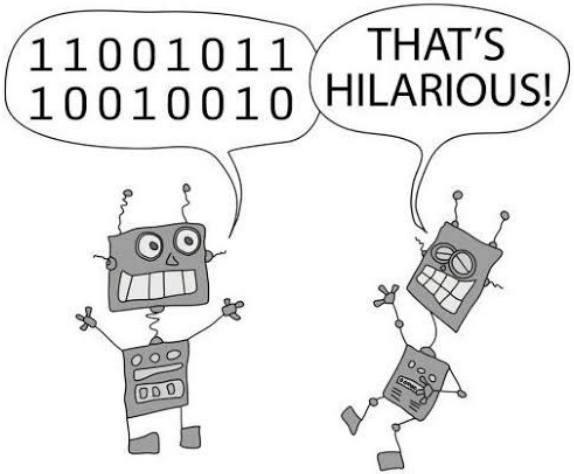
This is CS5045!

GCR:ioc7cdl

Machine Translation



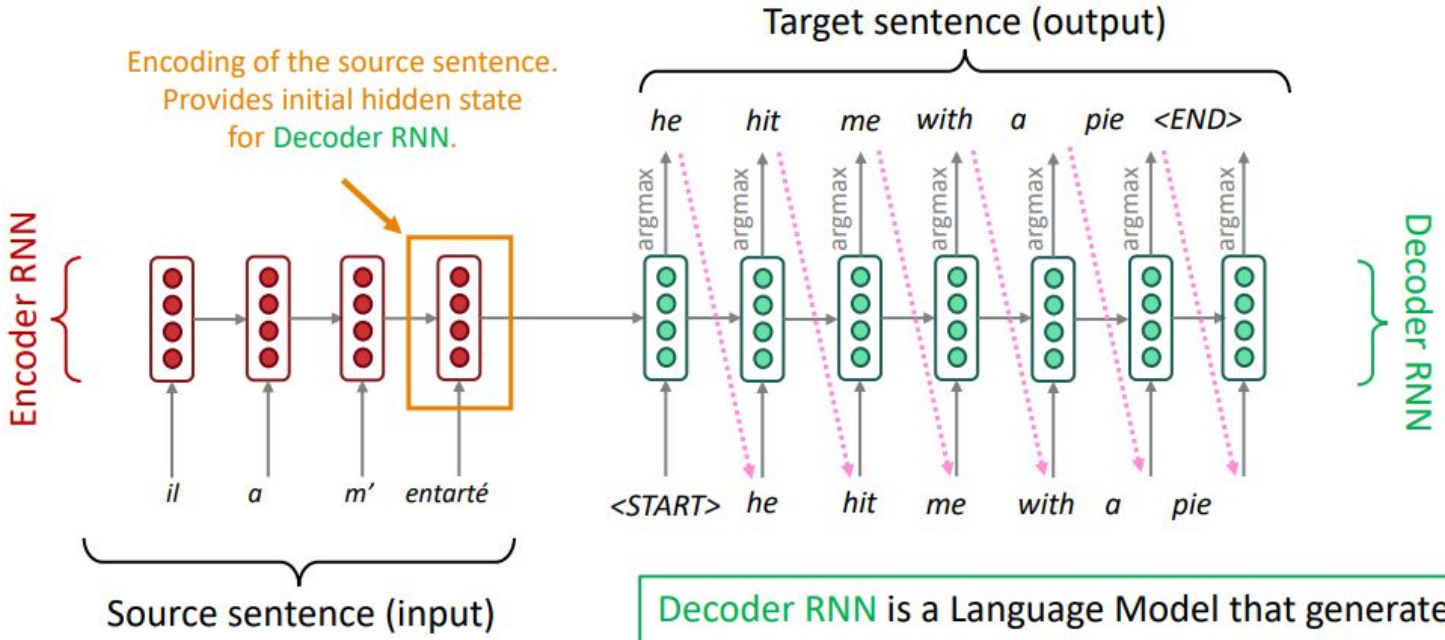
What is Neural Machine Translation?

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

- The neural network architecture is called a sequence-to-sequence model (aka seq2seq) and it involves *two* RNNs

Neural Machine Translation (NMT)

The sequence-to-sequence model



Encoder RNN produces an encoding of the source sentence.

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

Note: This diagram shows **test time** behavior: decoder output is fed in ⋯→ as next step's input

Sequence-to-sequence is versatile!

- The general notion here is an encoder-decoder model
  - One neural network takes input and produces a neural representation
  - Another network produces output based on that neural representation
  - If the input and output are sequences, we call it a seq2seq model

- Sequence-to-sequence is useful for *more than just MT*
- Many NLP tasks can be phrased as sequence-to-sequence:
  - Summarization (long text → short text)
  - Dialogue (previous utterances → next utterance)
  - Parsing (input text → output parse as sequence)
  - Code generation (natural language → Python code)

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*

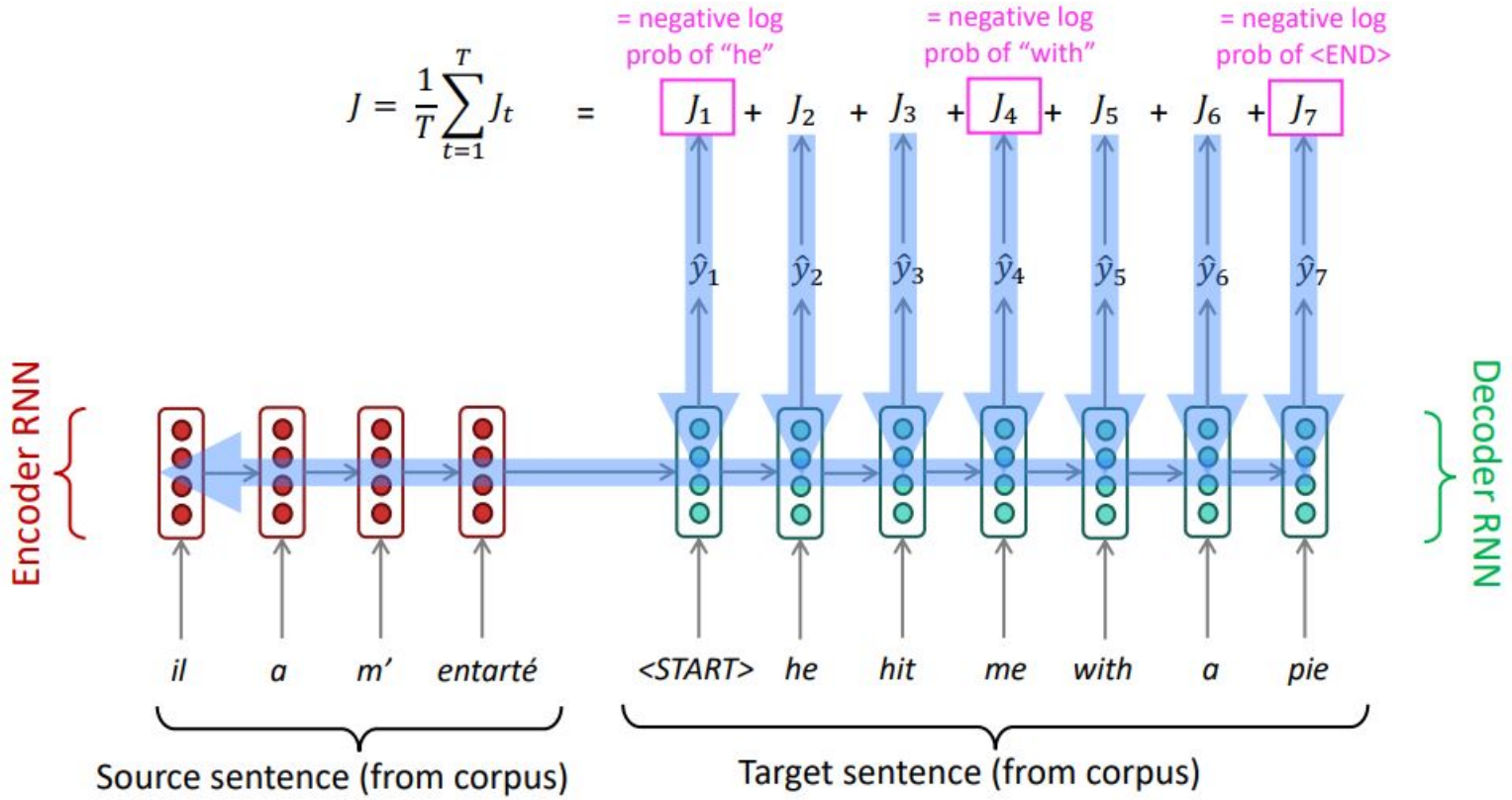
- NMT directly calculates $P(y|x)$:

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

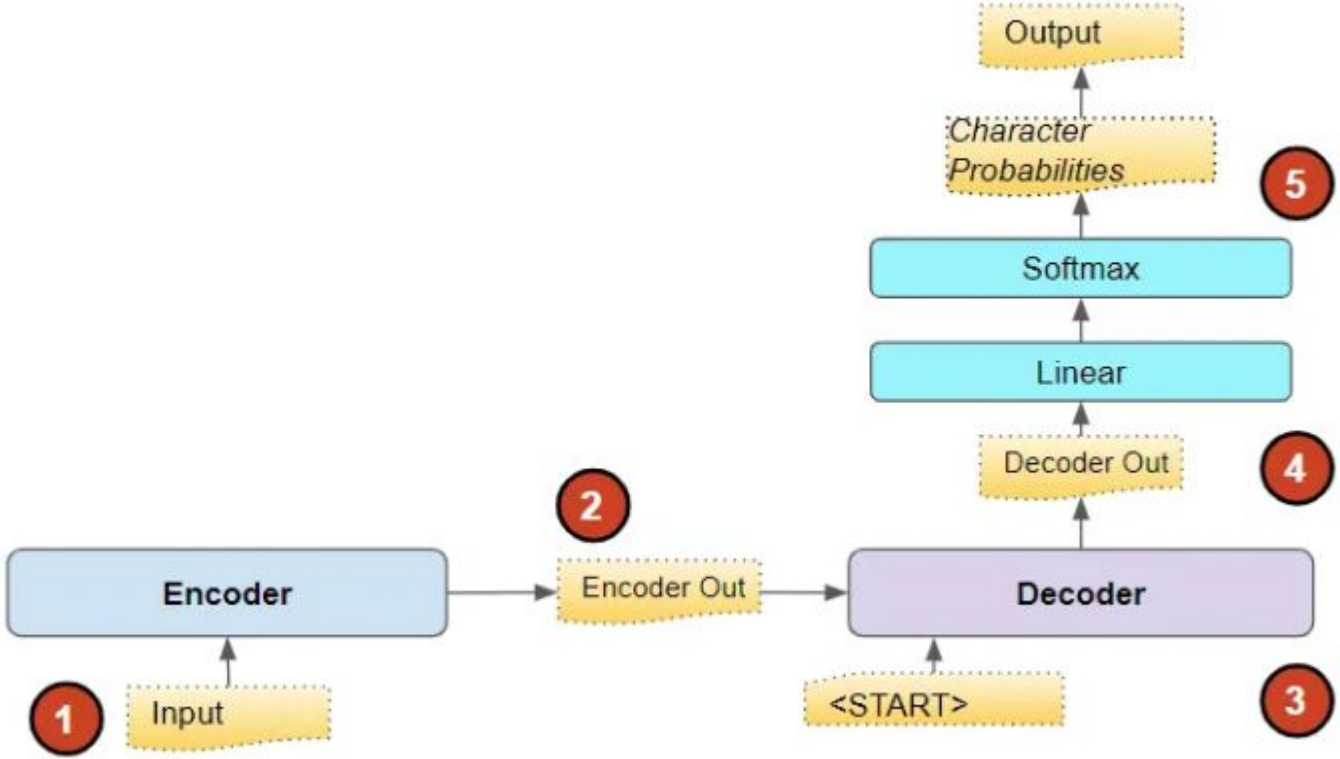
Probability of next target word, given target words so far and source sentence *x*

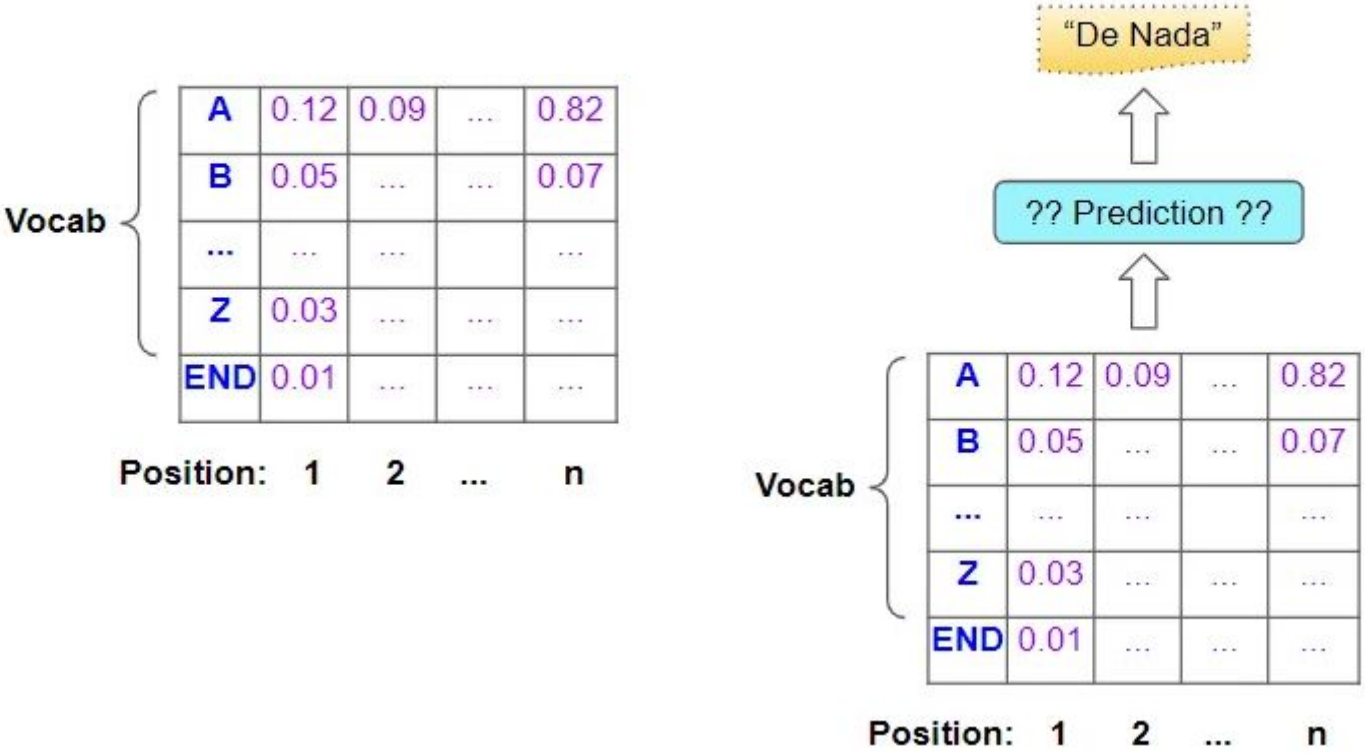
- **Question:** How to train an NMT system?
- **(Easy) Answer:** Get a big parallel corpus…
  - But there is now exciting work on "unsupervised NMT", data augmentation, etc.

