Seq2seq and Attention

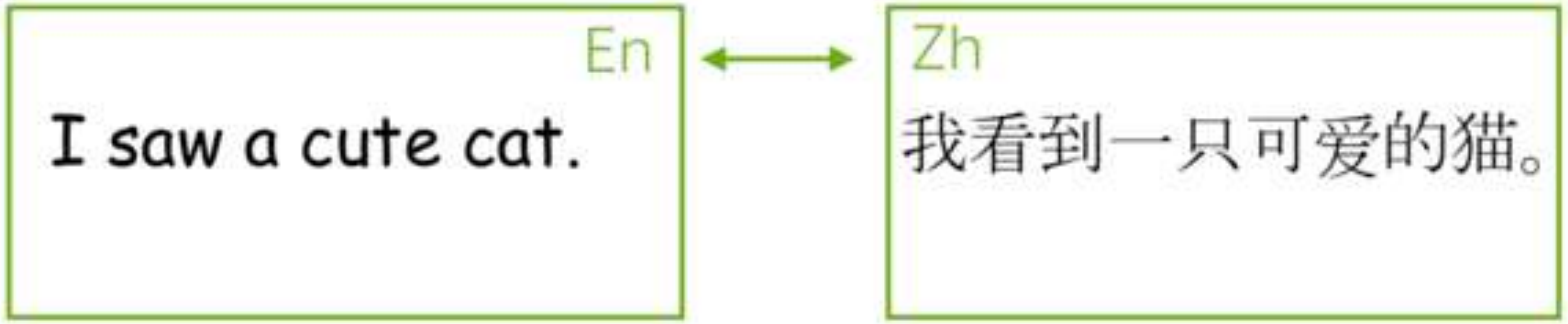
Lena Voita

Lecture-blog and lots of additional materials are here:
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Sequence to Sequence Task

- Translation between natural languages

- More generally, translation between any sequences



What is going to happen:

- Seq2seq Basics
- Attention
- Transformer
- Subword Segmentation: BPE
-  Analysis and Interpretability

What is going to happen:

- Seq2seq Basics
  - Machine Translation Task
  - Encoder-Decoder Framework
  - Conditional LMs
  - The Simplest RNN Model
  - Training
  - Inference
- Attention
- Transformer
- Subword Segmentation: BPE
- Analysis and Interpretability

Translation

<u>Human Translation</u>

$$y^* = \arg\max_y p(y|x)$$

The “probability” is intuitive and is given by a human translator’s expertise

Translation

<u>Human Translation</u>

$$y^* = \arg\max_y p(y|x)$$

The “probability” is intuitive and is given by a human translator’s expertise

<u>Machine Translation</u>

model parameters

$$y' = \arg\max_y p(y|x, \theta)$$

Translation

<u>Human Translation</u>

$$y^* = \arg\max_y p(y|x)$$

The "probability" is intuitive and is given by a human translator's expertise

<u>Machine Translation</u>

model parameters

$$y' = \arg\max_y p(y|x, \theta)$$

Questions we need to answer

- modeling

How does the model for $p(y|x,\theta)$ look like?

Translation

<u>Human Translation</u>

$$y^* = \arg\max_y p(y|x)$$

The “probability” is intuitive and is given by a human translator’s expertise

<u>Machine Translation</u>

model parameters

$$y' = \arg\max_y p(y|x,\theta)$$

Questions we need to answer

- modeling

How does the model for $p(y|x,\theta)$ look like?

- learning

How to find θ?

Translation

Human Translation

$$y^* = \arg\max_y p(y|x)$$

The "probability" is intuitive and is given by a human translator's expertise

Machine Translation

model parameters

$$y' = \arg\max_y p(y|x, \theta)$$

Questions we need to answer

- modeling

How does the model for $p(y|x,\theta)$ look like?

- learning

How to find θ?

- search

How to find the argmax?

Translation

Human Translation

$$y^* = \arg\max_y p(y|x)$$

The "probability" is intuitive and is given by a human translator's expertise

Machine Translation

model parameters

$$y' = \arg\max_y p(y|x,\theta)$$

Questions we need to answer

- modeling

How does the model for $p(y|x,\theta)$ look like?

- learning

How to find θ?

- search

How to find the argmax?

Translation

Human Translation

$$y^* = \arg\max_y p(y|x)$$

The "probability" is intuitive and is given by a human translator's expertise

Machine Translation

model parameters

$$y' = \arg\max_y p(y|x, \theta)$$

Questions we need to answer

- **modeling**

  How does the model for $p(y|x, \theta)$ look like?

- **learning**

  How to find θ?

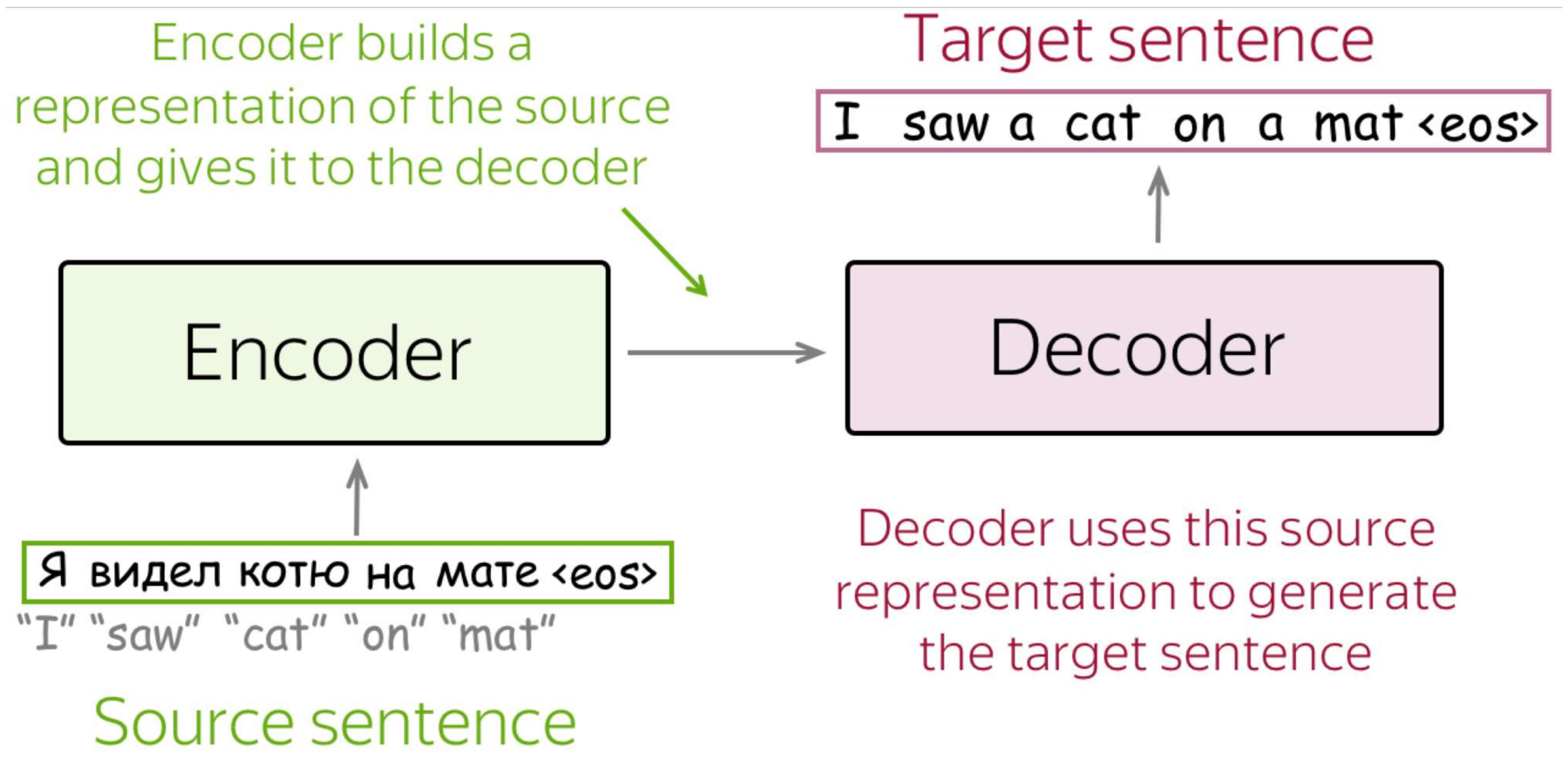
- **search**

  How to find the argmax?

Encoder-Decoder Framework

The standard modeling paradigm:

- Encoder – reads the source sentence and produces its representation

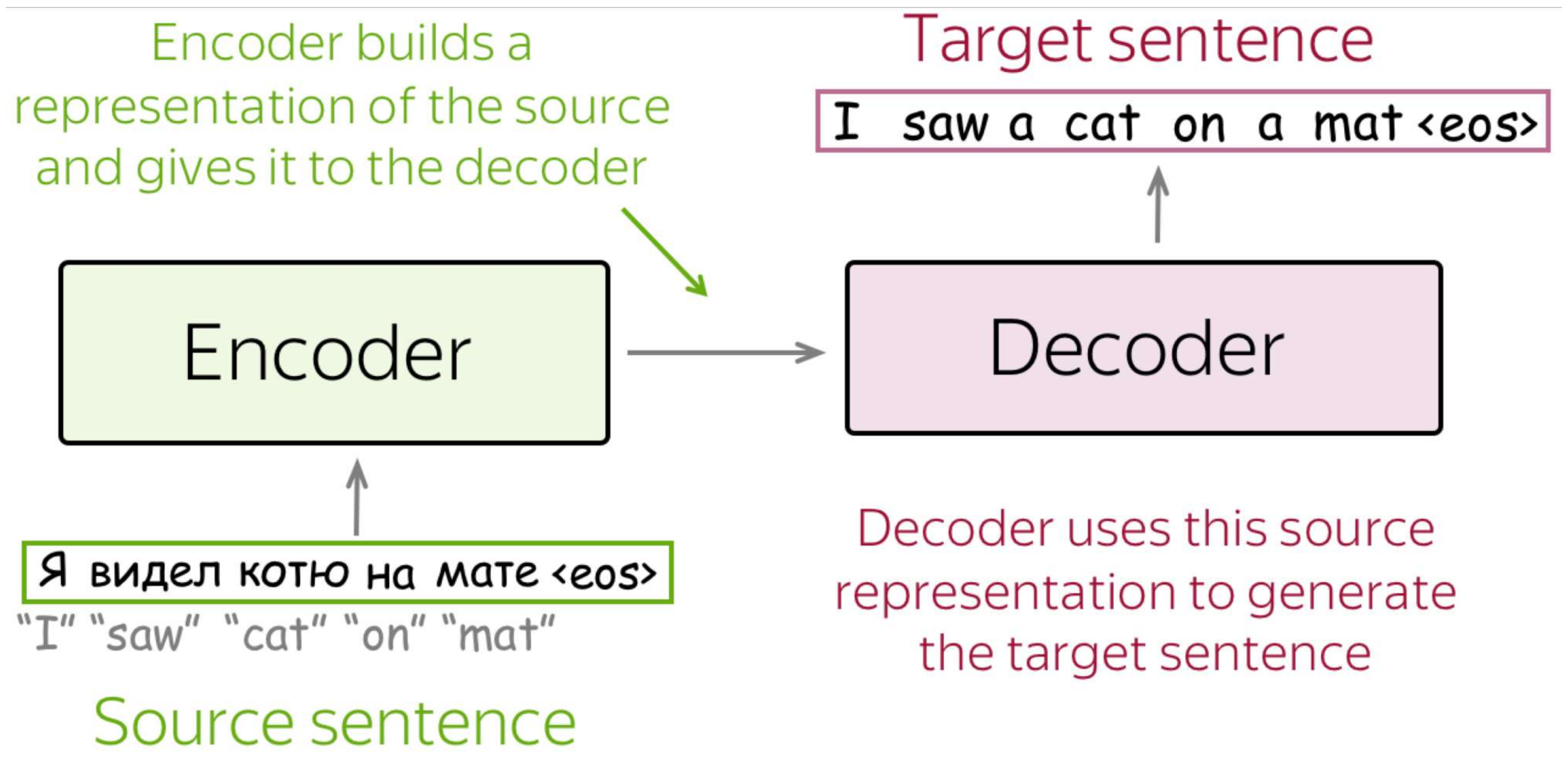


Encoder-Decoder Framework

The standard modeling paradigm:

- Encoder – reads the source sentence and produces its representation

- Decoder - uses source representation from the encoder to generate the target sequence.



