연세대학교 통계학회 ESC

20FALL Week4

Translation, Seq2Seq, Attention

ADSTOREPOST.COM

10. 2020

INDEX

Natural Language Processing

SMT(Statistical Machine Translation)

Source Language 문장
↓
Target Language 문장

Source Language 문장

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

Target Language 문장

Translation Model: Models how words and phrases should be translated

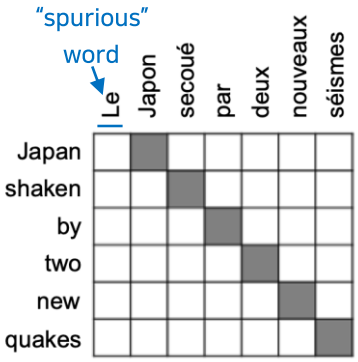
Language Model: Models how to write good target language

학습하기 위해서는 Parallel Data 필요

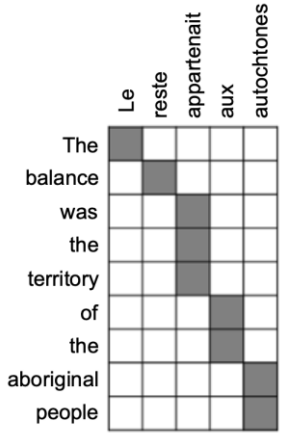
Learning Alignment

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

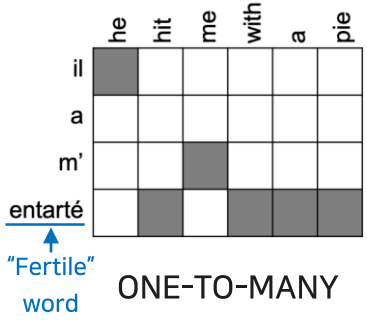
Source Language 단어들과
Target Language 단어들이
상응하는 관계



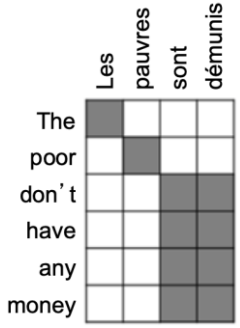
No COUNTERPART



MANY-TO-ONE



ONE-TO-MANY



MANY-TO-MANY

어떤 단어(들)에 상응되는지, 문장에서의 위치, fertility 등

Decoding for SMT

가장 좋은 번역 찾기 위해!

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

Translation Model

Language Model

어떻게 계산?

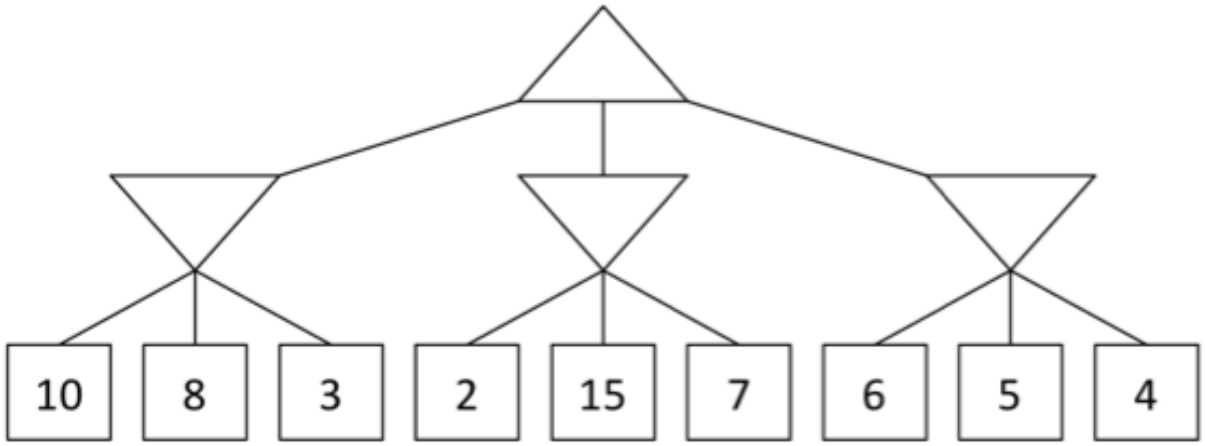
~~모든 경우의 수 계산~~

Heuristic Search Algorithm with Pruning

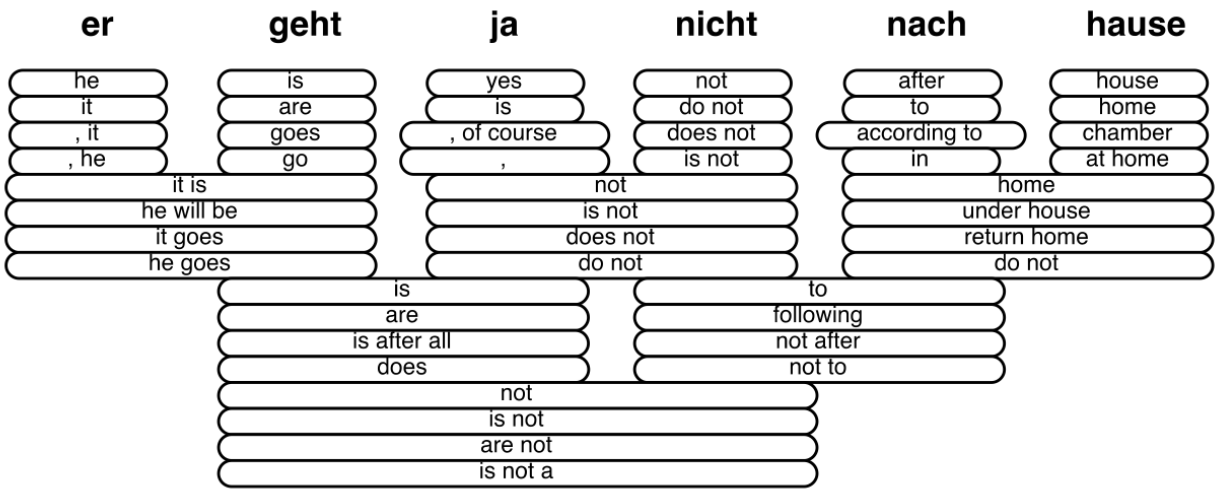
?

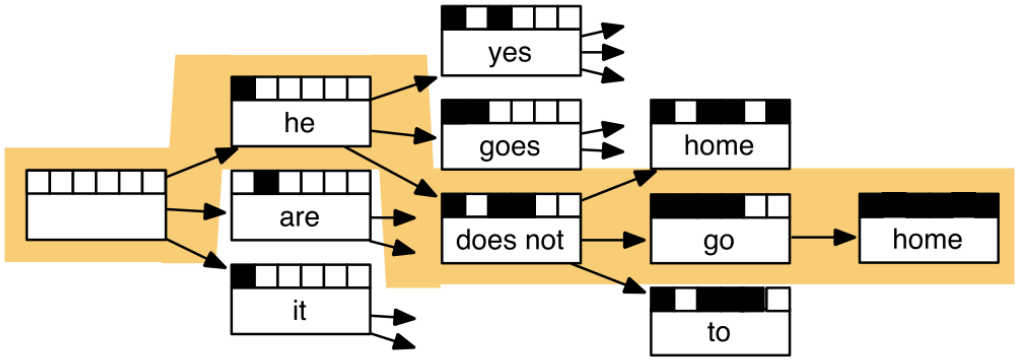
Alpha-Beta Pruning

Pruning의 역할을 직관적으로 이해해보자!



Decoding for SMT





Extremely COMPLEX!

Neural Machine Translation (NMT)

single neural network

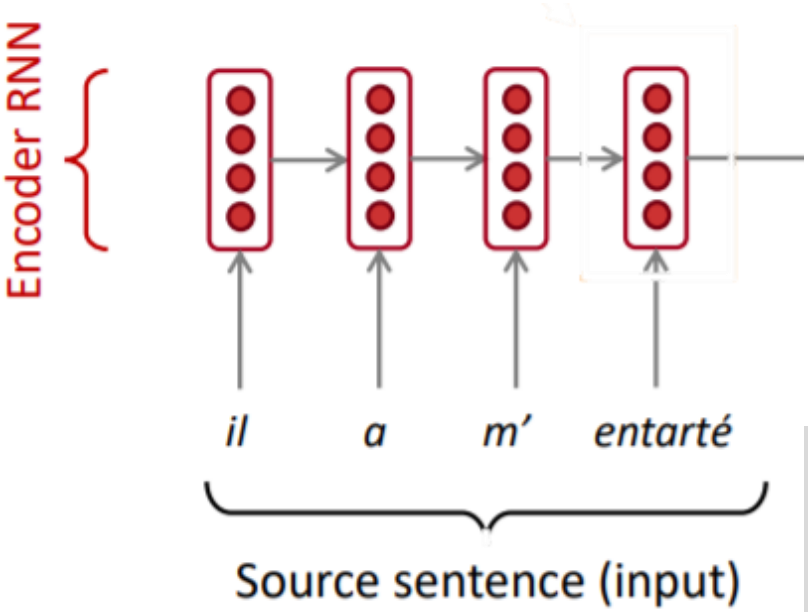
sequence-to sequence (seq2seq)

Two RNNs

NMT Testing

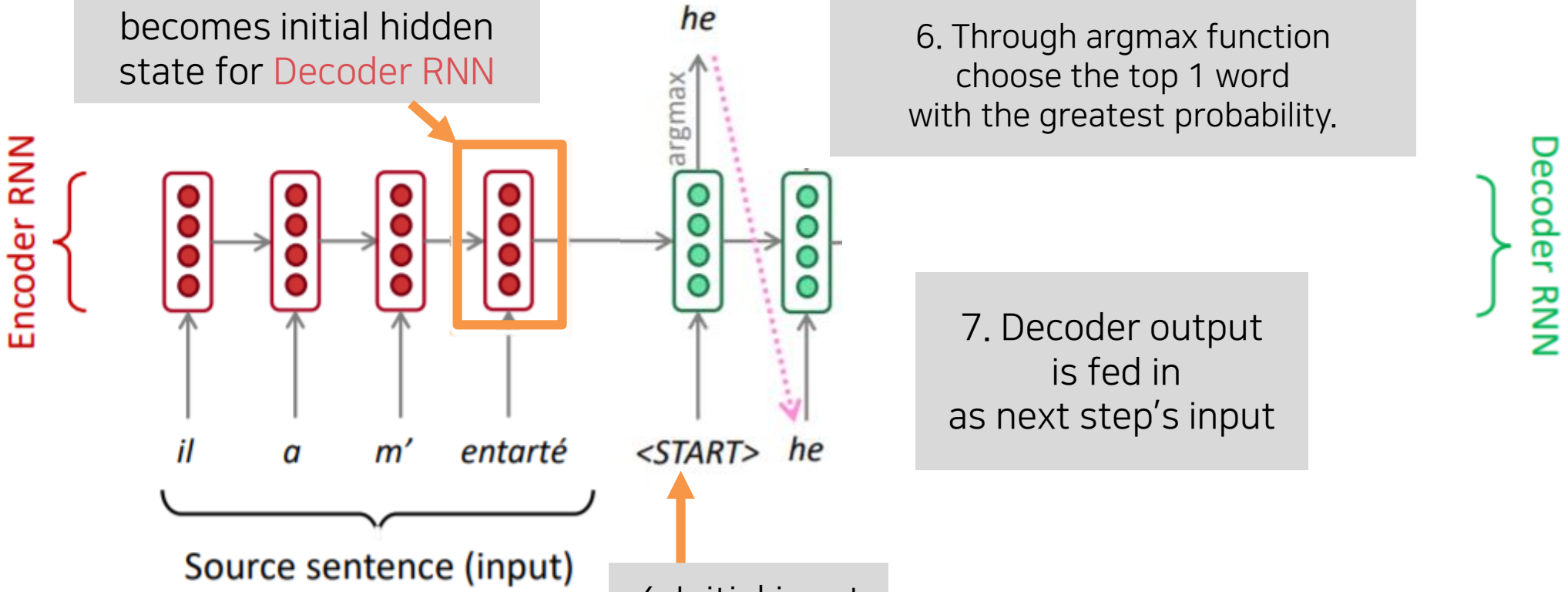
2. Encoder RNN
uni/bi- directional,
multi-layer, vanilla, LSTM, ···

