Machine Translation, Seq2Seq, Attention

2018.06.20

조수환

Statistical Machine Translation

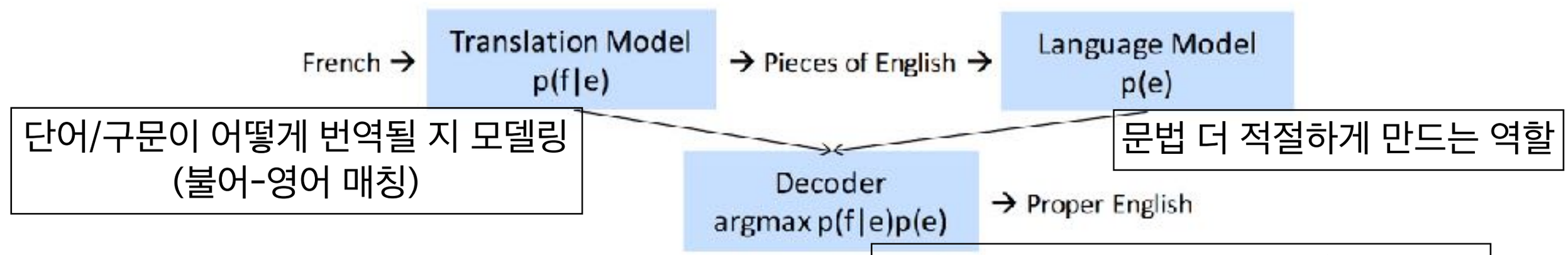
- Use parallel corpora (ex. Bible)
- Very complex (딥러닝 좋음)

Current statistical machine translation systems

- Source language f, e.g. French
- Target language e, e.g. English
- Probabilistic formulation (using Bayes rule)

$$\hat{e} = \text{argmax}_e p(e|f) = \text{argmax}_e p(f|e)p(e)$$

- Translation model p(f|e) trained on parallel corpus
- Language model p(e) trained on English only corpus (lots, free!)



단어/구문이 어떻게 번역될 지 모델링
(불어-영어 매칭)

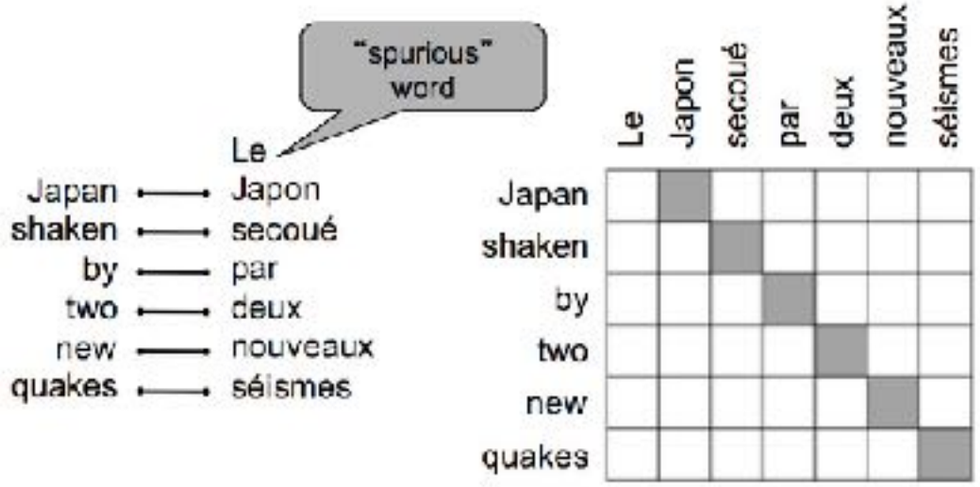
문법 더 적절하게 만드는 역할

ex. ‘go home’ is better than ‘go house’

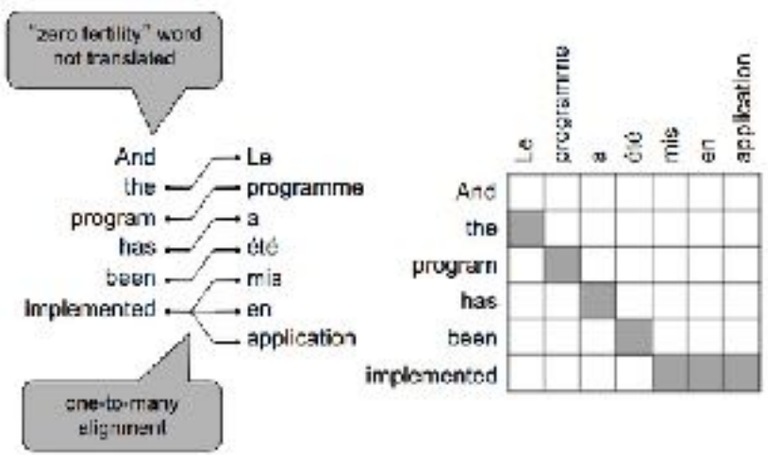
Statistical Machine Translation

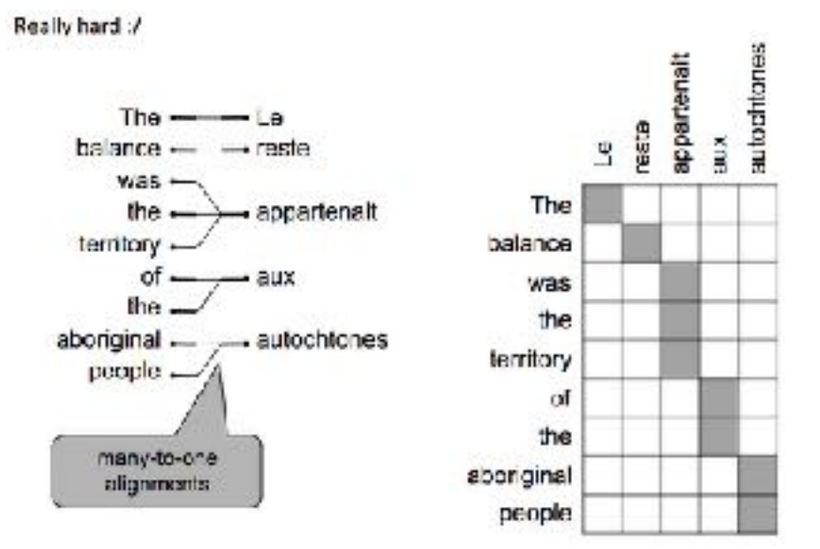
Step 1: Alignment source language — target language 매칭

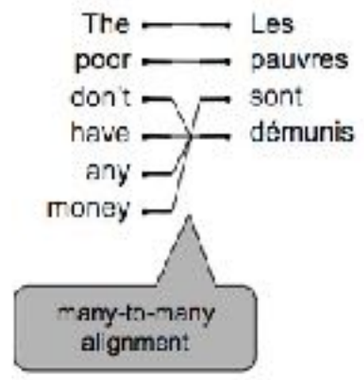
Goal: know which word or phrases in source language would translate to what words or phrases in target language? → Hard already!