Conditional Language Models

Language Models: $P\left(y_{1,}y_2, \ldots, y_n\right) = \prod_{t=1}^{n} p(y_t | y_{<t})$
(left-to-right)

Conditional Language Models

Language Models: $P(y_1, y_2, \dots, y_n) = \prod_{t=1}^{n} p(y_t | y_{<t})$
(left-to-right)

Conditional
Language Models: $P(y_1, y_2, \dots, y_n, | x) = \prod_{t=1}^{n} p(y_t | y_{<t}, x)$

condition on source x

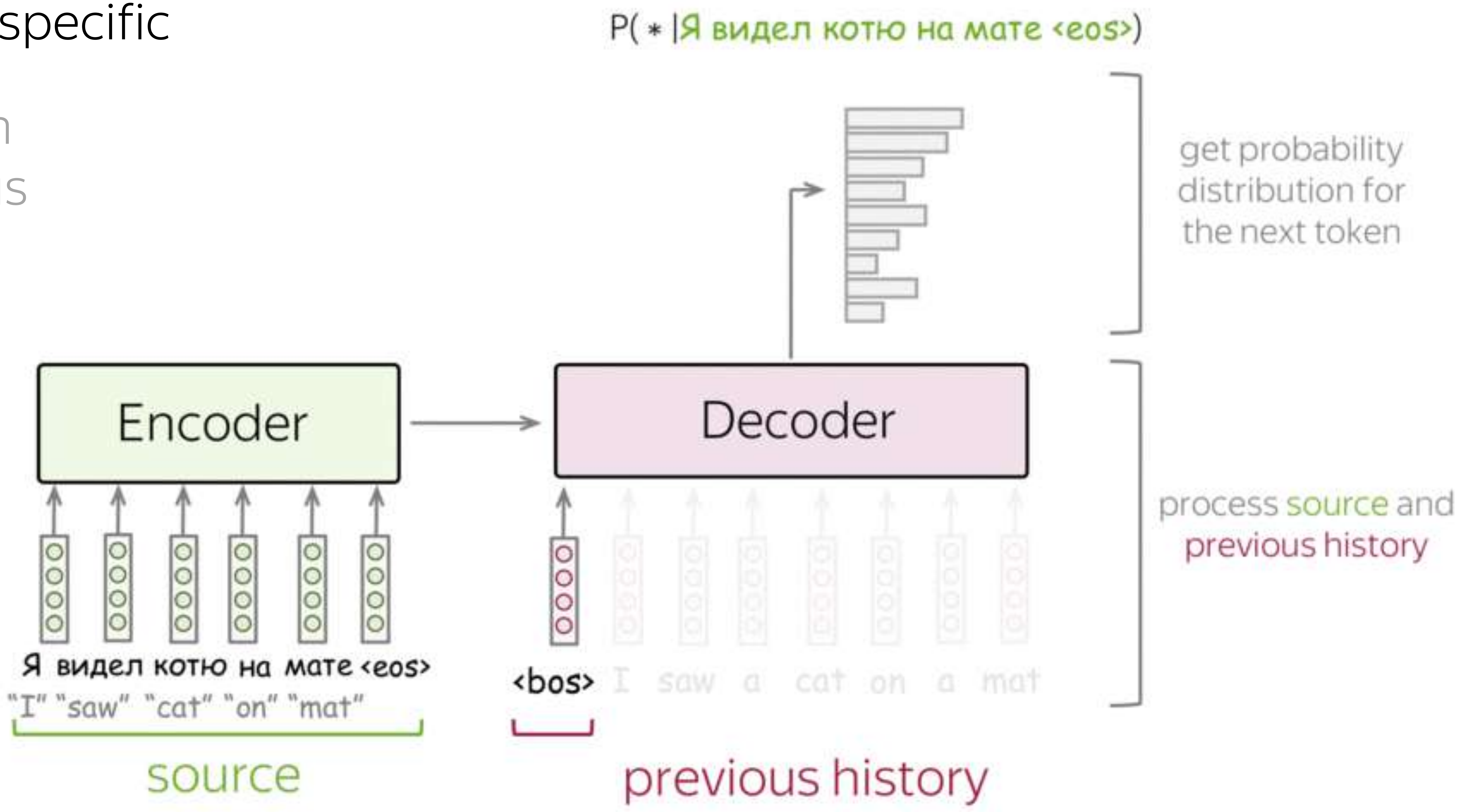
General View

- process context – model-specific

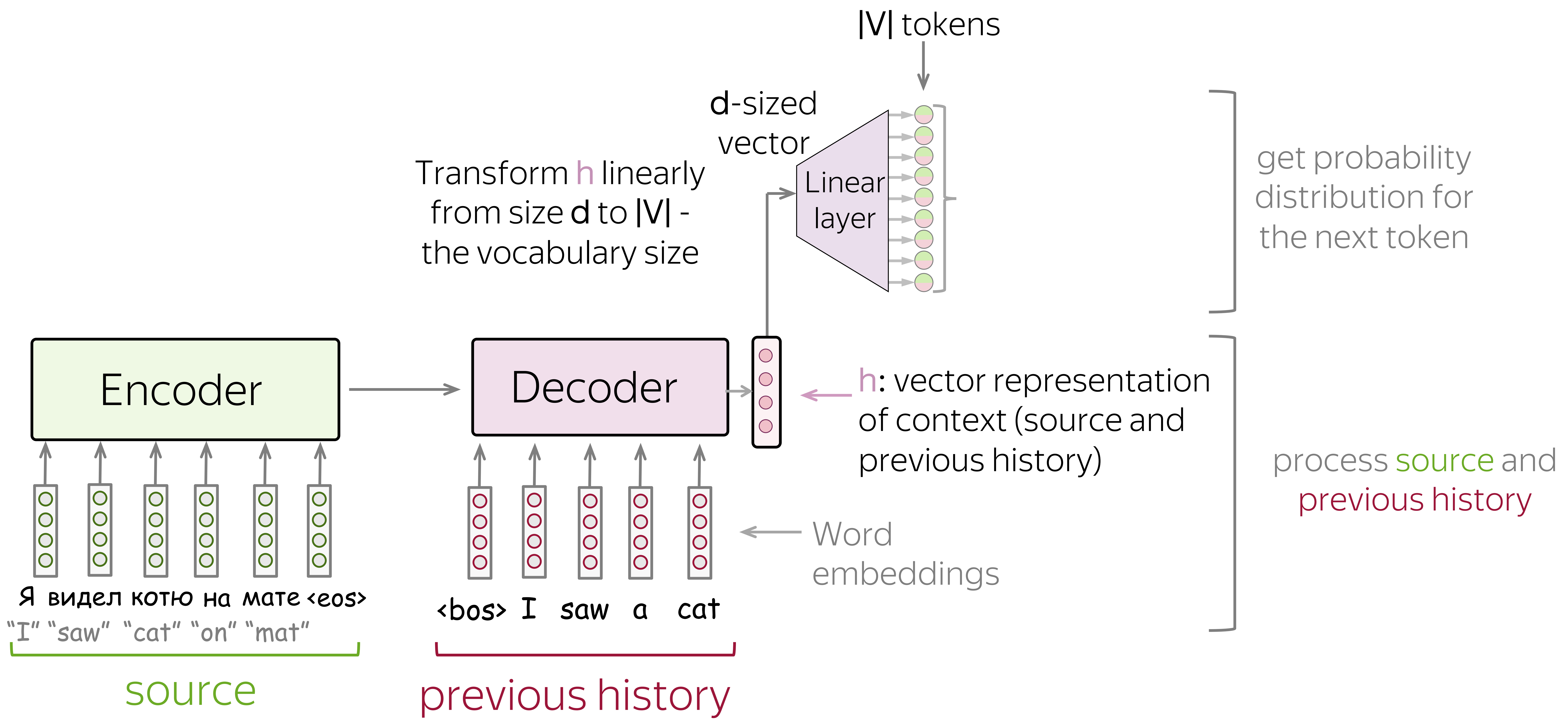
  Get vector representation of the source and previous target tokens

- evaluate probabilities – model-agnostic

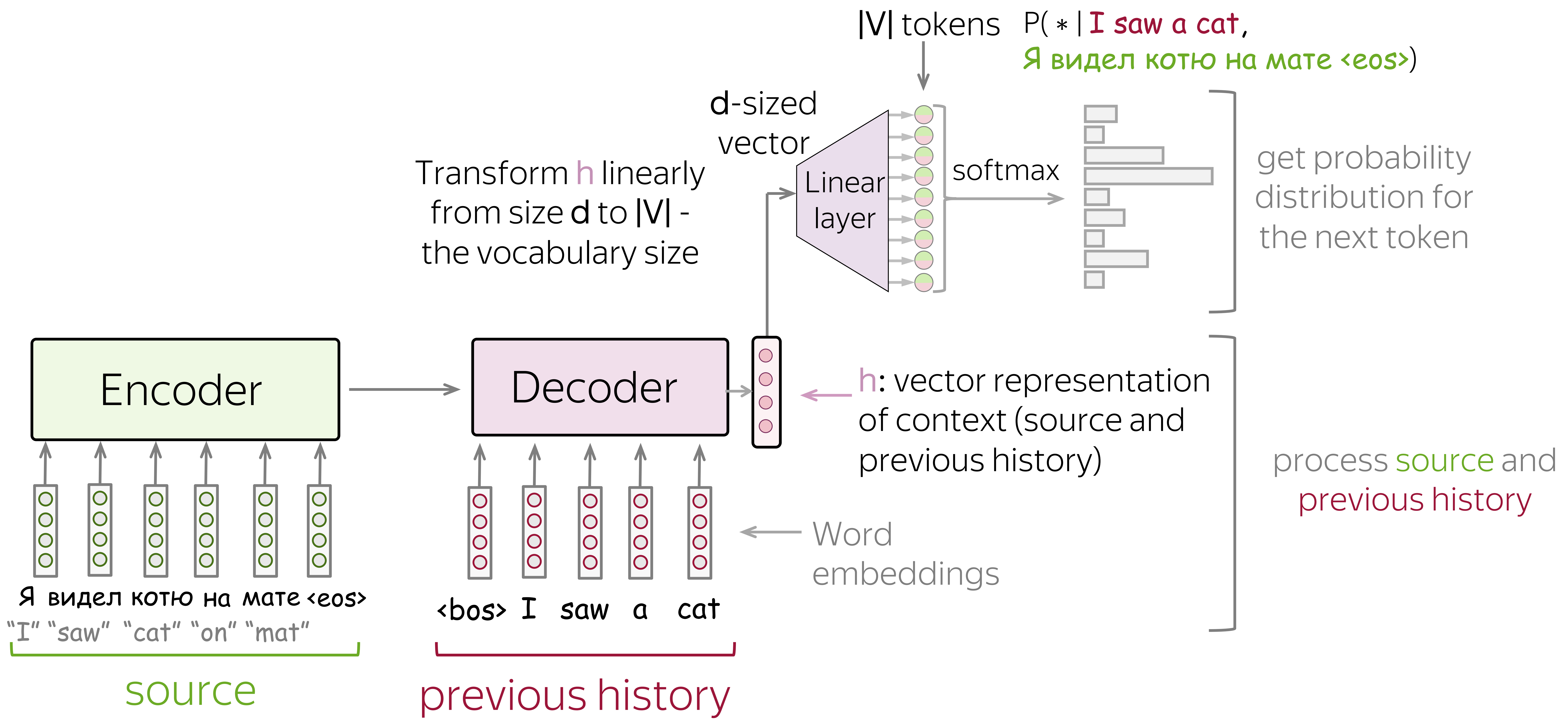
  Predict probability distribution for the next target token