→ Encoder RNN produces an encoding of the source sentence



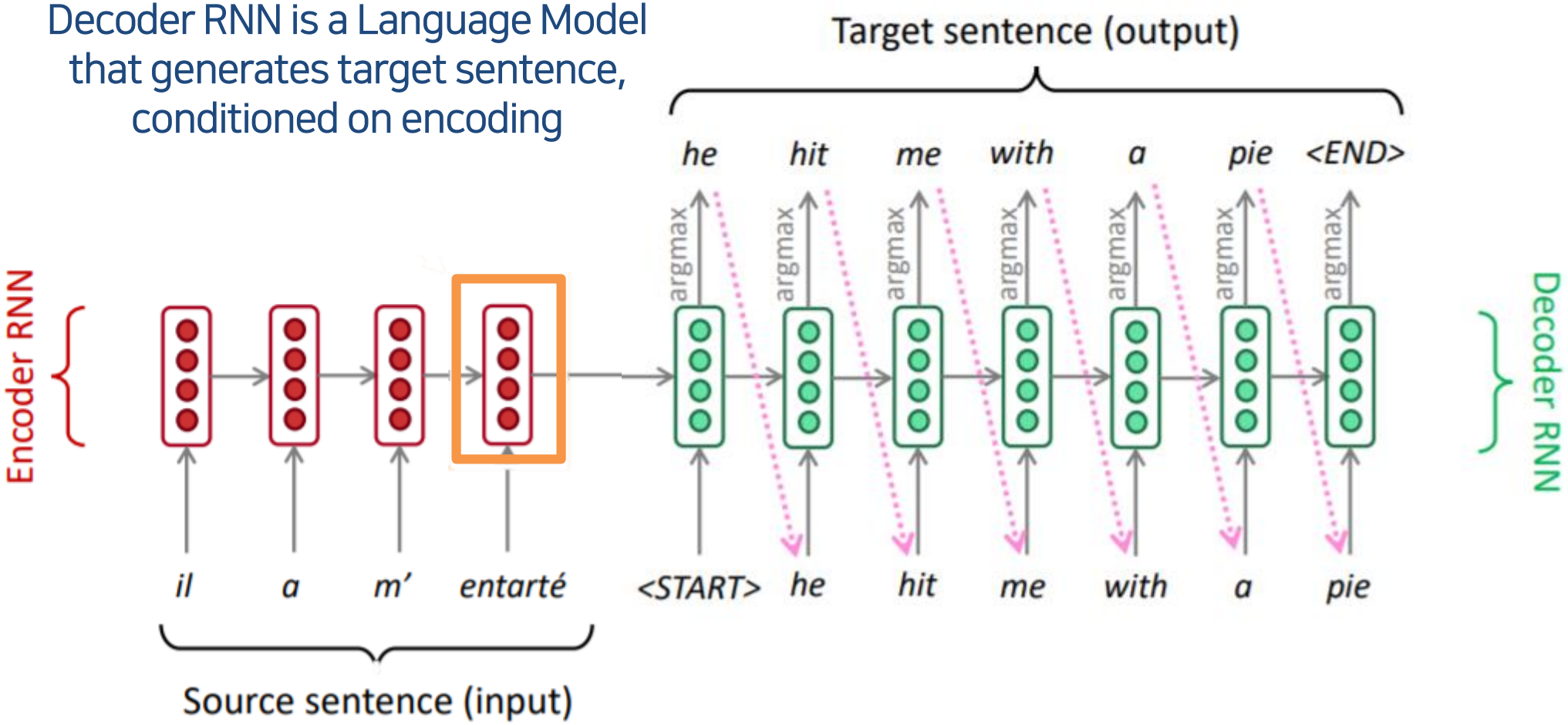
1. Input
번역하고자 하는 문장의
word embeddings

NMT Testing


3. Final hidden state of this encoder RNN becomes initial hidden state for Decoder RNN
5. Generate probability distribution of what word might come next
6. Through argmax function choose the top 1 word with the greatest probability.
Encoder RNN
Decoder RNN
argmax
he
il
a
m'
entarté
<START>
he
Source sentence (input)
4. Initial input
7. Decoder output is fed in as next step's input

NMT Testing

Decoder RNN is a Language Model that generates target sentence, conditioned on encoding



Neural Machine Translation (NMT)

- The sequence-to-sequence model is an example of a **Conditional Language Model**.
  - **Language Model** because the decoder is predicting the next word of the target sentence *y*
  - **Conditional** because its predictions are *also* conditioned on the source sentence *x*

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

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

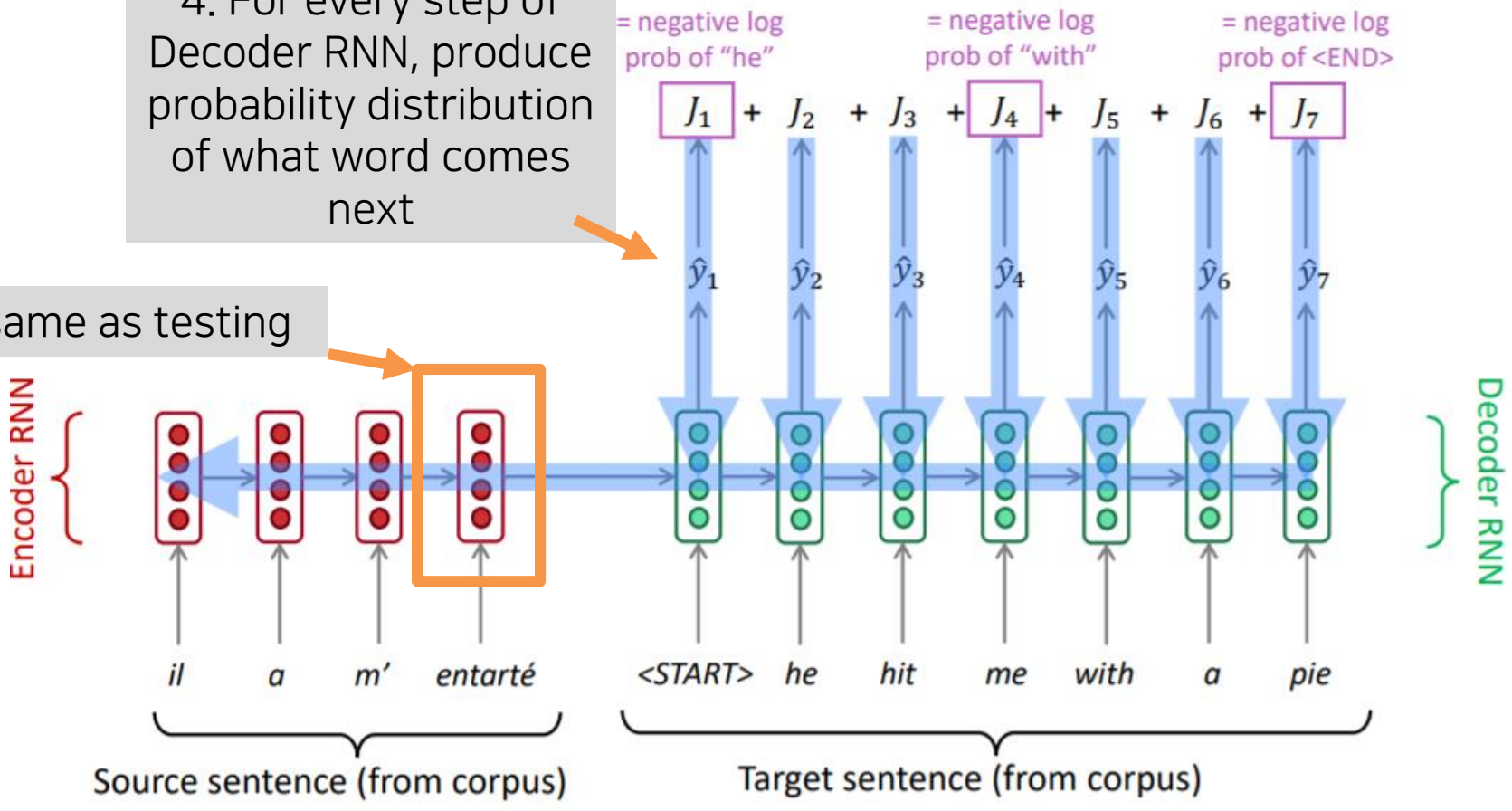
Probability of next target word, given target words so far and source sentence *x*

