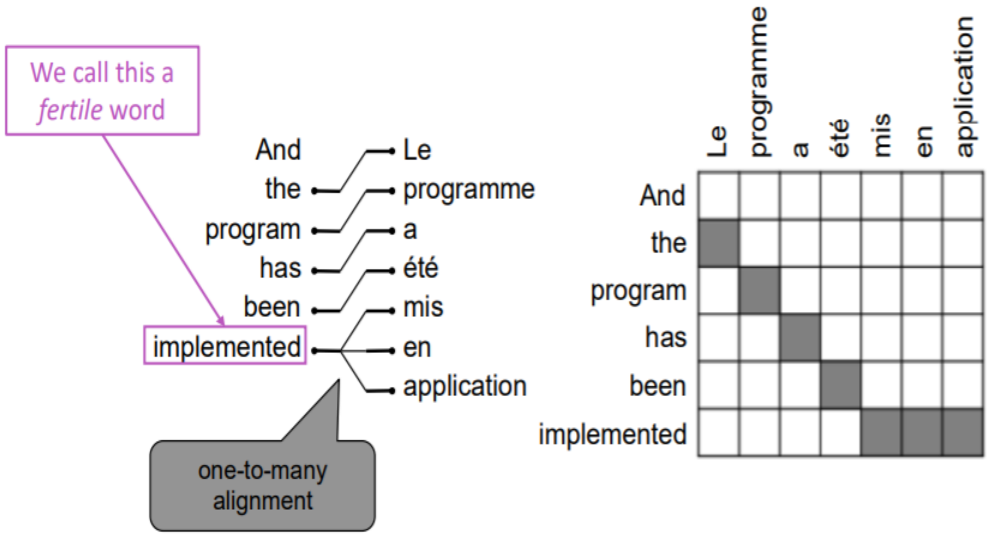
1. 정렬이란, 두 문장 사이에서 특정 단어쌍들의 대응
2.. 어떤 단어들은 대응되지 않을 수도..

Alignment is complex

Alignment can be many-to-one

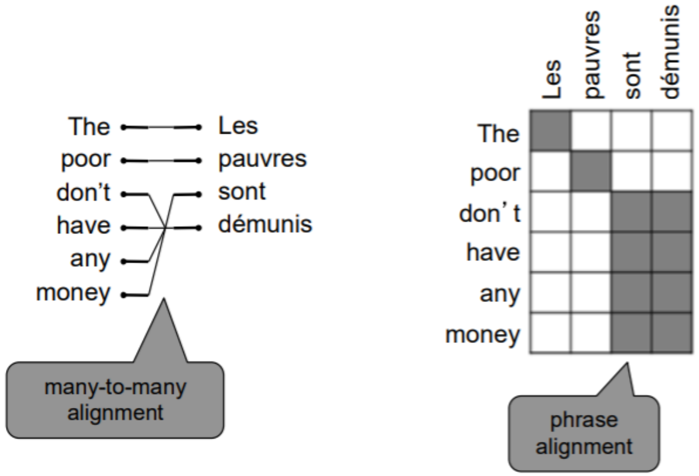


Alignment can be one-to-many



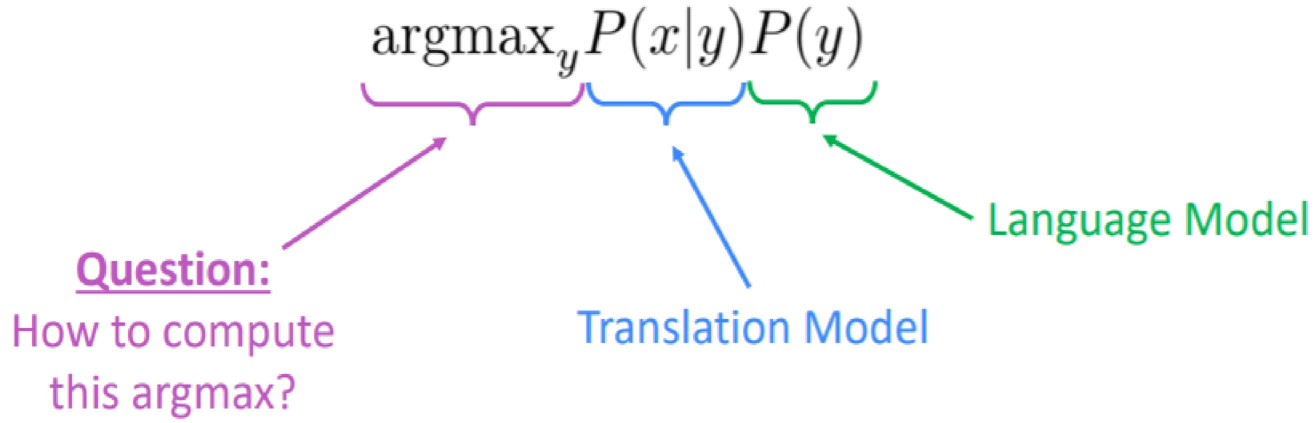
Alignment is complex, how learn?

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



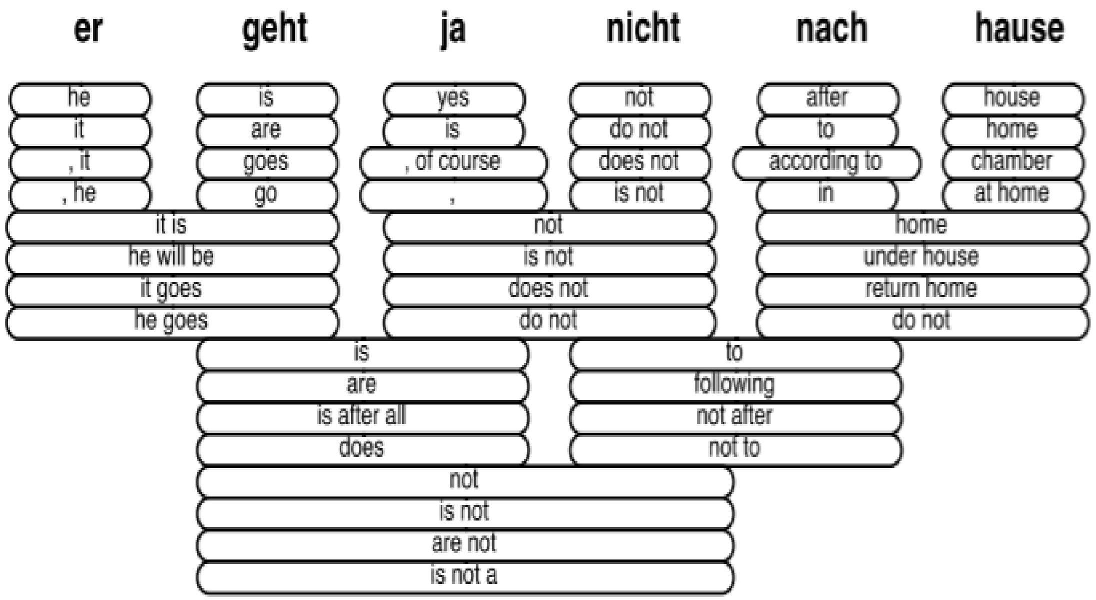
How learning?