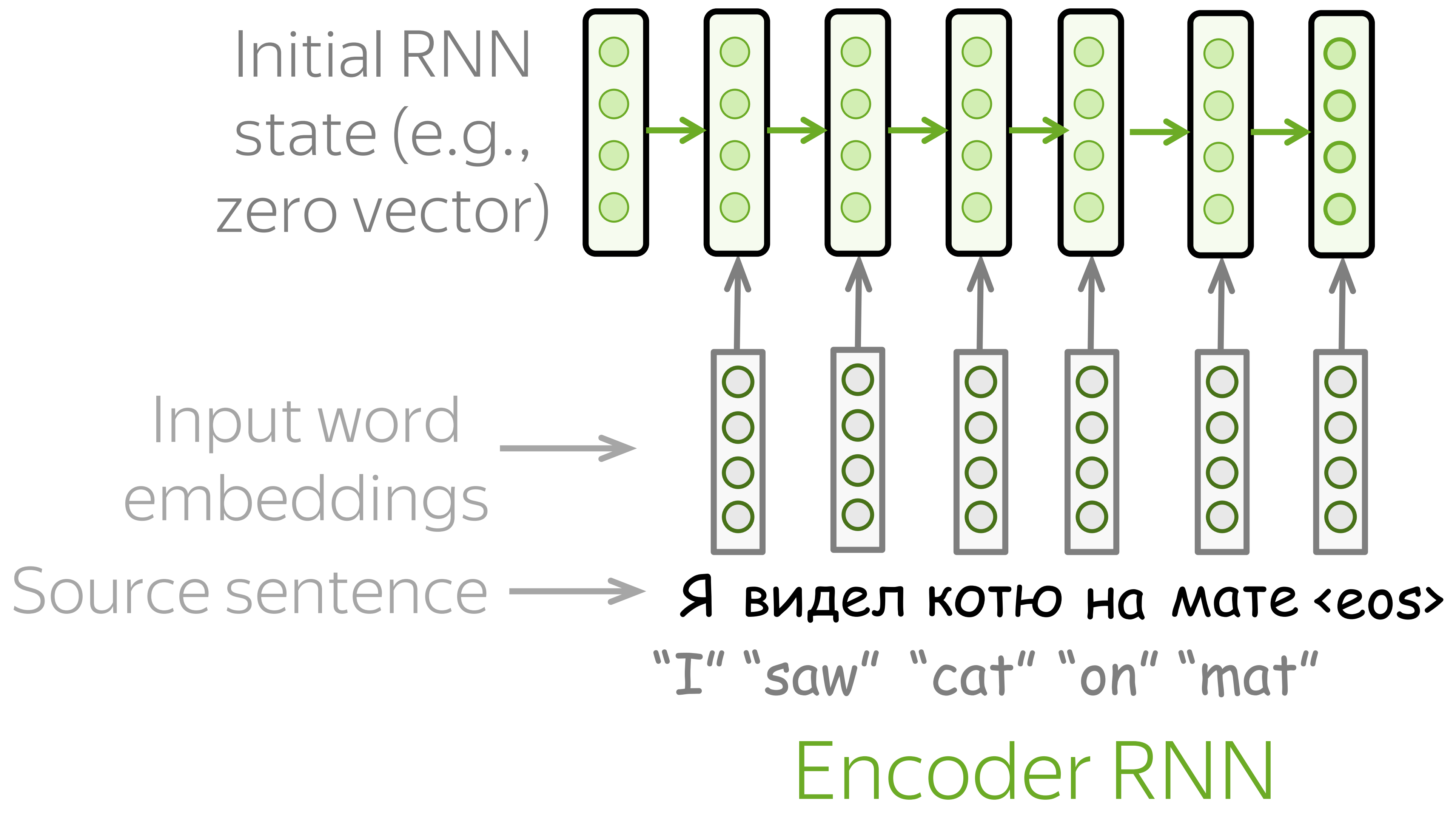
High-Level Pipeline



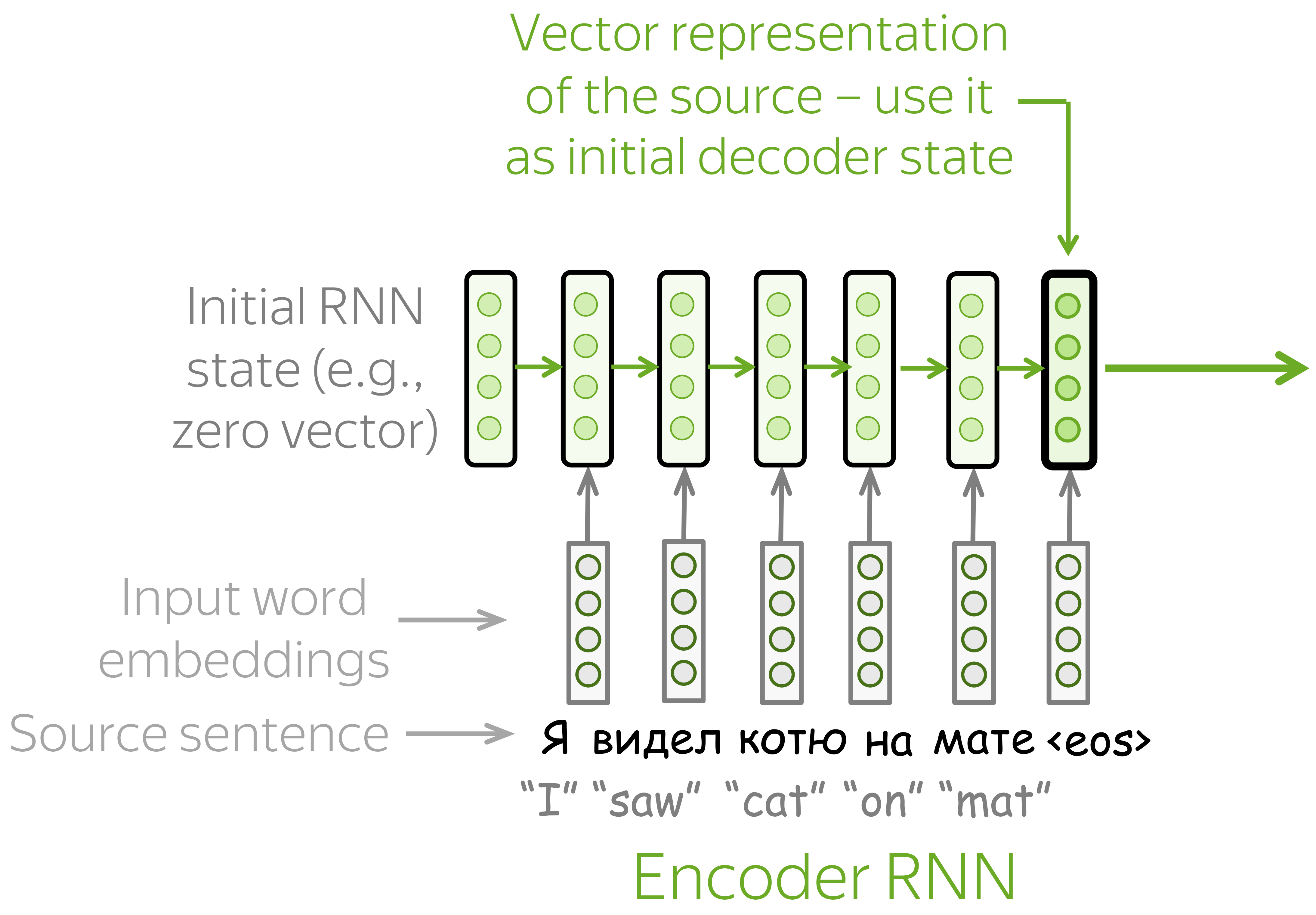
High-Level Pipeline



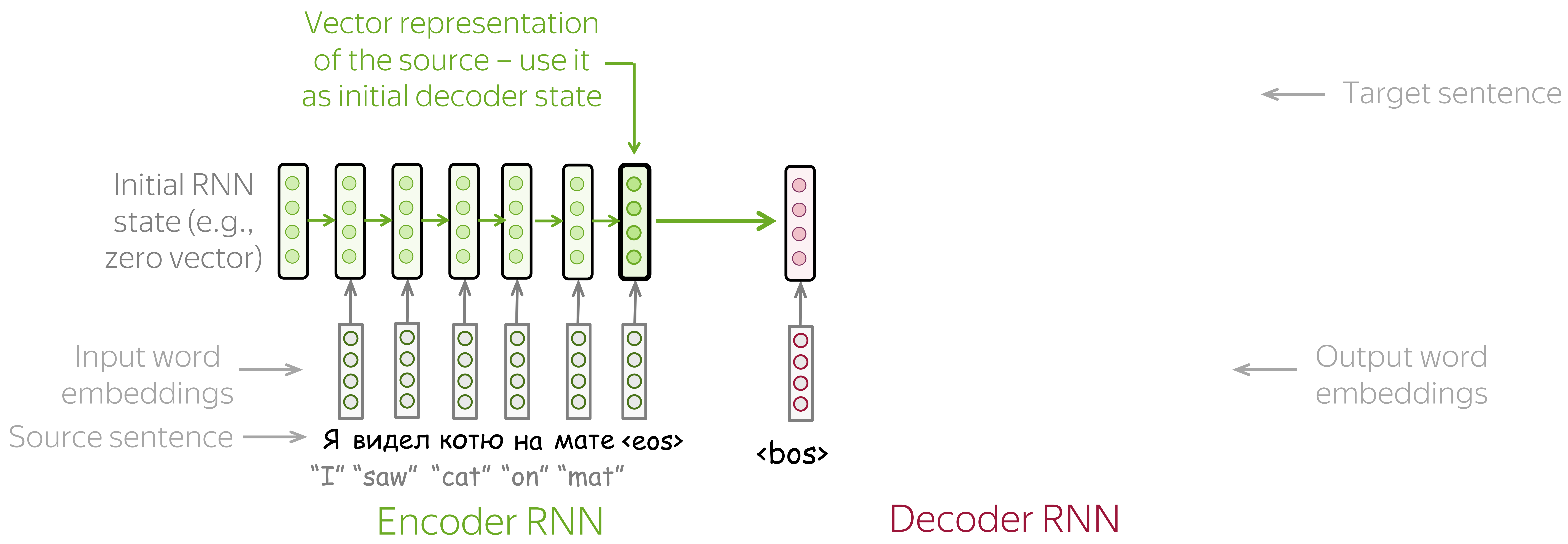
The Simplest Model: RNN Encoder and Decoder



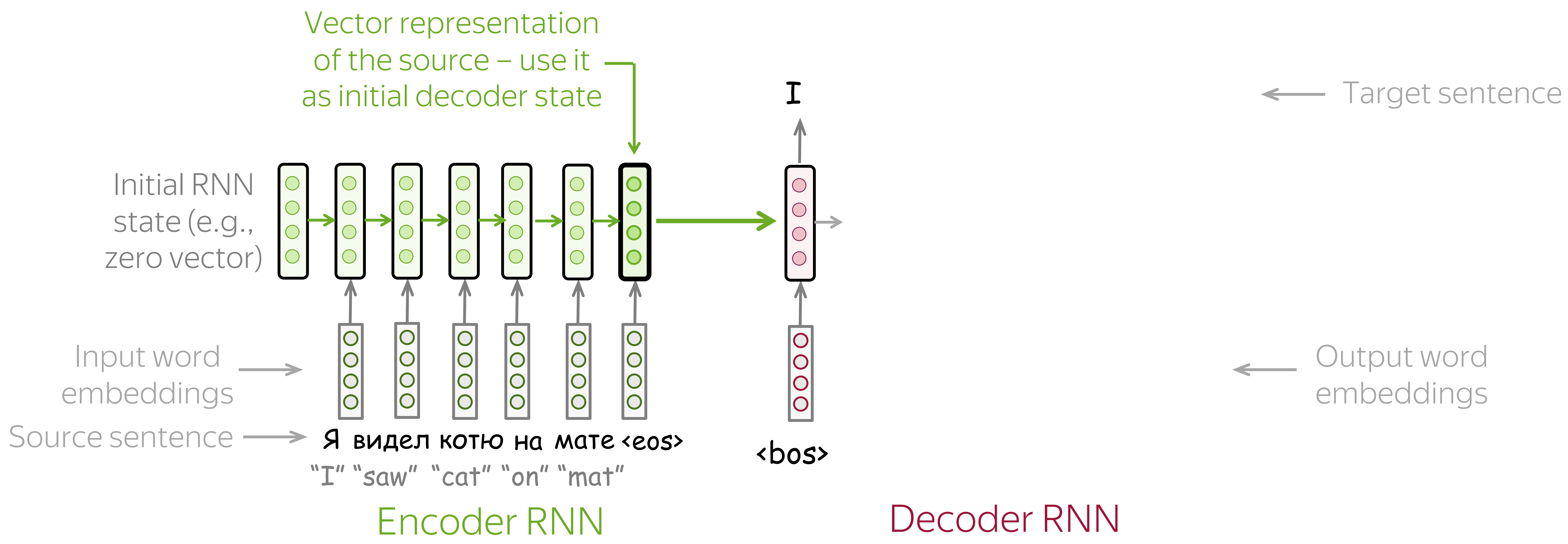
The Simplest Model: RNN Encoder and Decoder



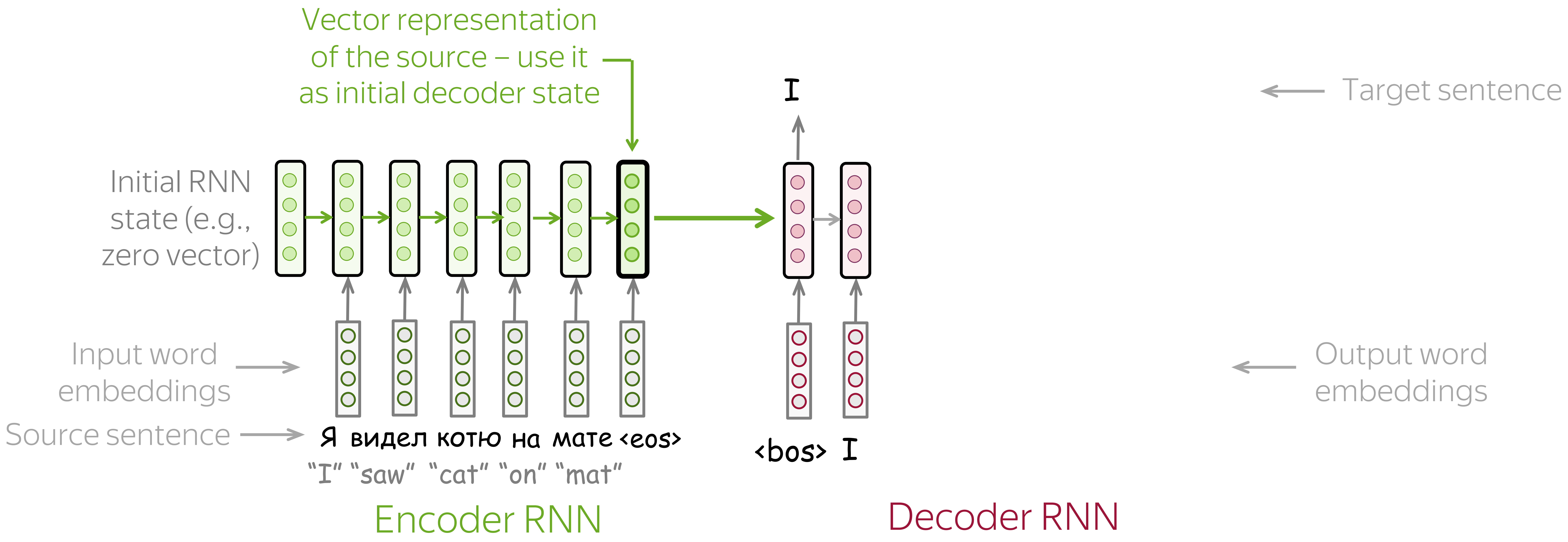
The Simplest Model: RNN Encoder and Decoder



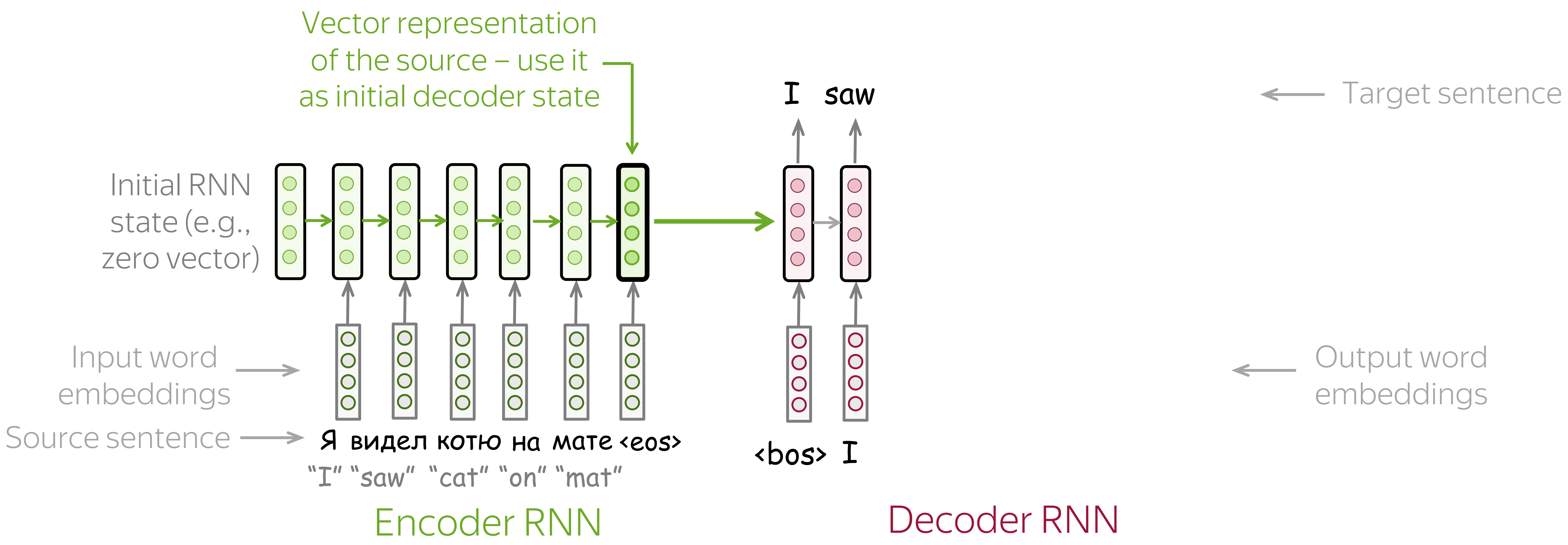
The Simplest Model: RNN Encoder and Decoder