Training a Neural Machine Translation system



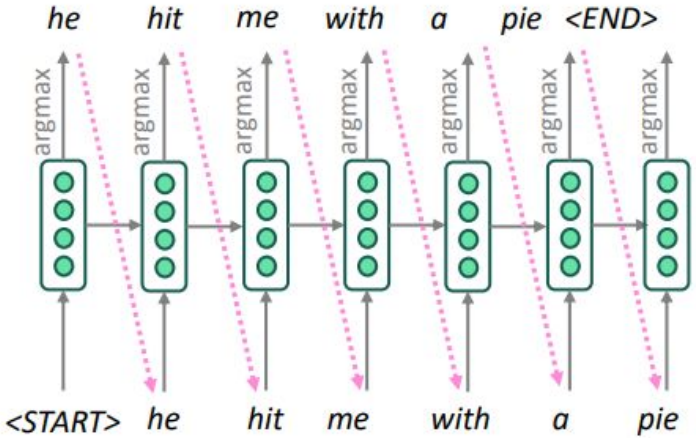
Seq2seq is optimized as a **single system.** Backpropagation operates "*end-to-end*".


Output
Character Probabilities
5
Softmax
Linear
Decoder Out
4
2
Encoder
Encoder Out
Decoder
1
Input
<START>
3


Vocab
A 0.12 0.09 ... 0.82
B 0.05 ... ... 0.07
... ...
Z 0.03 ... ...
END 0.01 ... ...
Position: 1 2 ... n
"De Nada"
?? Prediction ??
Vocab
A 0.12 0.09 ... 0.82
B 0.05 ... ... 0.07
... ...
Z 0.03 ... ...
END 0.01 ... ...
Position: 1 2 ... n

Greedy decoding

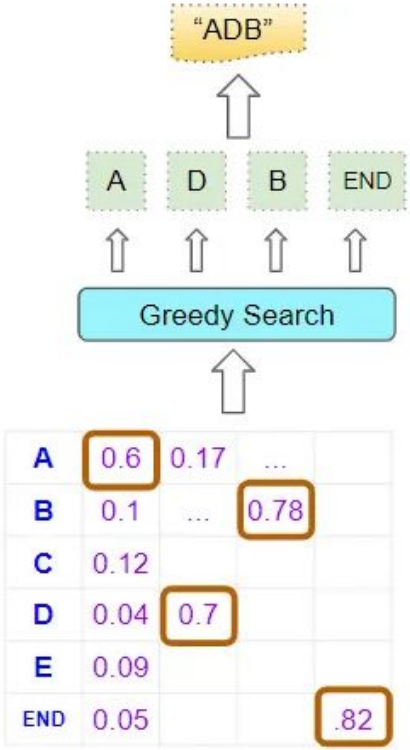
- We saw how to generate (or "decode") the target sentence by taking argmax on each step of the decoder



- This is greedy decoding (take most probable word on each step)
- **Problems with this method?**

Problems with 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…)*



Greedy Search (Image by Author)

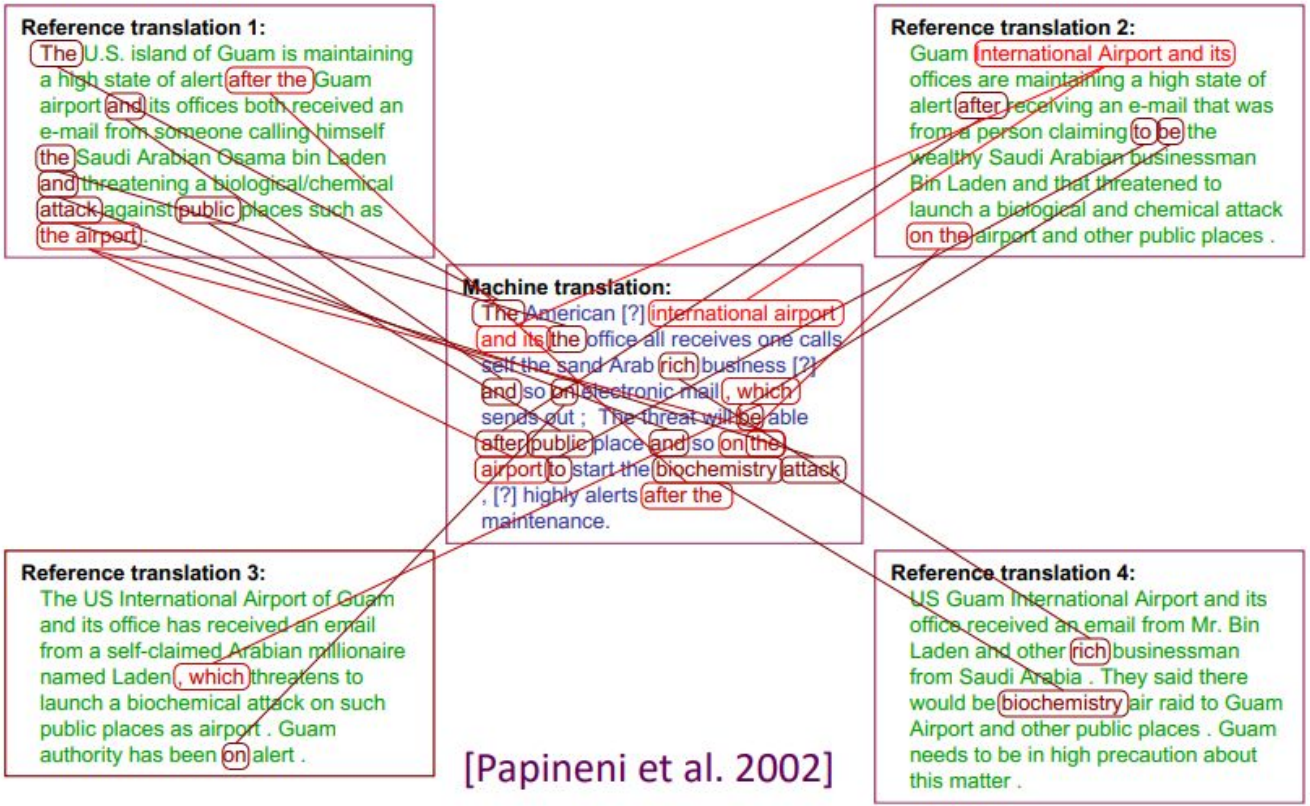
How do we evaluate Machine Translation?

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

- 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 ☹

BLEU score against 4 reference translations



[Papineni et al. 2002]

BLEU Score

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}.$$

Reference Text: It was raining heavily today

Then,

Predicted Text: It It It is raining heavily

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

Unigram:

Clipped Precision count= 3/6=1/2

Bigrams for reference text: ["It was", "was raining", "raining heavily", "heavily today"]

Bigrams for predicted text: ["It It", "It It", "It is", "is raining", "raining heavily"]

Clipped Precision: 1/5

BLEU Score

$$Global\ Average\ Precision = exp(\sum_{n=1}^{N} w_n log p_n) = \prod_{n=1}^{N} {p_n}^{w_n} = (p_1)^{1/2}.(p_2)^{1/2}$$

Usually we use N=4 and $w_n = 1/4$, but in this case we will use N=2 and $w_n = 1/2$

$$Brevity\ Penalty = \begin{cases} 1, & \text{if c>r} \\ e^{(1-r/c)}, & \text{if c<=r} \end{cases}$$

r = number of words in the reference text
c = number of words in the predicted text

Brevity penalty cannot be larger than 1, even if the predicted text is larger than the reference text.
r = 5 c = 6
Bleu = 0.316

Implementation for Bleu Score with Python

```
from nltk.translate.bleu_score import sentence_bleu

reference = ['It was raining heavily today'.split()]

candidate = 'It It It is raining heavily'.split()

print(sentence_bleu(reference, candidate, weights=(0.5, 0.5, 0, 0)))
```

https://www.scaler.com/topics/nlp/bleu-score-in-nlp/

NMT: perhaps the biggest success story of NLP Deep Learning?

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

- **2014**: First seq2seq paper published
- **2016**: Google Translate switches from SMT to NMT – and by 2018 everyone has

- This is amazing!
  - **SMT** systems, built by hundreds of engineers over many years, outperformed by NMT systems trained by a small group of engineers in a few months

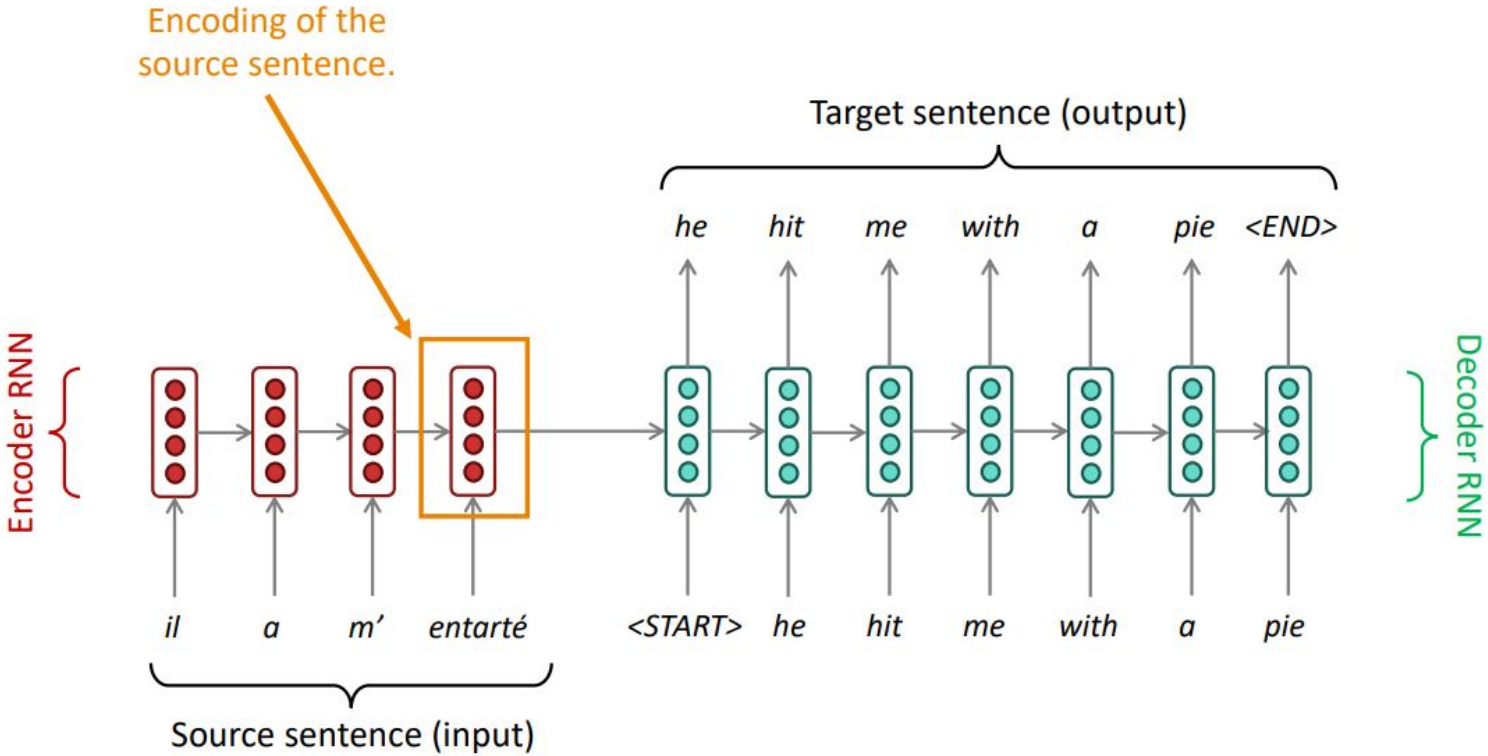
NMT research continues

NMT is a **flagship task** for NLP Deep Learning

- NMT research has pioneered many of the recent innovations of NLP Deep Learning
- NMT research continues to thrive
  - Researchers have found *many, many* improvements to the "vanilla" seq2seq NMT system we've just presented
  - But we'll present next one improvement so integral that it is the new vanilla…

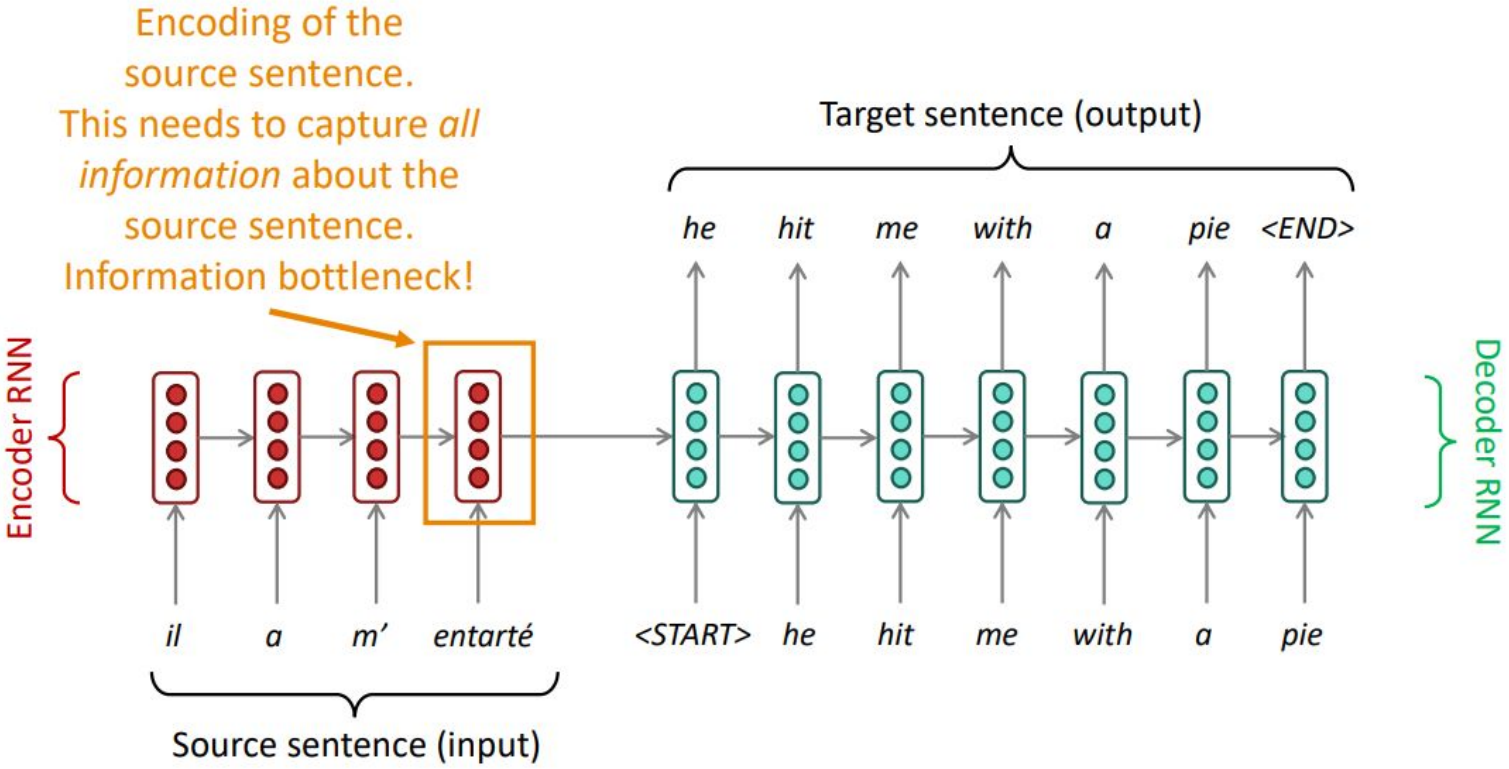
ATTENTION

1. Why attention? Sequence-to-sequence: the bottleneck problem



Problems with this architecture?