NMT Training

5. Compute loss

4. For every step of Decoder RNN, produce probability distribution of what word comes next

3. same as testing

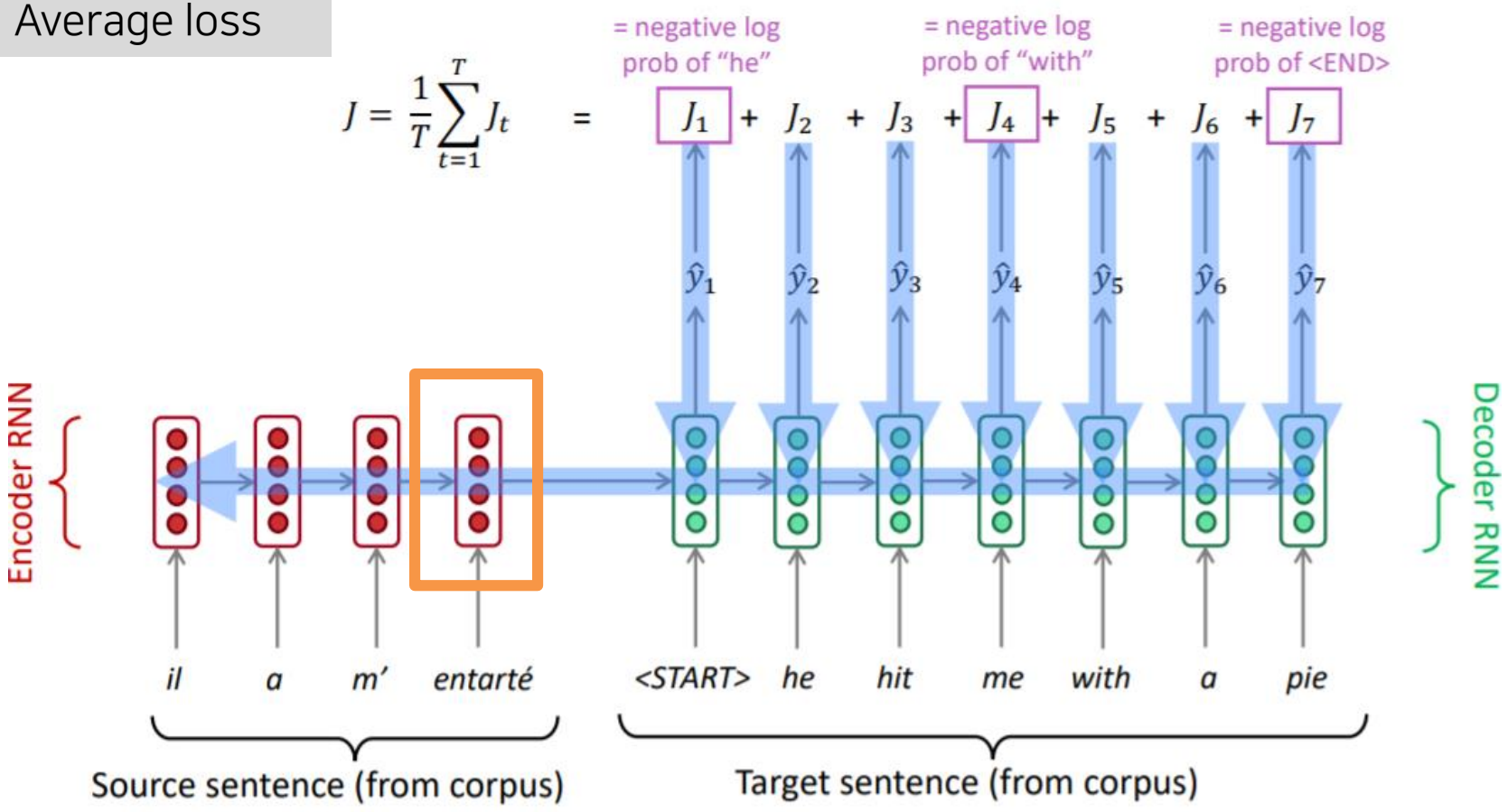


1. X as input of Encoder RNN

2. Y as input of Decoder RNN
Test와의 차이점

NMT Training

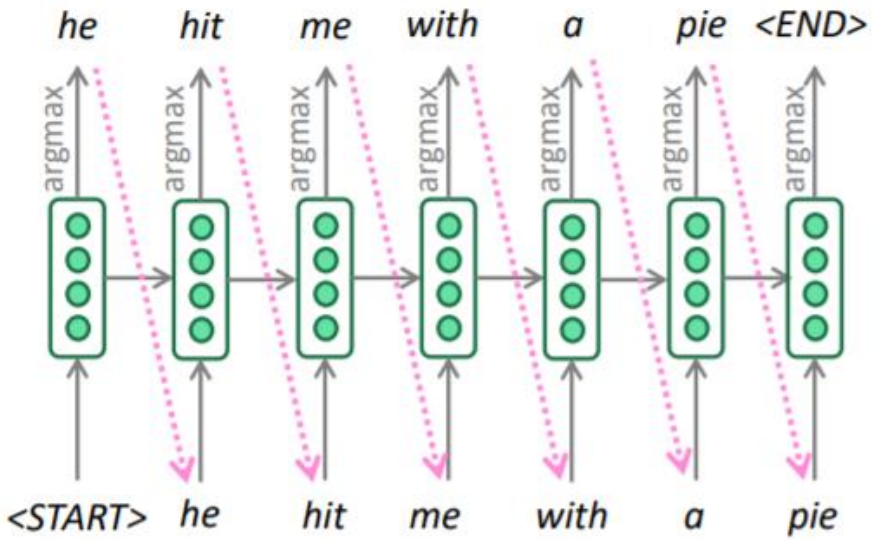
6. Average loss



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

Optimize as whole

Greedy decoding



By taking argmax on each step of the decoder, getting the most probable word on each step

- Input: *il a m'entarté* *(he hit me with a pie)*
- → *he* ____
- → *he hit* ____
- → *he hit a* ____

Problem: No way to undo decisions

Fix problem of Greedy decoding:
① Exhaustive search decoding

- Ideally we want to find a (length *T*) translation *y* that maximizes

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

- We could try computing all possible sequences *y*
  - This means that on each step *t* of the decoder, we're tracking V^t possible partial translations, where *V* is vocab size
  - This $O(V^T)$ complexity is far too expensive!

Fix problem of Greedy decoding: ① Beam search decoding

- Core idea: On each step of decoder, keep track of the *k* most probable partial translations (which we call *hypotheses*)
  - *k* is the beam size (in practice around 5 to 10)

- A hypothesis $y_1, \dots, y_t$ has a score which is its log probability:

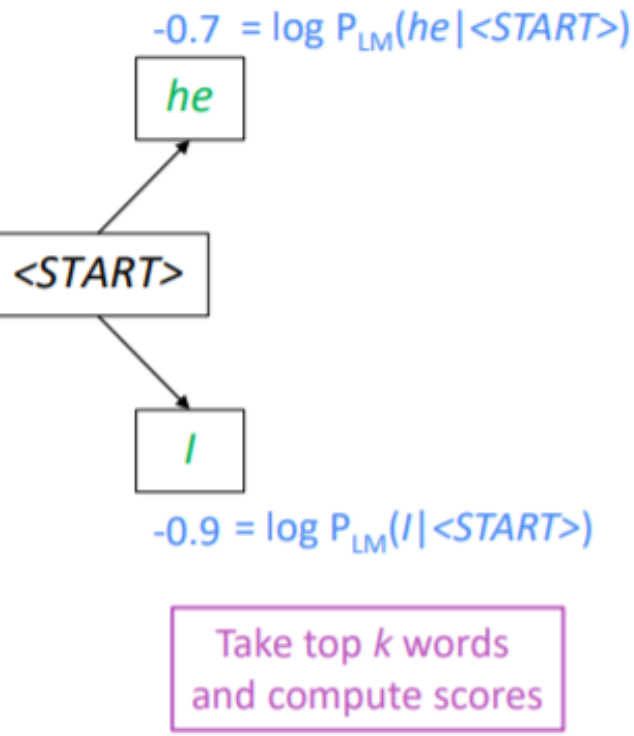
$$\text{score}(y_1, \dots, y_t) = \log P_{\text{LM}}(y_1, \dots, y_t|x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i|y_1, \dots, y_{i-1}, x)$$

  - Scores are all negative, and higher score is better
  - We search for high-scoring hypotheses, tracking top *k* on each step

- Beam search is not guaranteed to find optimal solution
- But much more efficient than exhaustive search!