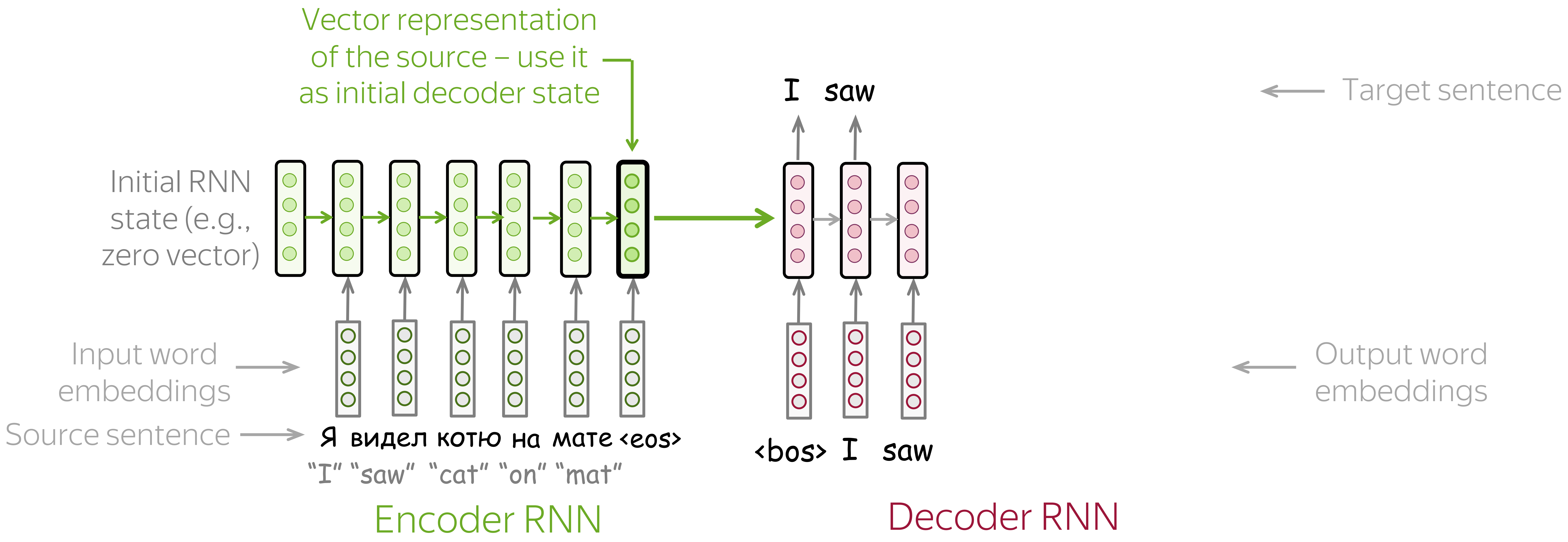
The Simplest Model: RNN Encoder and Decoder



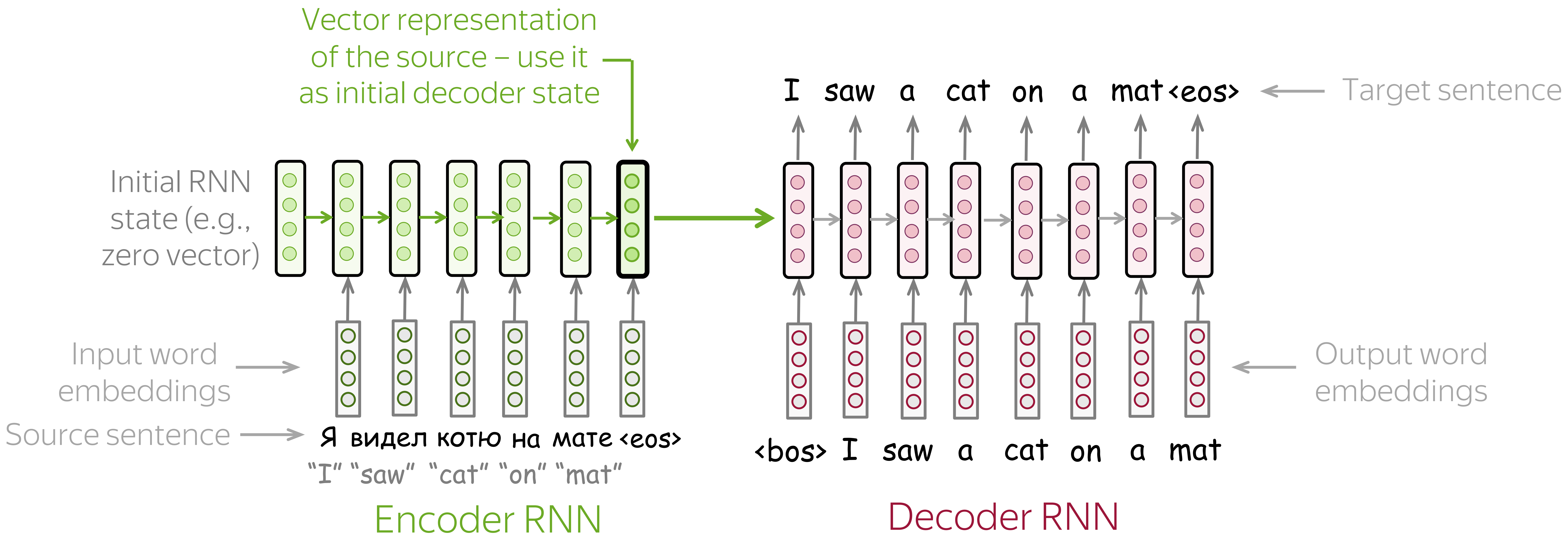
The Simplest Model: RNN Encoder and Decoder



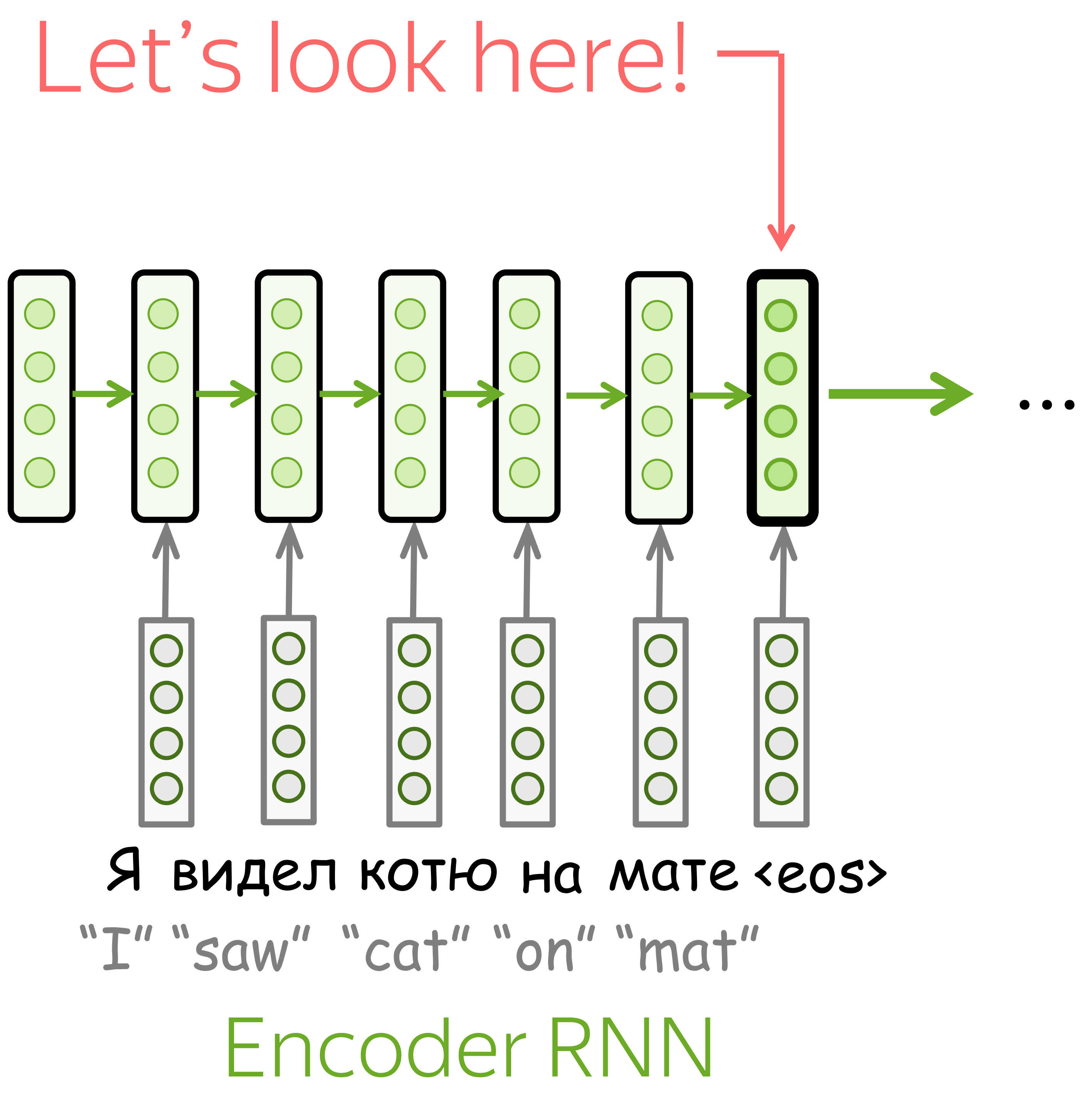
The Simplest Model: RNN Encoder and Decoder