1. Why attention? Sequence-to-sequence: the bottleneck problem



Attention

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

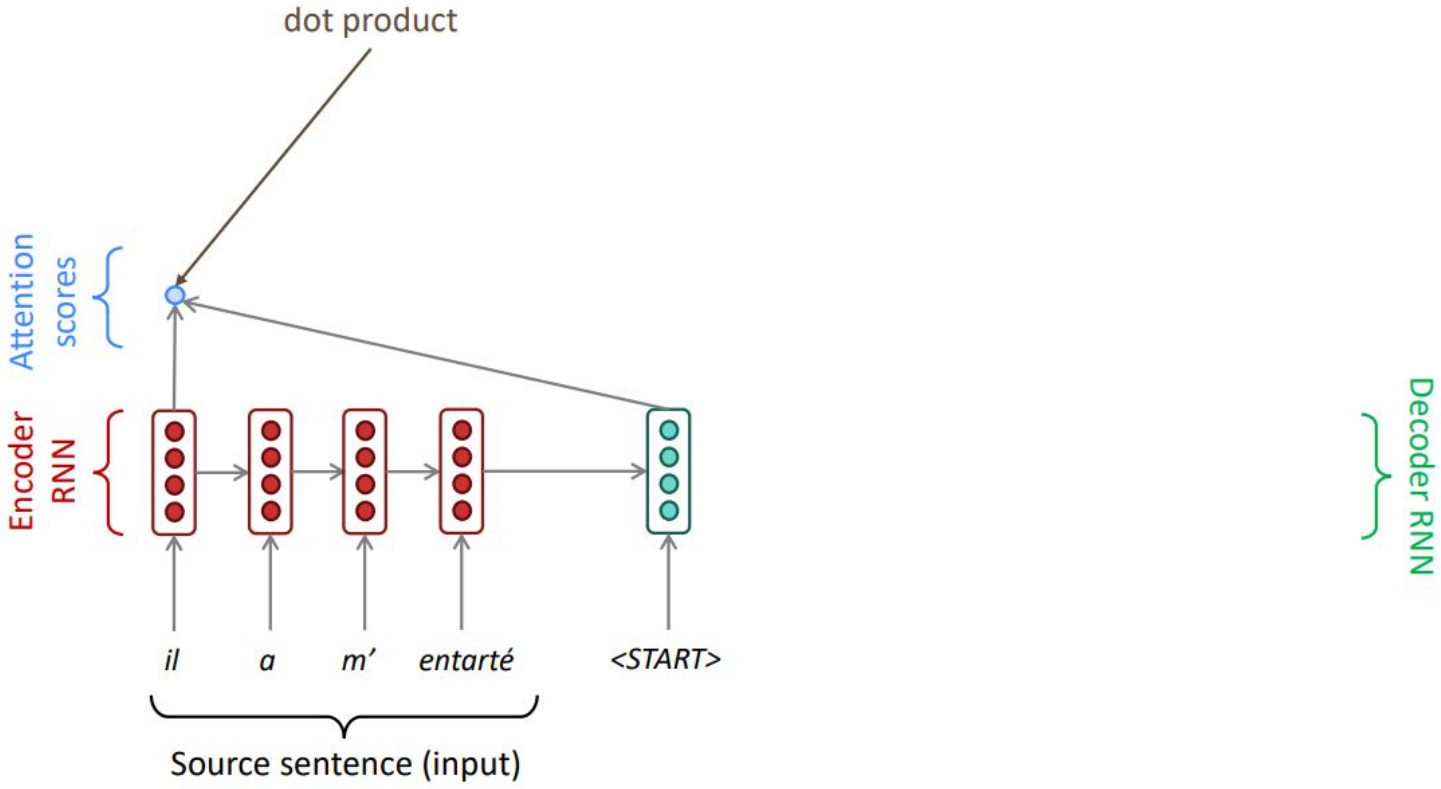
- Core idea: on each step of the decoder, use *direct connection to the encoder* to *focus on a particular part* of the source sequence

- First, we will show via diagram (no equations), then we will show with equations

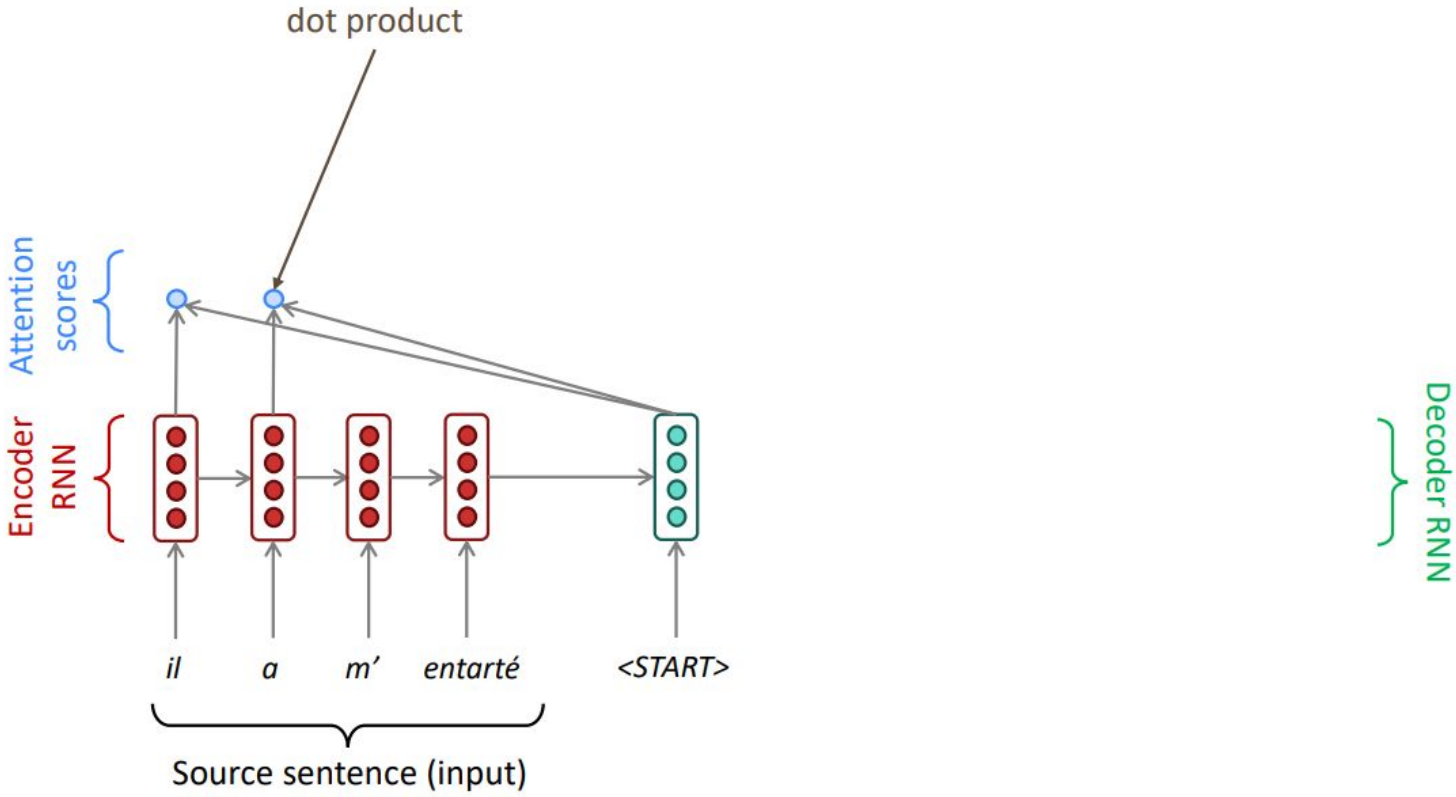
Attention:

At different steps, let a model "focus" on different parts of the input.