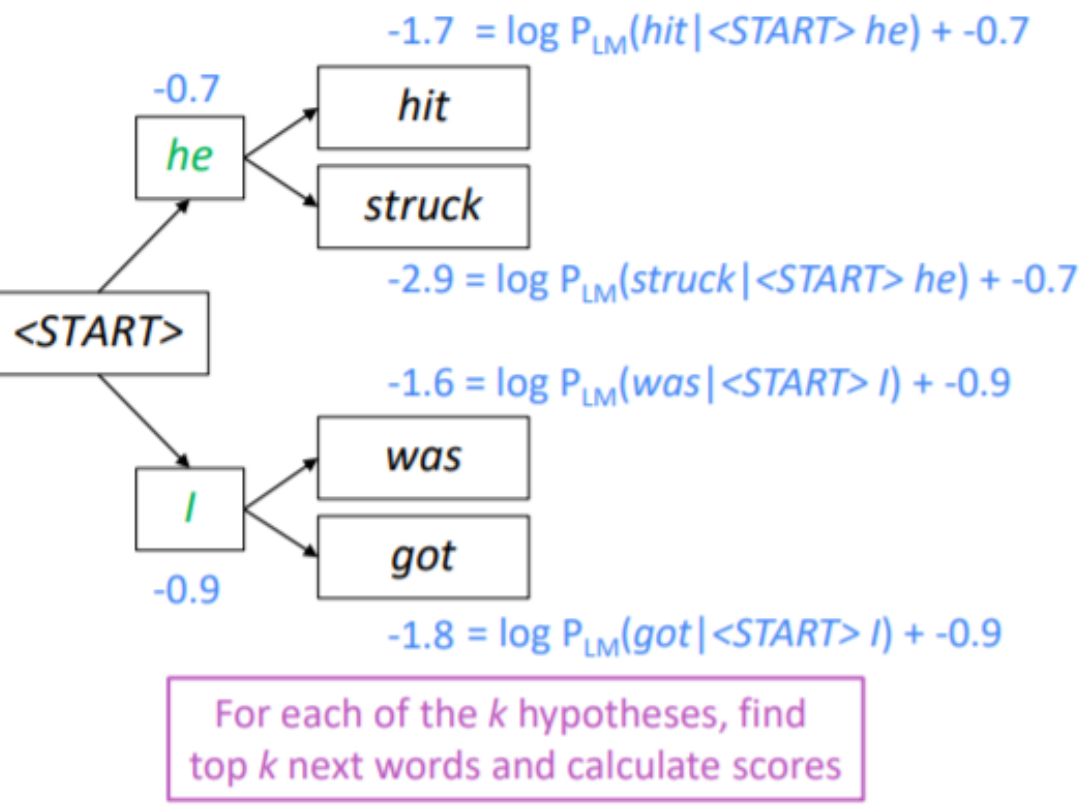
Beam search decoding

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



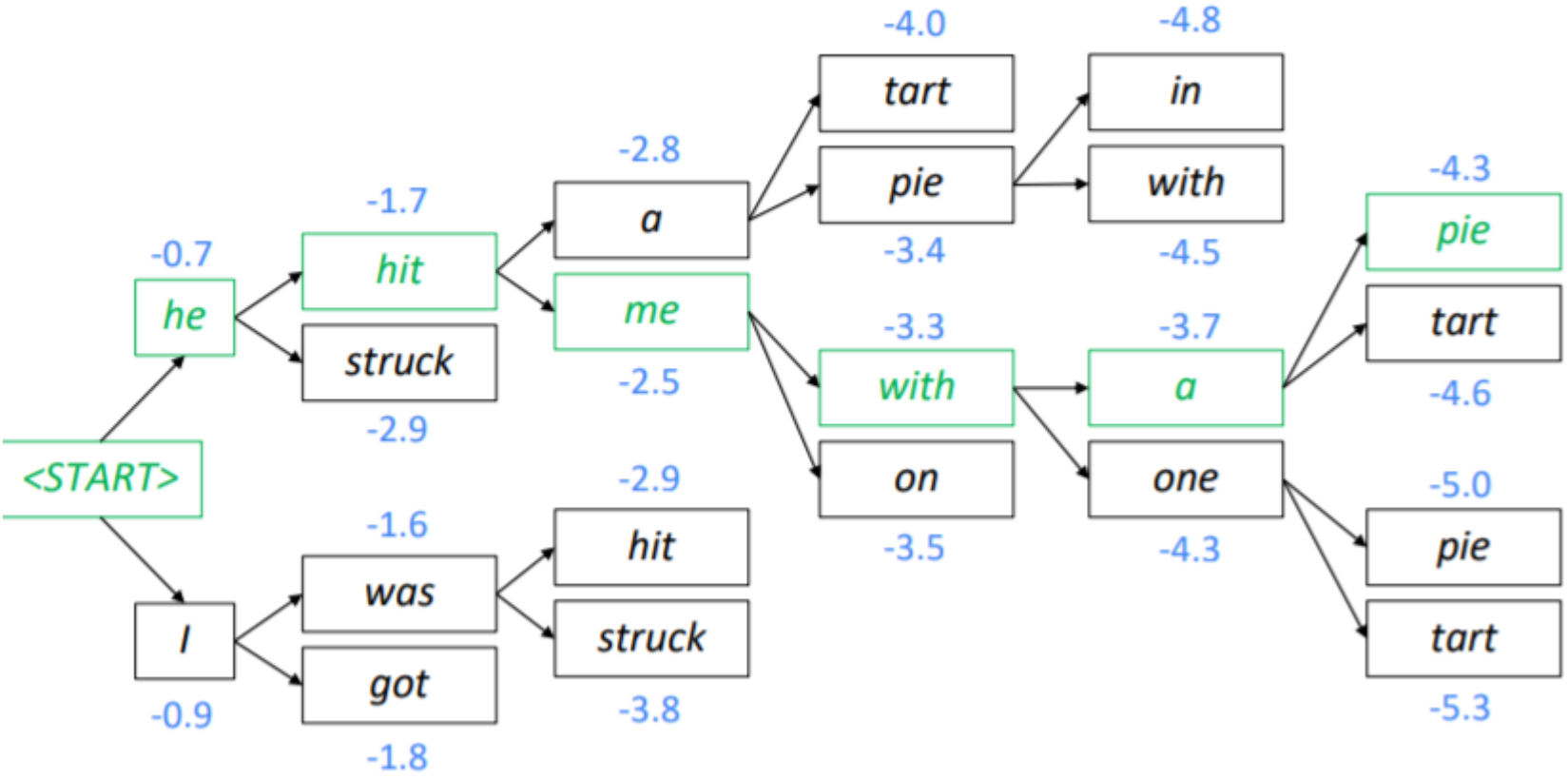
Beam search decoding

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

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

- In beam search decoding, different hypotheses may produce <END> tokens on different timesteps
  - When a hypothesis produces <END>, that hypothesis is complete.
  - Place it aside and continue exploring other hypotheses via beam search.

- Usually we continue beam search until:
  - We reach timestep *T* (where *T* is some pre-defined cutoff), or
  - We have at least *n* completed hypotheses (where *n* is pre-defined cutoff)

Beam search decoding: Normalization

- Each hypothesis $y_1, \ldots, y_t$ on our list has a score

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

- Problem with this: longer hypotheses have lower scores

- Fix: Normalize by length. Use this to select top one instead:

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

Advantages of NMT

- Better performance
  - More fluent
  - Better use of context
  - Better use of phrase similarities
- A single neural network to be optimized end-to-end
  - No subcomponents to be individually optimized
- Requires much less human engineering effort
  - No feature engineering
  - Same method for all language pairs

Disadvantages of NMT

- NMT is less interpretable
  - Hard to debug
- NMT is difficult to control
  - For example, can't easily specify rules or guidelines for translation
  - Safety concerns!

Evaluating Machine Translation

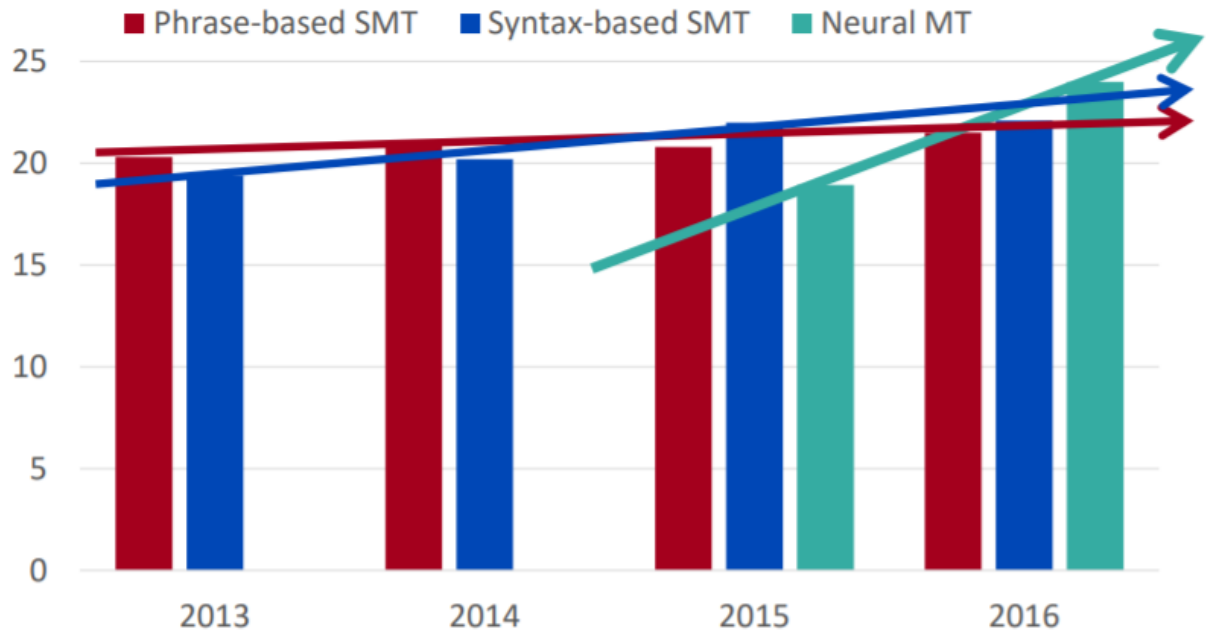
BLEU (**Bi**lingual **E**valuation **U**nderstudy)

You'll see BLEU in detail in Assignment 4!

- BLEU compares the machine-written translation to one or several human-written translation(s), and computes a similarity score based on:
  - *n*-gram precision (usually for 1, 2, 3 and 4-grams)
  - Plus a penalty for too-short system translations

- BLEU is useful but imperfect
  - There are many valid ways to translate a sentence
  - So a good translation can get a poor BLEU score because it has low *n*-gram overlap with the human translation ☹

MT progress over time



Neural Machine Translation went from a fringe research activity in **2014** to the leading standard method in **2016**

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

Problem of Machine Translation

- **Nope!**
- Many difficulties remain:
  - Out-of-vocabulary words
  - Domain mismatch between train and test data
  - Maintaining context over longer text
  - Low-resource language pairs

- Using common sense is still hard
- NMT picks up biases in training data
- Uninterpretable systems do strange things

3. Attention
from Stanford cs224n Winter2019

Attention

Attention is ...

- Behavioral and cognitive process of ***selectively concentrating*** on a discrete aspect of information while ***ignoring other*** perceivable information.
- State of ***arousal***
- It is the taking ***possession by the mind*** in clear and vivid form of one out of what seem several simultaneous objects or trains of thought.
- Focalization, the concentration of ***consciousness***
- ***The allocation*** of limited cognitive processing resources

Philosophy, Education, Psychology, Neuroscience

Reference: Attention (1): What is Attention? Youtube-Information and Intelligence