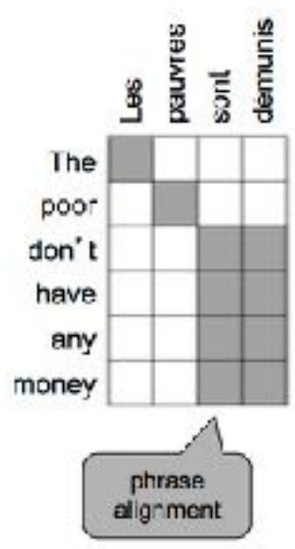


어렵다!
- source에 있지만 target에 없는 단어
- source에 없는데 번역할 때 필요한 단어
- 일대다 / 다대일 / 다대다 alignments
- 이러한 모든 조합에 대한 확률을 구하여 combine하는 것은 매우 복잡함





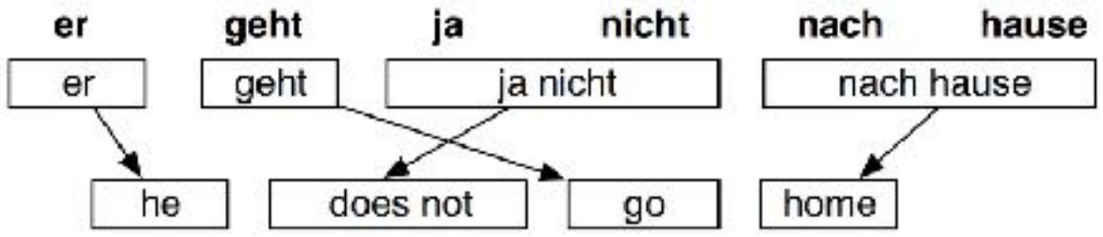




Statistical Machine Translation

Step 1: Alignment

- We could spend an entire lecture on alignment models
- Not only single words but could use phrases, syntax
- Then consider reordering of translated phrases

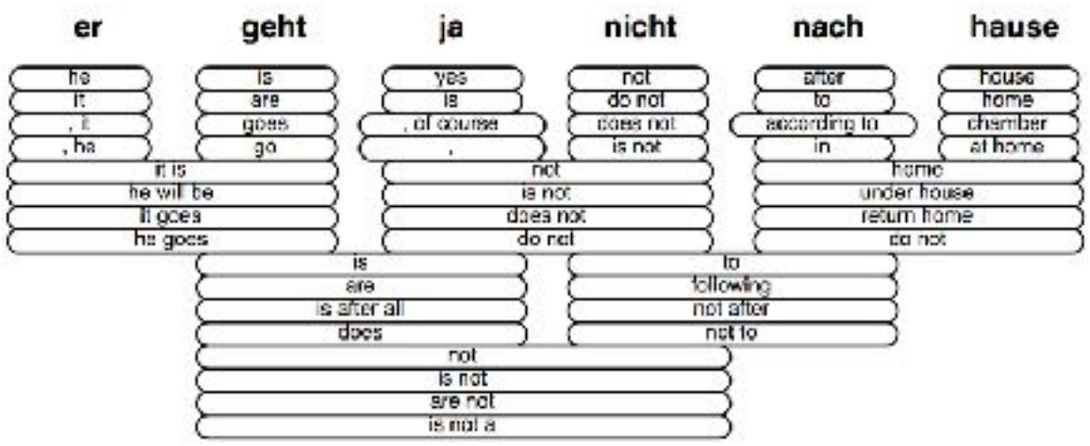


단어 뿐만 아니라 가능한 phrase 조합별로
번역하여 가능한 candidate를 모두 구한 후

Beam search를 통해 가장 자연스러운 조합 택

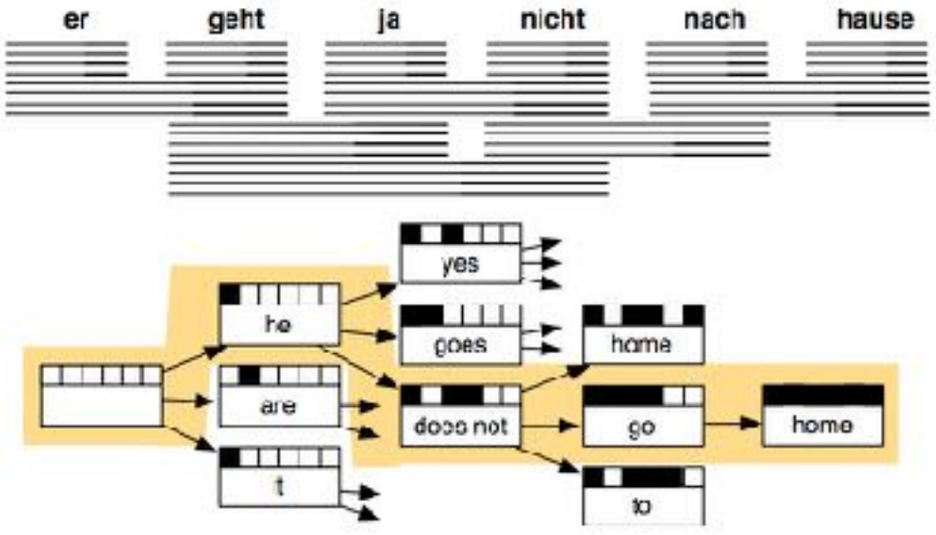
After many steps

Each phrase in source language has many possible translations resulting in large search space:



Decode: Search for best of many hypotheses

Hard search problem that also includes language model



Statistical Machine Translation

- Hundreds of important details we haven't mentioned here
- Systems have many separately-designed subcomponents
- Lots of feature engineering
  - Need to design features to capture particular language phenomena
- Require compiling and maintaining extra resources
  - Like tables of equivalent phrases
- Lots of human effort to maintain
  - Repeated effort for each language pair!

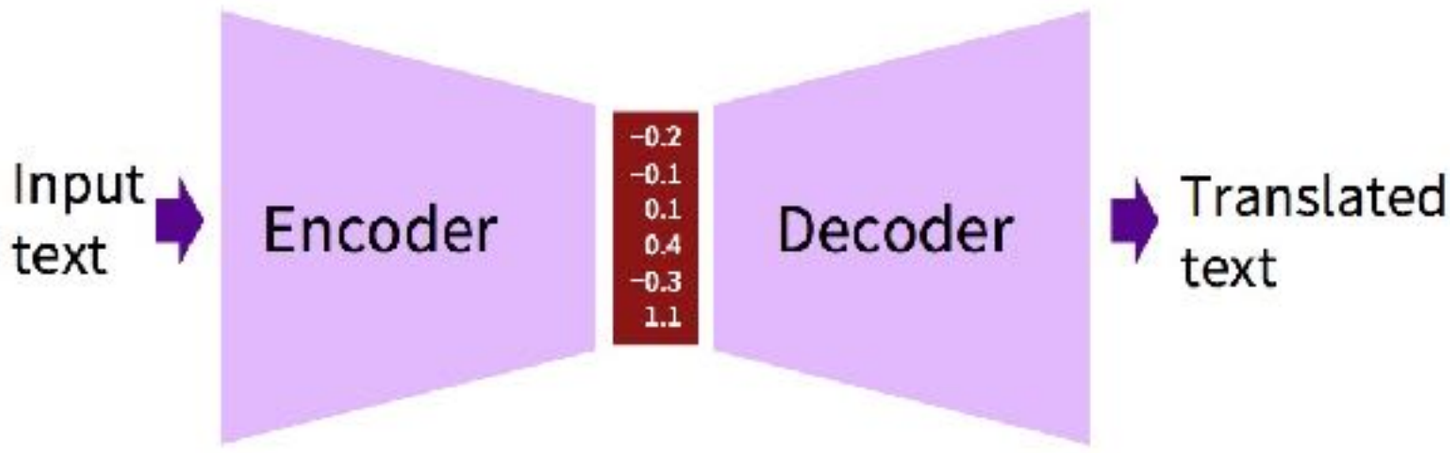
ex. alignment, reordering …

한 마디로, 너무 복잡하고 손도 많이 간다!