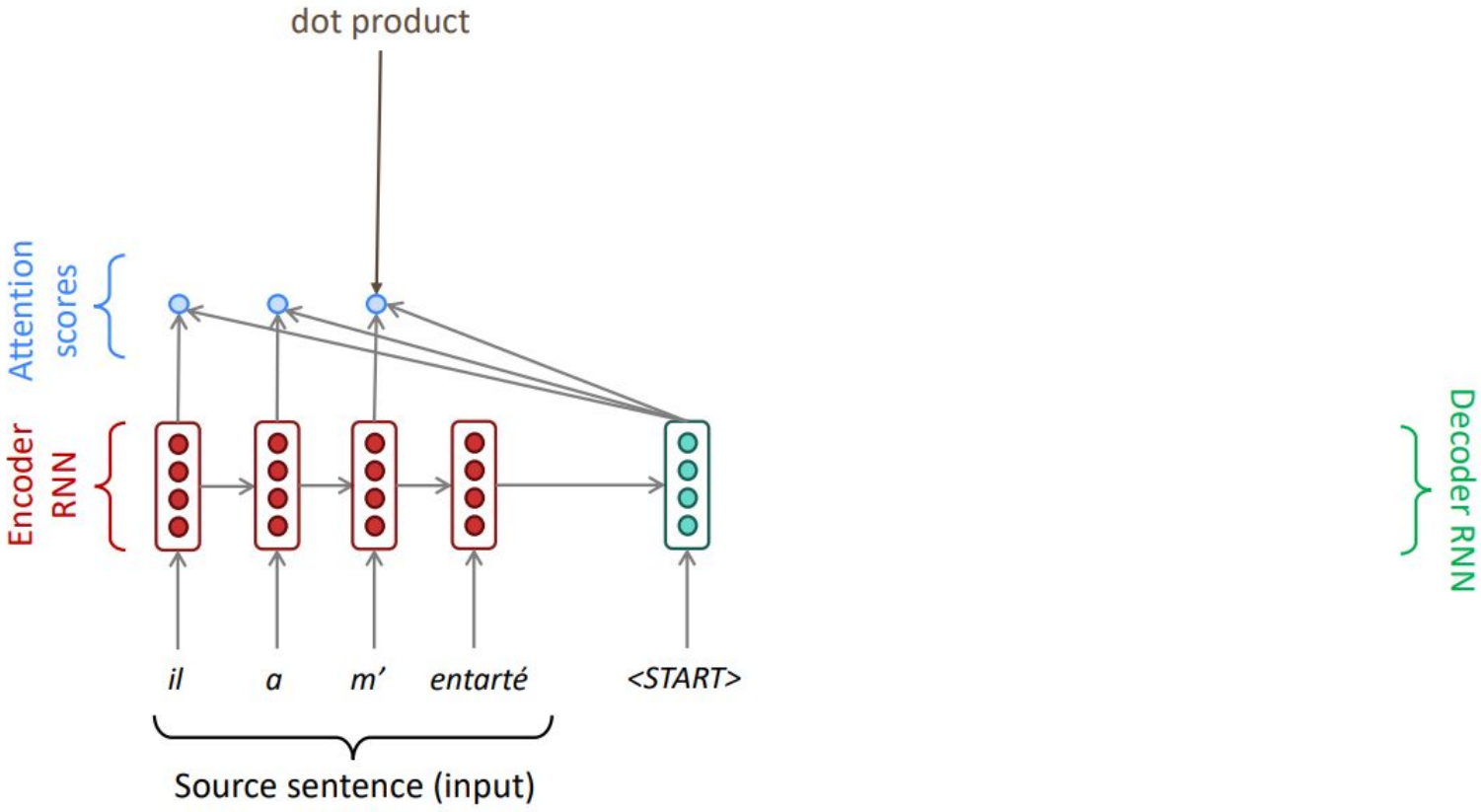
Sequence-to-sequence with attention



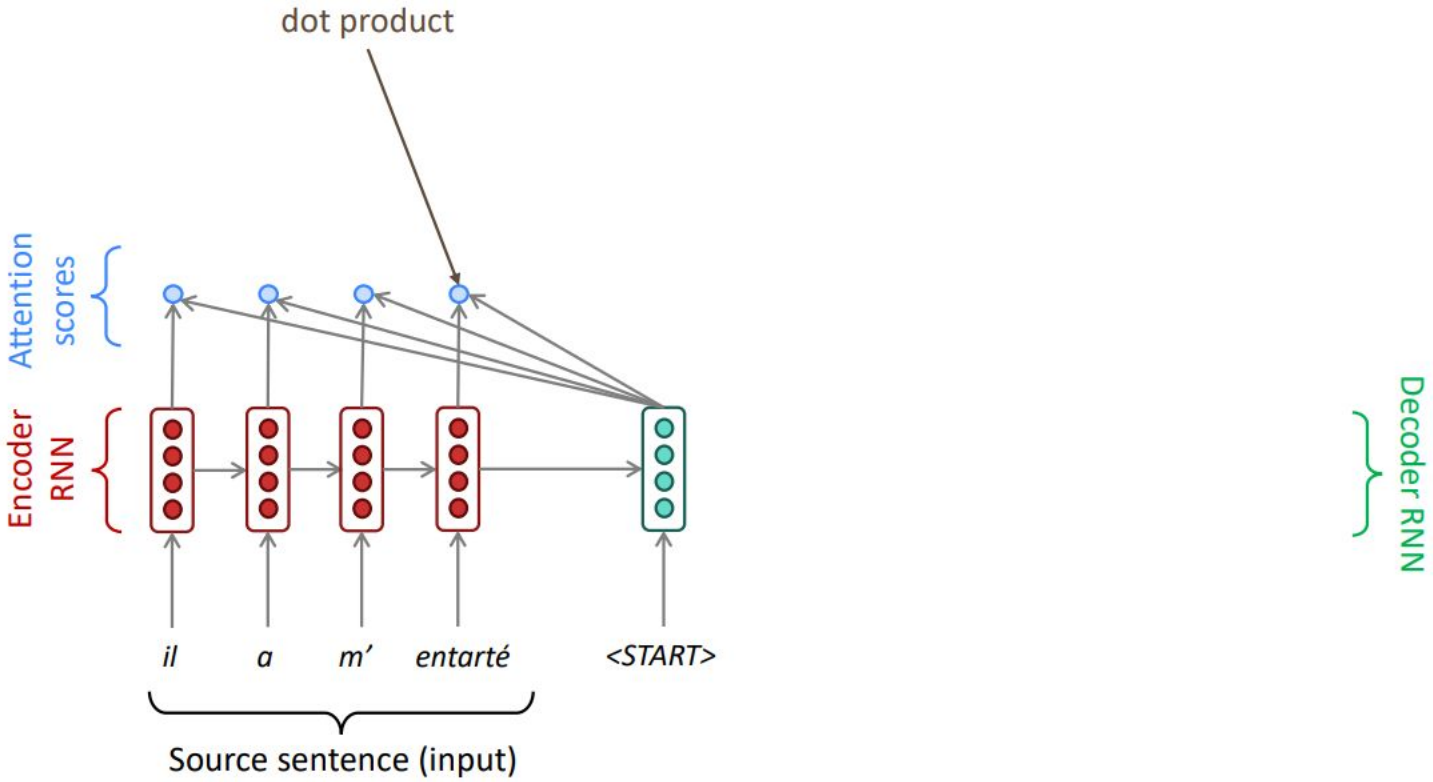
Sequence-to-sequence with attention



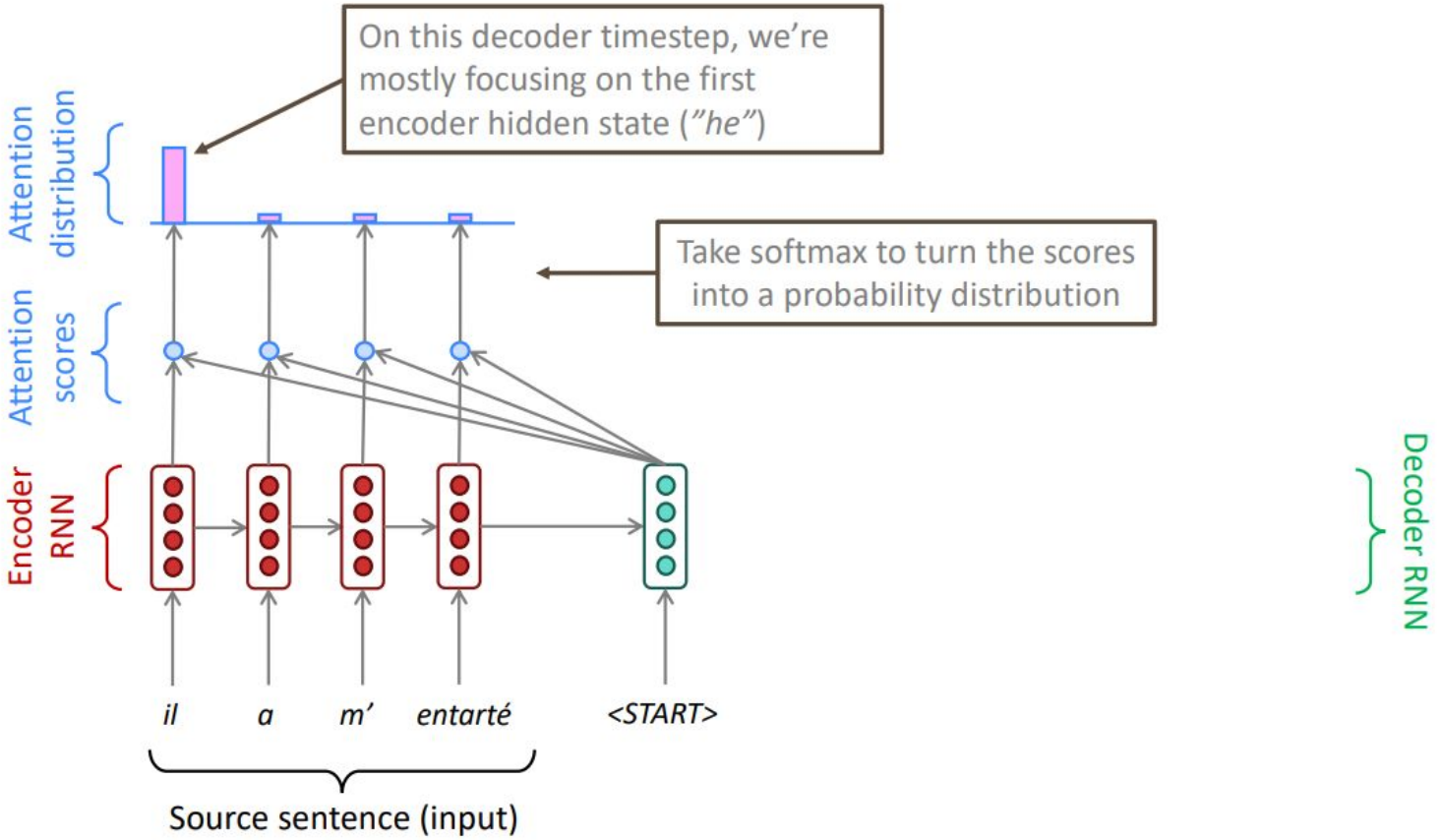
Sequence-to-sequence with attention



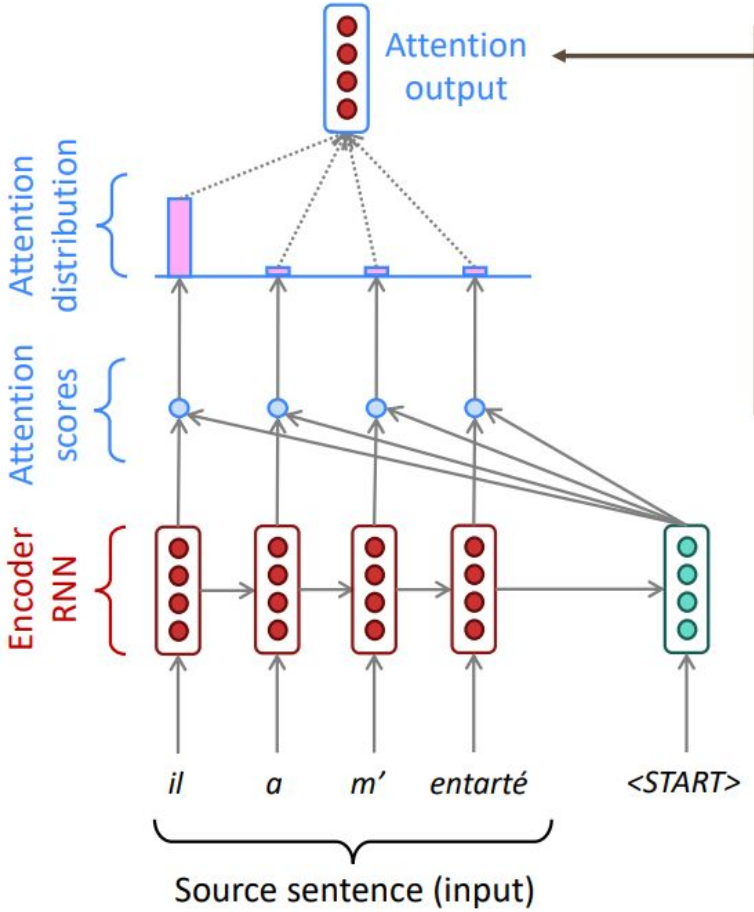
Sequence-to-sequence with attention



Sequence-to-sequence with attention



Sequence-to-sequence with attention

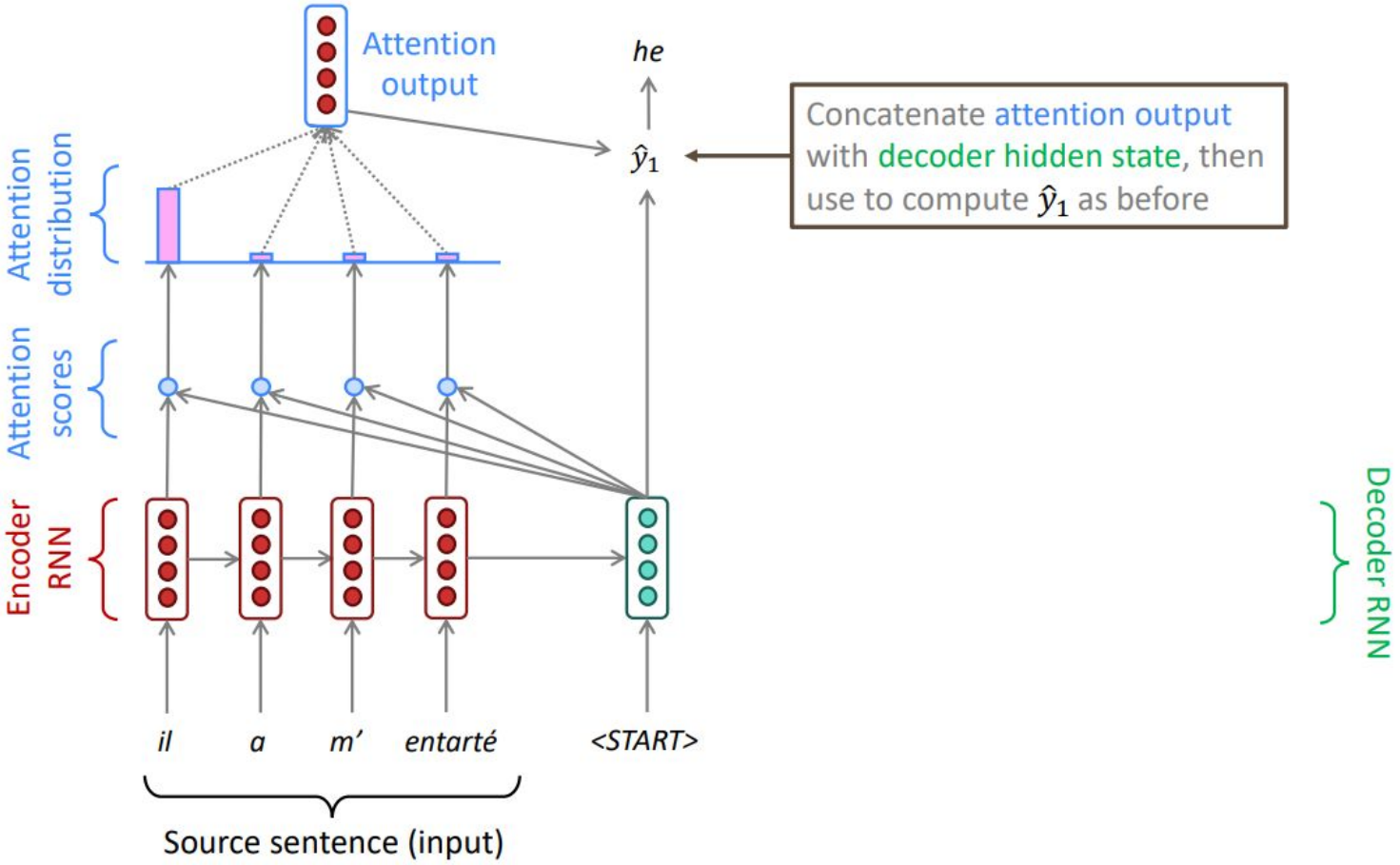


Use the attention distribution to take a **weighted sum** of the encoder hidden states.

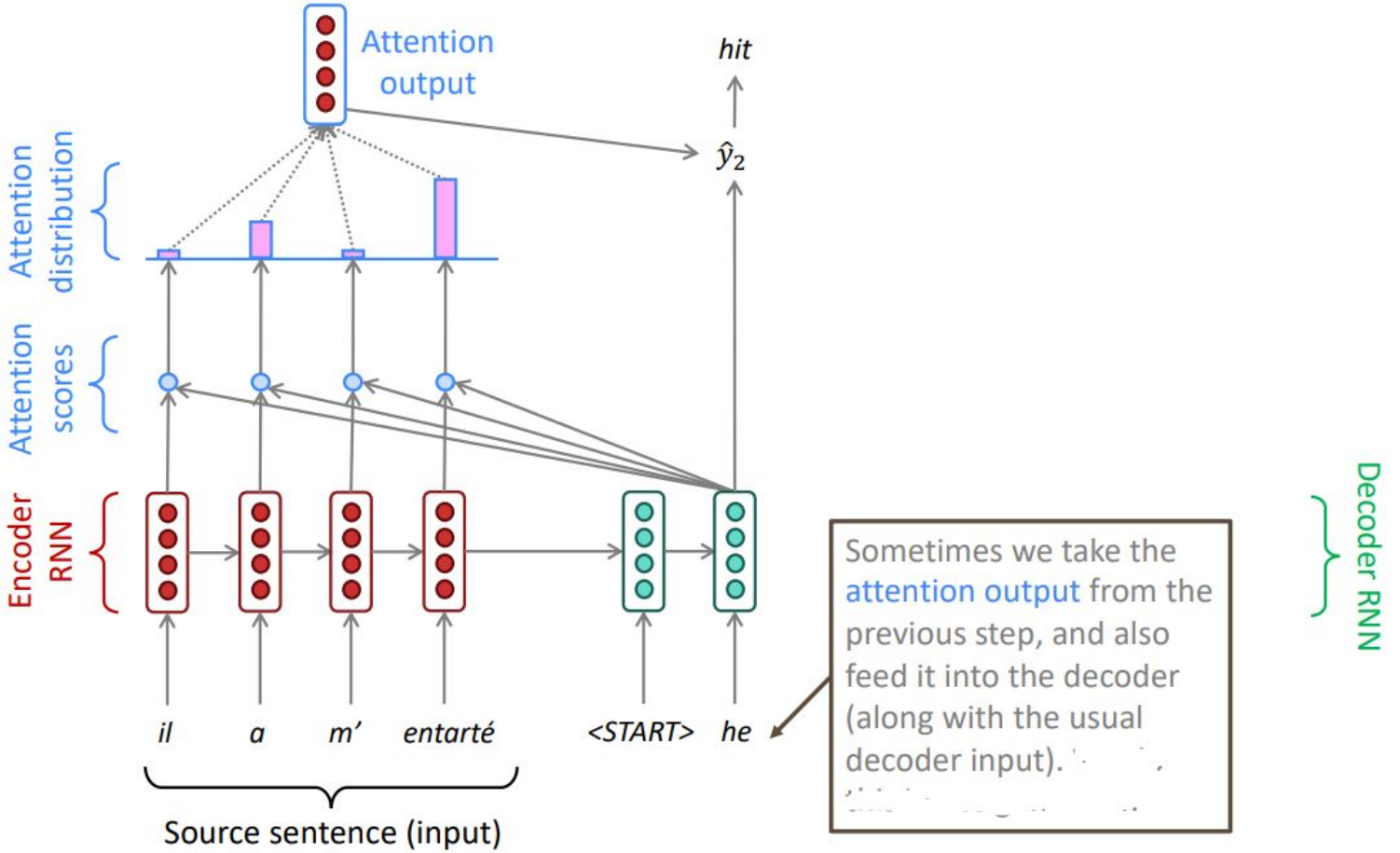
The attention output mostly contains information from the hidden states that received high attention.