Attention

Attentional component

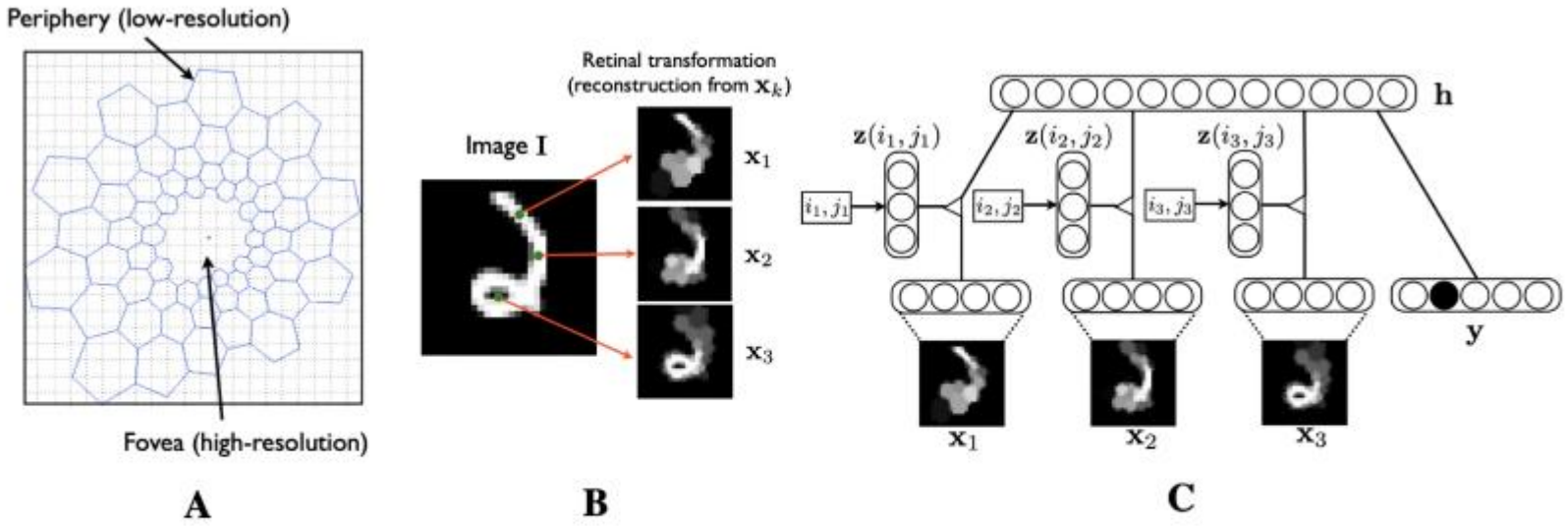
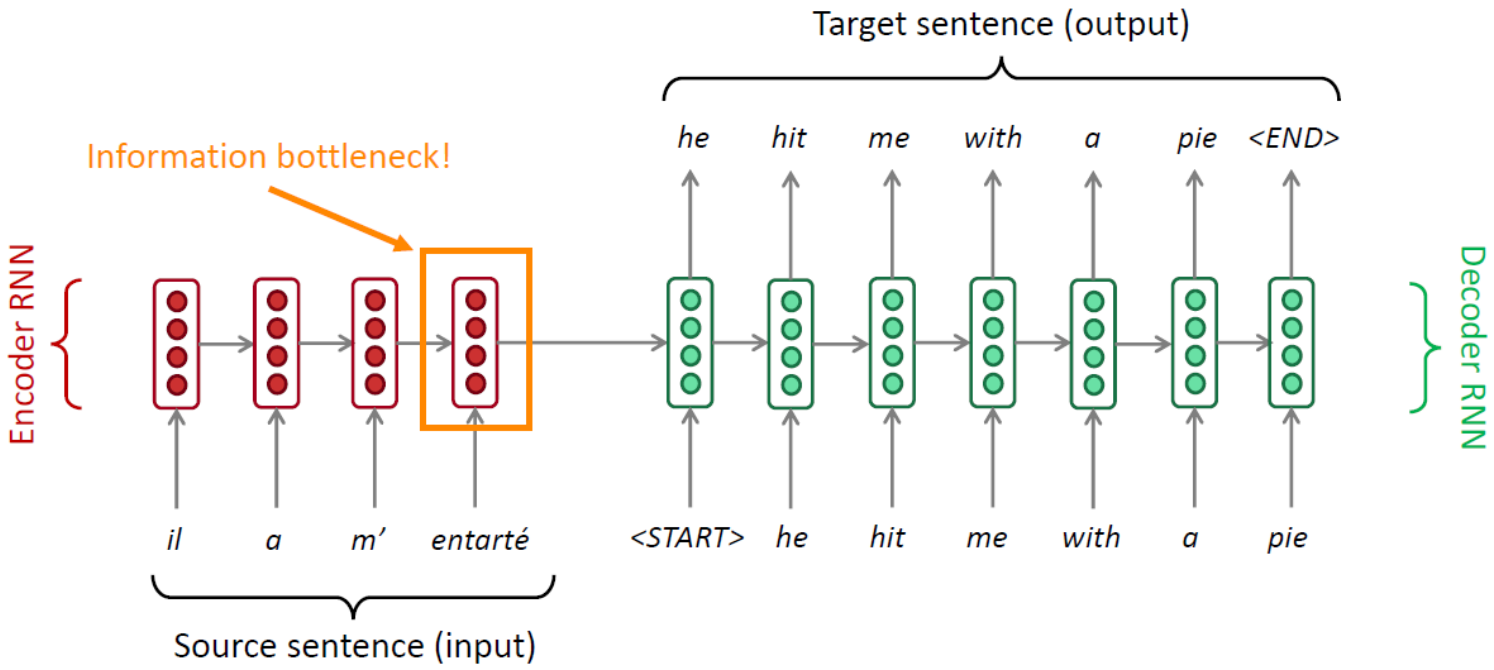


Figure 1: **A:** Illustration of the retinal transformation $\mathbf{r}(\mathbf{I}, (i, j))$. The center dot marks the pixel at position (i, j) (pixels are drawn as dotted squares). **B:** examples of glimpses computed by the retinal transformation, at different positions (visualized through reconstructions). **C:** Illustration of the multi-fixation RBM.

http://papers.nips.cc/paper/4089-learning-to-combine-foveal-glimpses-with-a-third-order-boltzmann-machine.pdf

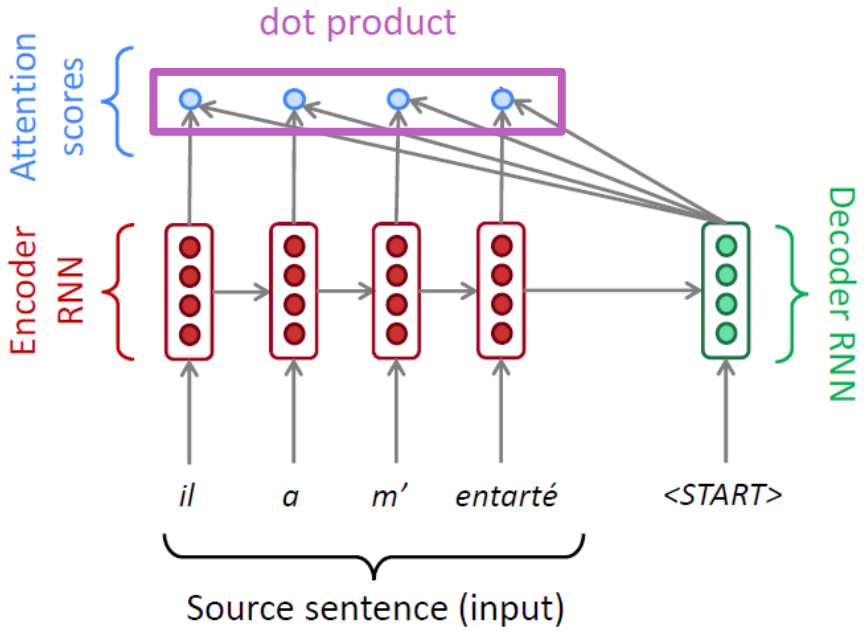
Reference: Attention (3): 1st Encounter with Artificial Neural Networks? Youtube-Information and Intelligence

Sequence-to-sequence



입력 받은 모든 단어들을 하나의 벡터로 압축하는 과정에서
'정보손실'의 문제 발생

Sequence-to-sequence with attention



1. Attention Score 구하기

hidden state of decoder : s_t
hidden state of encoder : h_i

dot product

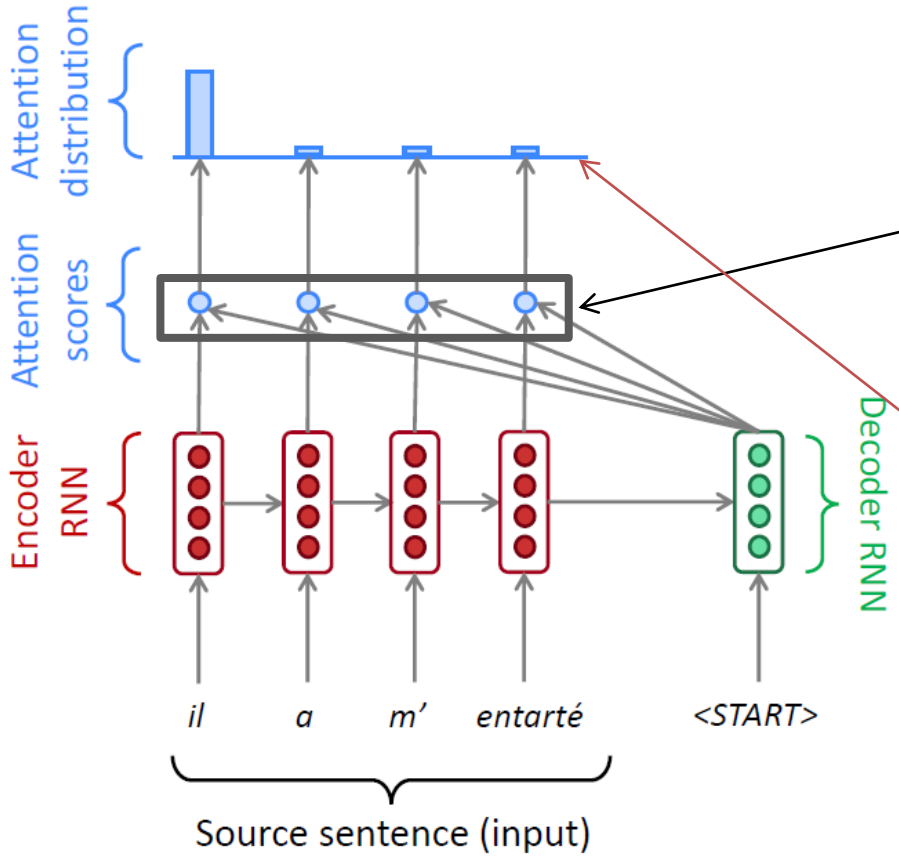
$$score(s_t{}^T, h_i) = s_t{}^T h_i$$

Sequence-to-sequence with attention

dot product 뿐만 아니라…

- Basic dot-product attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$
  - Note: this assumes $d_1 = d_2$
  - This is the version we saw earlier
- Multiplicative attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$
  - Where $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$ is a weight matrix
- Additive attention: $\boldsymbol{e}_i = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}) \in \mathbb{R}$
  - Where $\boldsymbol{W}_1 \in \mathbb{R}^{d_3 \times d_1}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices and $\boldsymbol{v} \in \mathbb{R}^{d_3}$ is a weight vector.
  - d_3 (the attention dimensionality) is a hyperparameter