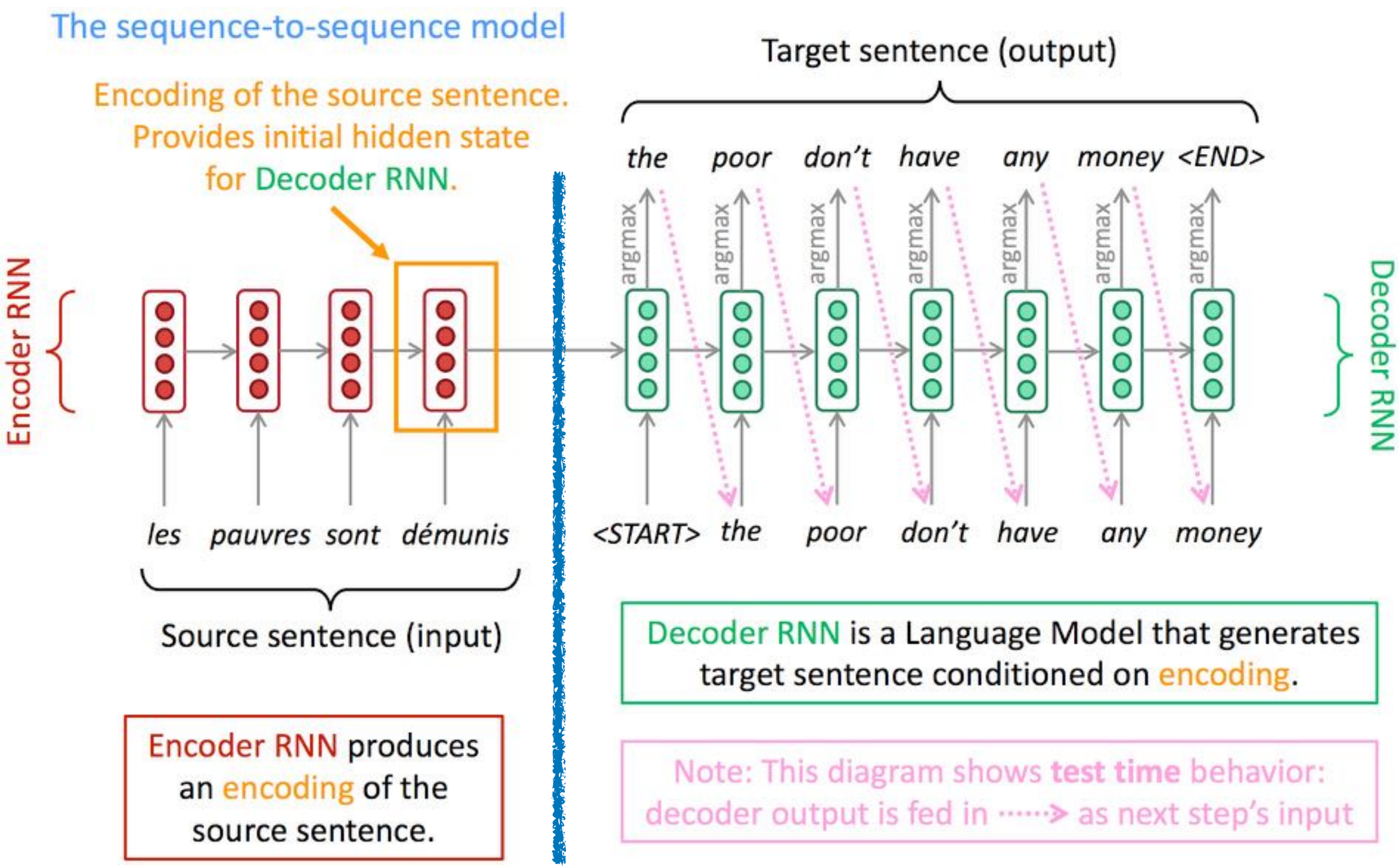
Neural Machine Translation

- Neural Machine Translation (NMT) is a way to do Machine Translation with a *single neural network*

- The neural network architecture is called sequence-to-sequence (aka seq2seq) and it involves *two* RNNs. (Encoder / Decoder)



Neural Machine Translation



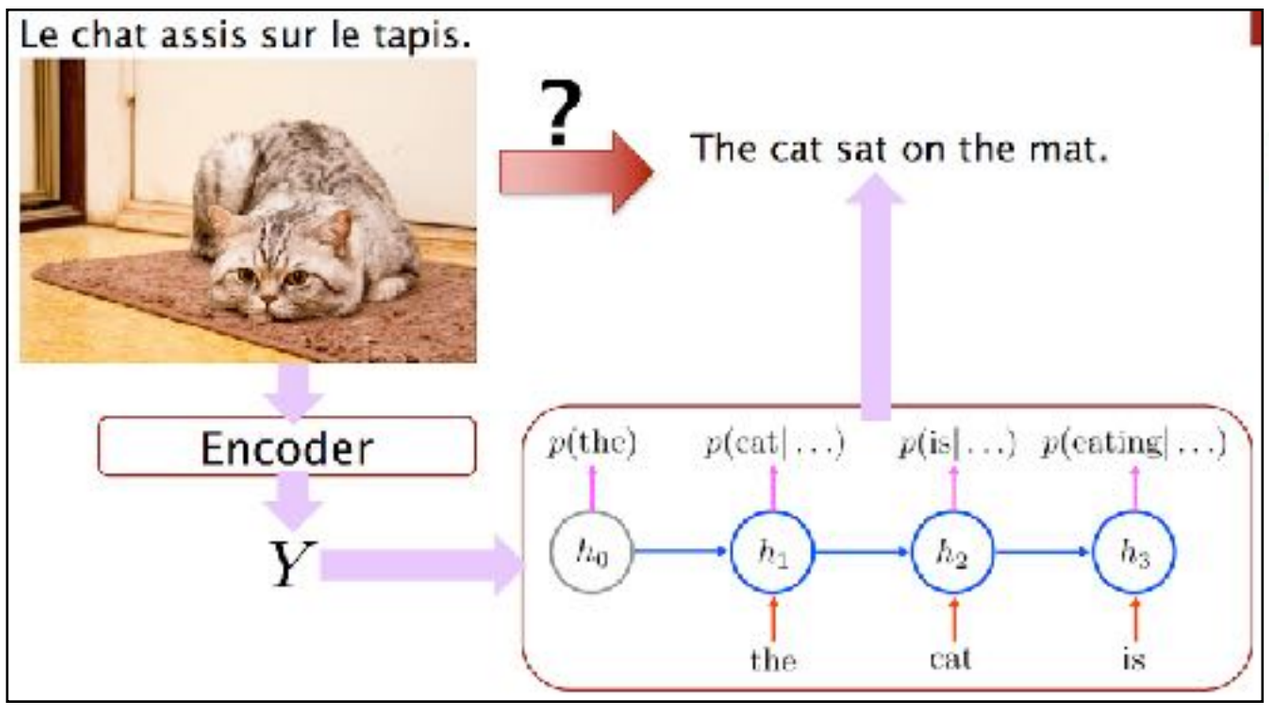
Neural Machine Translation

- 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*



Encoder의 마지막 hidden state와 이전 timestep까지 번역된 단어들에 대한 조건부 확률

Training a NMT system

$$J = \frac{1}{T}\sum_{t=1}^{T} J_t = J_1 + J_2 + J_3 + J_4 + J_5 + J_6 + J_7$$

= negative log prob of "the"