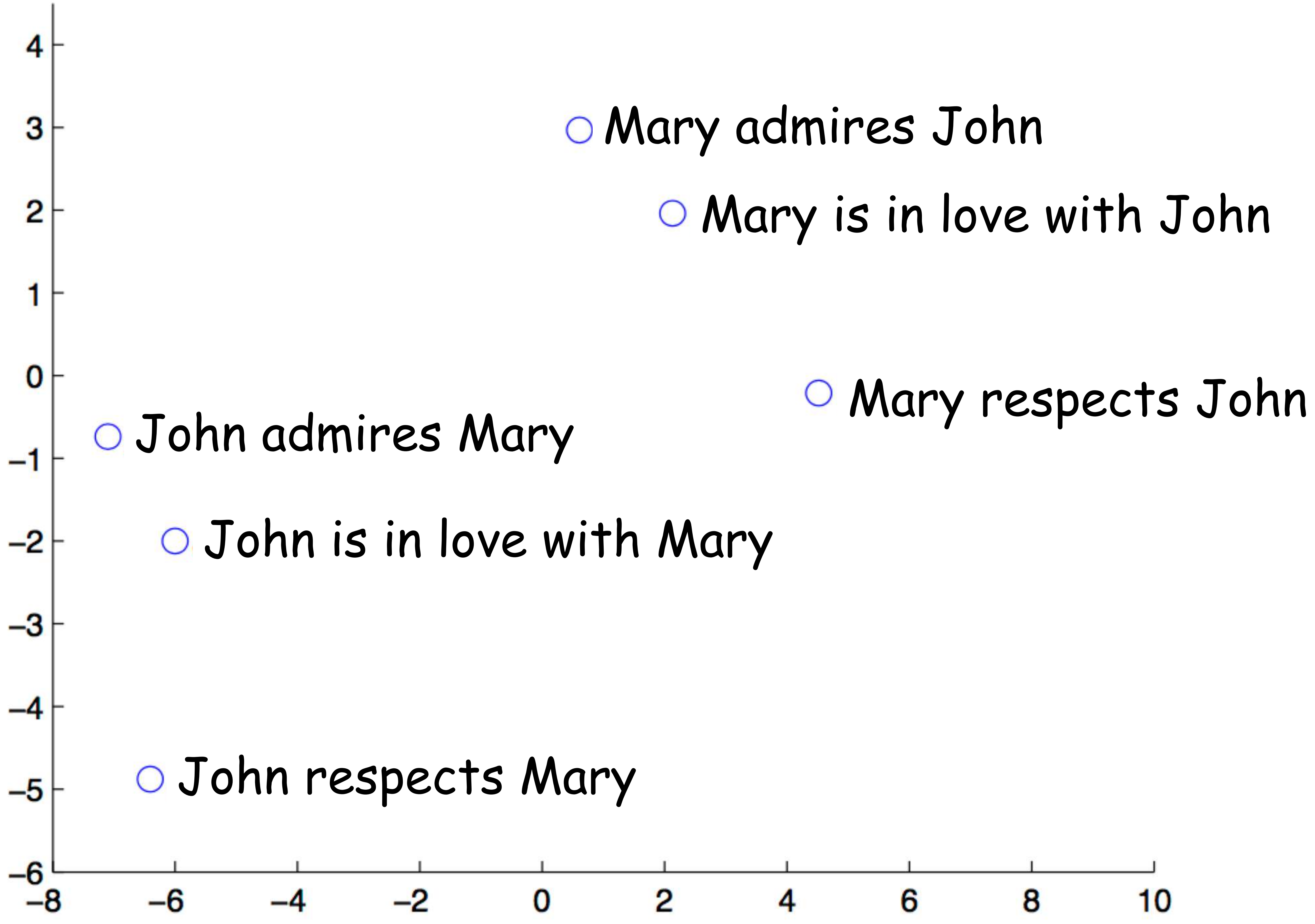


The Simplest Model: RNN Encoder and Decoder



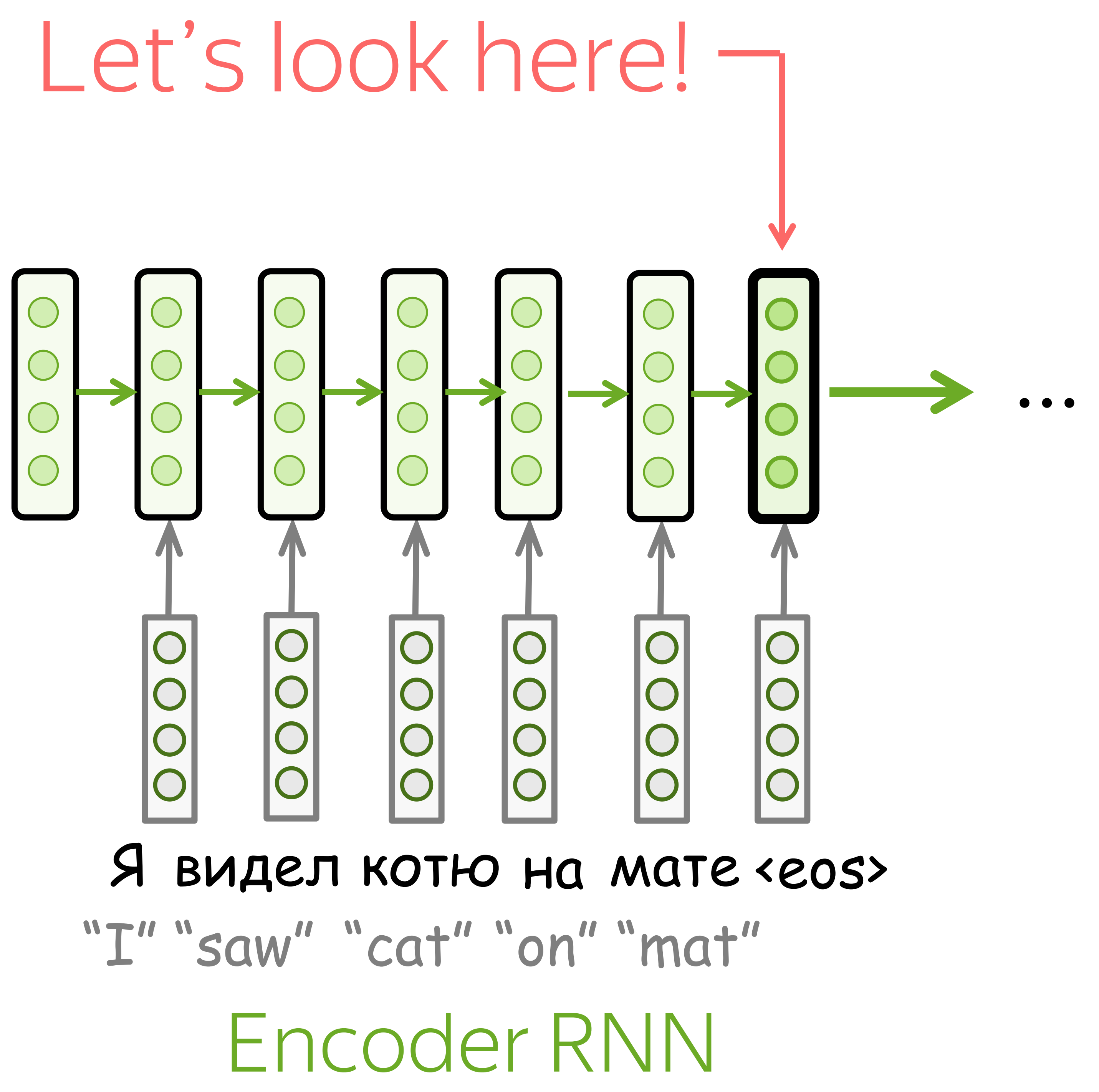
What does final encoder state represent?

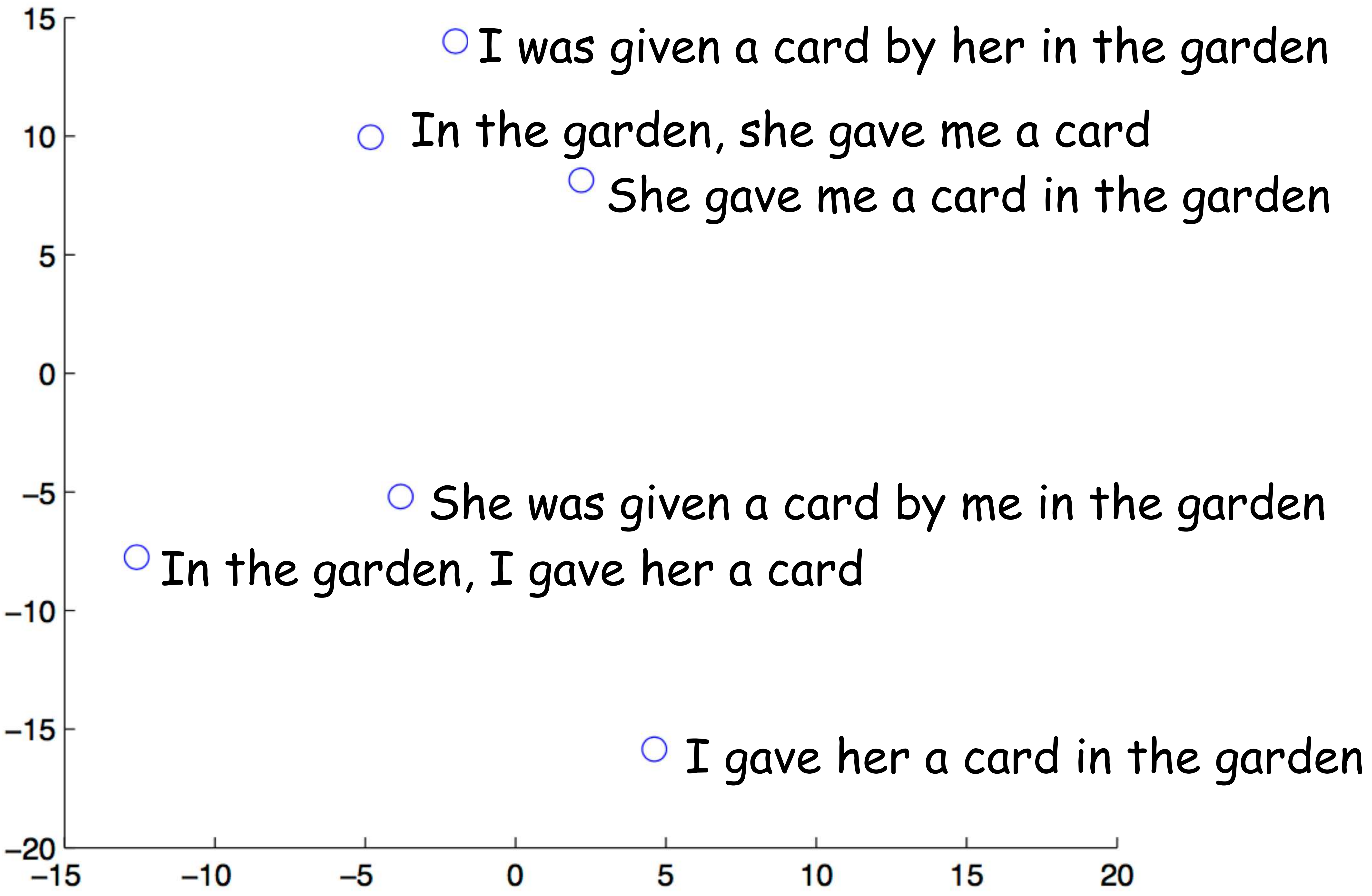




The examples are from the paper Sequence to Sequence Learning with Neural Networks

What does final encoder state represent?





The examples are from the paper Sequence to Sequence Learning with Neural Networks

Translation

Human Translation

$$y^* = \arg\max_y p(y|x)$$

The "probability" is intuitive and is given by a human translator's expertise

Machine Translation

model parameters

$$y' = \arg\max_y p(y|x, \theta)$$

Questions we need to answer

- modeling

How does the model for $p(y|x, \theta)$ look like?

- learning

How to find θ?

- search

How to find the argmax?

Translation

Human Translation

$$y^* = \arg\max_y p(y|x)$$

The "probability" is intuitive and is given by a human translator's expertise

Machine Translation

model parameters

$$y' = \arg\max_y p(y|x, \theta)$$

Questions we need to answer

- modeling

How does the model for $p(y|x, \theta)$ look like?

- learning

How to find θ?

- search

How to find the argmax?

Cross-Entropy – again!

Source sequence:

Я видел котю на мате <eos>

"I" "saw" "cat" "on" "mat"

Target sequence: ← one training example

I saw a cat on a mat <eos> ← one step for this example

previous tokens

we want the model to predict this

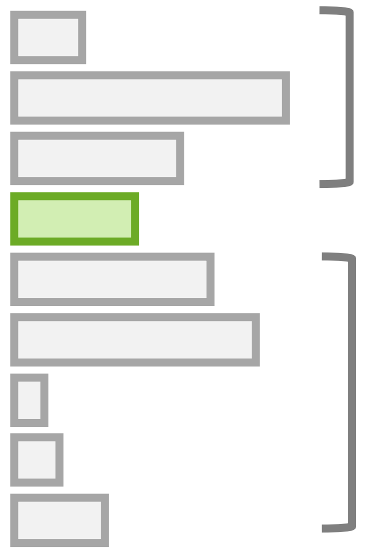
Model prediction: p(∗ |I saw a, Я … <eos>)

← cat →

Target

0
0
0
1
0
0
0
0
0

Loss = -log (p(cat)) → min



decrease

increase

decrease

Translation

Human Translation

$$y^* = \arg\max_y p(y|x)$$

The "probability" is intuitive and is given by a human translator's expertise

Machine Translation

model parameters

$$y' = \arg\max_y p(y|x, \theta)$$

Questions we need to answer

- modeling

How does the model for $p(y|x, \theta)$ look like?

- learning

How to find θ?

- search

How to find the argmax?

Translation

Human Translation

$$y^* = \arg\max_y p(y|x)$$

The "probability" is intuitive and is given by a human translator's expertise

Machine Translation

model parameters

$$y' = \arg\max_y p(y|x, \theta)$$

Questions we need to answer

- modeling

How does the model for $p(y|x, \theta)$ look like?

- learning

How to find θ?

- search

How to find the argmax?

How To Generate a Translation?

$$y' = \arg\max_y p(y|x) = \arg\max_y \prod_{t=1}^{n} p(y_t|y_{<t}, x)$$

Greedy Decoding

$$y' = \arg\max_y p(y|x) = \arg\max_y \prod_{t=1}^{n} p(y_t|y_{<t}, x)$$

Straightforward:

- **greedy** - at each step, pick token with the highest probability

Greedy Decoding

$$y' = \arg\max_y p(y|x) = \arg\max_y \prod_{t=1}^{n} p(y_t|y_{<t}, x)$$

Straightforward:

- **greedy** - at each step, pick token with the highest probability

Wait a minute...

Greedy Decoding

$$y' = \arg\max_{y} p(y|x) = \arg\max_{y} \prod_{t=1}^{n} p(y_t|y_{<t}, x)$$

Straightforward:

- **greedy** - at each step, pick token with the highest probability

$$\boxed{\arg\max_{y} \prod_{t=1}^{n} p(y_t|y_{<t}, x) \neq \prod_{t=1}^{n} \arg\max_{y_t} p(y_t|y_{<t}, x)}$$

- this is bad!

Beam Search

- At each step, keep several best hypotheses

Beam Search

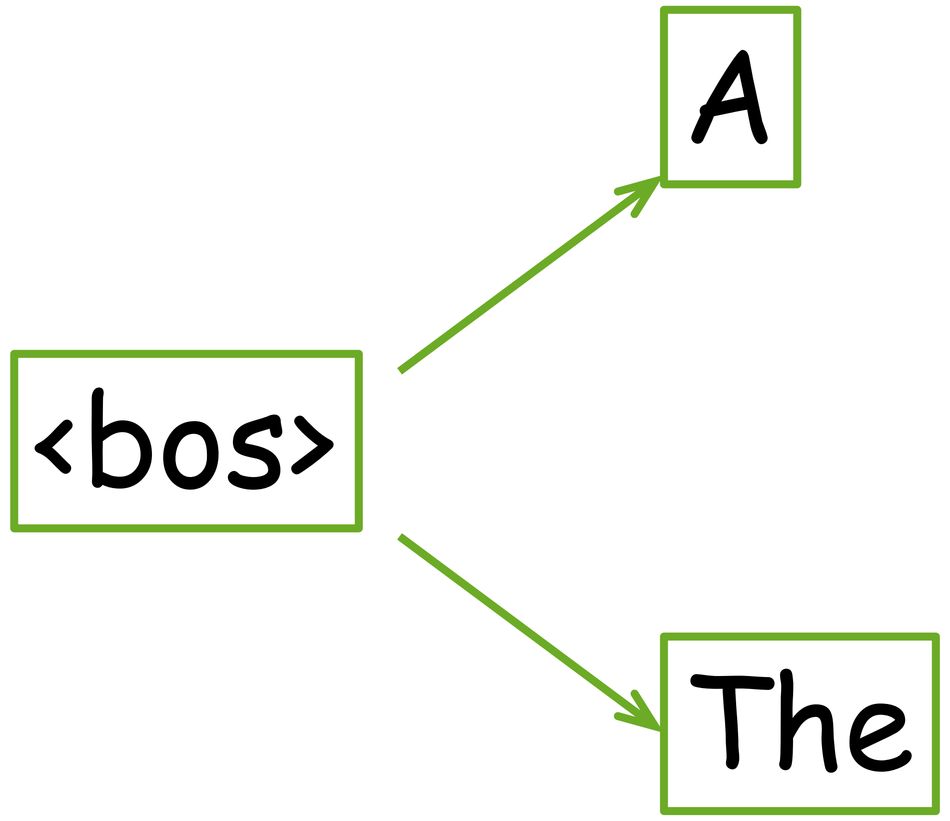
- At each step, keep several best hypotheses