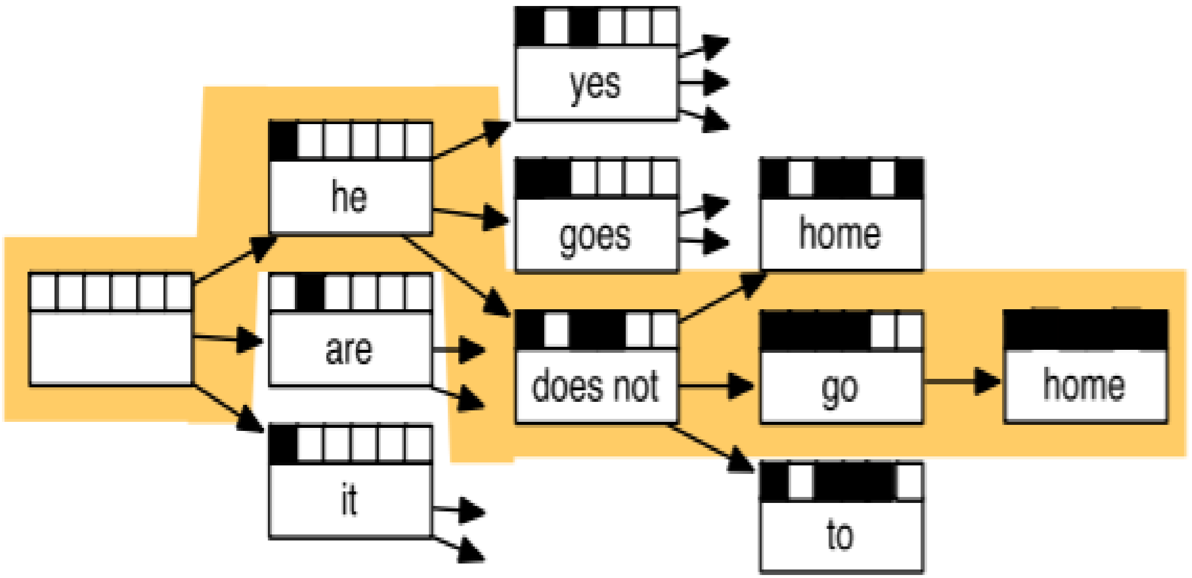
Decoding for SMT



1. 무차별 대입 솔루션
2. Heuristic 알고리즘

Decoding for SMT





1990s-2010s: Statistical Machine Translation

$$= \text{argmax}_y \underbrace{P(x|y)}_{\text{Translation Model}} \underbrace{P(y)}_{\text{Language Model}}$$

Translation Model

Models how words and phrases should be translated (*fidelity*).
Learnt from parallel data.

Language Model

Models how to write good English (*fluency*).
Learnt from monolingual data.

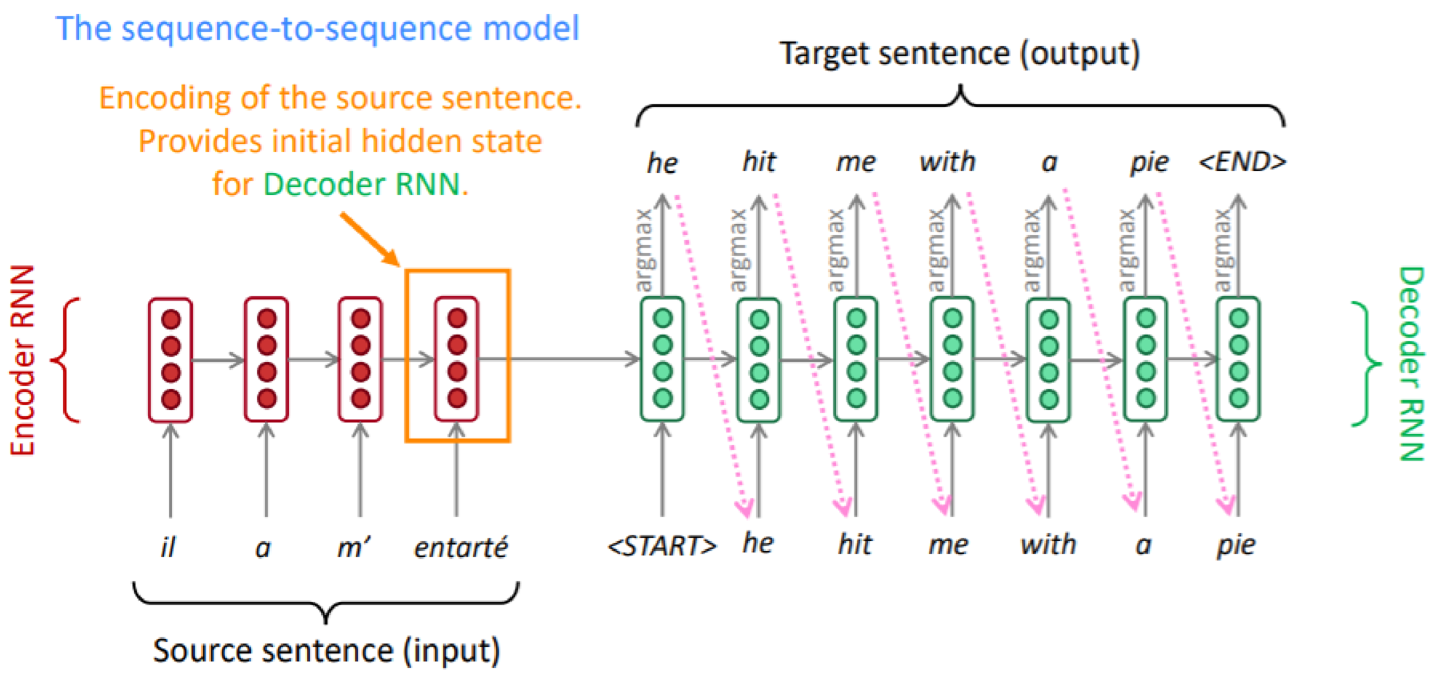
1. 좋은 성능을 내지만 매우 복잡한 구조
2. 각 system은 각 부분으로 나눠서 sub-system들이 모여 있는 형태
3. 많은 feature engineering이 필요
4. 추가적인 많은 자료 필요
5. 사람의 손을 많이 거쳐야함

14기 정규세션

ToBig's 14기 고경태

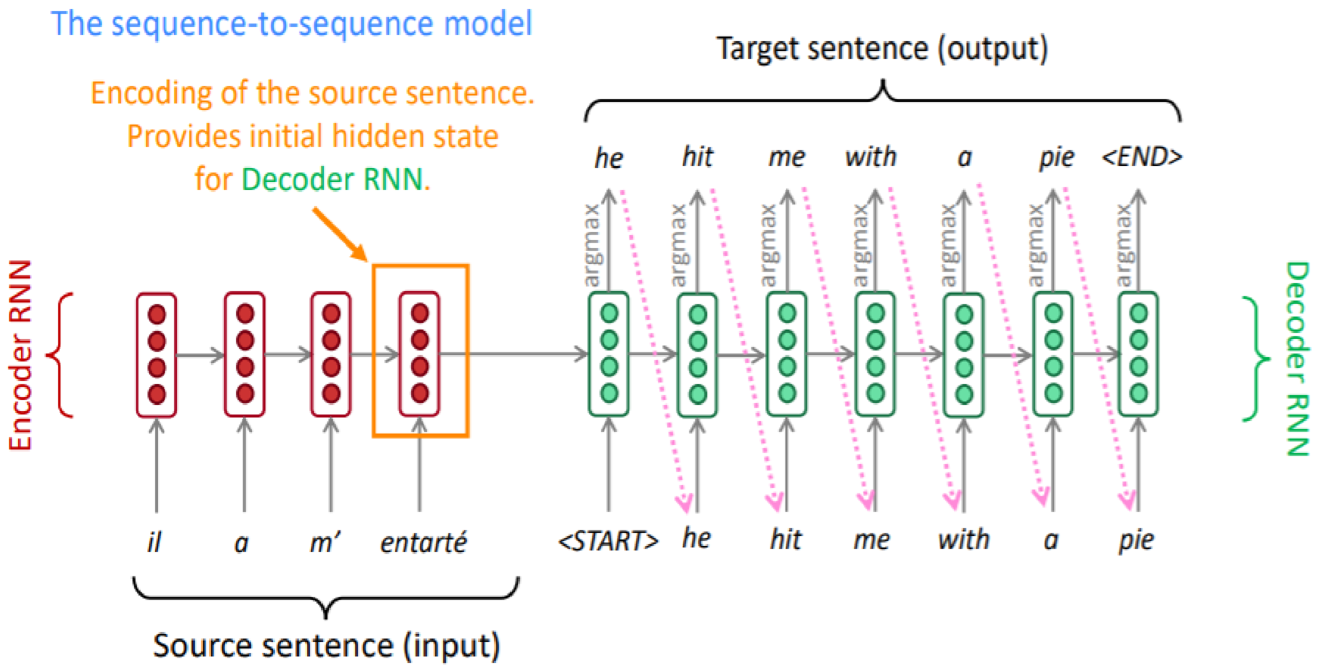
Neural Machine Translation
(Sequence to Sequence)