Sequence-to-sequence with attention



2. Attention distribution (α^t) 구하기

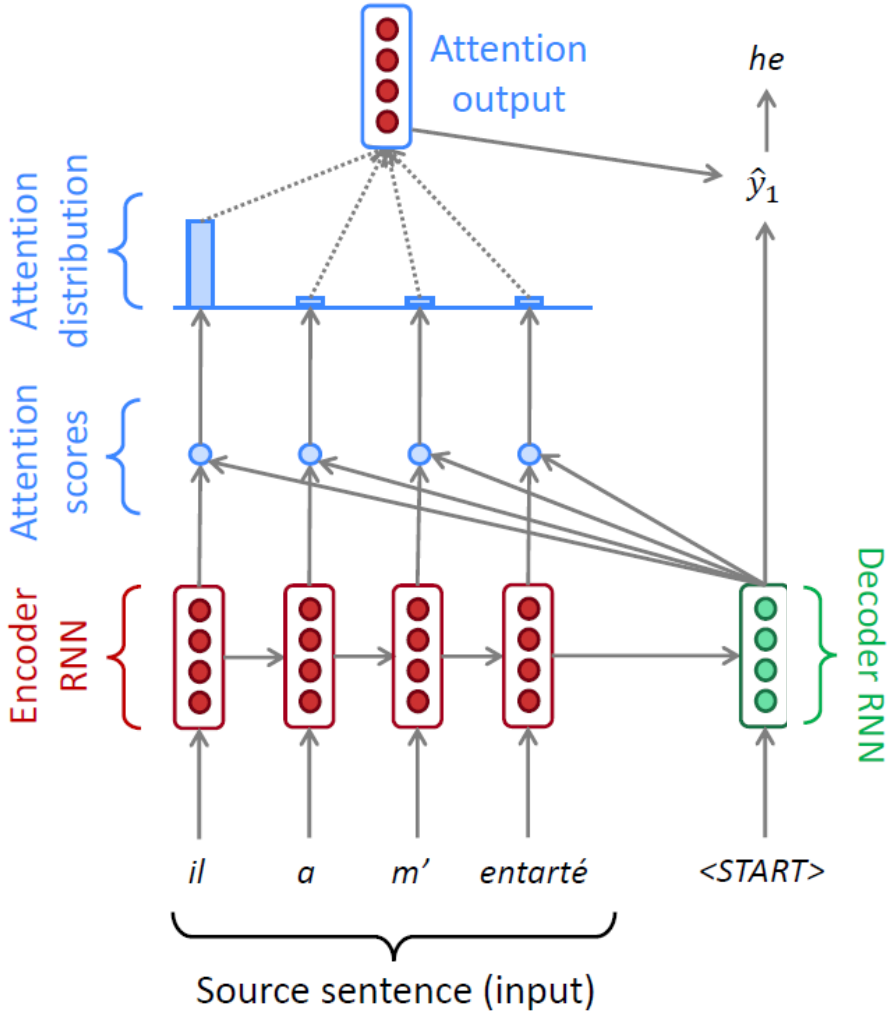
$$e^t = [\, s_t{}^T h_1, s_t{}^T h_2, \cdots, s_t{}^T h_N]$$

softmax 함수로 확률 분포 만들기

$$\alpha^t = \text{softmax}(e^t), \quad \sum_{i=1}^{N} \alpha_i{}^t = 1$$

→ 디코더의 입력단어와 얼마나 연관성이 있는지에 대한 가중치

Sequence-to-sequence with attention



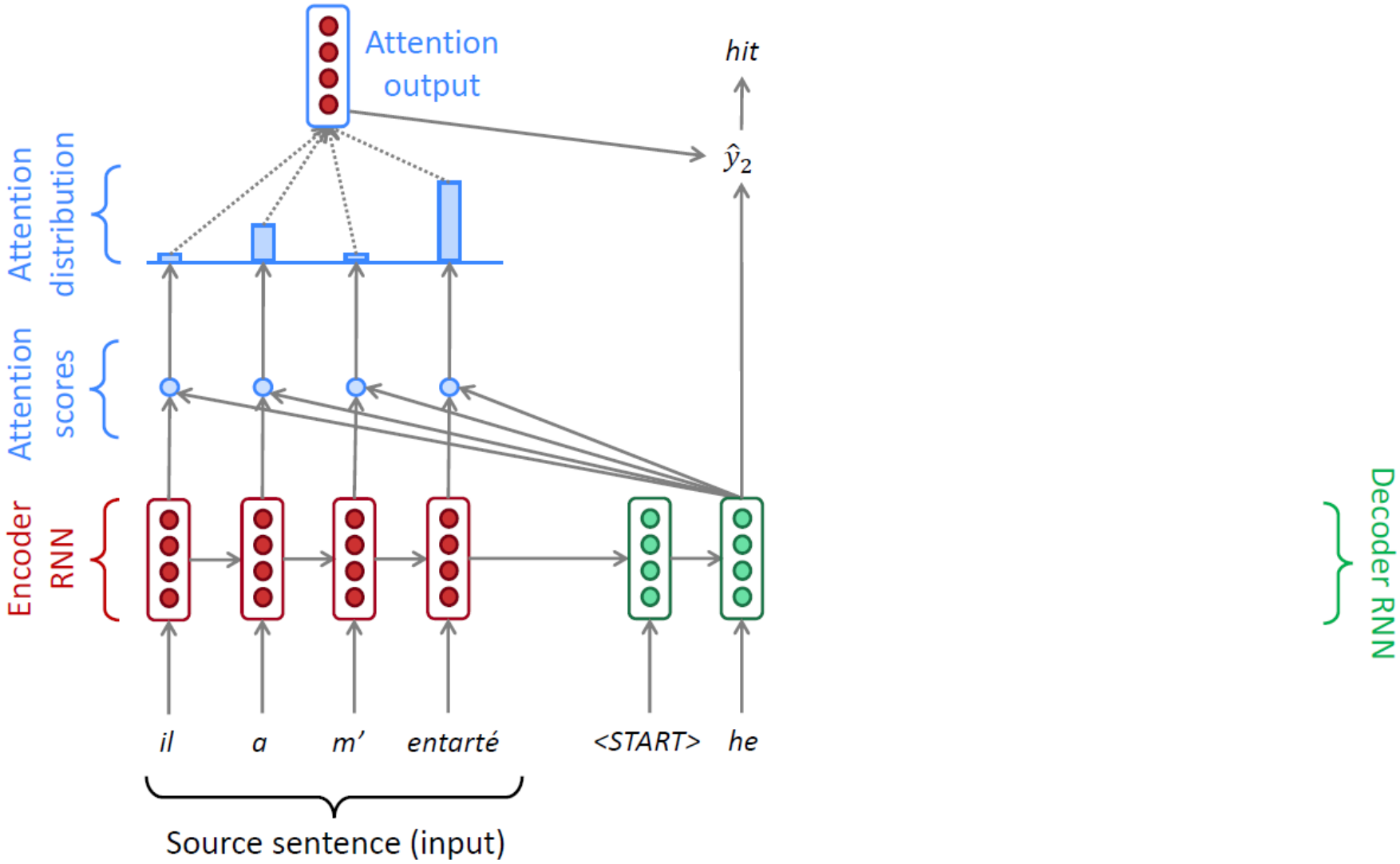
3. Attention output (a_t) 과
디코더의 hidden state (s_t) 결합

Attention output :

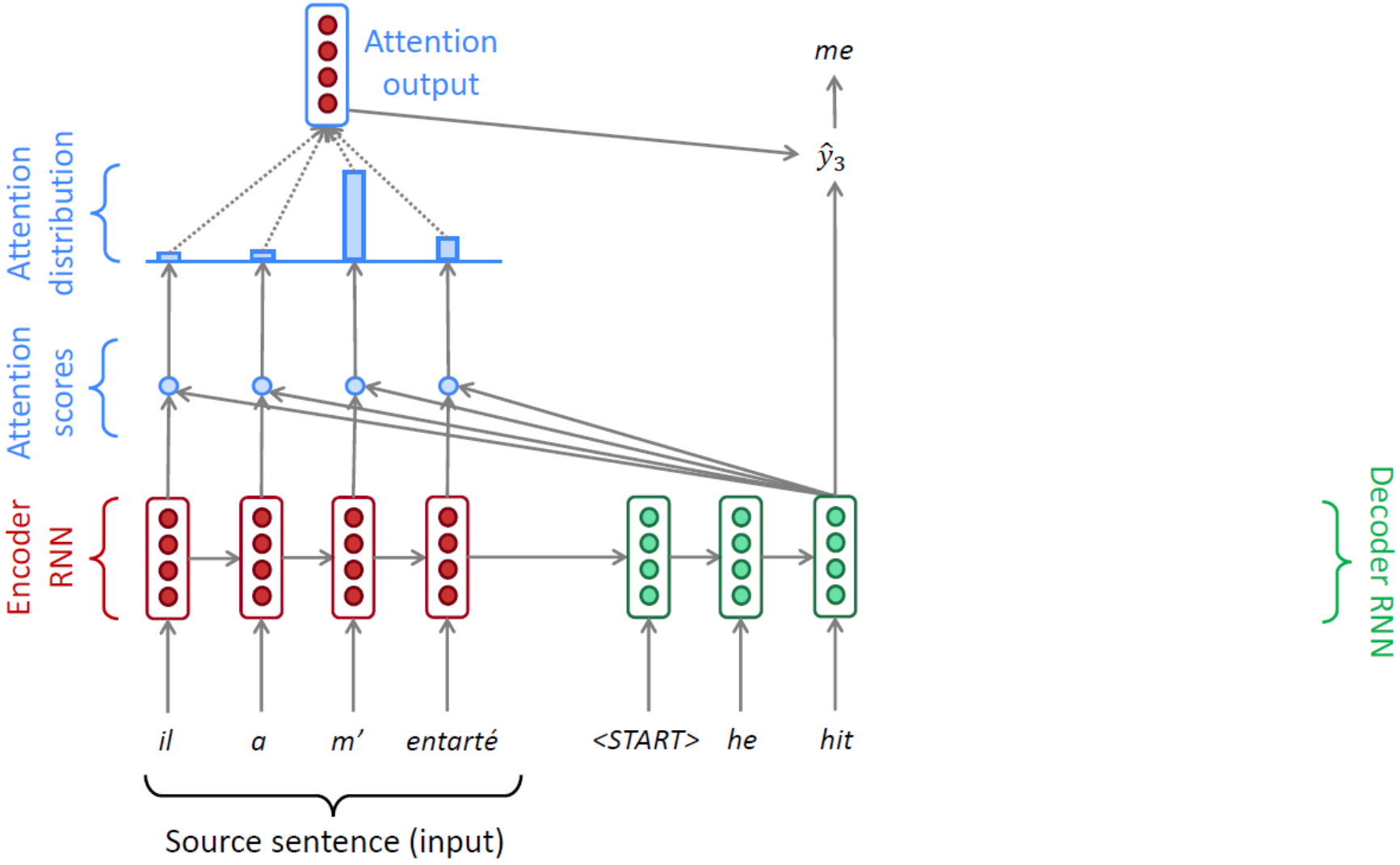
$$a_t = \sum_{i=1}^{N} \alpha_i{}^t h_i$$