Decoder RNN

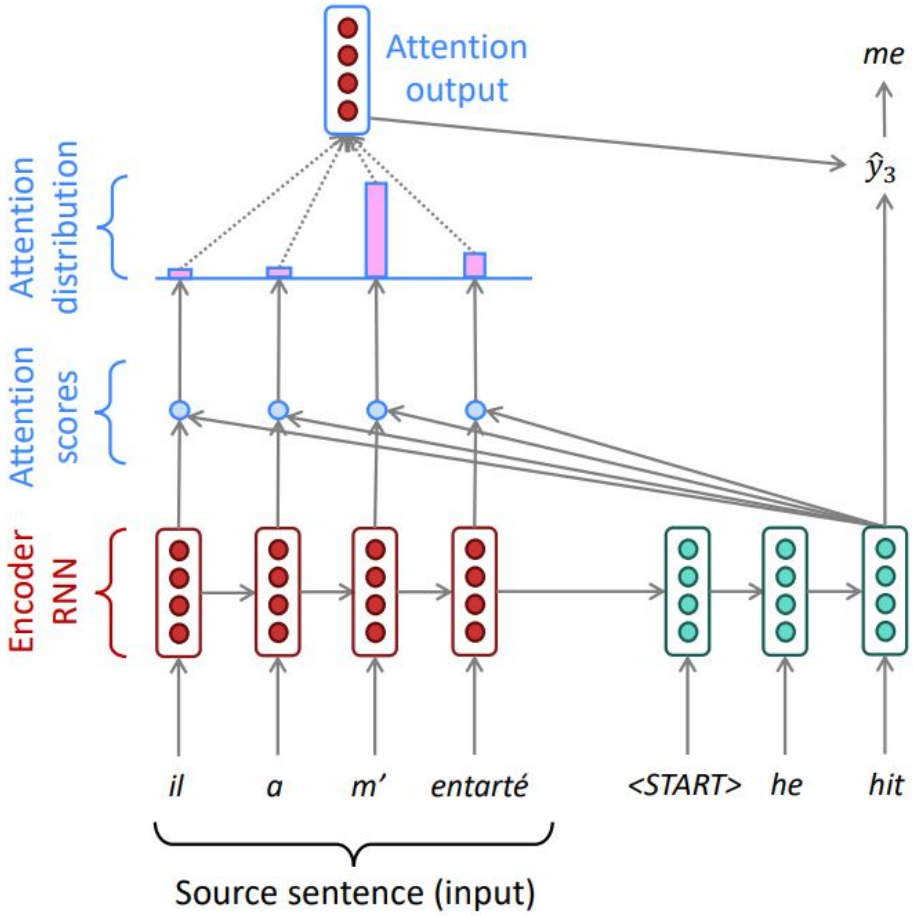
Sequence-to-sequence with attention



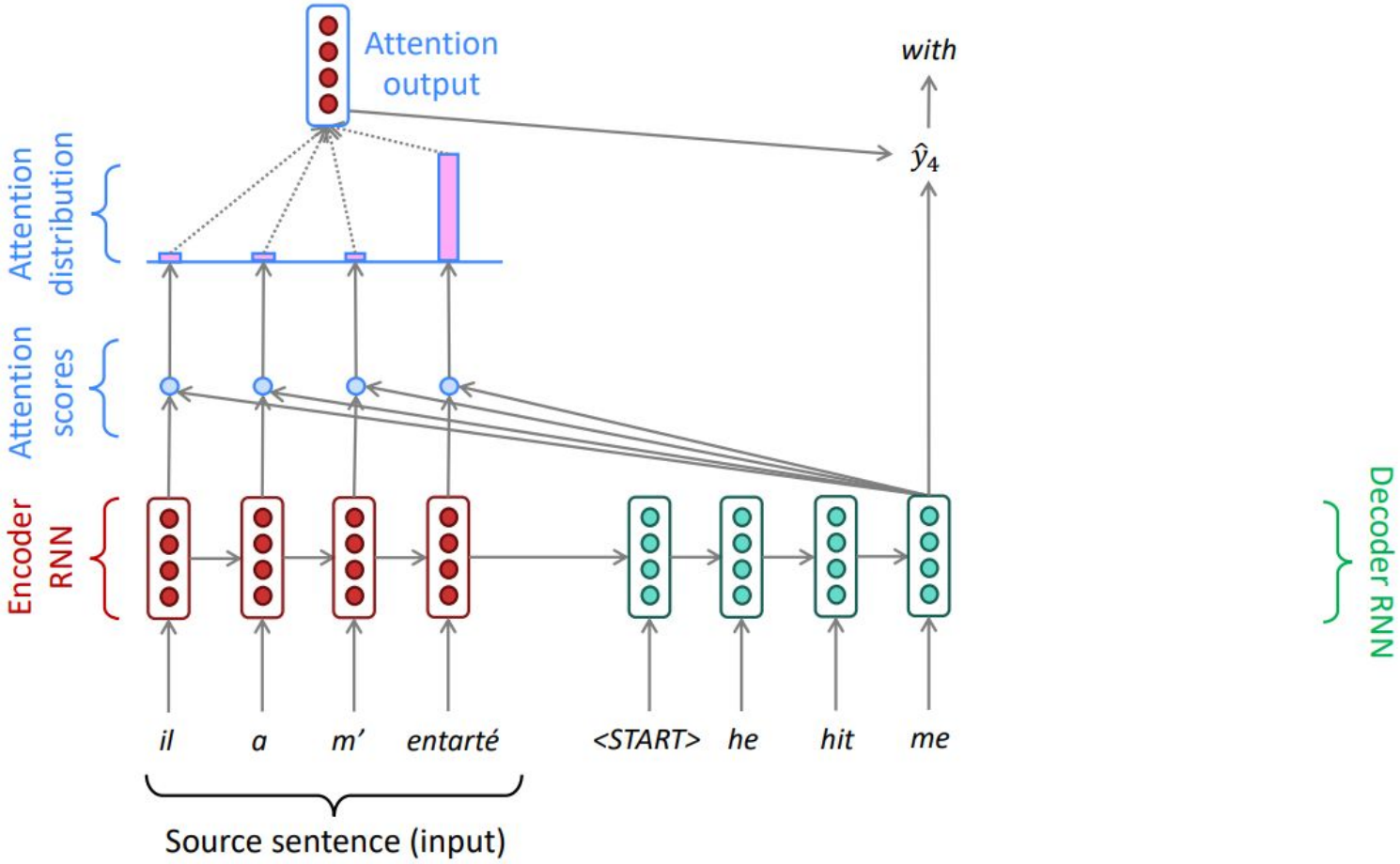
Sequence-to-sequence with attention



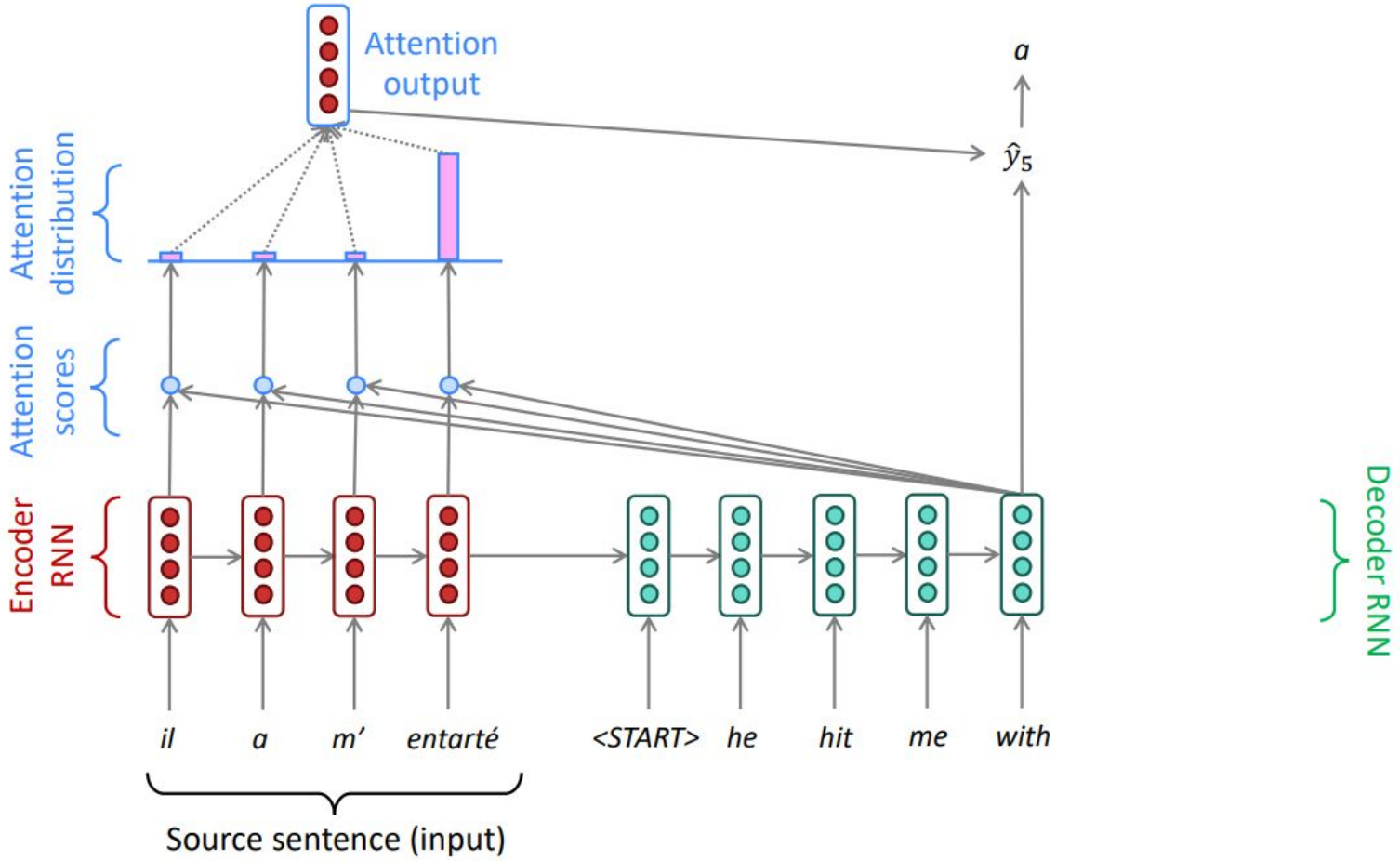
Sequence-to-sequence with attention



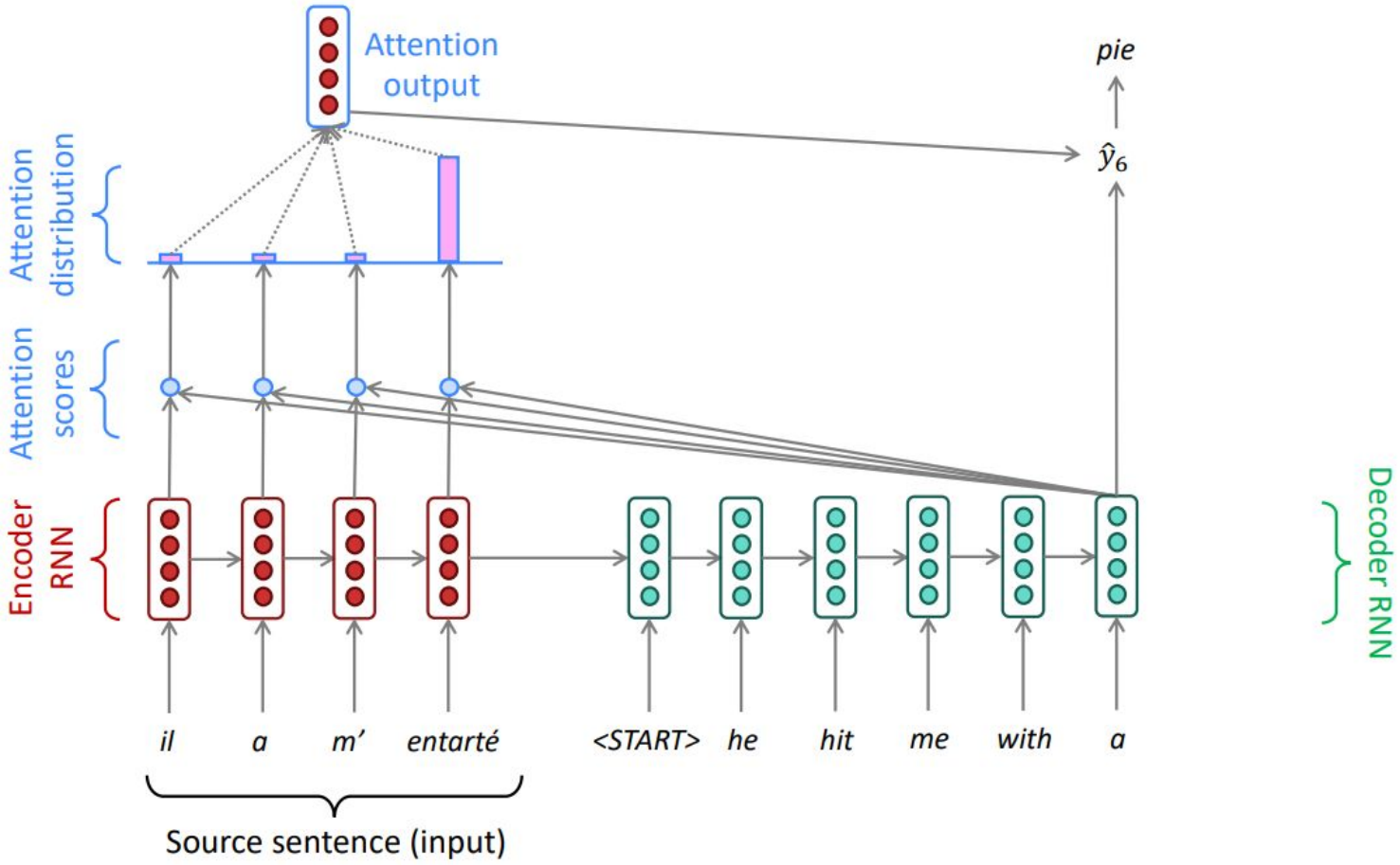
Sequence-to-sequence with attention



Sequence-to-sequence with attention

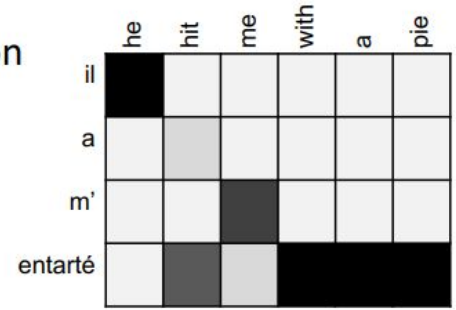


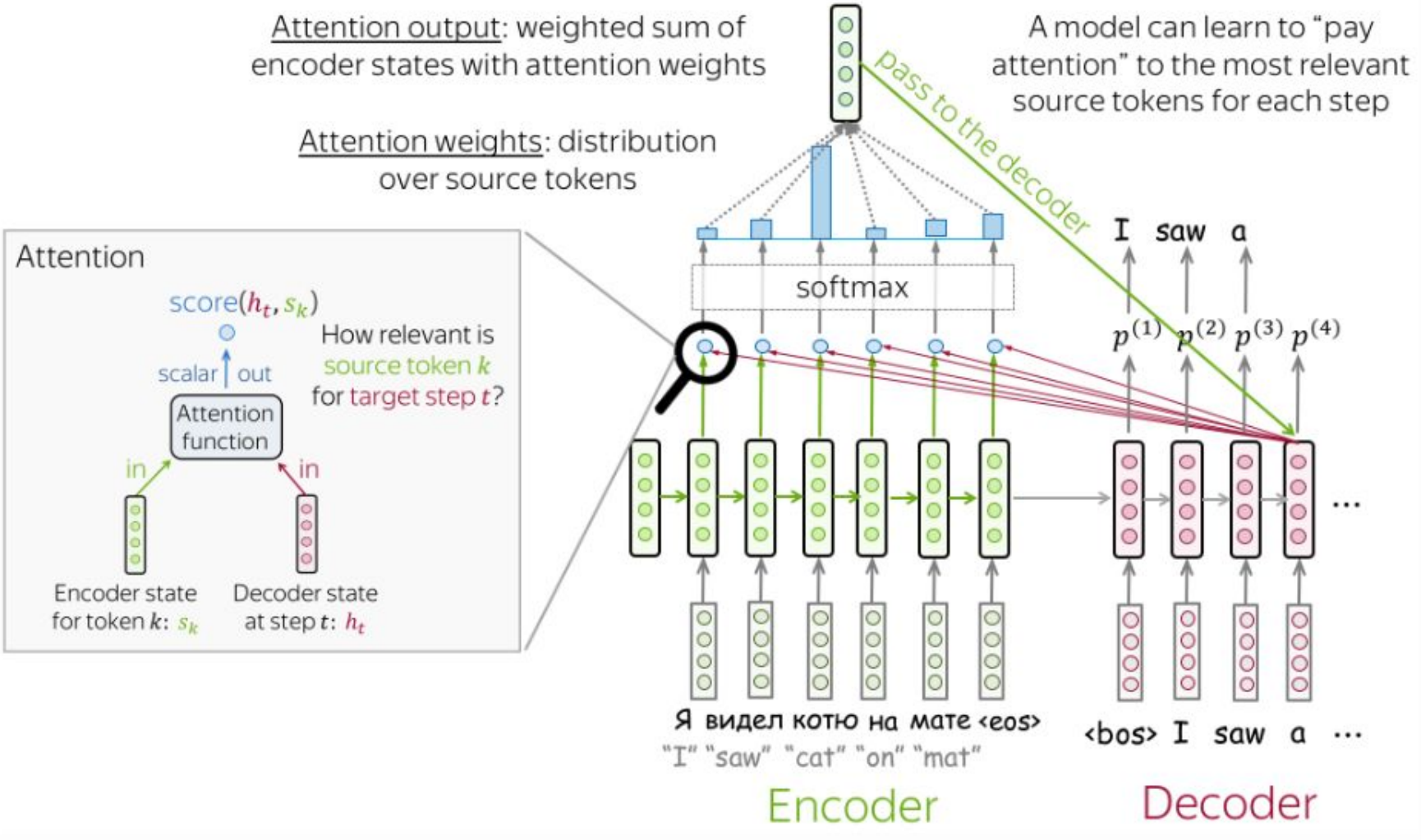
Sequence-to-sequence with attention



Attention is great!

- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention provides more "human-like" model of the MT process
  - You can look back at the source sentence while translating, rather than needing to remember it all
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with the vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we see what the decoder was focusing on
  - We get (soft) alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself




Attention output: weighted sum of encoder states with attention weights
Attention weights: distribution over source tokens
A model can learn to "pay attention" to the most relevant source tokens for each step
pass to the decoder
softmax
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$
Attention
score(h_t, s_k)
scalar out
Attention function
in
in
How relevant is source token k for target step t?
Encoder state for token k: s_k
Decoder state at step t: h_t
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
Encoder
Decoder

Attention output

$$c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$$

"source context for decoder step t"

↑ (weighted sum)

Attention weights

$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \text{k} = 1..\text{m}$$

"attention weight for source token k at decoder step t"

↑ (softmax)

Attention scores

$$\text{score}(h_t, s_k), \text{k} = 1..\text{m}$$

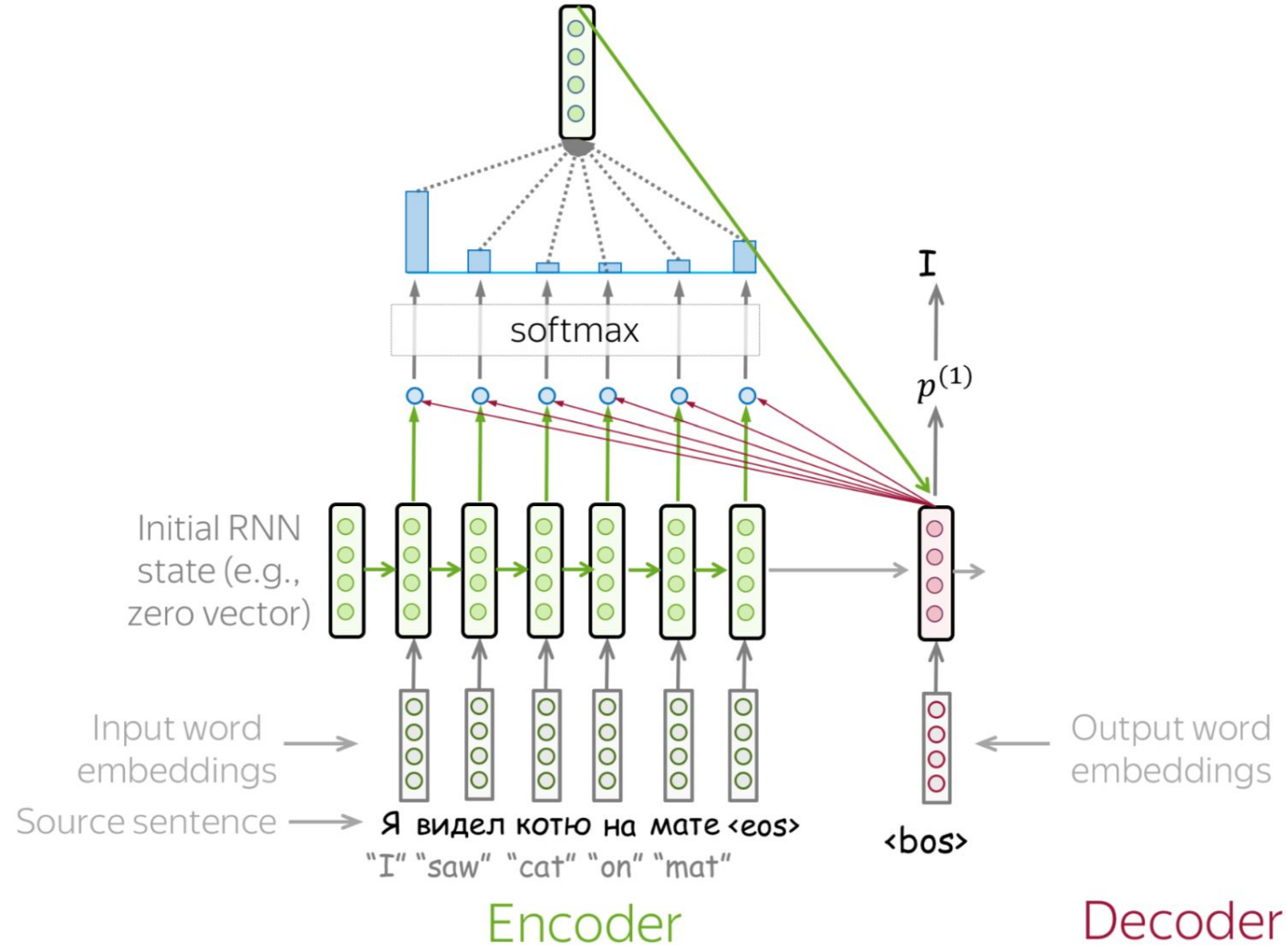
"How relevant is source token k for target step t?"