= negative log prob of "have"

= negative log prob of <END>

$\hat{y}_1$ $\hat{y}_2$ $\hat{y}_3$ $\hat{y}_4$ $\hat{y}_5$ $\hat{y}_6$ $\hat{y}_7$

Encoder RNN

Decoder RNN

les *pauvres* *sont* *démunis* *<START>* *the* *poor* *don't* *have* *any* *money*

Source sentence (from corpus)

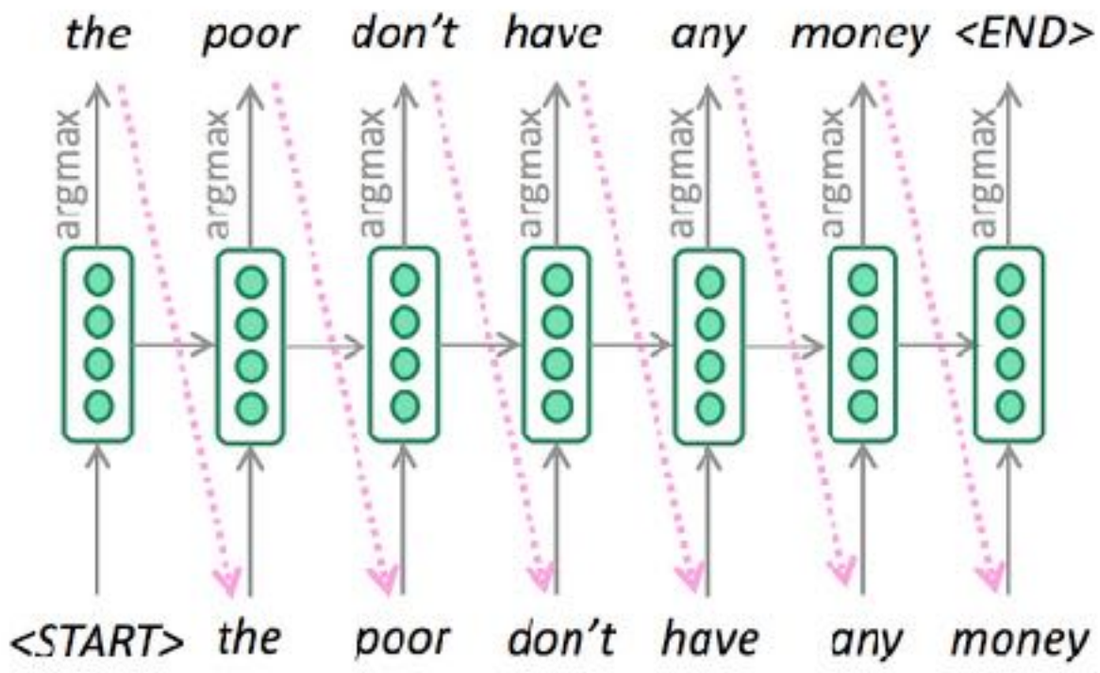
Target sentence (from corpus)

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

Training a NMT system

Better-than-greedy decoding?

- We showed how to generate (or "decode") the target sentence by taking argmax on each step of the decoder



- This is greedy decoding (take most probable word on each step)

Greedy decoding에서는 한번 잘못 번역하면 되돌릴 수 없다
→Beam search!

Training a NMT system

Beam search decoding

- Ideally we want to find *y* that maximizes

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

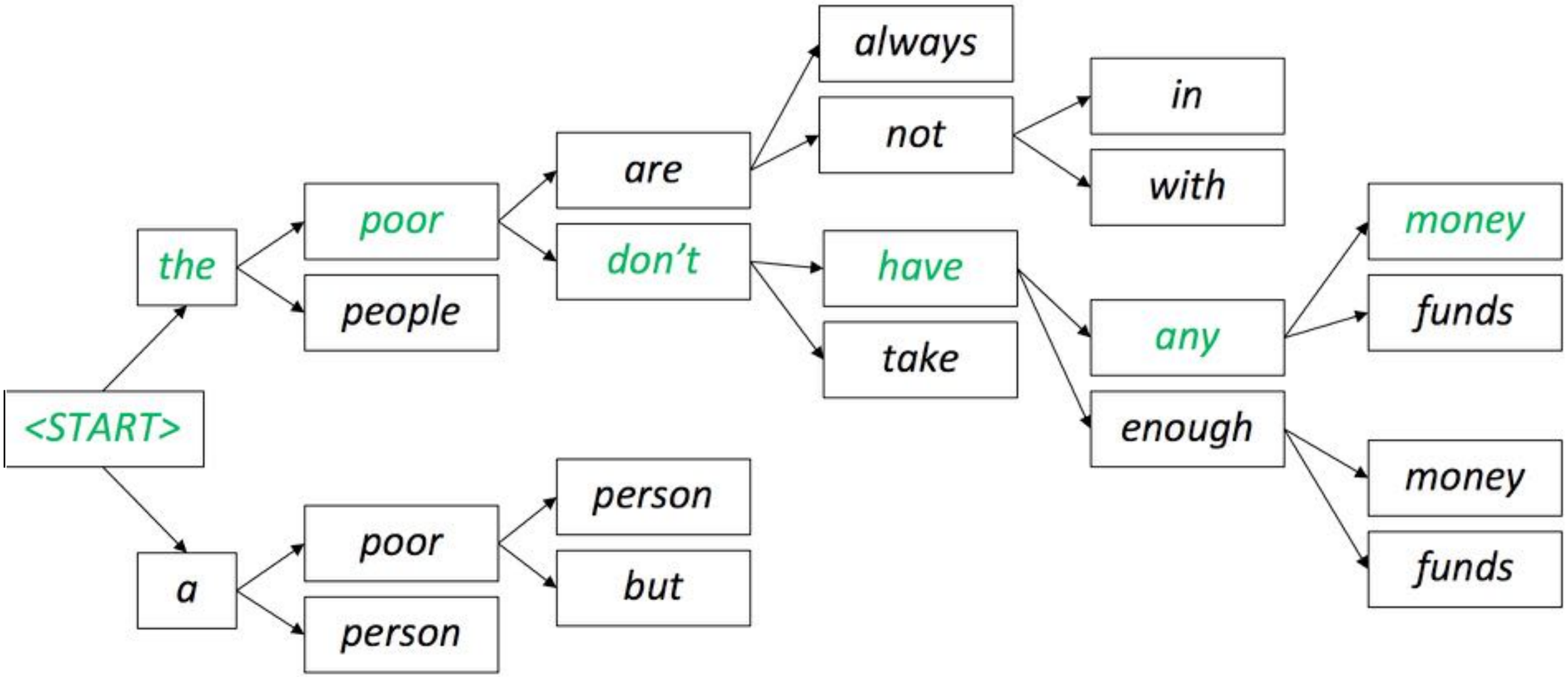
- We could try enumerating all *y* → too expensive!
  - Complexity $O(V^T)$ where *V* is vocab size and *T* is target sequence length

- **Beam search**: On each step of decoder, keep track of the *k* most probable partial translations
  - *k* is the beam size (in practice around 5 to 10)
  - Not guaranteed to find optimal solution
  - But much more efficient!