Neural Machine Translation (NMT)



1. 단일 신경망으로 기계번역을 하는 방법
2. 두개의 RNN을 포함한 seq-to-seq
3. 언어모델인 Decoder RNN은 target sentence를 생성
4. Output이 다음 단계의 input이 됨.

Sequence-to-sequence is versatile!



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

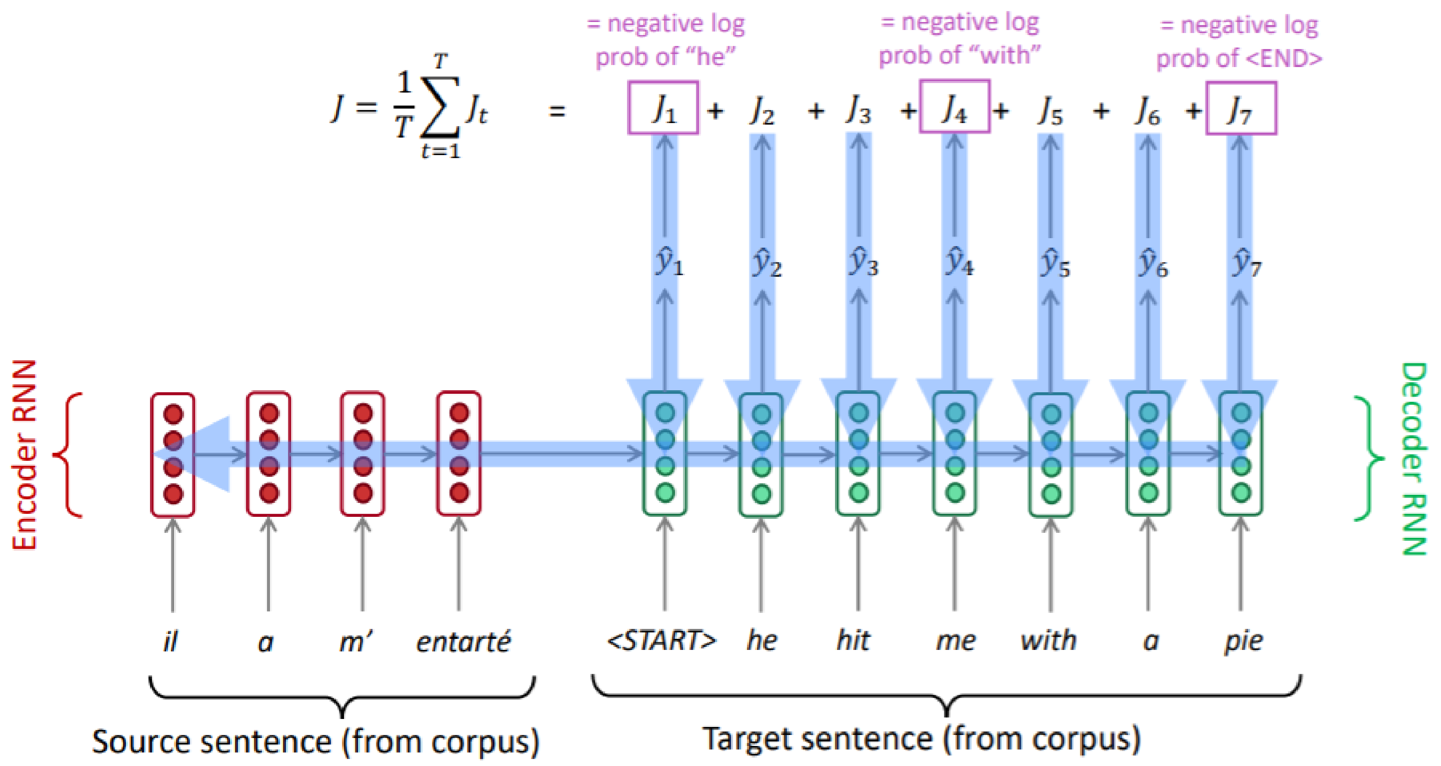
Neural Machine Translation (NMT)

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

$$P(y|x) = P(y_1|x)\,P(y_2|y_1, x)\,P(y_3|y_1, y_2, x) \ldots \underbrace{P(y_T|y_1, \ldots, y_{T-1}, x)}_{\text{Probability of next target word, given target words so far and source sentence } x}$$

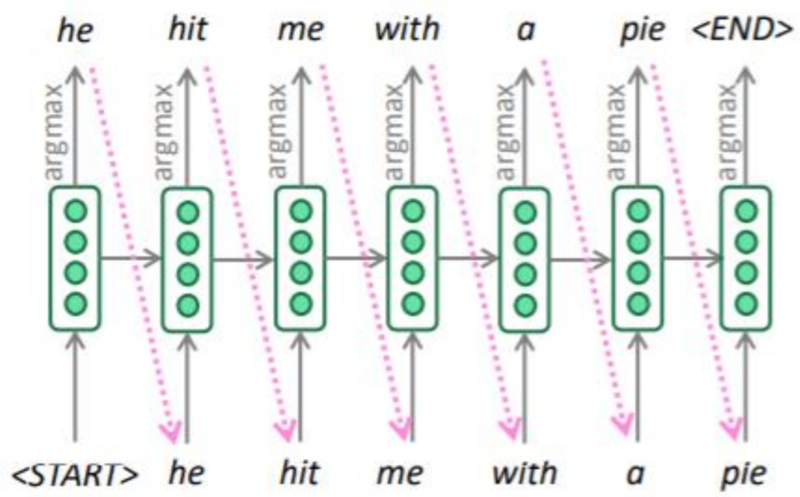
요약하자면 조건부 언어모델!

Training a Neural Machine Translation system



End- to-End
시스템 전반적으로 최적화!

Greedy decoding



- 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…)

1. Exhaustive search decoding

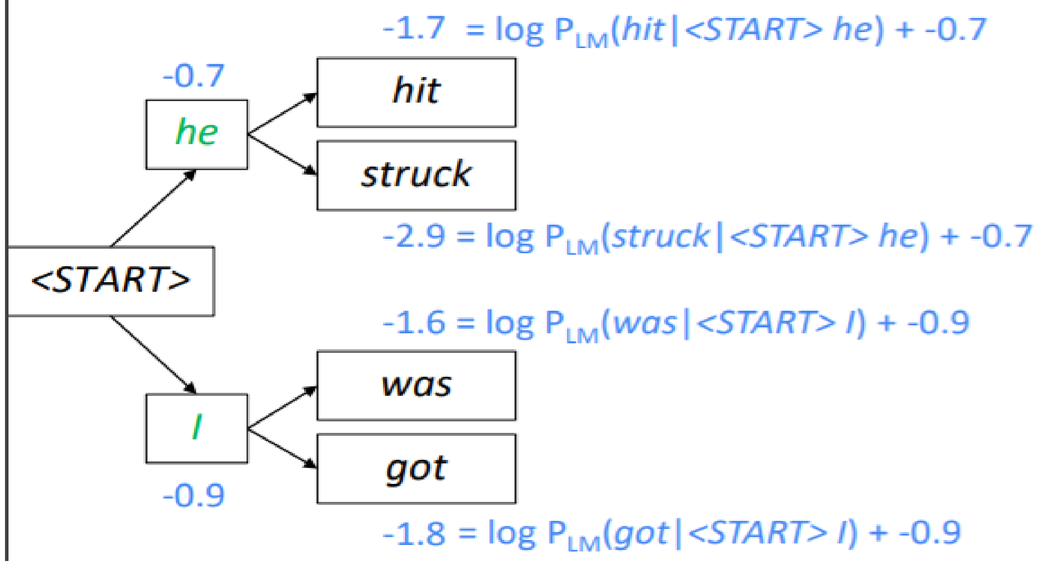
$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots,P(y_T|y_1,\ldots,y_{T-1},x)$$
$$= \prod_{t=1}^{T} P(y_t|y_1,\ldots,y_{t-1},x)$$