→ a_t와 s_t을 결합해서 하나의 벡터로 만들고 $\hat{y}$ 계산에 활용

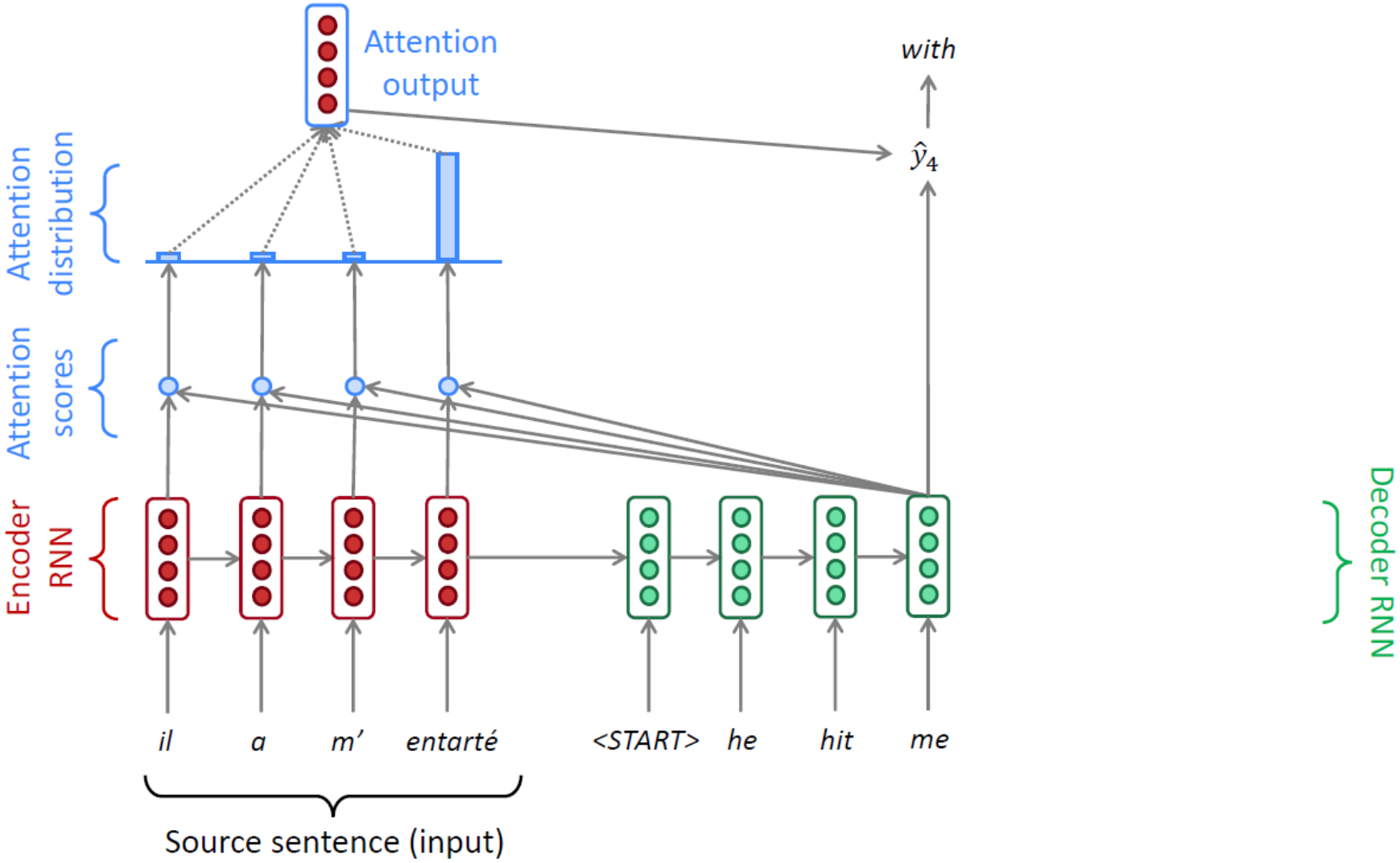
Sequence-to-sequence with attention



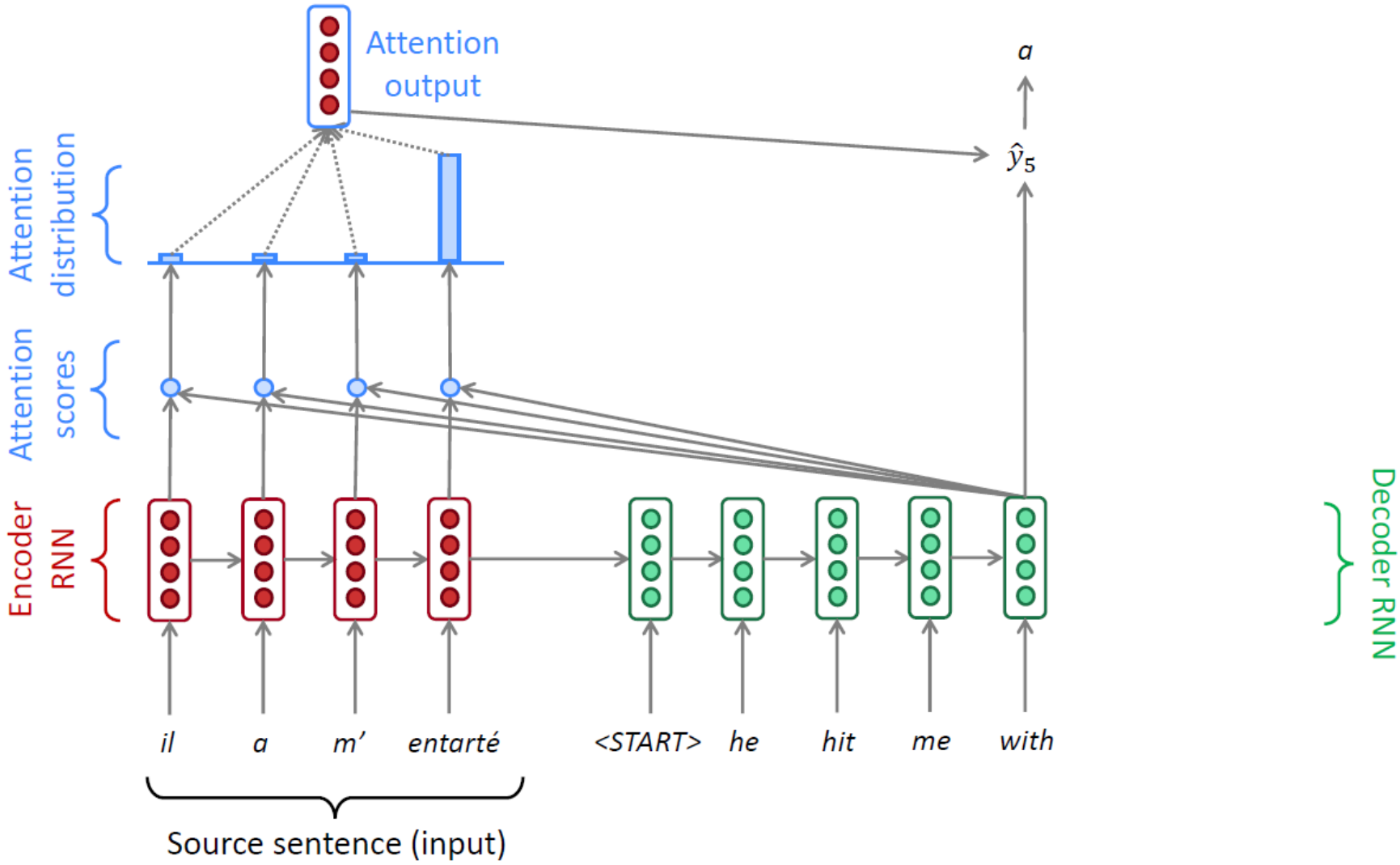
Sequence-to-sequence with attention



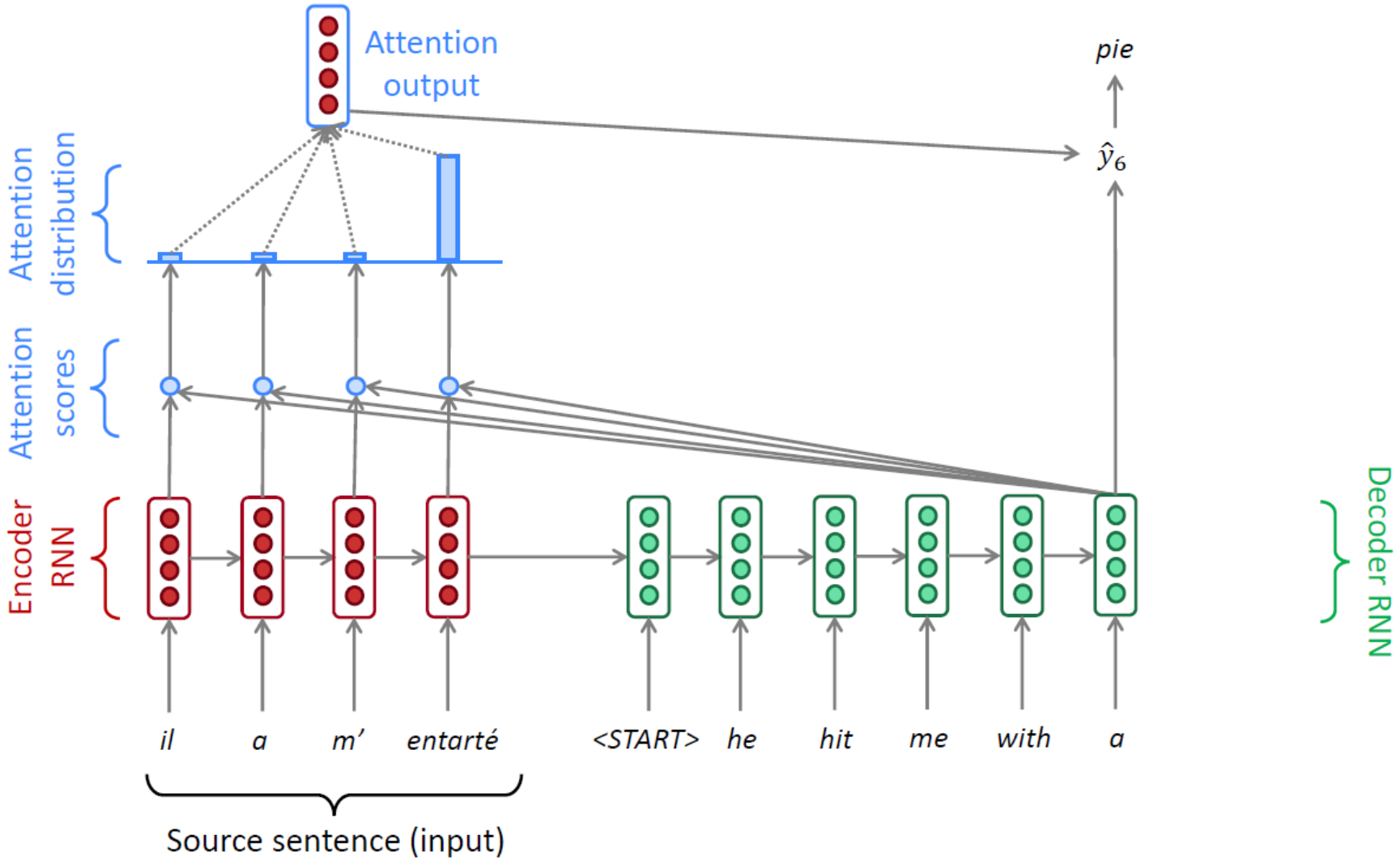
Sequence-to-sequence with attention



Sequence-to-sequence with attention



Sequence-to-sequence with attention



Advantages of Attention

- Improve NMT performance
- Solve the bottleneck problem
- Help with vanishing gradient problem
- Provide some interpretability

Advantages of Attention

→ Also,
attention can be used in many architectures and many tasks.