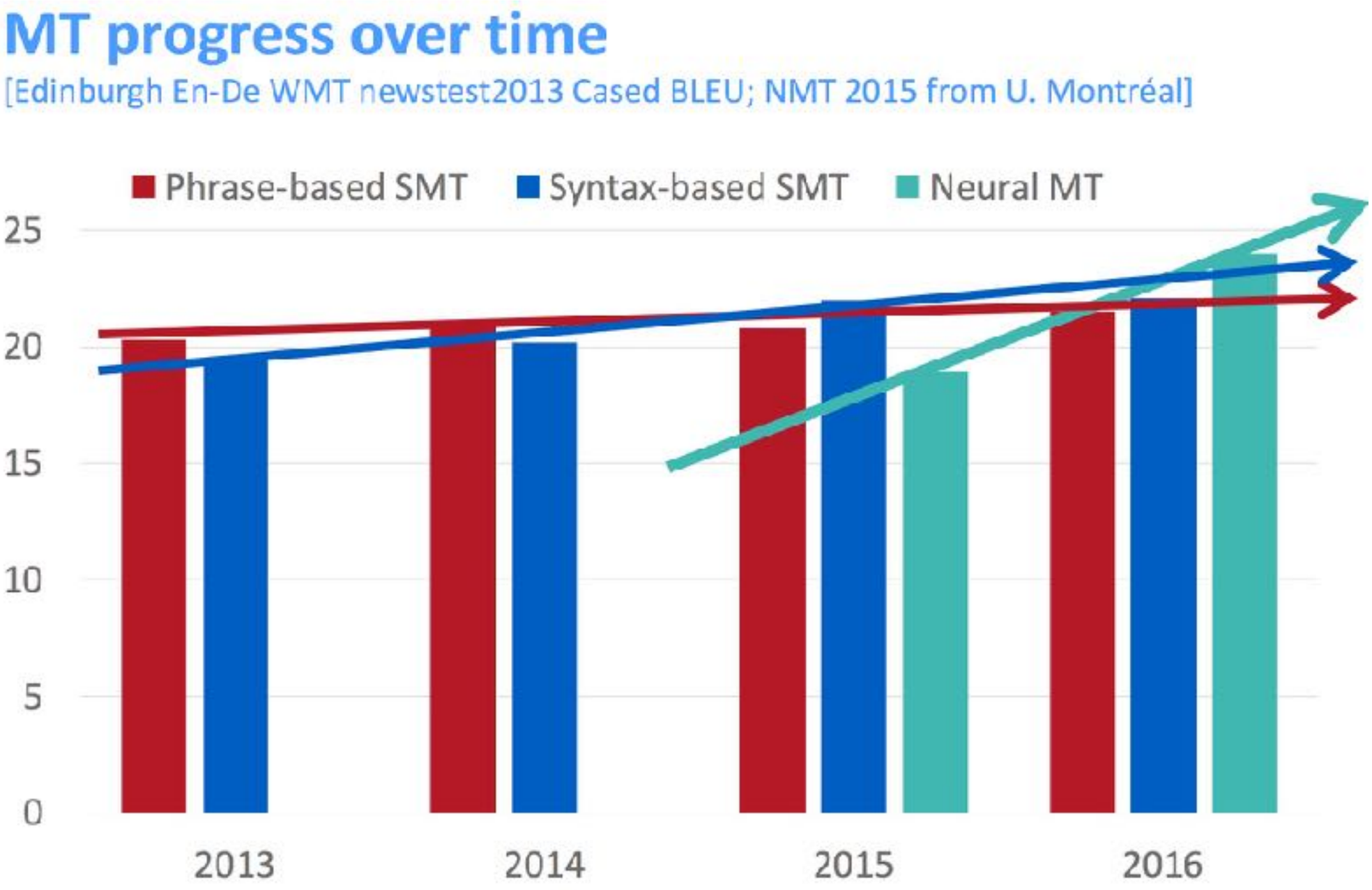
매 스텝마다 가장 좋아보이는 k개 tracking

Training a NMT system

Beam size = 2



Good NMT…



SMT보다도 훨씬 빠르게 성능 향상되고 있으며, 이미 SMT는 넘어섬

Advantages / Disadvantages

Advantages

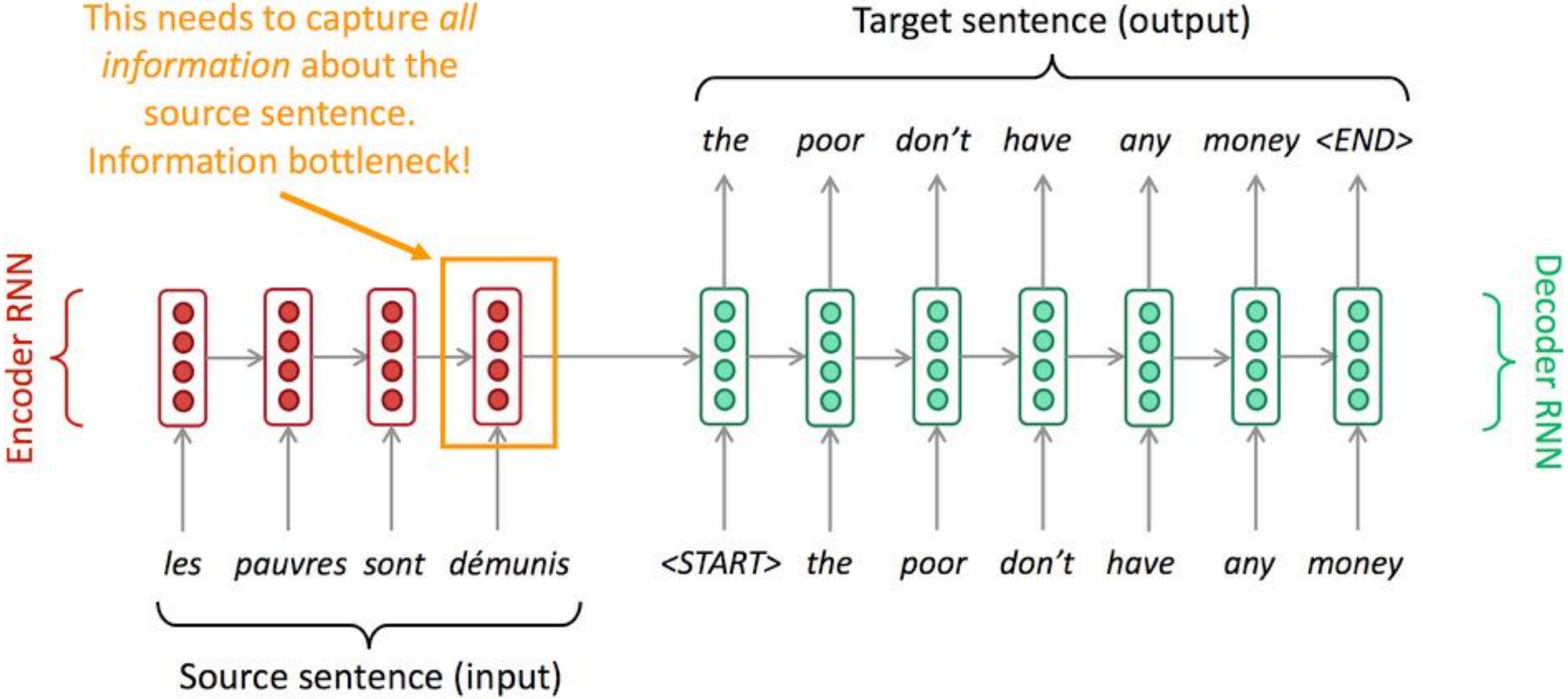
- Better performance
  - More fluent
  - Better use of context
  - Better use of phrase similarities
- A single neural network to be optimized end-to-end
  - No subcomponents to be individually optimized
- Requires much less human engineering effort
  - No feature engineering
  - Same method for all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

Disadvantages

- 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!