<bos>

Start with the begin of sentence token or with an empty sequence

Beam Search

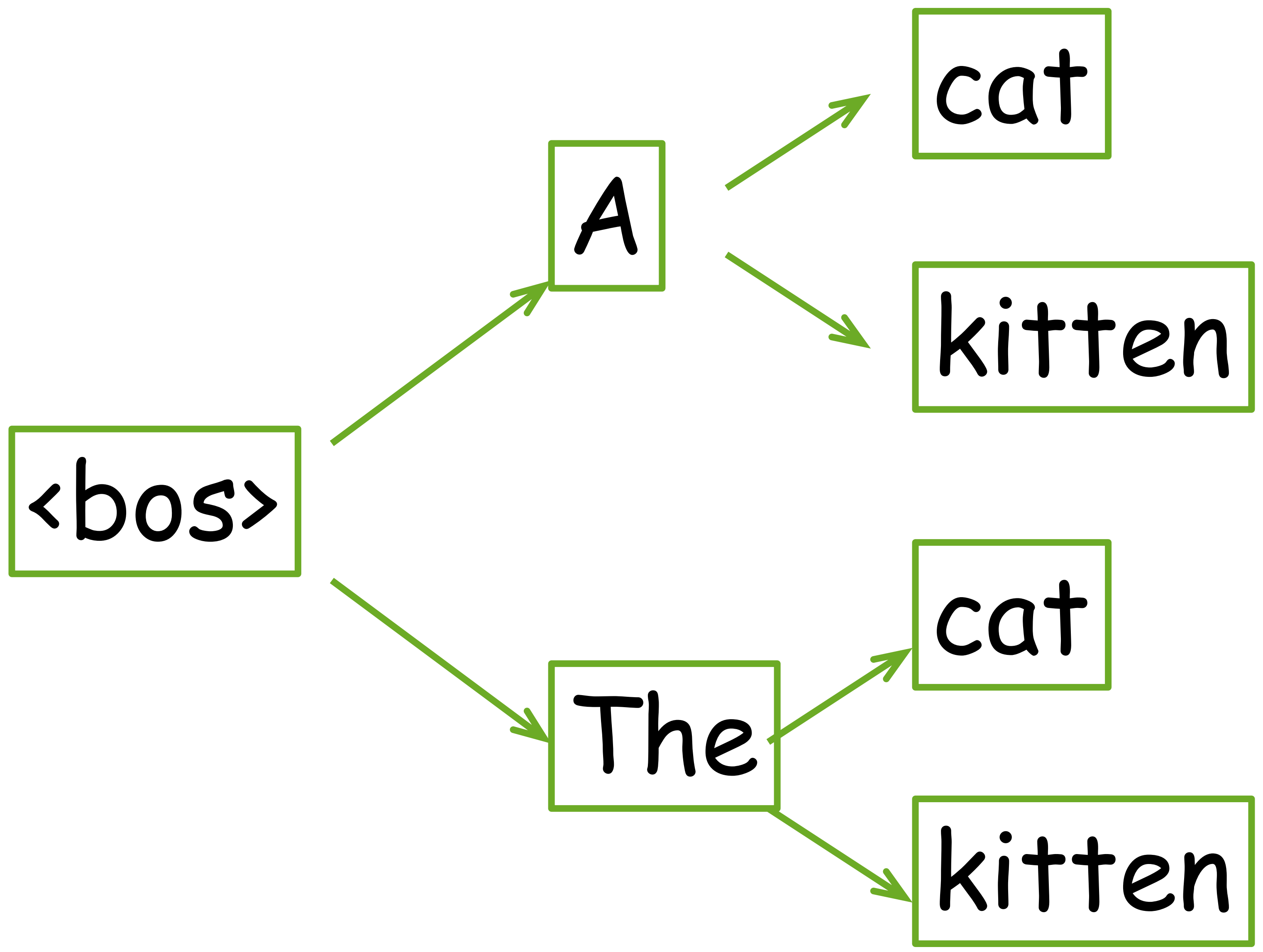
- At each step, keep several best hypotheses



Generate: **`beam_size`** most probable tokens

Beam Search

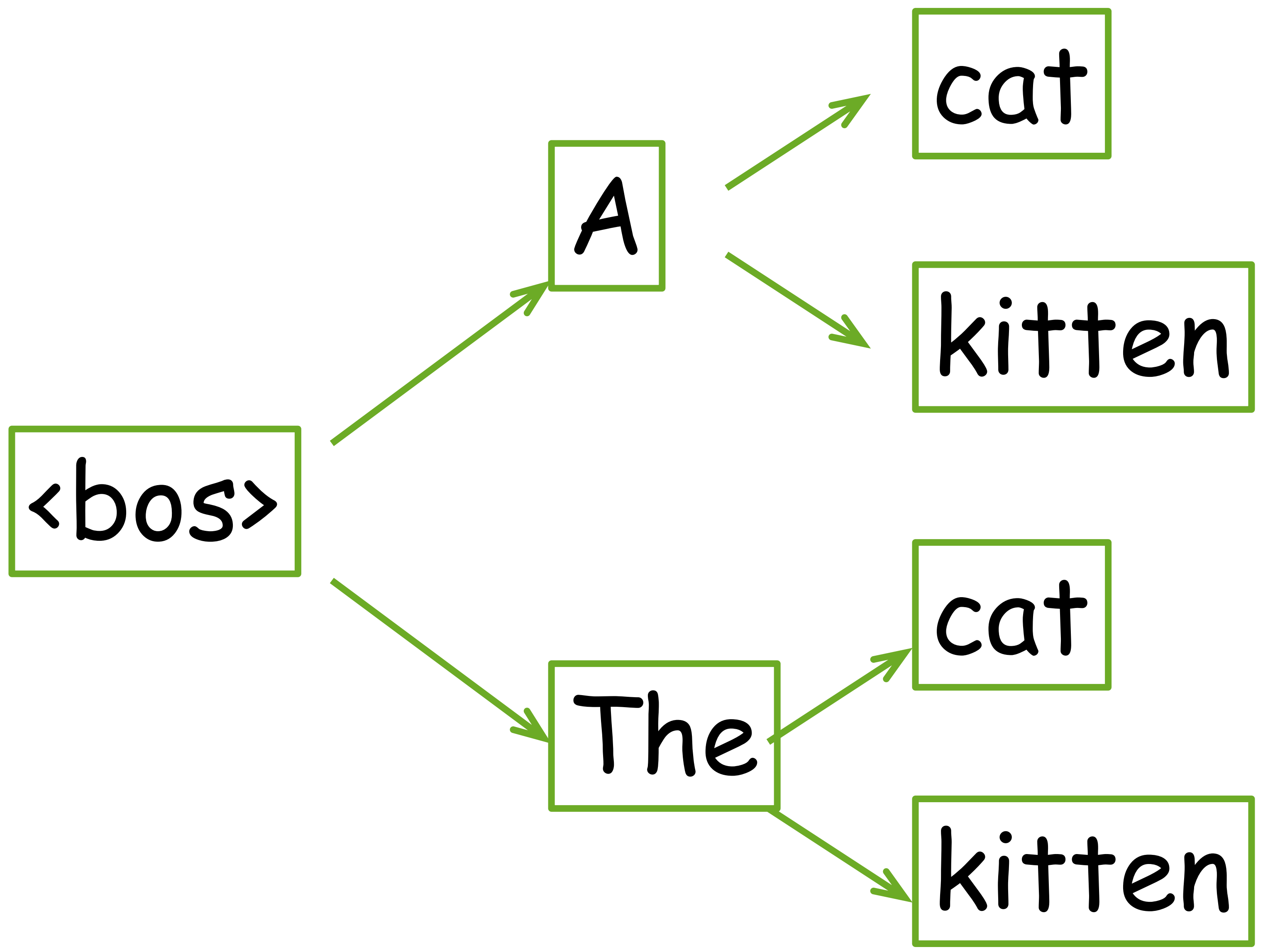
- At each step, keep several best hypotheses



Generate: `beam_size` most probable tokens

Beam Search

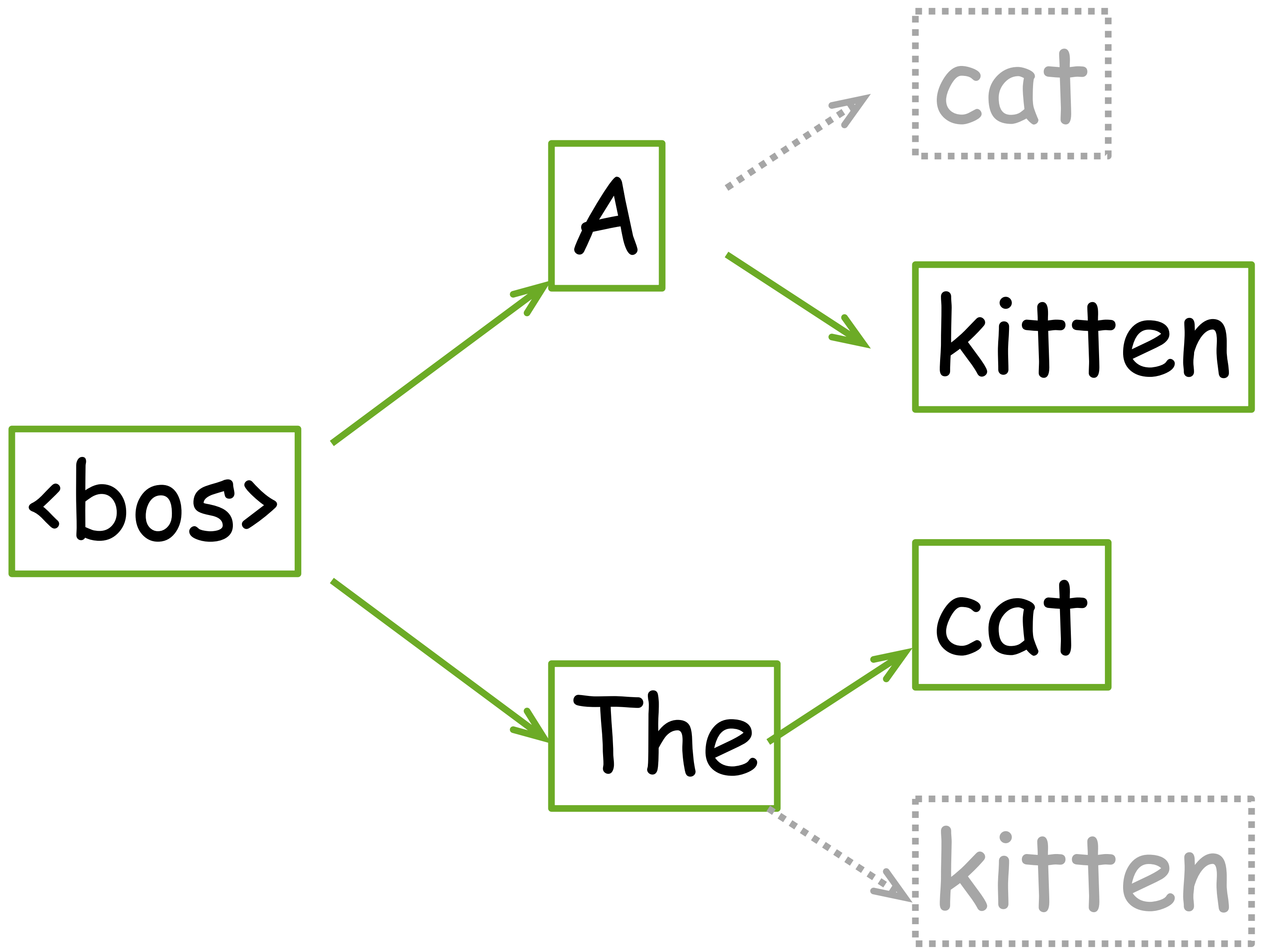
- At each step, keep several best hypotheses



Look at probabilities: $\underbrace{P(\text{The cat}) > P(\text{A kitten})}_{\text{Top 2 hypos}} > P(\text{A cat}) > P(\text{The kitten})$

Beam Search

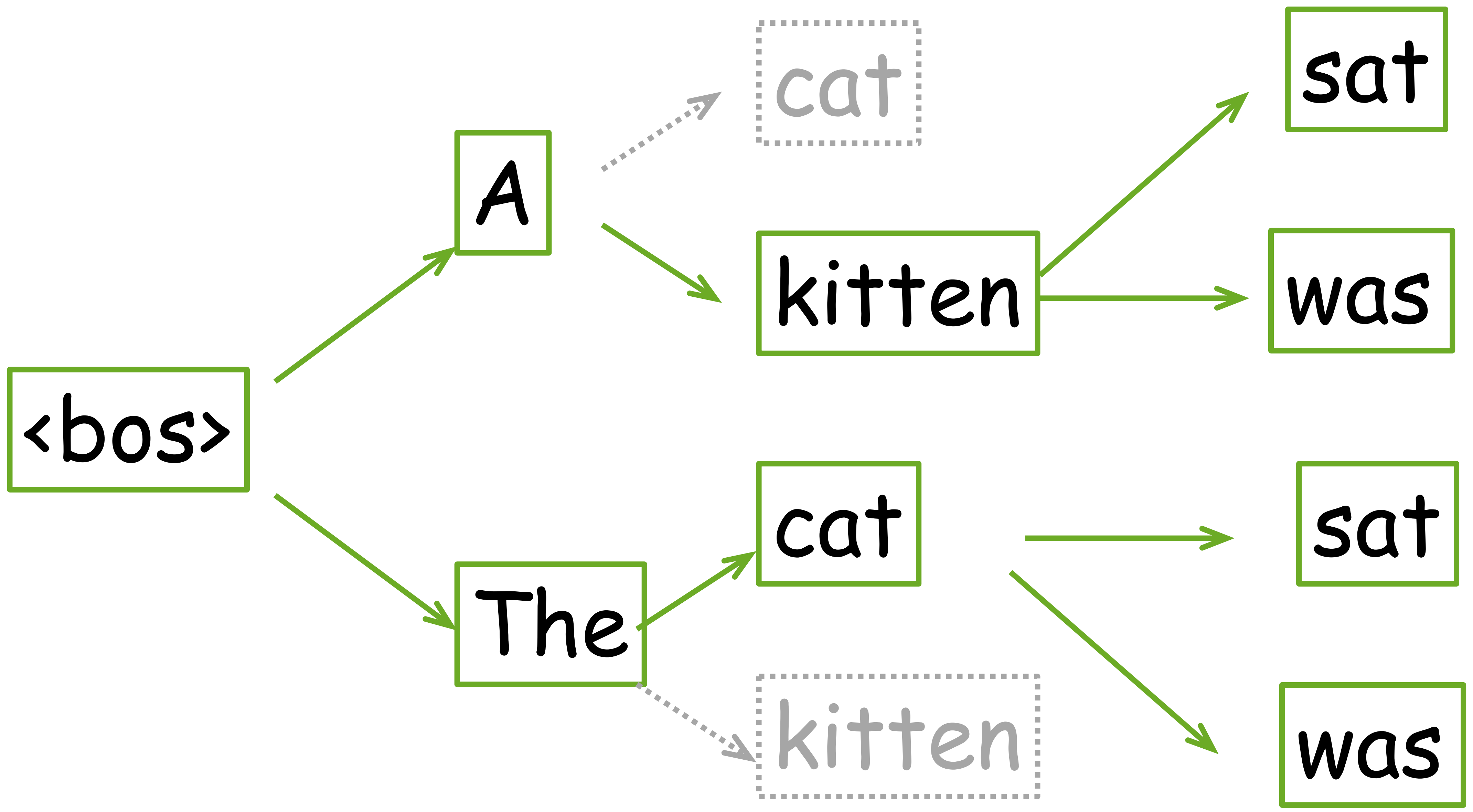
- At each step, keep several best hypotheses