2. Beam search decoding

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

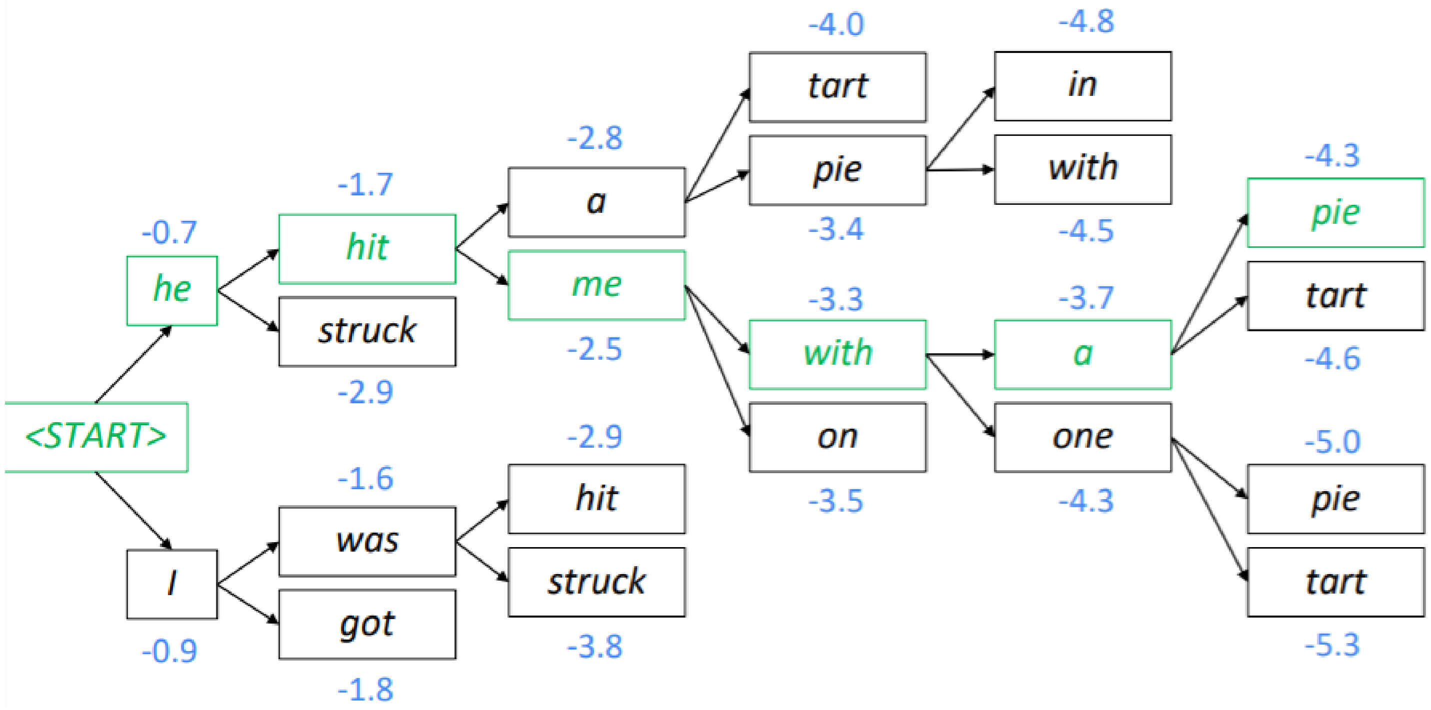
Beam search decoding: example

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



Beam search decoding: example

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

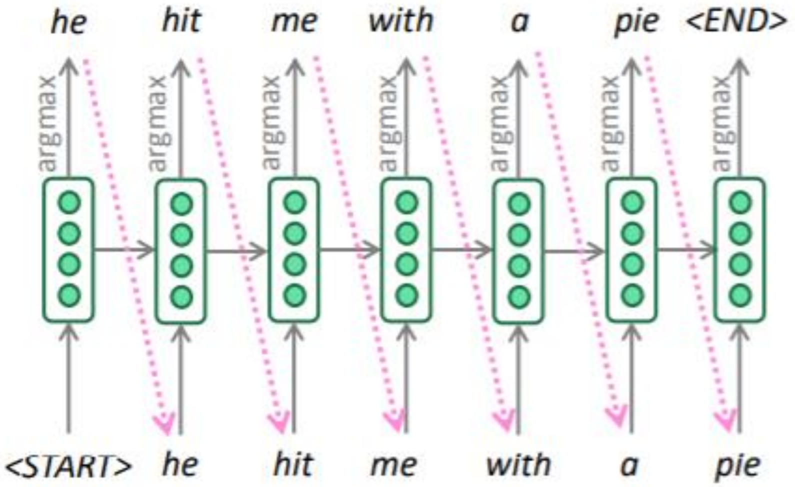


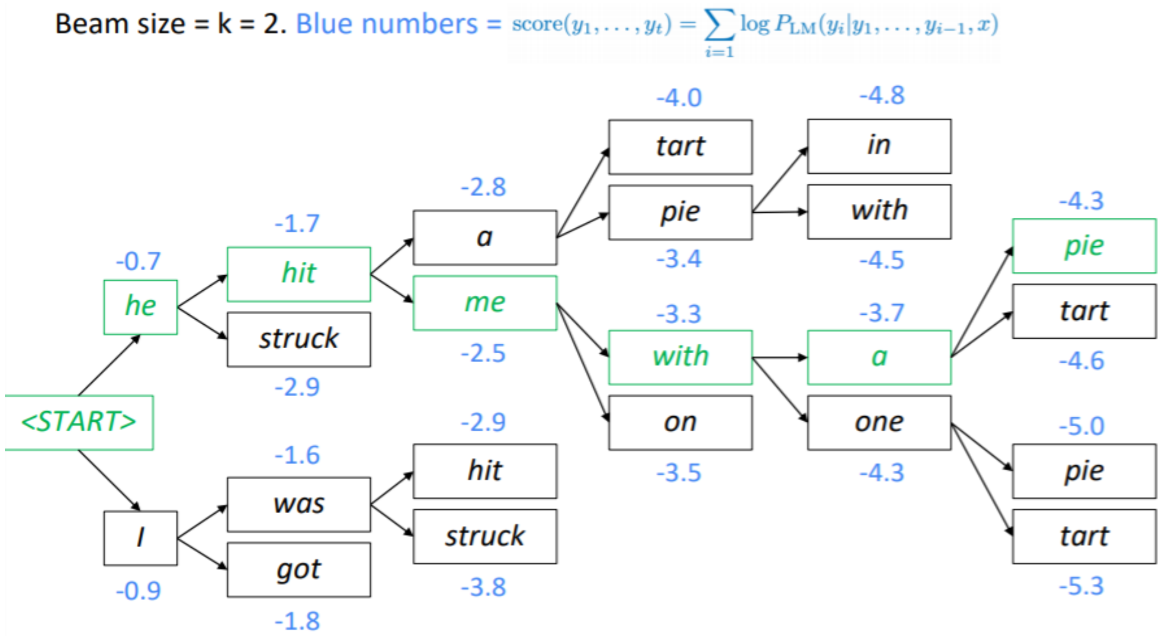
Beam search decoding: stopping criterion