Bottleneck problem

맨 마지막 hidden state만으로
모든 번역문 생성하는 것은 무리



→ Encoder의 다른 hidden states도 활용하자

Attention

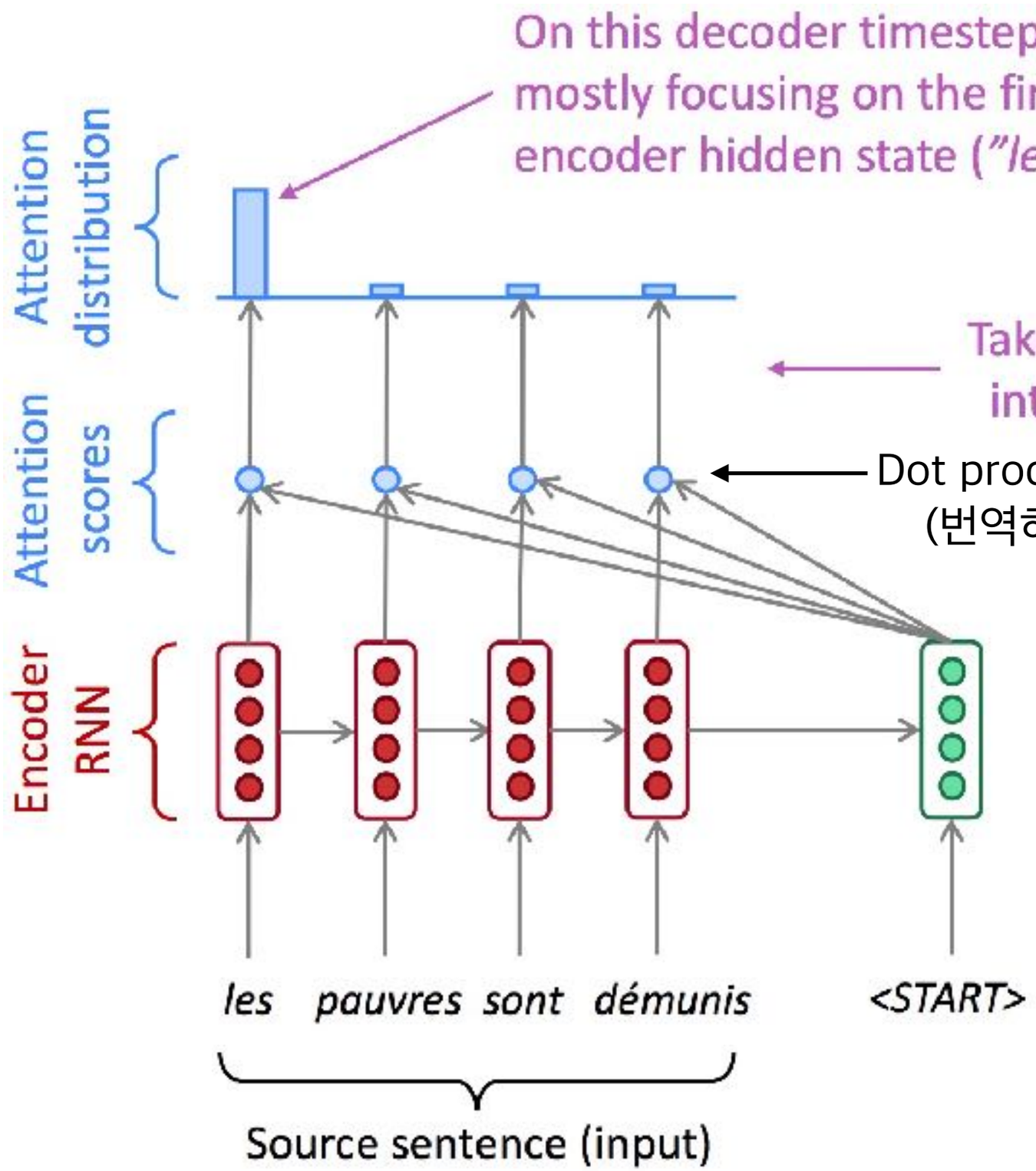
- **Attention** provides a solution to the bottleneck problem.

- Core idea: on each step of the decoder, *focus on a particular part* of the source sequence

Source sentence의 어느 부분을 다음 timestep에서 번역할 것인가?

Source sentence와 target sentence 간의 implicit alignment
→ 사람이 번역할 때와 유사하다

Attention



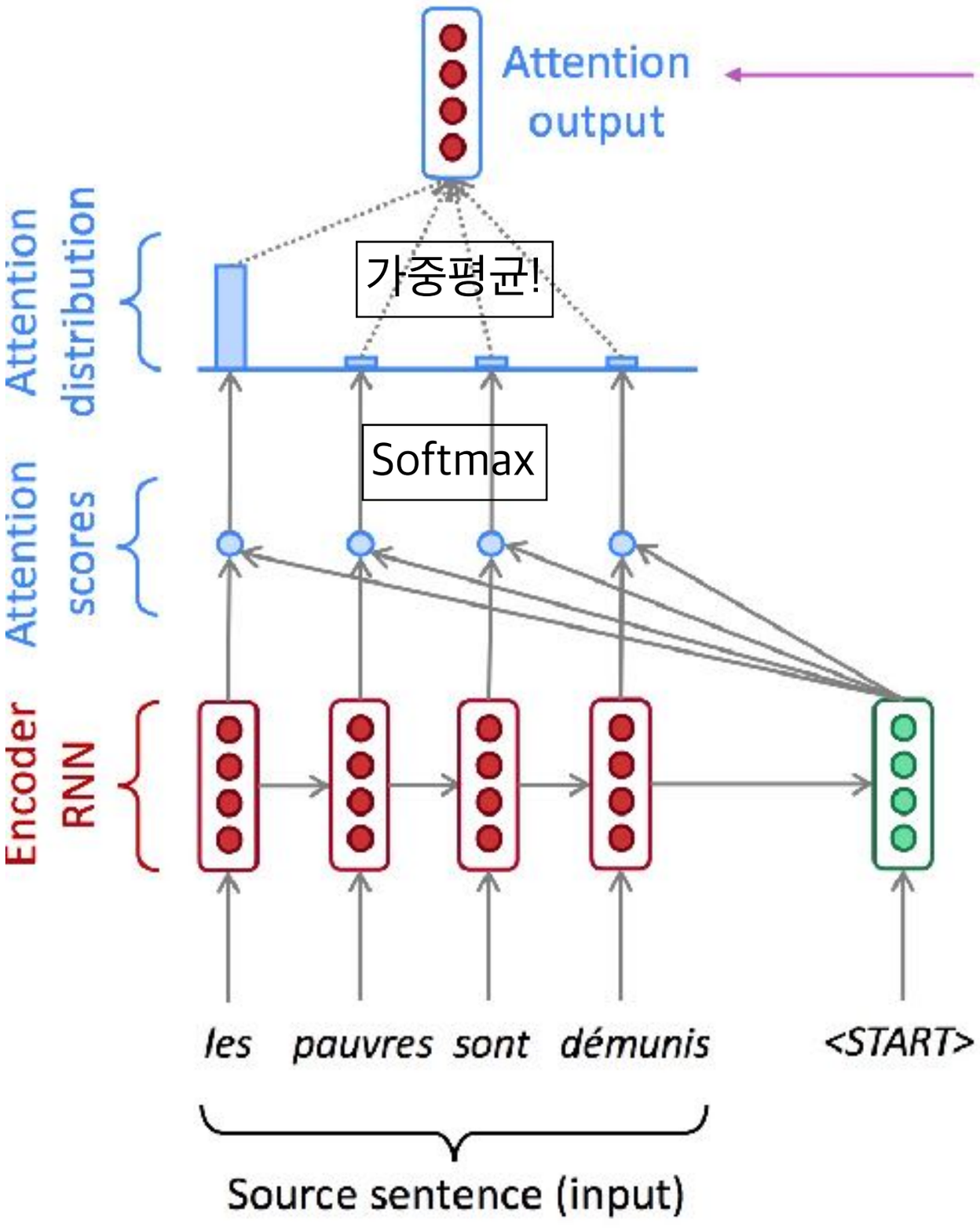
이번 timestep에서는 encoder의 첫 번째 hidden에 포커스를 맞추겠다