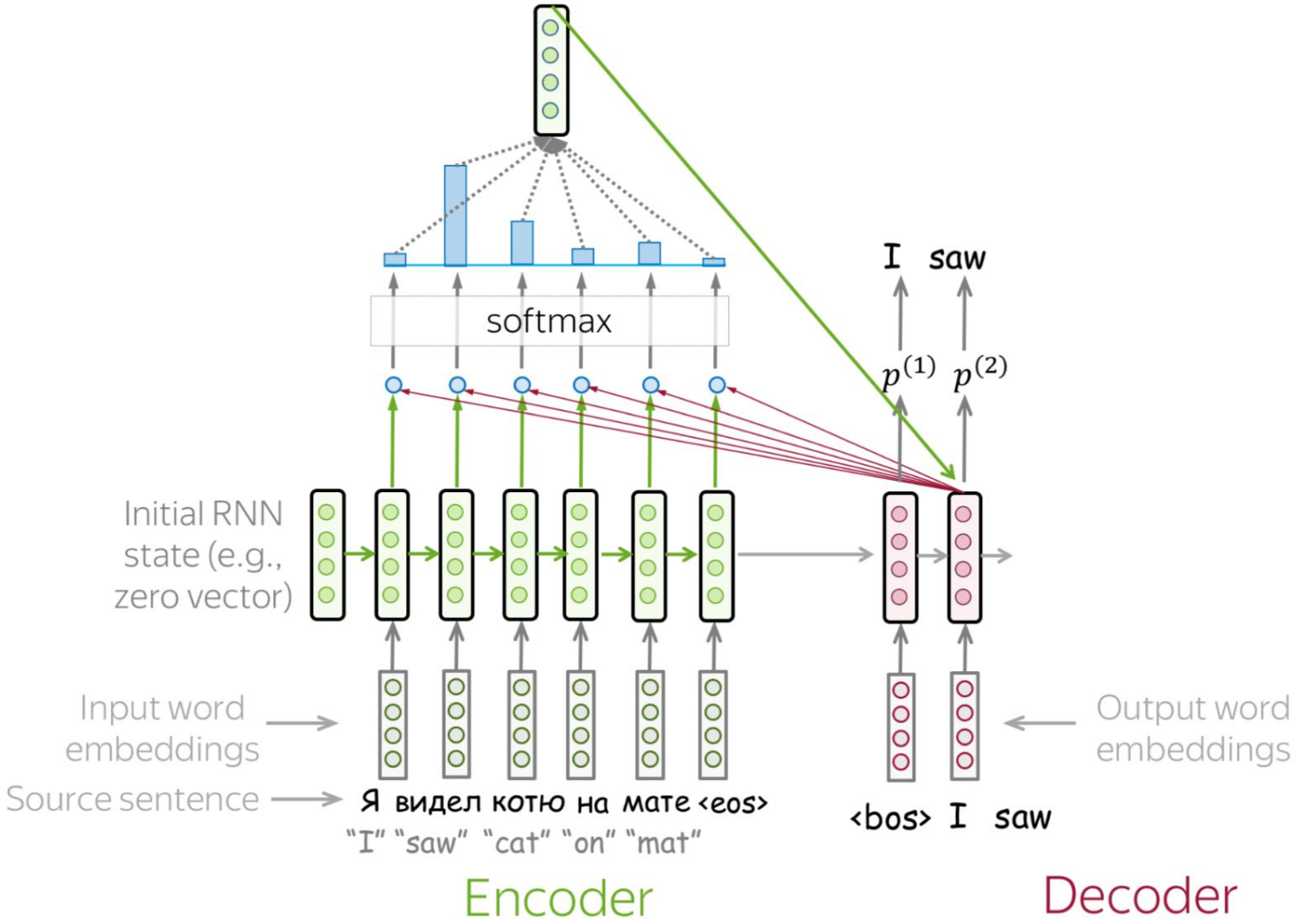
↑

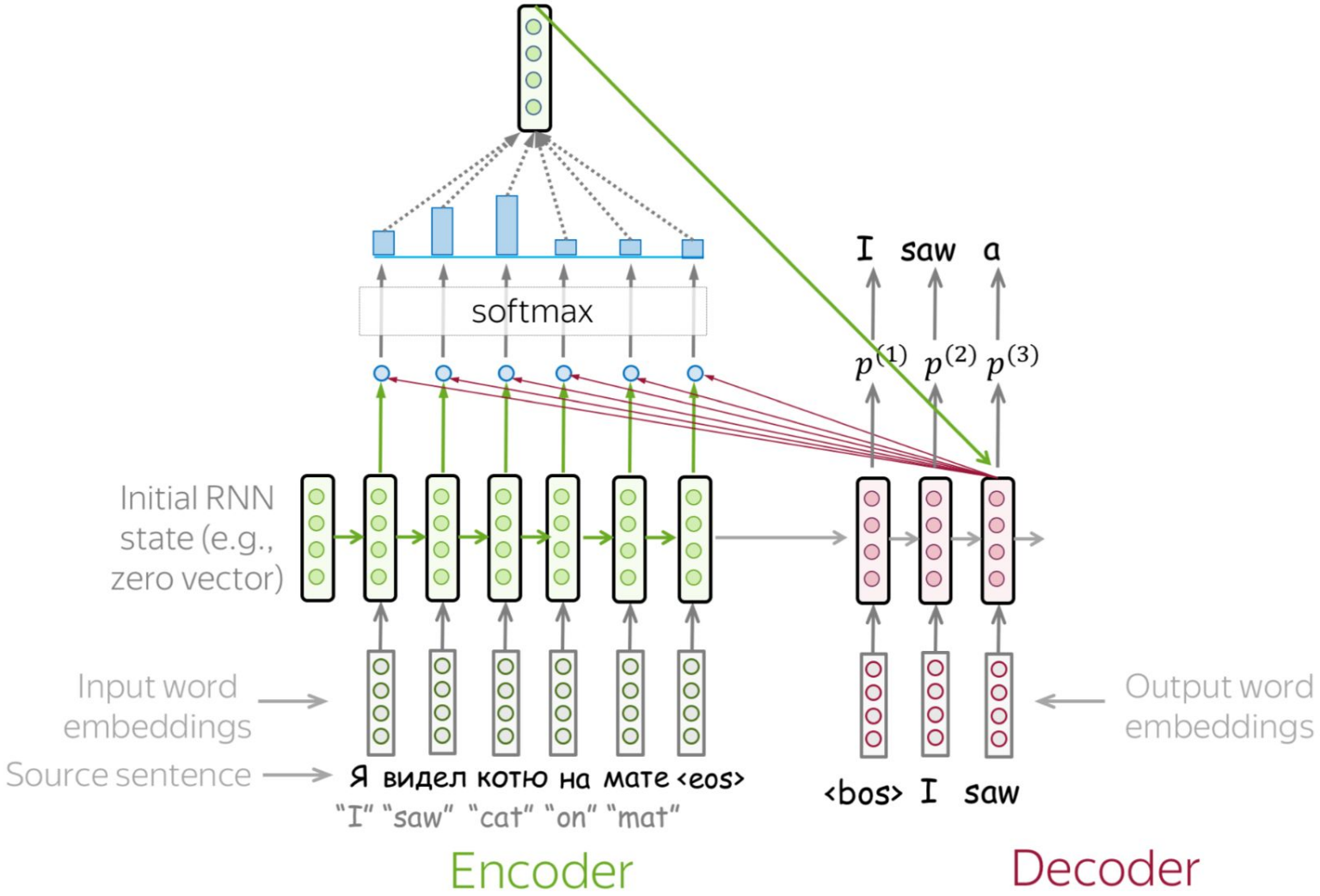
Attention input

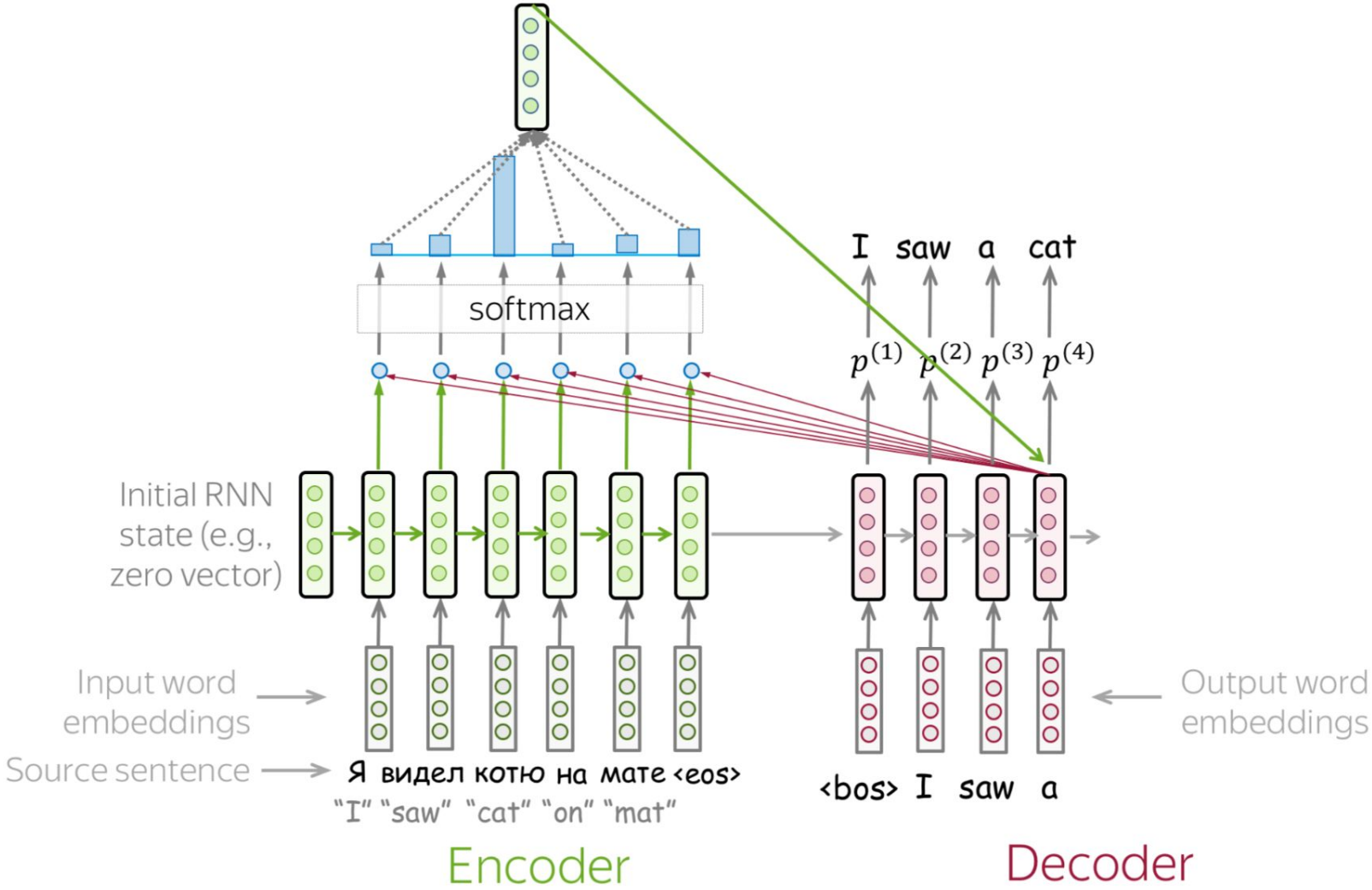
$s_1, s_2, \ldots, s_m$ all encoder states

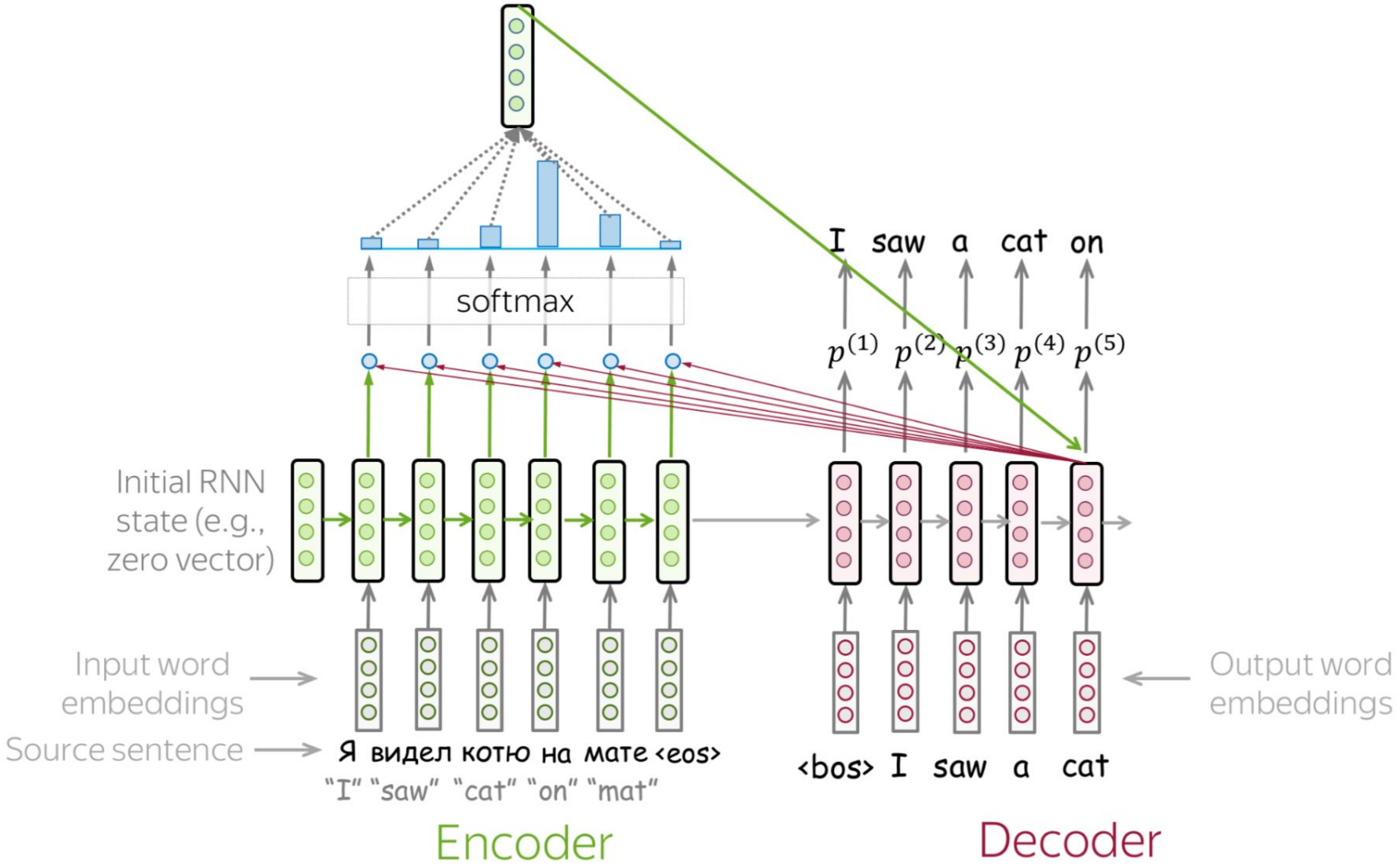
h_t one decoder state

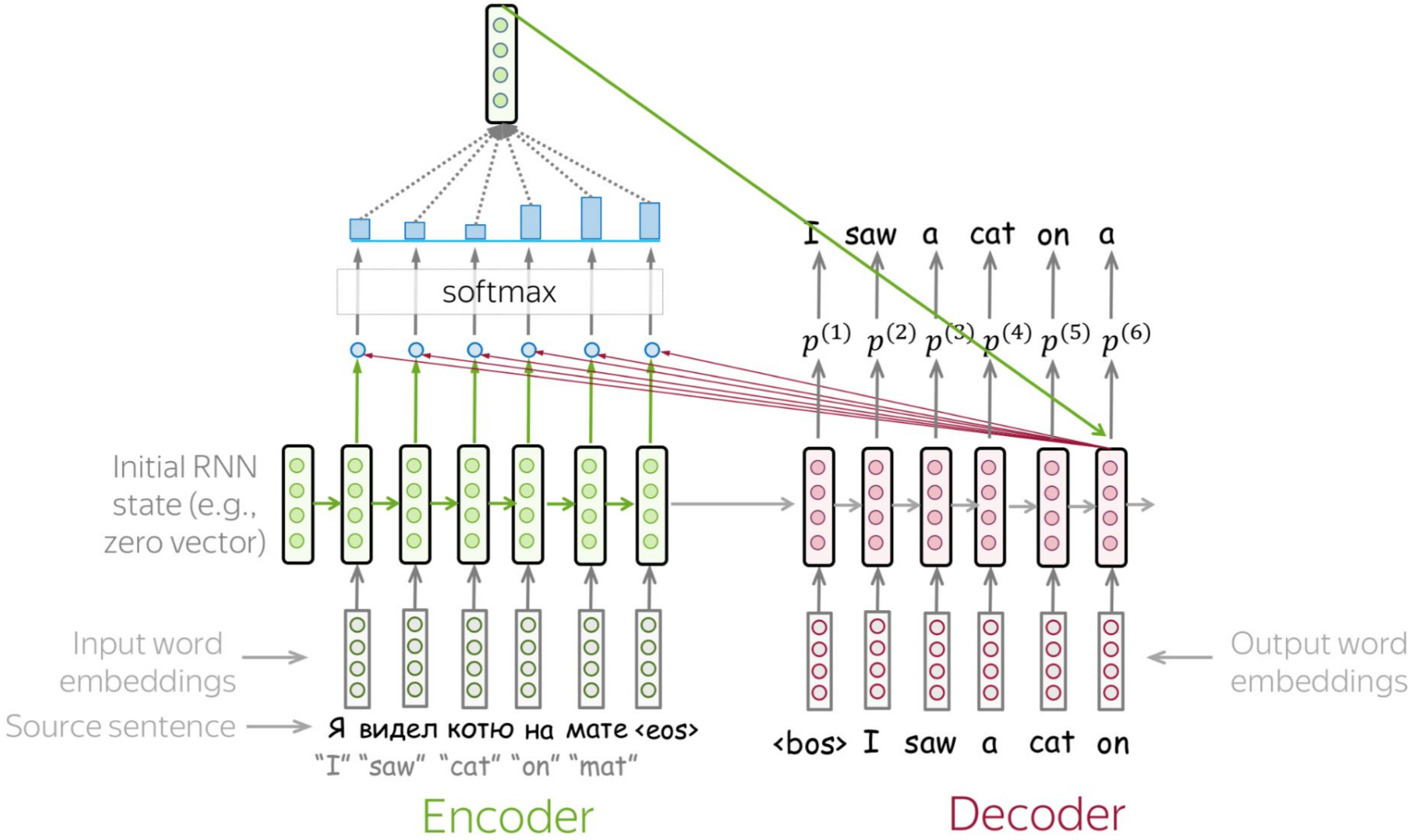

softmax
I
$p^{(1)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Output word embeddings
<bos>
Encoder
Decoder

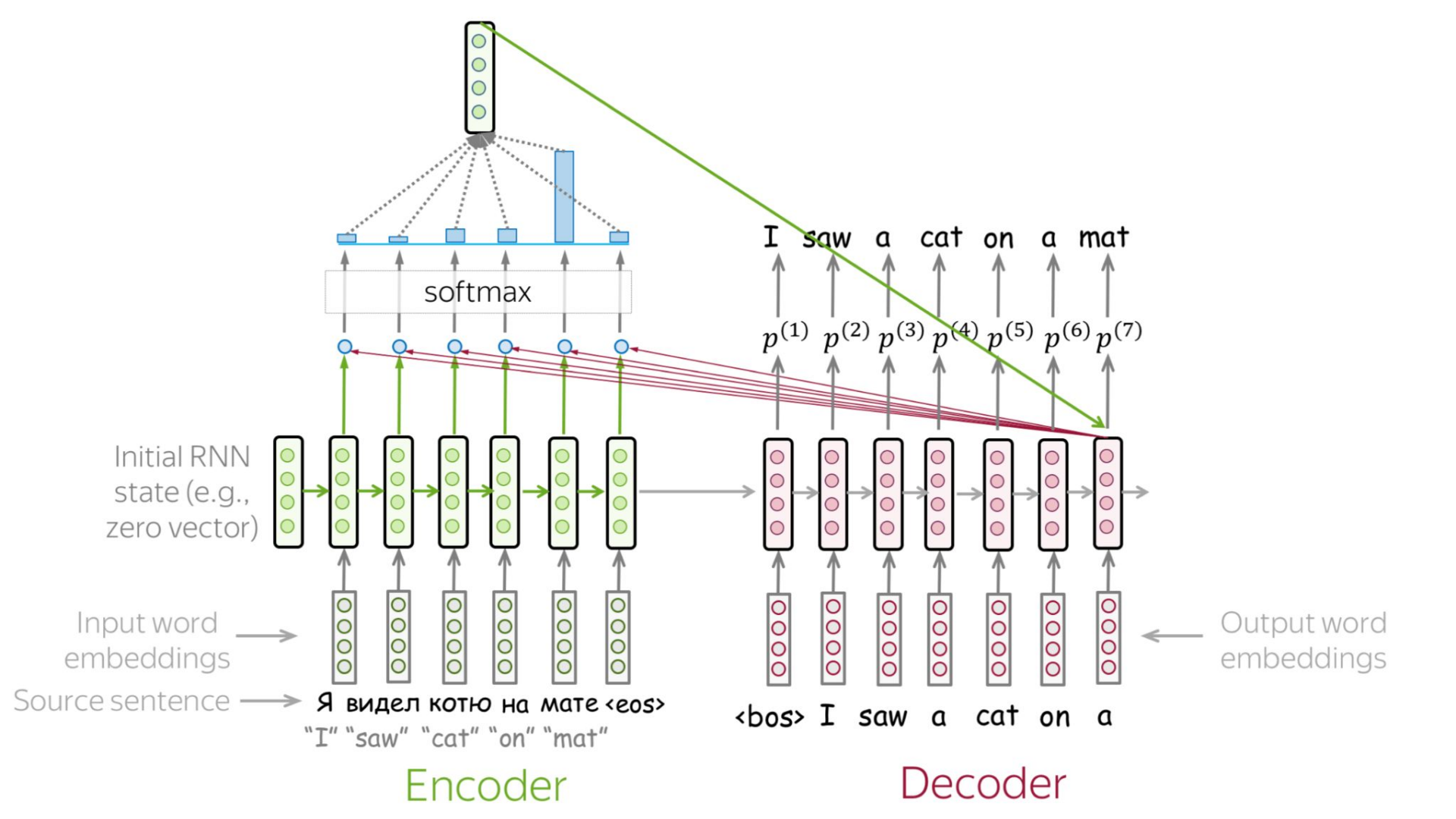

I saw
softmax
$p^{(1)}$ $p^{(2)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Output word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw
Encoder
Decoder