Greedy decoding

vs

Beam search decoding





Beam search decoding: finishing up

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

$$\downarrow$$

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

Advantages, Disadvantages of NMT

Advantages

1. 더 나은 성능
2. Single neural network to be optimized end-to-end
   (하부구조가 개별적으로 optimized될 필요 X)
3. 인간의 노력 덜 필요

Disadvantages

1. Hard to debug
2. Difficult to control

How do we evaluate Machine Translation? BLEU (Bilingual Evaluation Understudy)

- `예측된 sentence`: **빛이 쐬는** 노인은 **완벽한** 어두운곳에서 **잠든 사람과 비교할 때** 강박증이 **심해질** 기회가 **훨씬 높았다**
- `true sentence`: **빛이 쐬는** 사람은 **완벽한** 어둠에서 **잠든 사람과 비교할 때** 우울증이 **심해질** 가능성이 **훨씬 높았다**

- 1-gram precision: $\frac{\text{일치하는}1-gram\text{의 수(예측된 }sentence\text{중에서)}}{\text{모든}1-gram\text{쌍 (예측된 }sentence\text{중에서)}} = \frac{10}{14}$
- 2-gram precision: $\frac{\text{일치하는}2-gram\text{의 수(예측된 }sentence\text{중에서)}}{\text{모든}2-gram\text{쌍 (예측된 }sentence\text{중에서)}} = \frac{5}{13}$
- 3-gram precision: $\frac{\text{일치하는}3-gram\text{의 수(예측된 }sentence\text{중에서)}}{\text{모든}3-gram\text{쌍 (예측된 }sentence\text{중에서)}} = \frac{2}{12}$
- 4-gram precision: $\frac{\text{일치하는}4-gram\text{의 수(예측된 }sentence\text{중에서)}}{\text{모든}4-gram\text{쌍 (예측된 }sentence\text{중에서)}} = \frac{1}{11}$

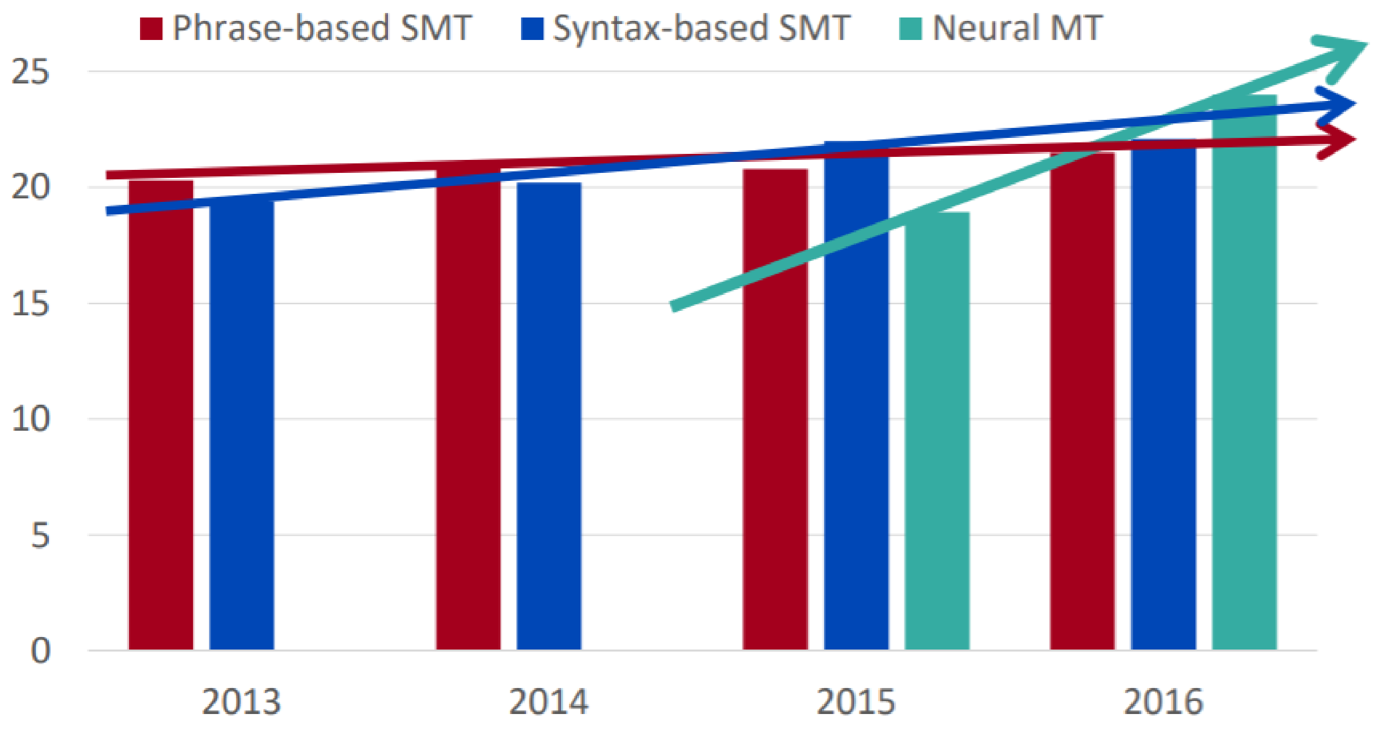
$$(\prod_{i=1}^{4} precision_i)^{\frac{1}{4}} = (\frac{10}{14} \times \frac{5}{13} \times \frac{2}{12} \times \frac{1}{11})^{\frac{1}{4}}$$

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

$$= min(1, \frac{14}{14}) \times (\frac{10}{14} \times \frac{5}{13} \times \frac{2}{12} \times \frac{1}{11})^{\frac{1}{4}}$$

How do we evaluate Machine Translation?

BLEU (**Bi**lingual **E**valuation **U**nderstudy)



NMT : the biggest success story of NLP Deep Learning

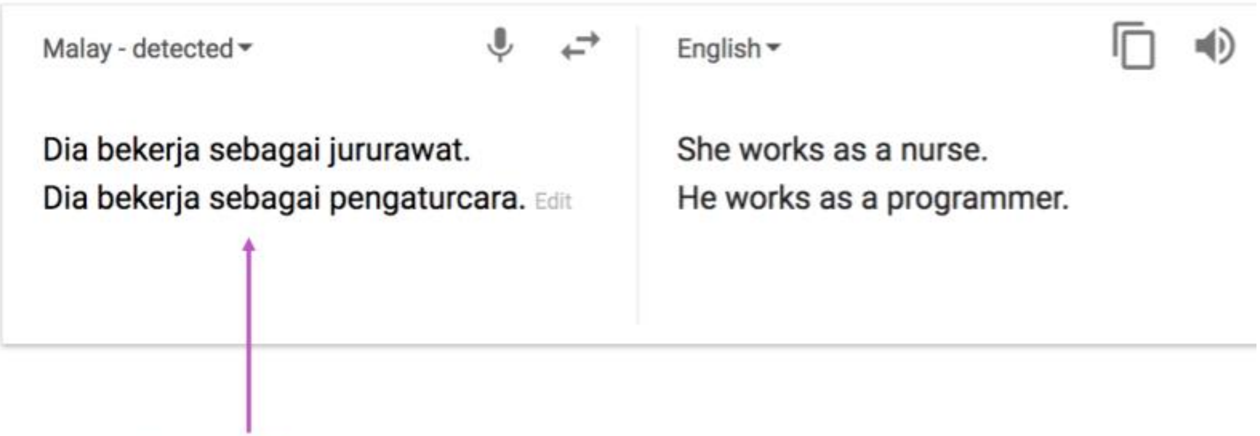
- **2014**: First seq2seq paper published
- **2016**: Google Translate switches from SMT to NMT

하지만 한계점이 존재..