(번역하려는 지점과 유사도가 높은 부분)

Dot product 외에도 다양한 score 이용 가능

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a [\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) \end{cases}$$

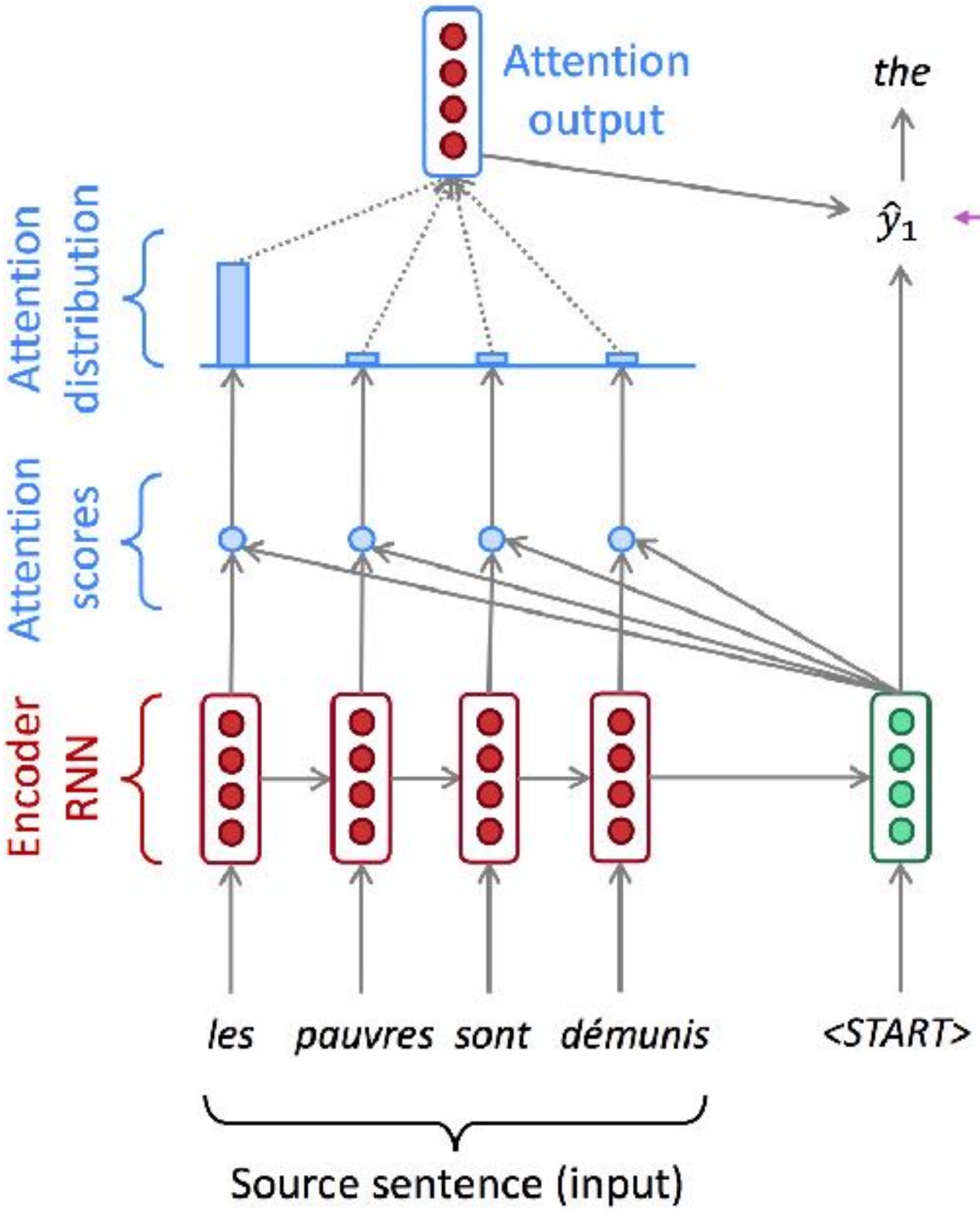
Attention



Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information the **hidden states** that received high attention.