Pick top **`beam_size`** hypos, terminate the rest

Beam Search

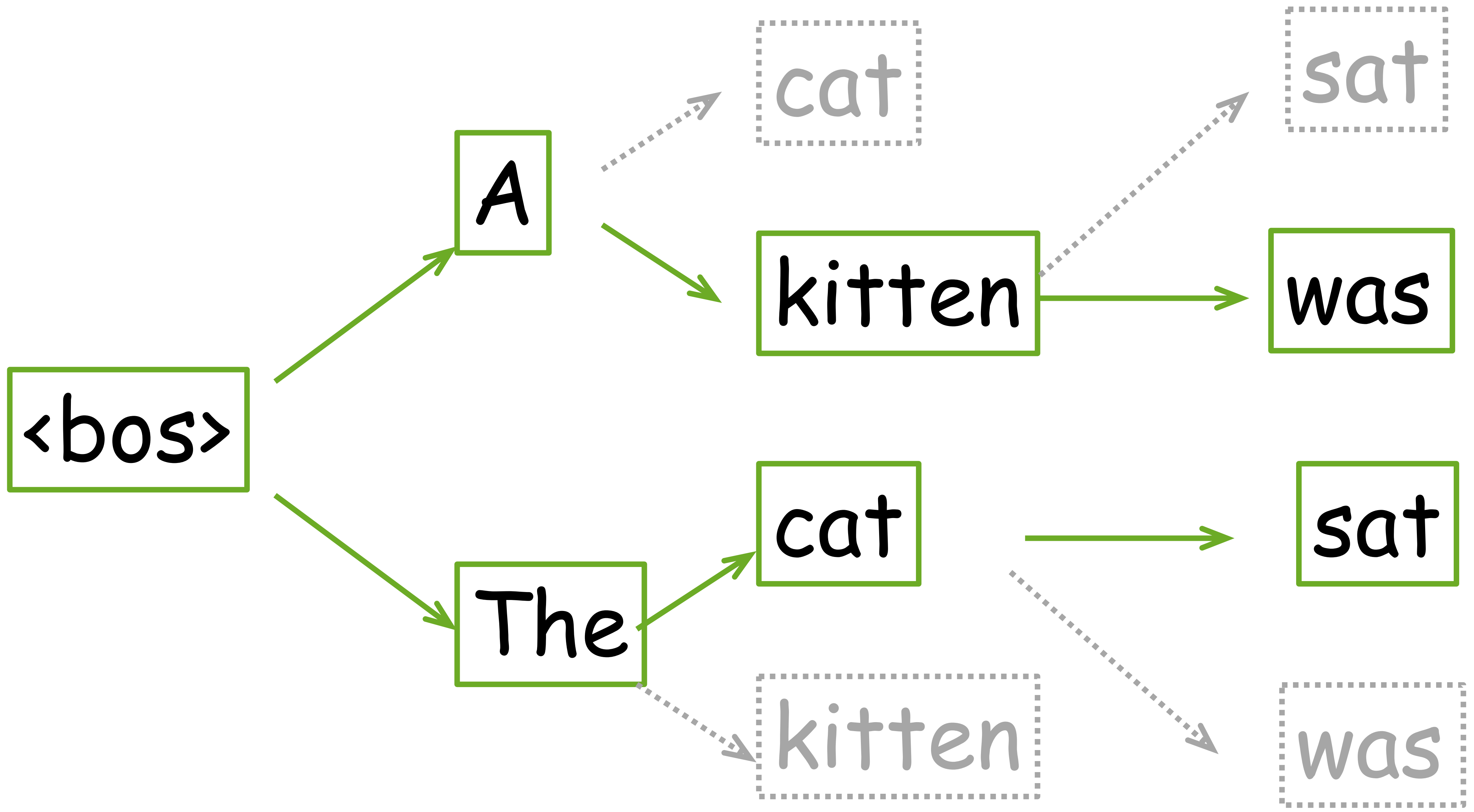
- At each step, keep several best hypotheses



Generate: `beam_size` most probable tokens

Beam Search

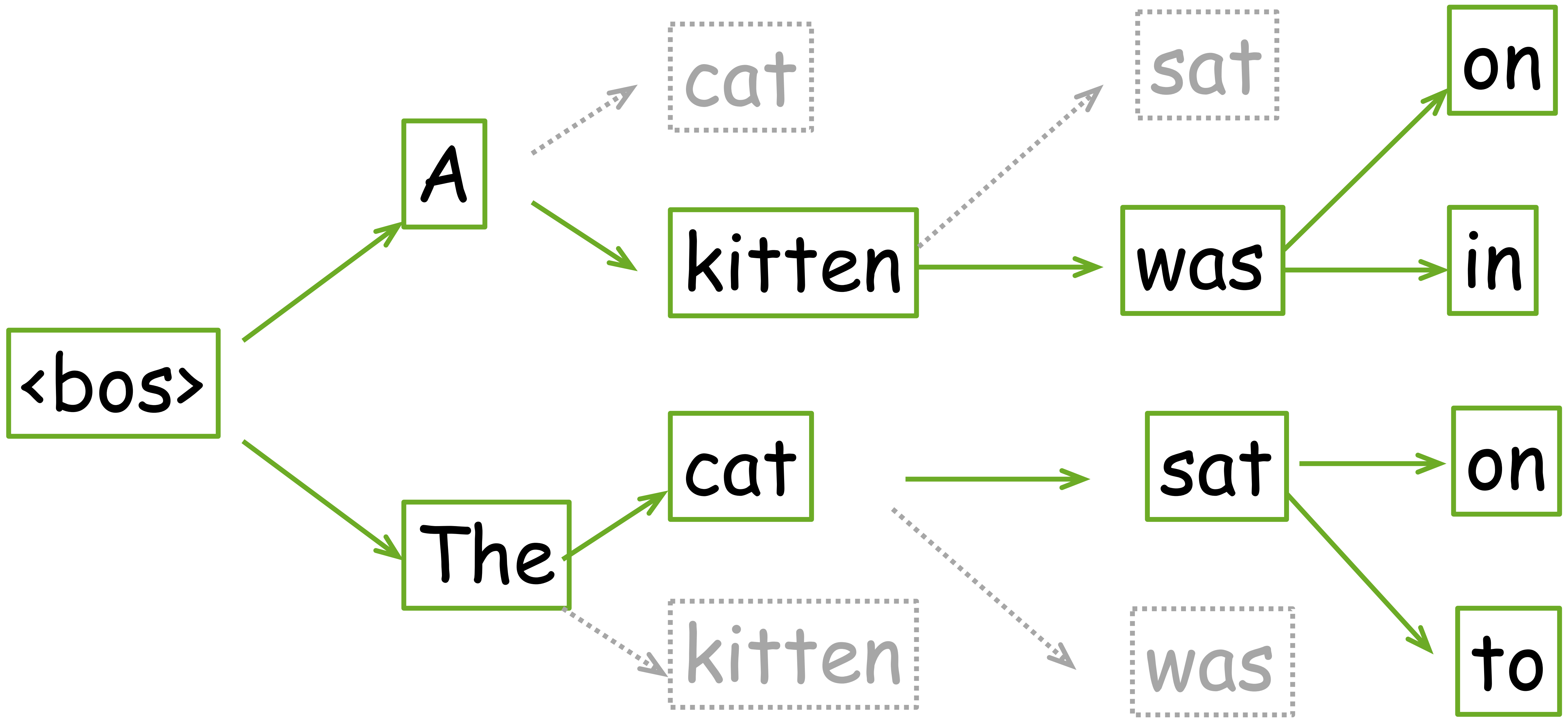
- At each step, keep several best hypotheses



Pick top `beam_size` hypos, terminate the rest

Beam Search

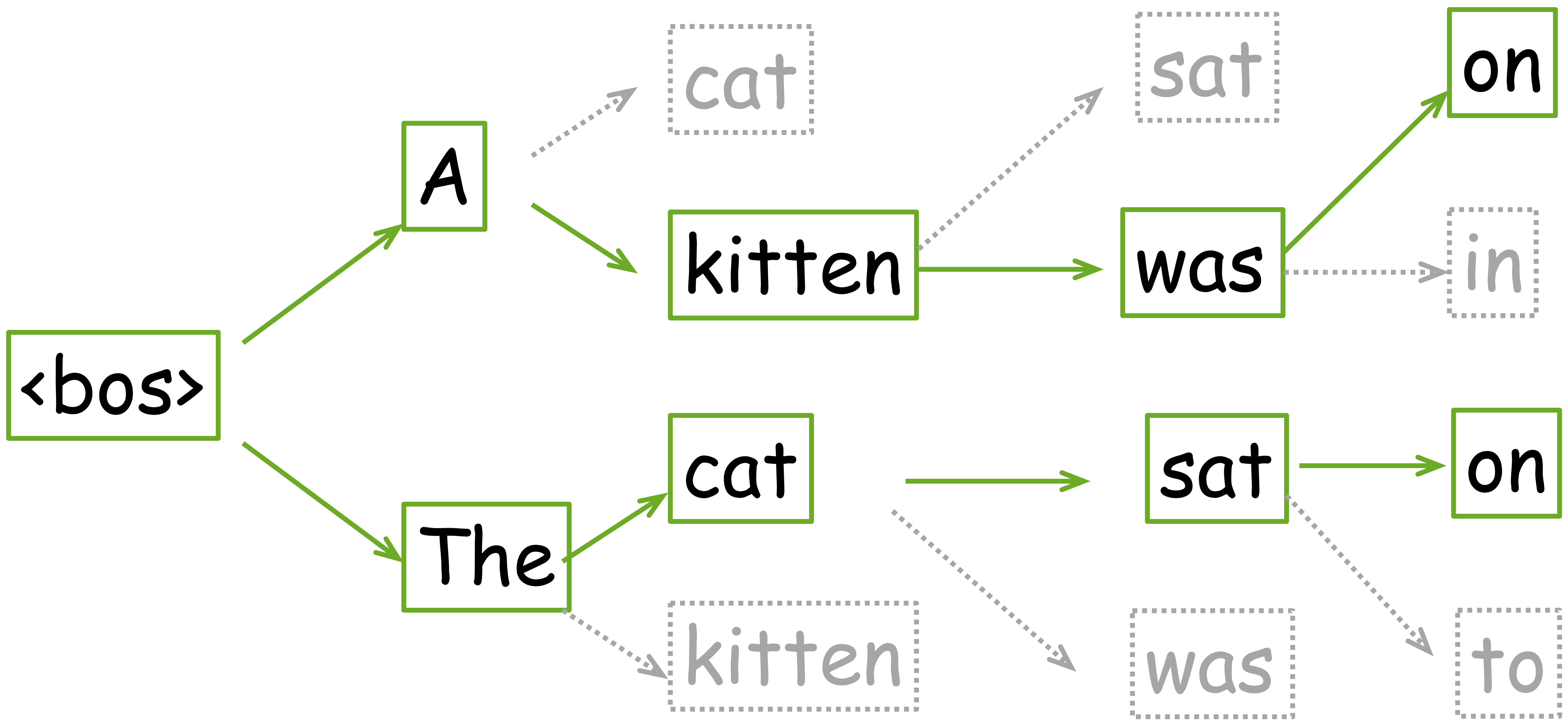
- At each step, keep several best hypotheses



Generate: **`beam_size`** most probable tokens

Beam Search

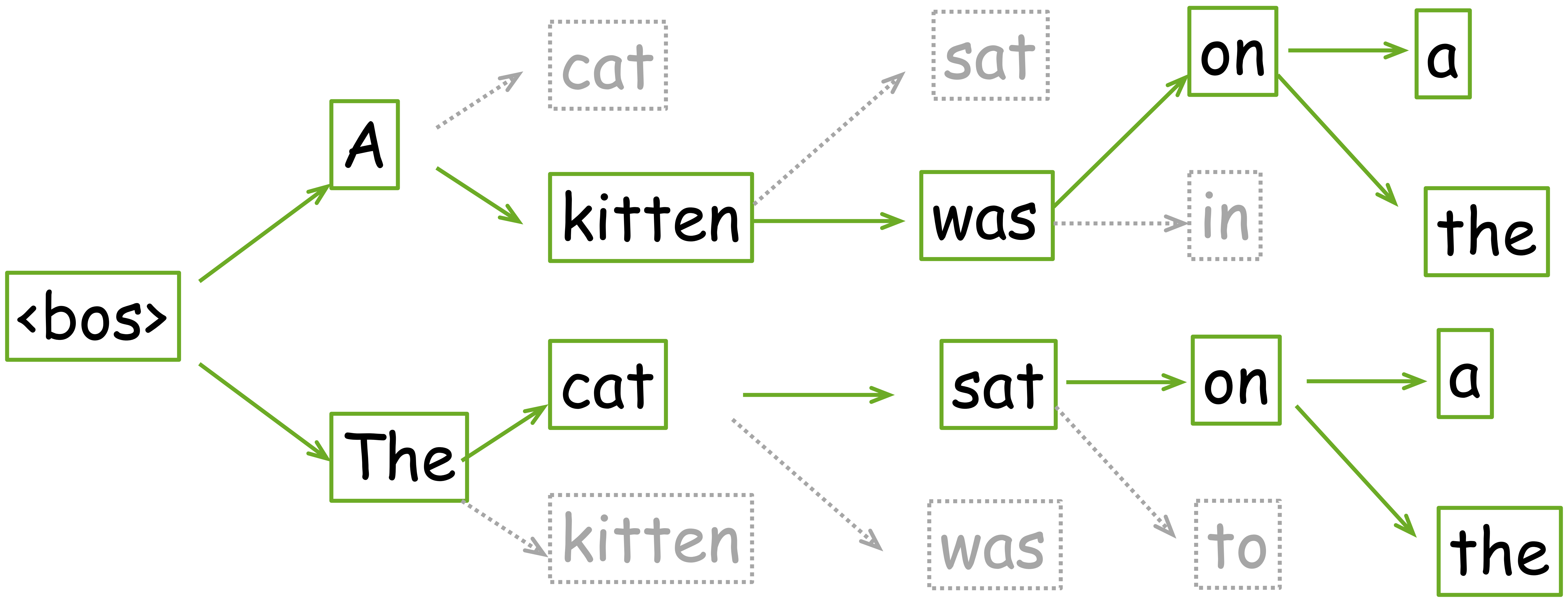
- At each step, keep several best hypotheses