1. 목표 어휘에 없는 단어 생성 불가
2. 도메인 불일치
3. 긴 텍스트에 대한 문맥 유지
4. 리소스가 부족한 언어 쌍

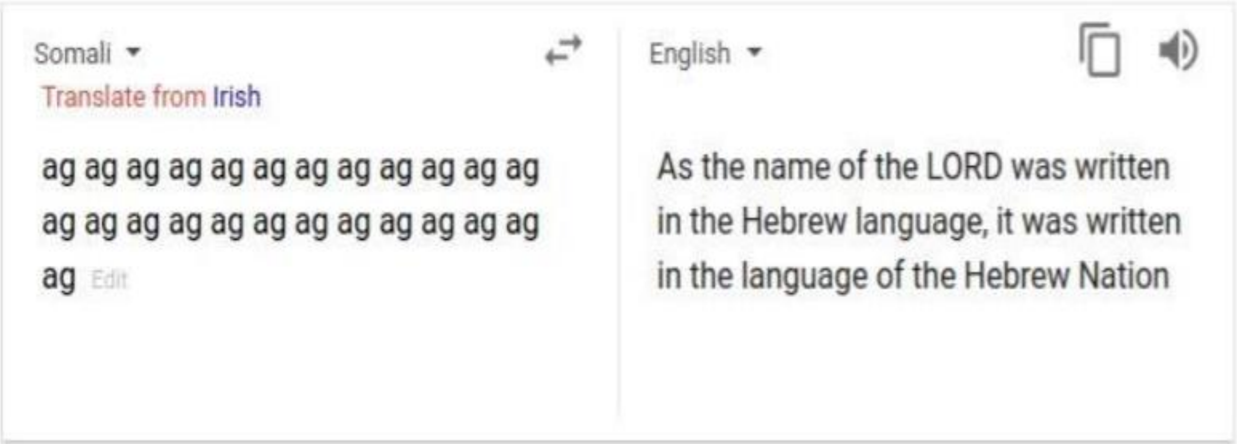
So is Machine Translation solved?

?



Didn’t specify gender

So is Machine Translation solved?