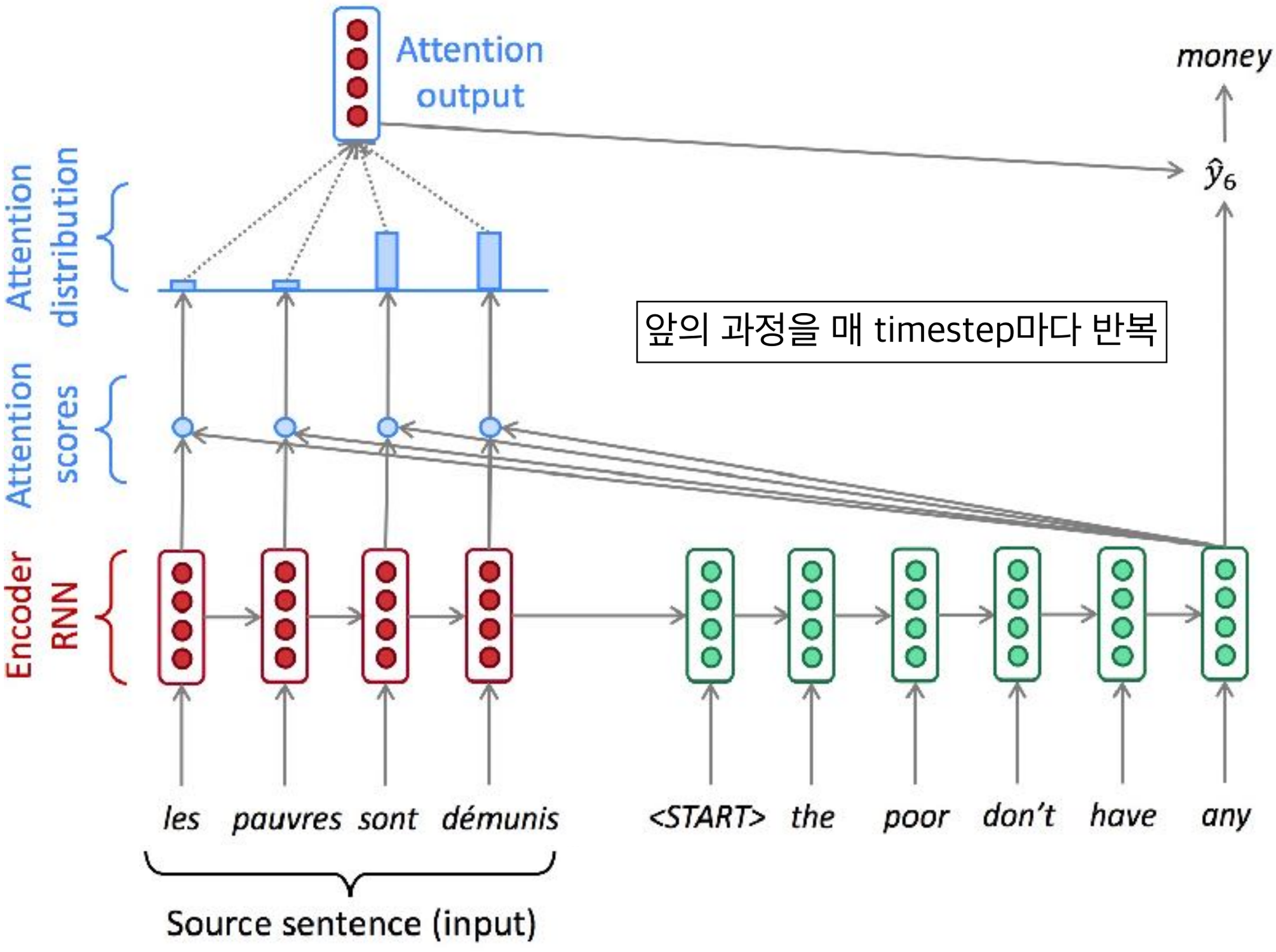
Attention



Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

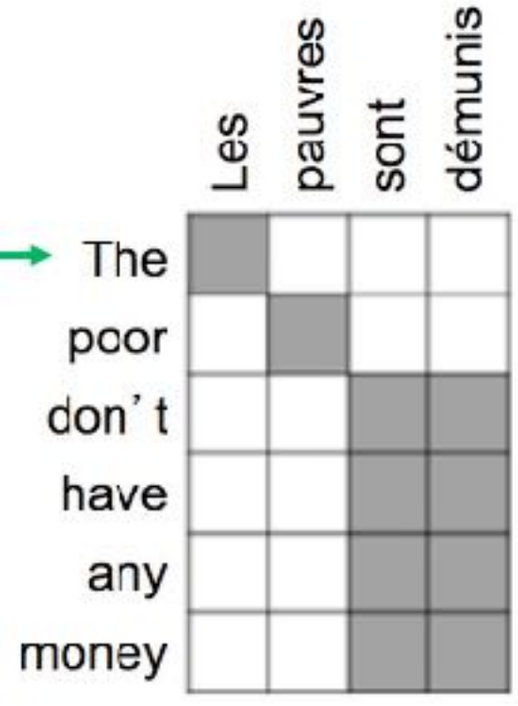
이렇게 구한 attention output을 기존의 decoder hidden state와 concat한다

Attention


Attention output
money
$\hat{y}_6$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
앞의 과정을 매 timestep마다 반복
les pauvres sont démunis
<START> the poor don't have any
Source sentence (input)

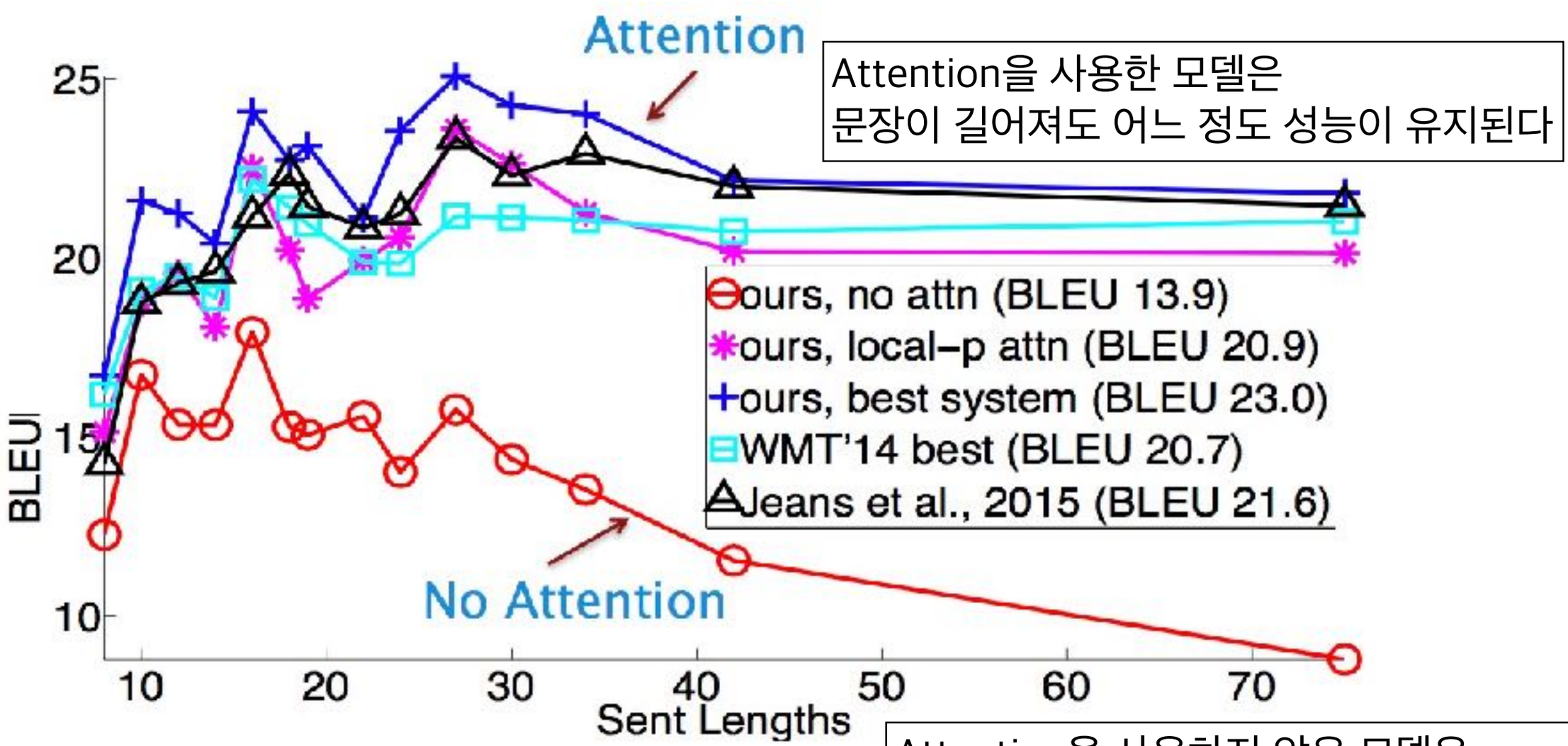
Attention is Good!

- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - We get alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself



alignment를 따로 해주지 않았지만
모델이 알아서 alignment, translation 함께 수행

Attention is Good!



Attention을 사용한 모델은
문장이 길어져도 어느 정도 성능이 유지된다

Attention을 사용하지 않은 모델은
문장 길이가 길어지면 급격하게 성능 저하

감사합니다