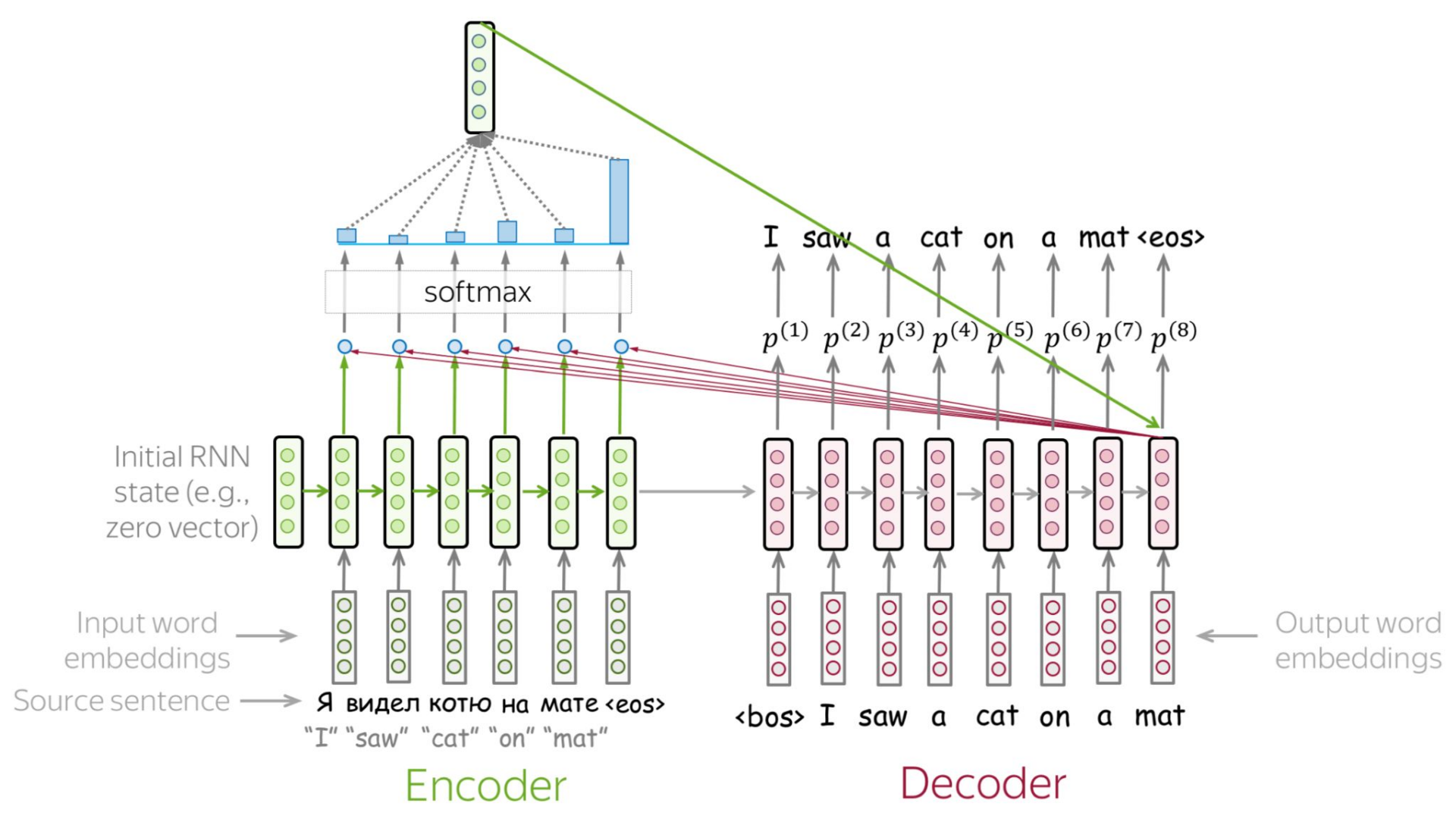

softmax
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Output word embeddings
<bos> I saw
Encoder
Decoder


softmax
I saw a cat
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Output word embeddings
<bos> I saw a
Encoder
Decoder

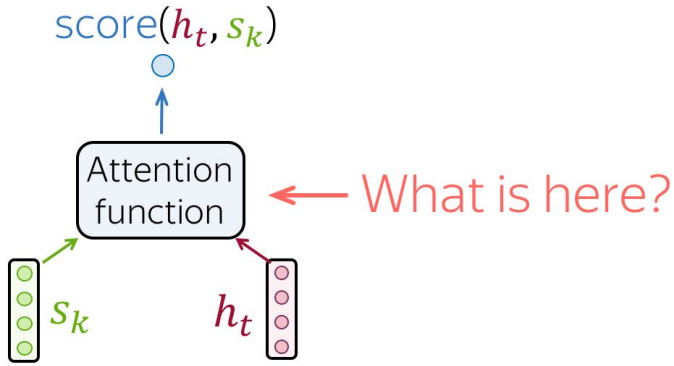

softmax
I saw a cat on
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$ $p^{(5)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Output word embeddings
<bos> I saw a cat
Encoder
Decoder


softmax
I saw a cat on a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$ $p^{(5)}$ $p^{(6)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Output word embeddings
<bos> I saw a cat on
Encoder
Decoder

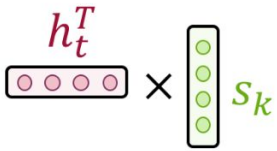

softmax
I saw a cat on a mat
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$ $p^{(5)}$ $p^{(6)}$ $p^{(7)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a cat on a
Output word embeddings
Encoder
Decoder


softmax
I saw a cat on a mat <eos>
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$ $p^{(5)}$ $p^{(6)}$ $p^{(7)}$ $p^{(8)}$
Initial RNN state (e.g., zero vector)
Input word embeddings
Source sentence
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Output word embeddings
<bos> I saw a cat on a mat
Encoder
Decoder

How to Compute Attention Score?

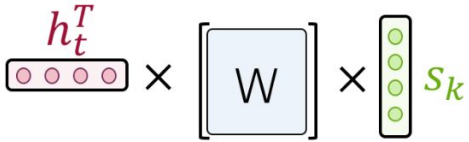


Dot-product



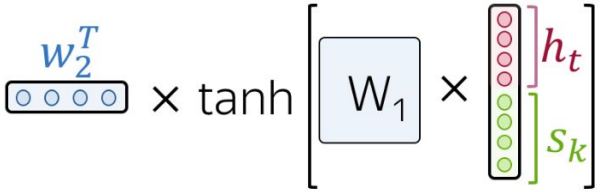
$$\text{score}(h_t, s_k) = h_t^T \; s_k$$

Bilinear



$$\text{score}(h_t, s_k) = h_t^T \; W s_k$$

Multi-Layer Perceptron



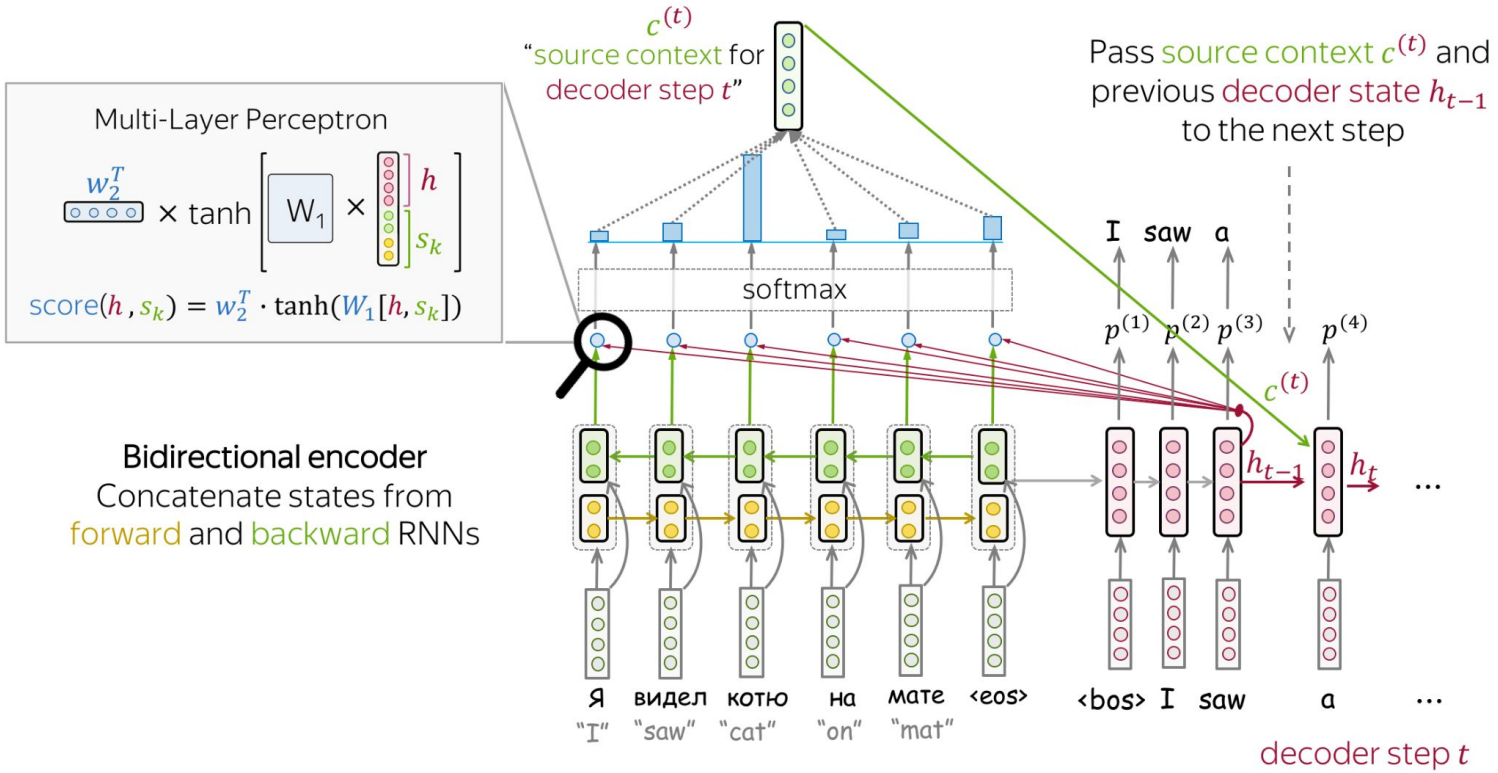
$$\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1[h_t, s_k])$$

How to Compute Attention Score?

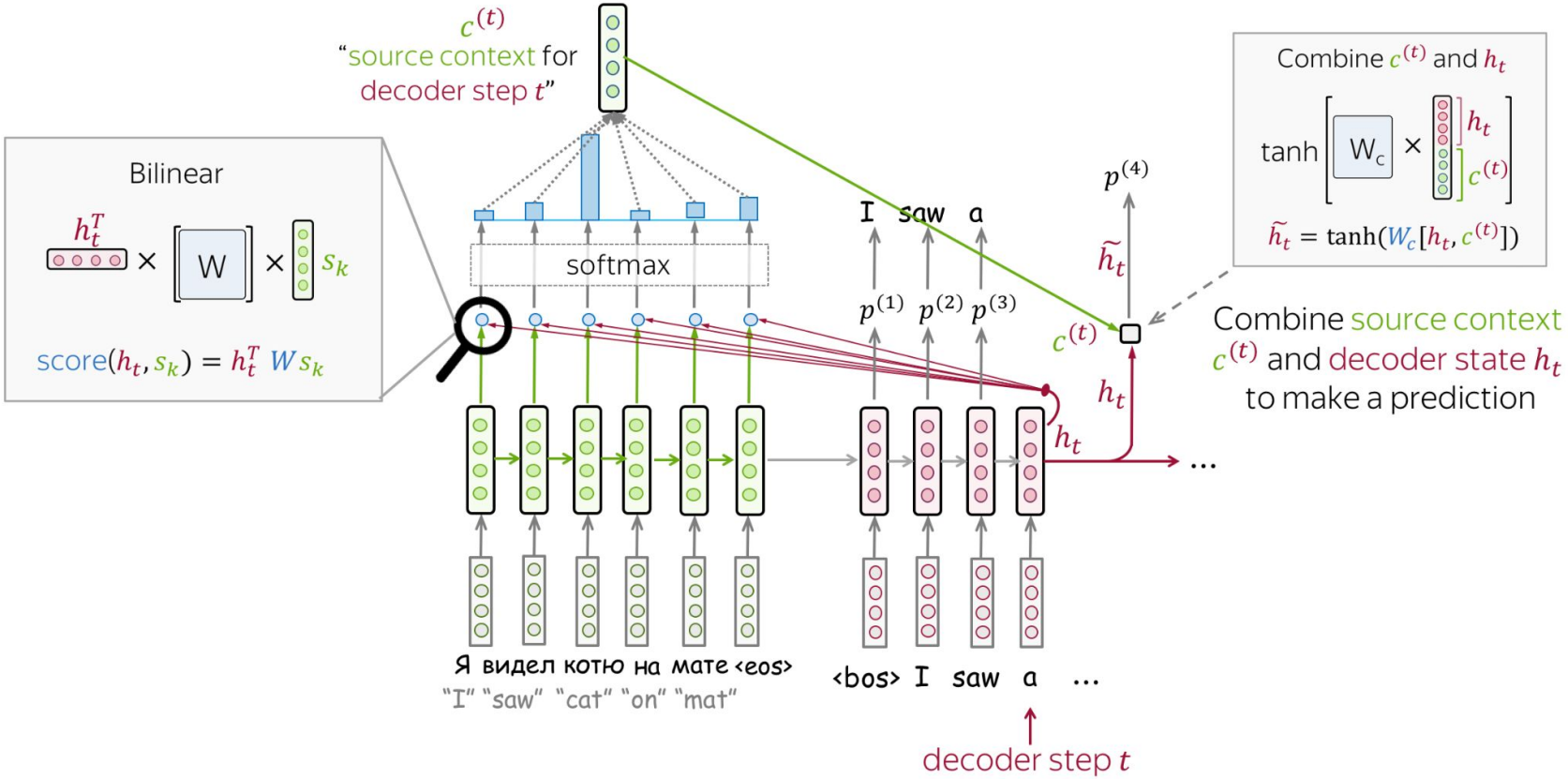
The most popular ways to compute attention scores are:

- dot-product - the simplest method
- bilinear function (aka "Luong attention") - used in the paper Effective Approaches to Attention-based Neural Machine Translation
- multi-layer perceptron (aka "Bahdanau attention") - the method proposed in the original paper (NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE).

Bahdanau Model



Luong Model


$c^{(t)}$
"source context for decoder step t"
Bilinear
h_t^T × W × s_k
score$(h_t, s_k) = h_t^T W s_k$
softmax
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$
$\tilde{h}_t$
$c^{(t)}$
h_t
Combine $c^{(t)}$ and h_t
tanh [W_c × h_t ; $c^{(t)}$]
$\tilde{h}_t = \tanh(W_c[h_t, c^{(t)}])$
Combine source context $c^{(t)}$ and decoder state h_t to make a prediction
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
decoder step t

There are *several* attention variants

- We have some *values* $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and a *query* $\boldsymbol{s} \in \mathbb{R}^{d_2}$

- Attention always involves:
  1. Computing the *attention scores* $\boldsymbol{e} \in \mathbb{R}^N$ ← There are multiple ways to do this
  2. Taking softmax to get *attention distribution* α:

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

  3. Using attention distribution to take weighted sum of values:

$$\boldsymbol{a} = \sum_{i=1}^{N} \alpha_i \boldsymbol{h}_i \in \mathbb{R}^{d_1}$$

thus obtaining the *attention output* $\boldsymbol{a}$ (sometimes called the *context vector*)

Attention is a *general* Deep Learning technique

- 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.

Intuition:

- The weighted sum is a *selective summary* of the information contained in the values, where the query determines which values to focus on.
- Attention is a way to obtain a *fixed-size representation of an arbitrary set of representations* (the values), dependent on some other representation (the query).

Upshot:

- Attention has become the powerful, flexible, general way pointer and memory manipulation in all deep learning models. A new idea from after 2010! From NMT!

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com


ATTENTION IS ALL YOU NEED?
FALSE. YOU NEED WATER AND RATIONS.
imgflip.com

References

https://web.stanford.edu/class/cs224n/slides/cs224n-2022-lecture08-final-project.pdf

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html