More general definition of Attention :

Given a set of vector values, and a vector query, attention is a technique to compute a weighted sum of the values, dependent on the query.

Attention

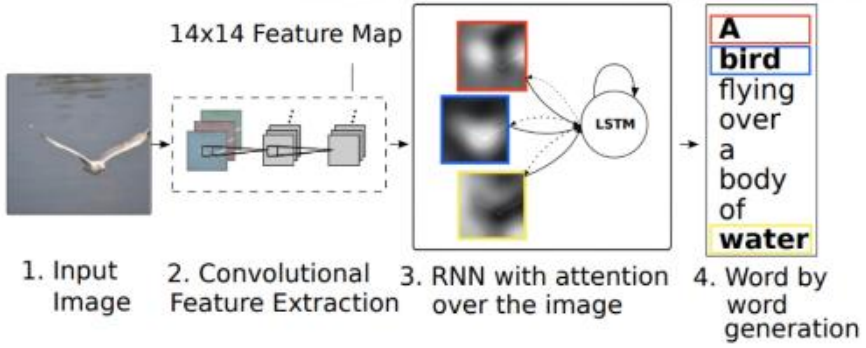
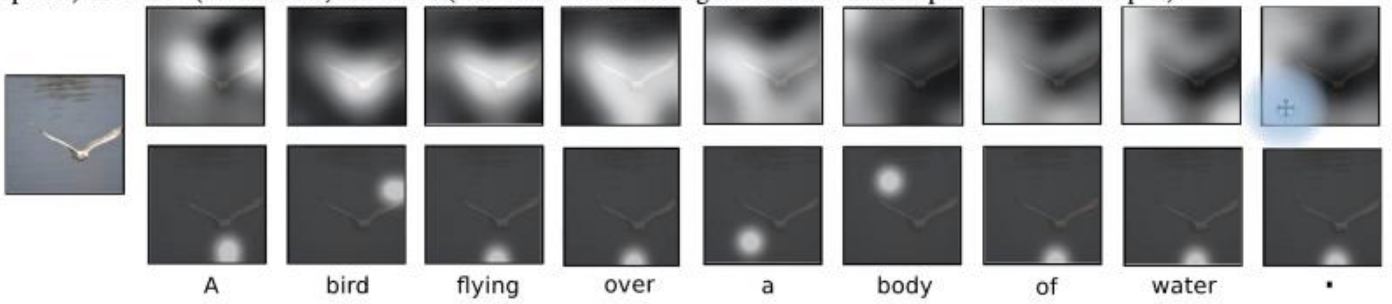


Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention. (Note that both models generated the same captions in this example.)



Reference: Attention (5): Show, Attend and Tell Youtube-Information and Intelligence

Self-Attention

Self-Attention (Intra-Attention)

- Attention mechanism relating ***different positions of a single sequence*** in order to compute a representation of the same sequence.

- It has been shown to be very useful in **machine reading, abstractive summarization,** or **image description generation**.

Reference: Attention (6): Self-Attention Youtube-Information and Intelligence

Self-Attention

Long Short-Term Memory-Networks for Machine Reading

Jianpeng Cheng, Li Dong, Mirella Lapata

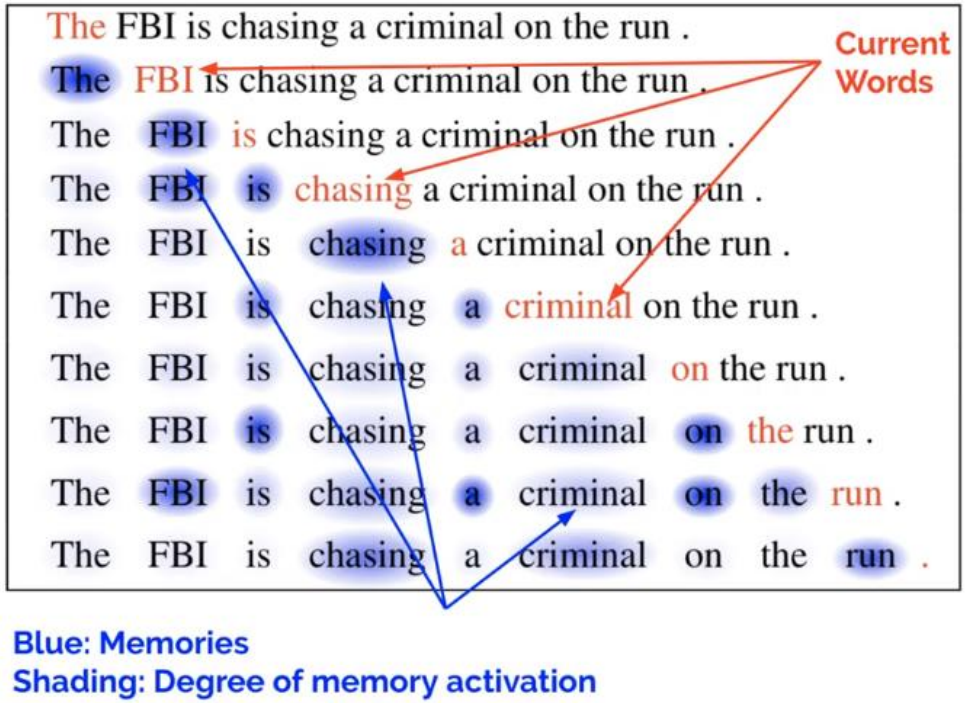
(Submitted on 25 Jan 2016 (v1), last revised 20 Sep 2016 (this version, v7))

In this paper we address the question of how to render sequence-level networks better at handling structured input. We propose a machine reading simulator which processes text incrementally from left to right and performs shallow reasoning with memory and attention. The reader extends the Long Short-Term Memory architecture with a memory network in place of a single memory cell. This enables adaptive memory usage during recurrence with neural attention, offering a way to weakly induce relations among tokens. The system is initially designed to process a single sequence but we also demonstrate how to integrate it with an encoder-decoder architecture. Experiments on language modeling, sentiment analysis, and natural language inference show that our model matches or outperforms the state of the art.

- Self-attention to do machine reading
- Self-attention mechanism to learn the ***correlation between the current words and the previous part of the sentence***.

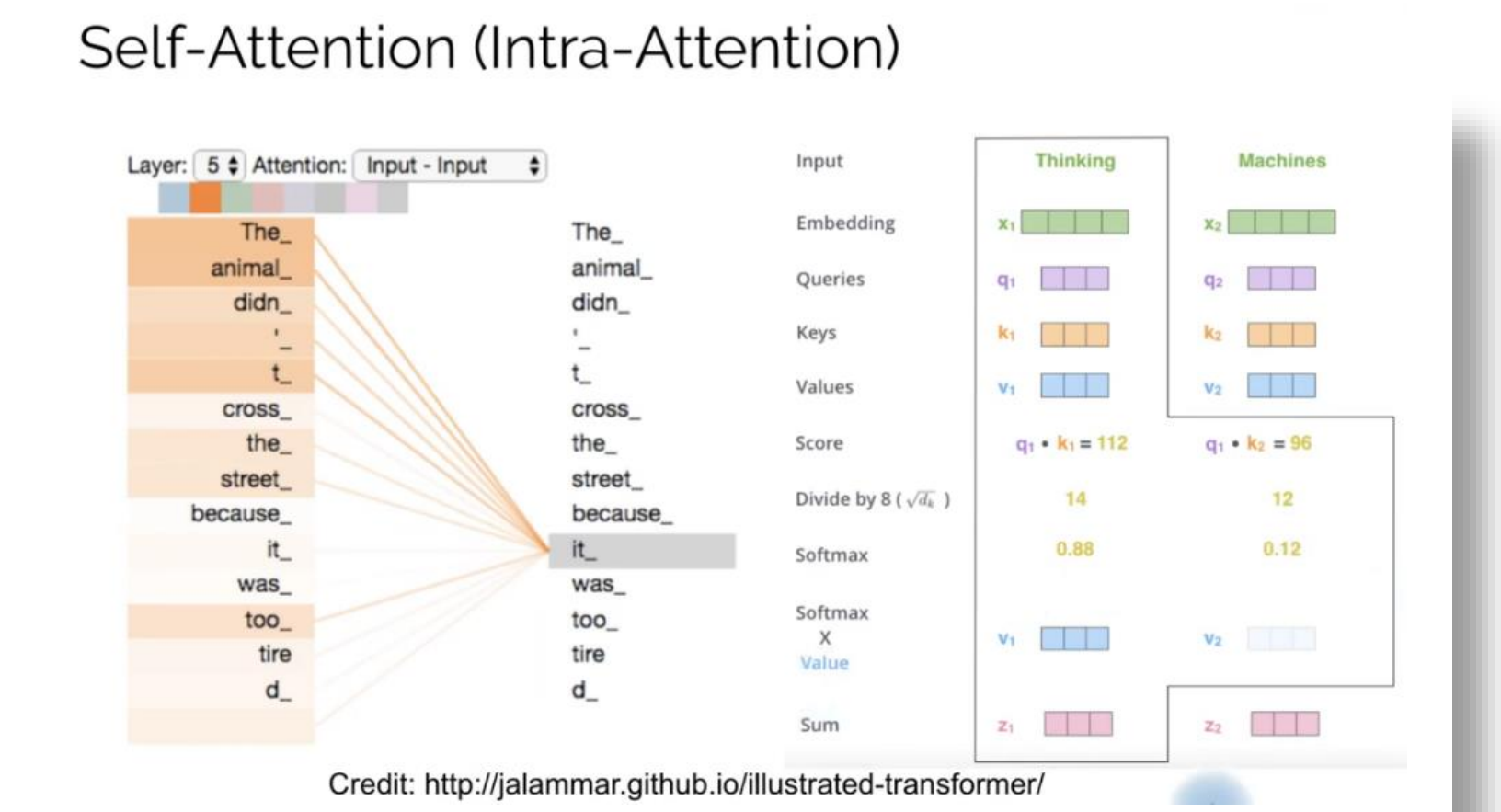
Reference: Attention (6): Self-Attention Youtube-Information and Intelligence

Self-Attention



Reference: Attention (6): Self-Attention Youtube-Information and Intelligence

Self-Attention



Reference: Attention (7): Attention is All You Need, Transformers(1) Youtube-Information and Intelligence

END

감사합니다.

01

Pre-NMT

02

NMT

03

Attention

04

Thank you