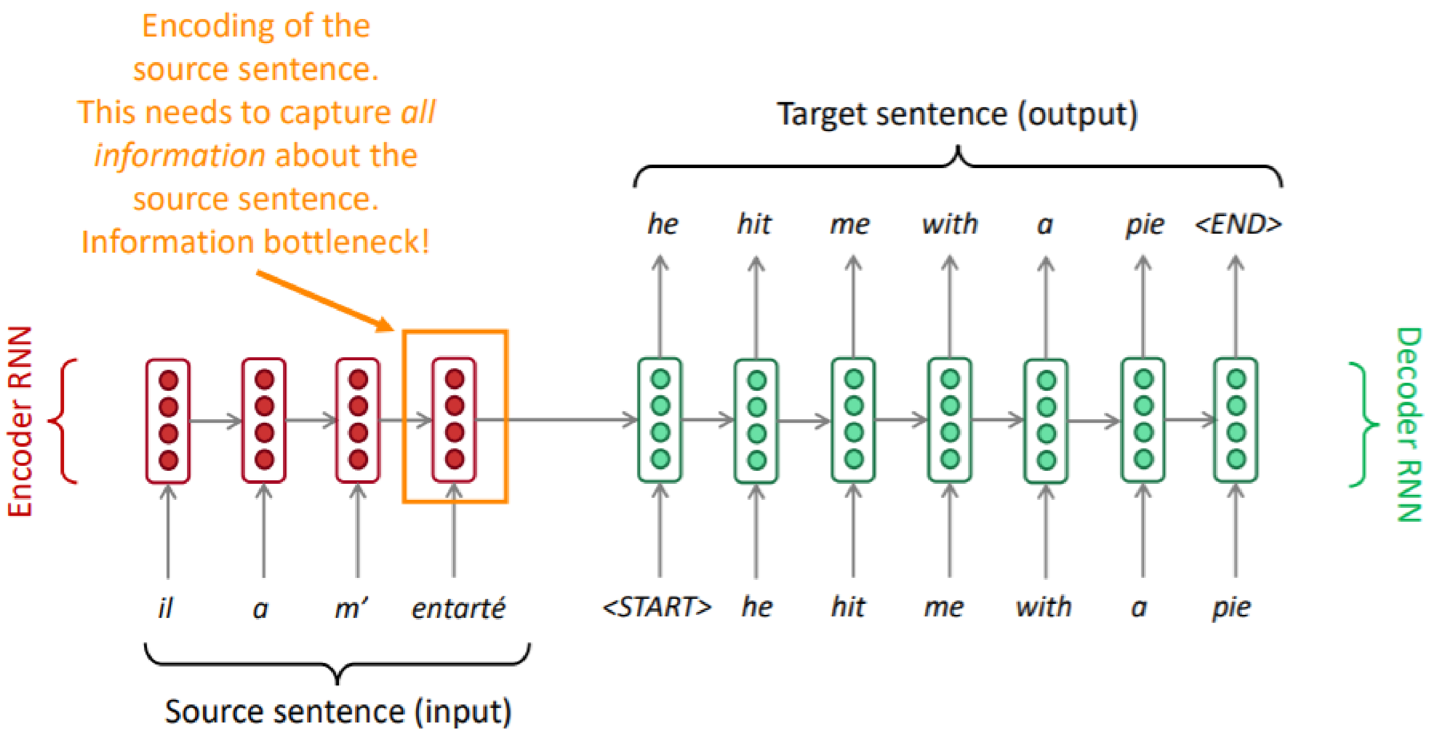
→ **ATTENTION**

14기 정규세션

ToBig's 14기 고경태

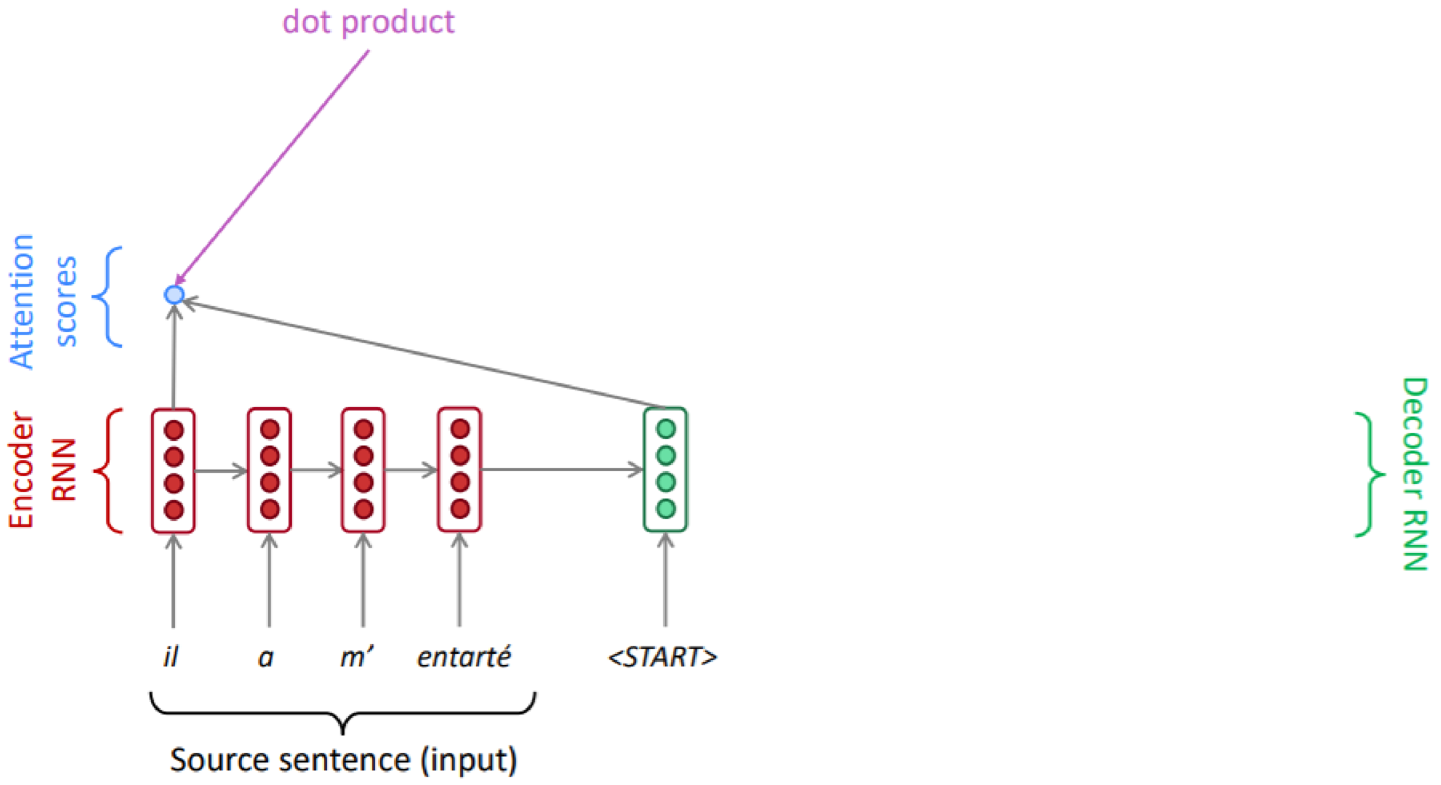
Attention

Sequence-to-sequence : the bottleneck problem

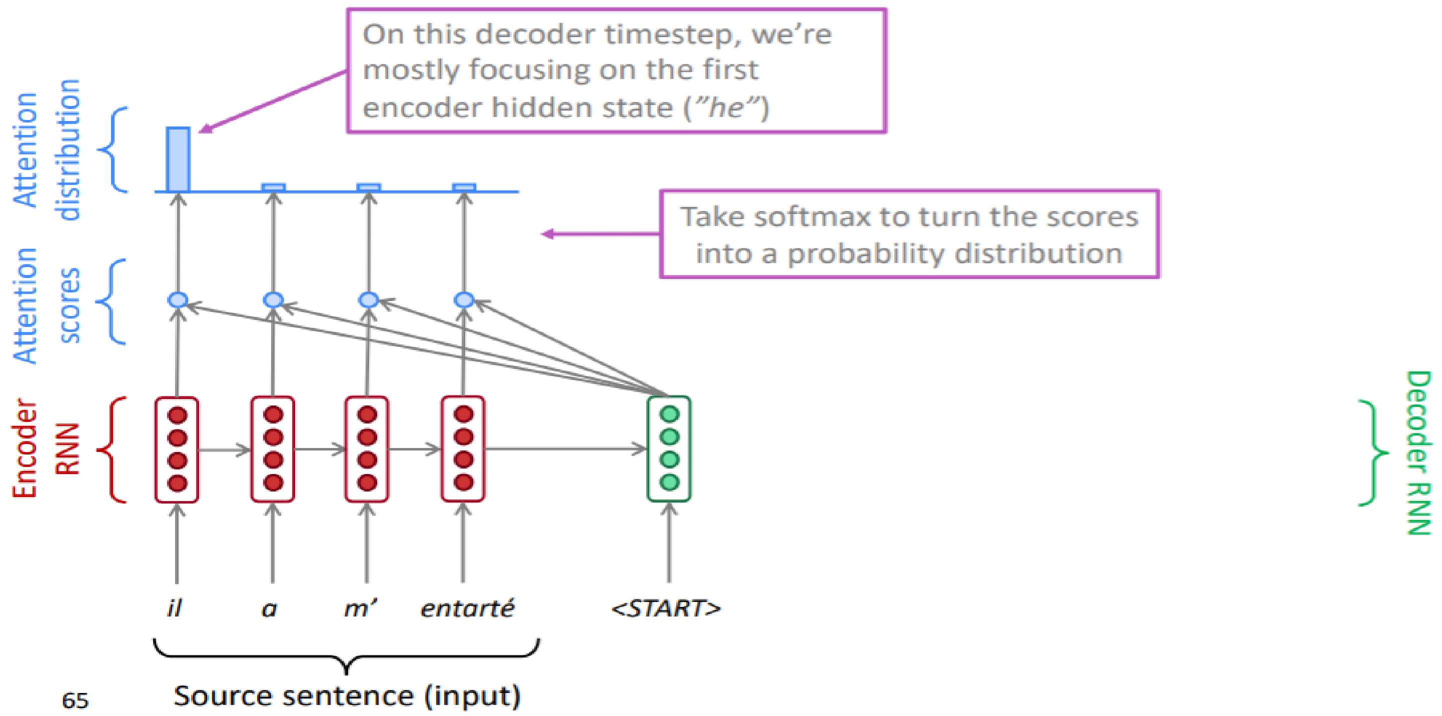


맨 끝에서 모든 정보를 캡쳐 강요
→ 너무 많은 압력 → 병목문제

Sequence-to-sequence with attention

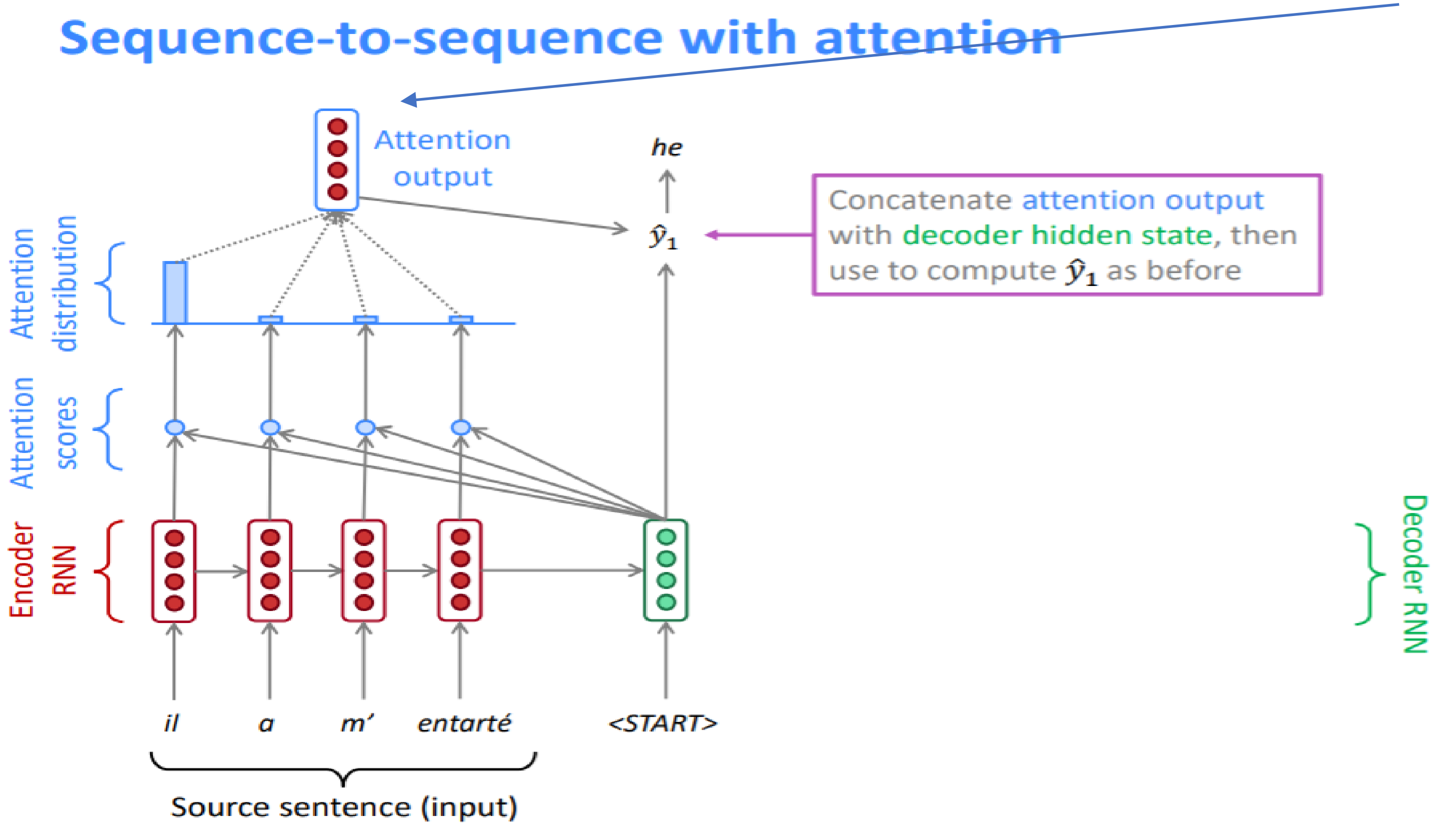


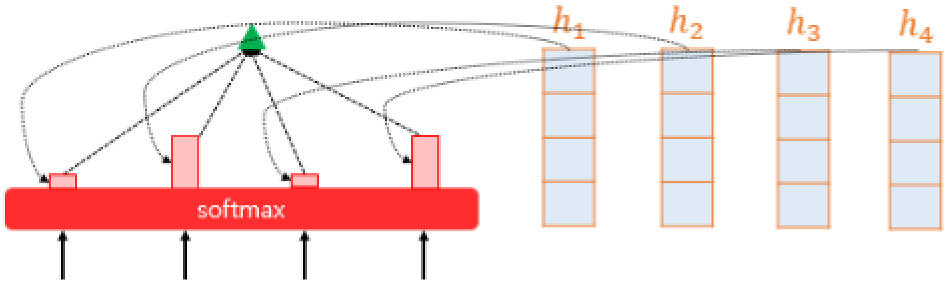
Sequence-to-sequence with attention



모든 인코더의 step마다 반복!
Attention을 주는 것

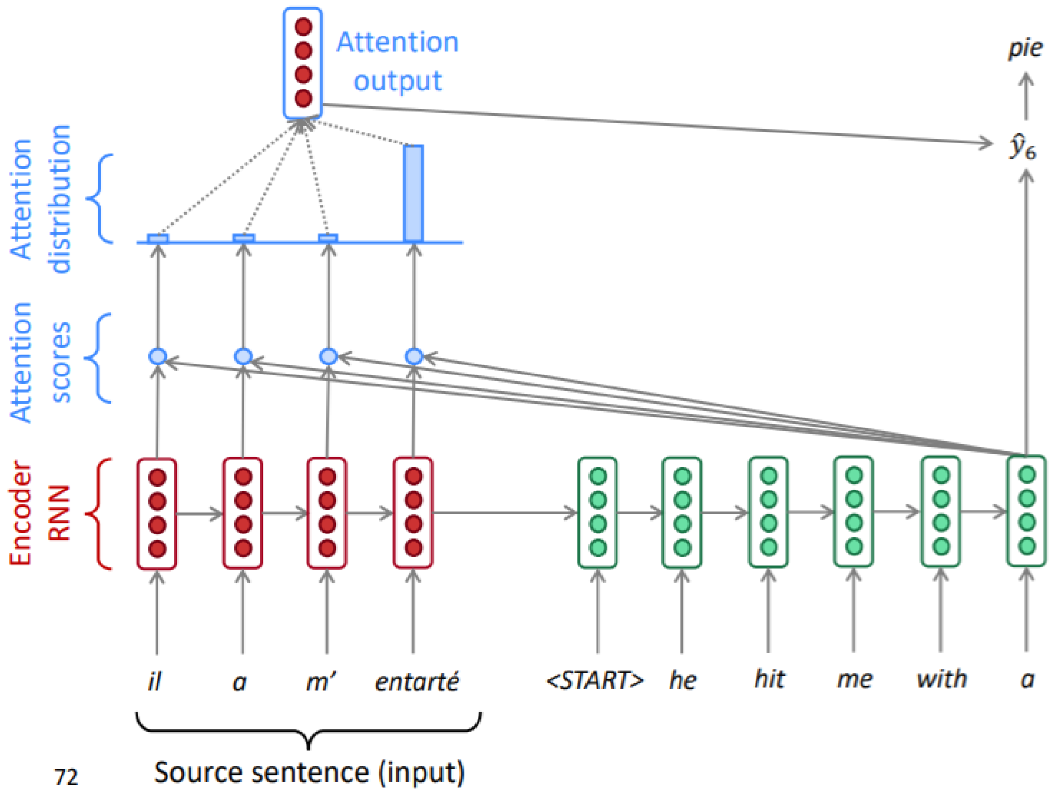
Sequence-to-sequence with attention





Attention의 output과 decode의 hidden state의 결합 → y1(hat)을 계산

Sequence-to-sequence with attention



decoder에서도 같은 행동을 반복!

Attention : in equations

- We have encoder hidden states $h_1, \dots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \dots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \mathrm{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

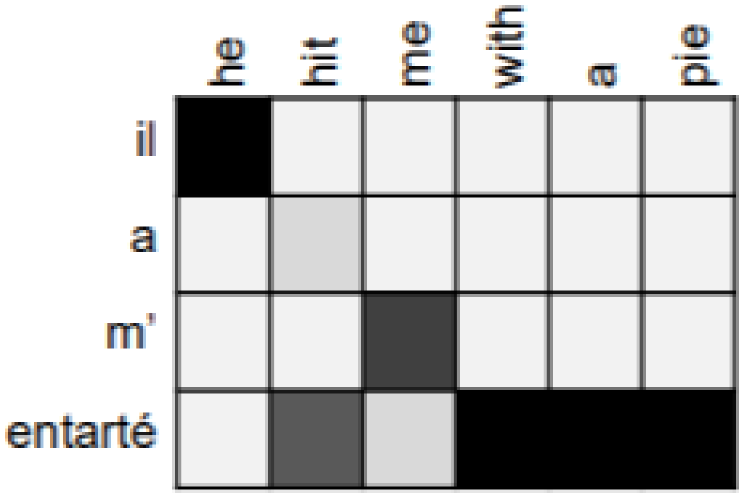
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

1. Encoder hidden states
2. Decoder hidden state
3. Softmax
4. Attention output
5. Y hat

Attention is great



1. NMT 성능을 향상시킴
2. 병목문제 해결
3. 기울기 소실 문제 해결
4. 추적 가능성

참고 자료

https://donghwa-kim.github.io/BLEU.html
https://www.programmersought.com/article/89135232797/
https://jeongukjae.github.io/posts/cs224n-lecture-8-machine-translation,-seq2seq,-attention/
https://pongdangstory.tistory.com/424
https://wikidocs.net/22893
자언어->코드 : https://blogs.oracle.com/meena/code-generation-using-lstm-long-short-term-memory-rnn-network

Q & A

들어주셔서 감사합니다.