Pick top `beam_size` hypos, terminate the rest

Beam Search

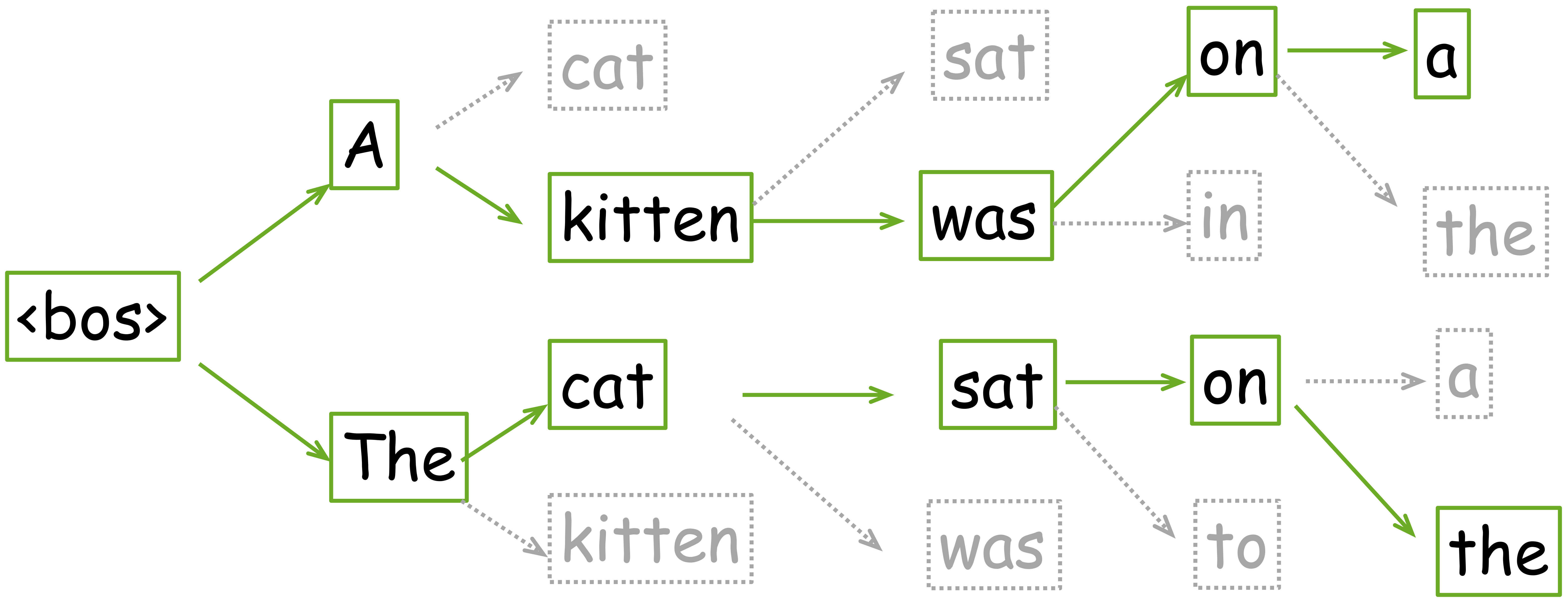
- At each step, keep several best hypotheses



Generate: `beam_size` most probable tokens

Beam Search

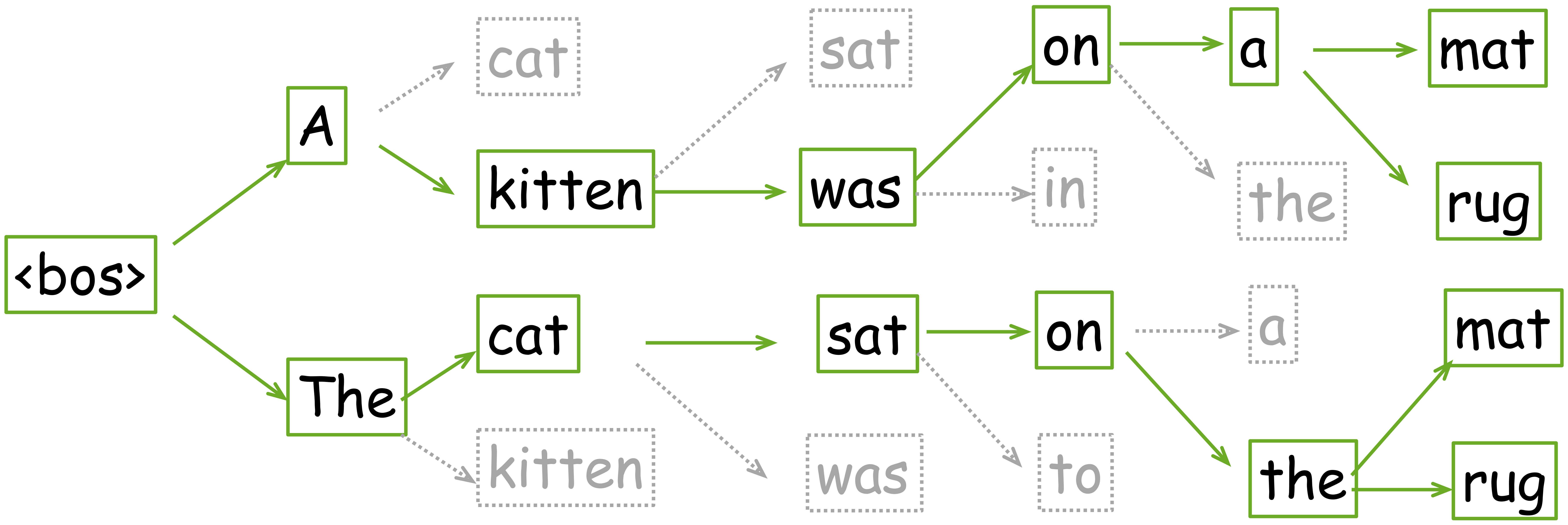
- At each step, keep several best hypotheses



Pick top **`beam_size`** hypos, terminate the rest

Beam Search

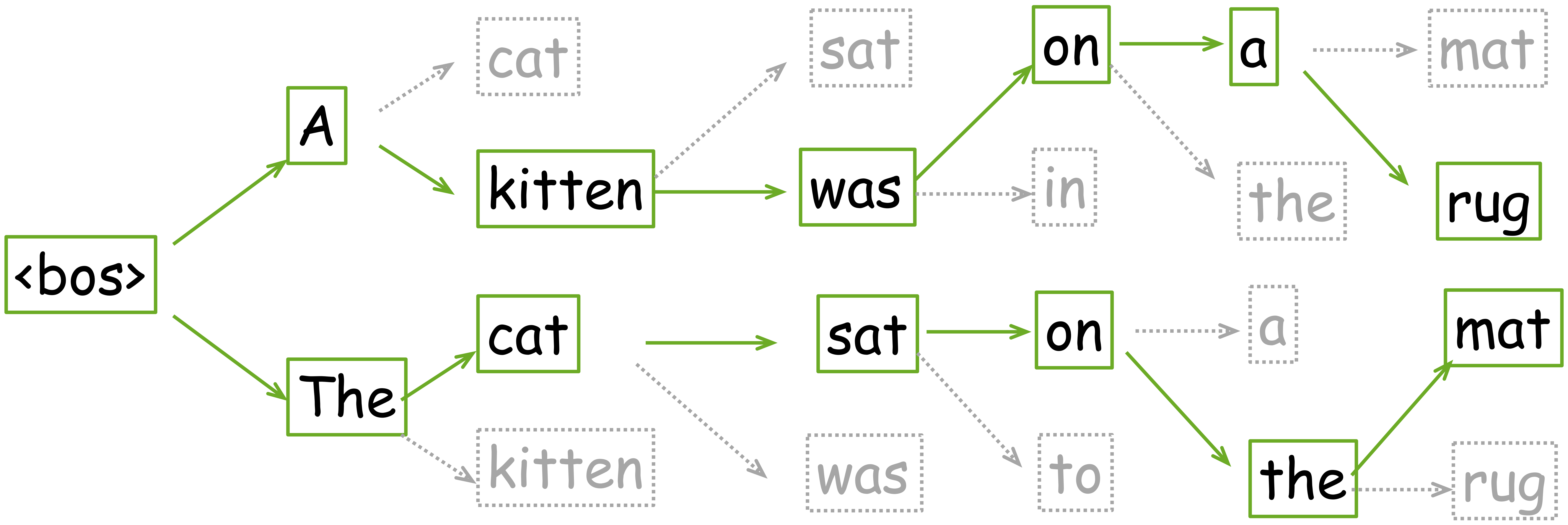
- At each step, keep several best hypotheses



Generate: **`beam_size`** most probable tokens

Beam Search

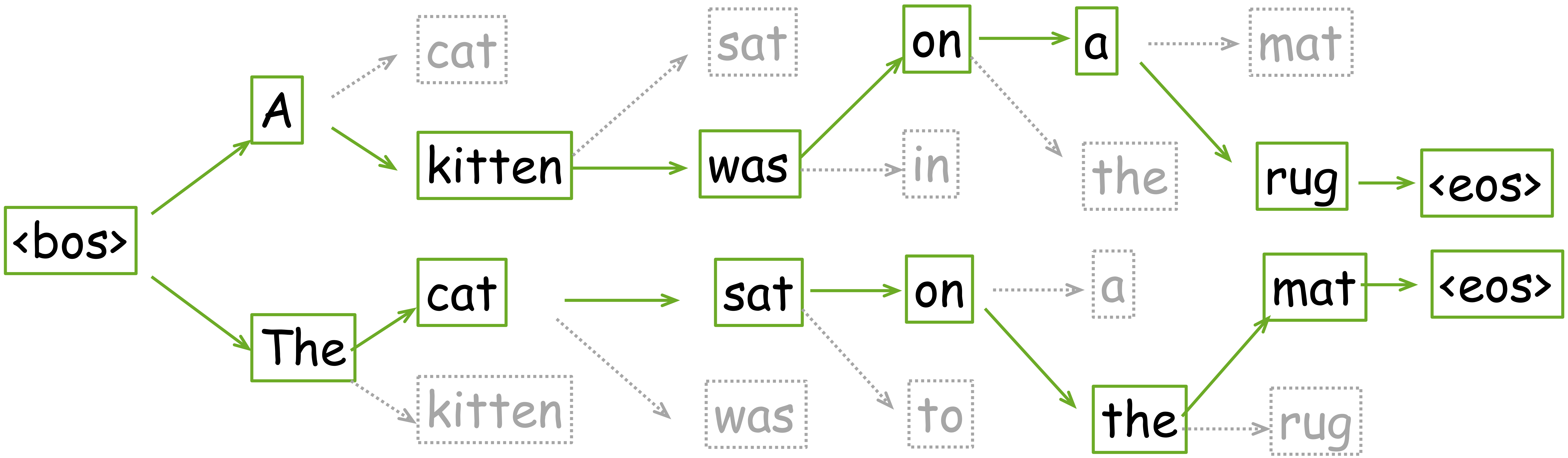
- At each step, keep several best hypotheses



Pick top `beam_size` hypos, terminate the rest

Beam Search

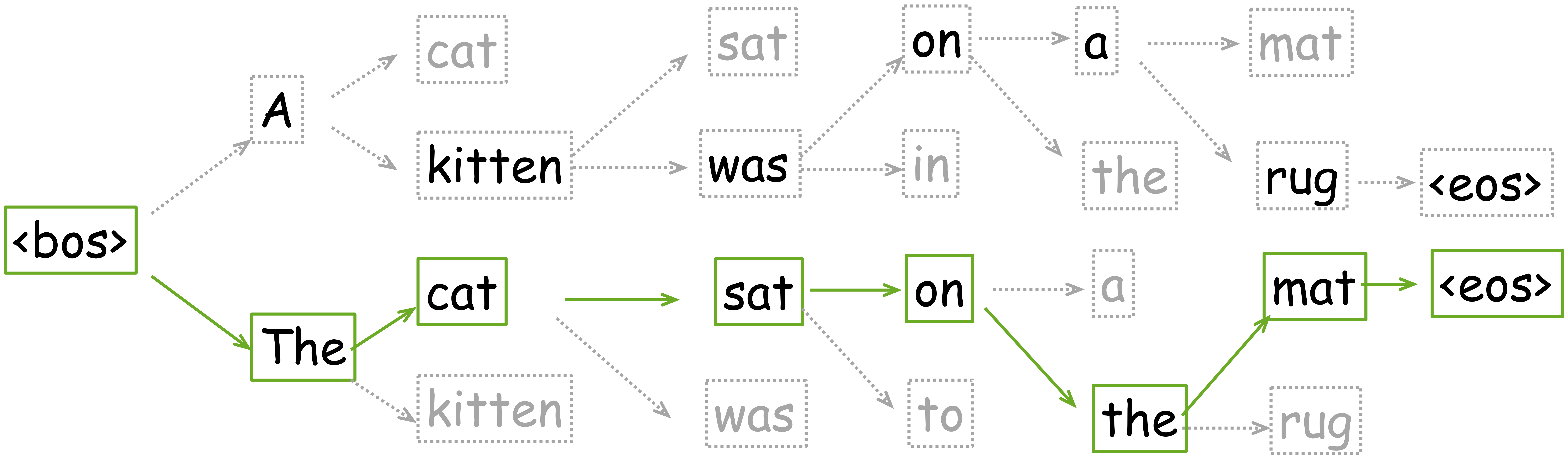
- At each step, keep several best hypotheses



All hypotheses are complete - generation ended

Beam Search

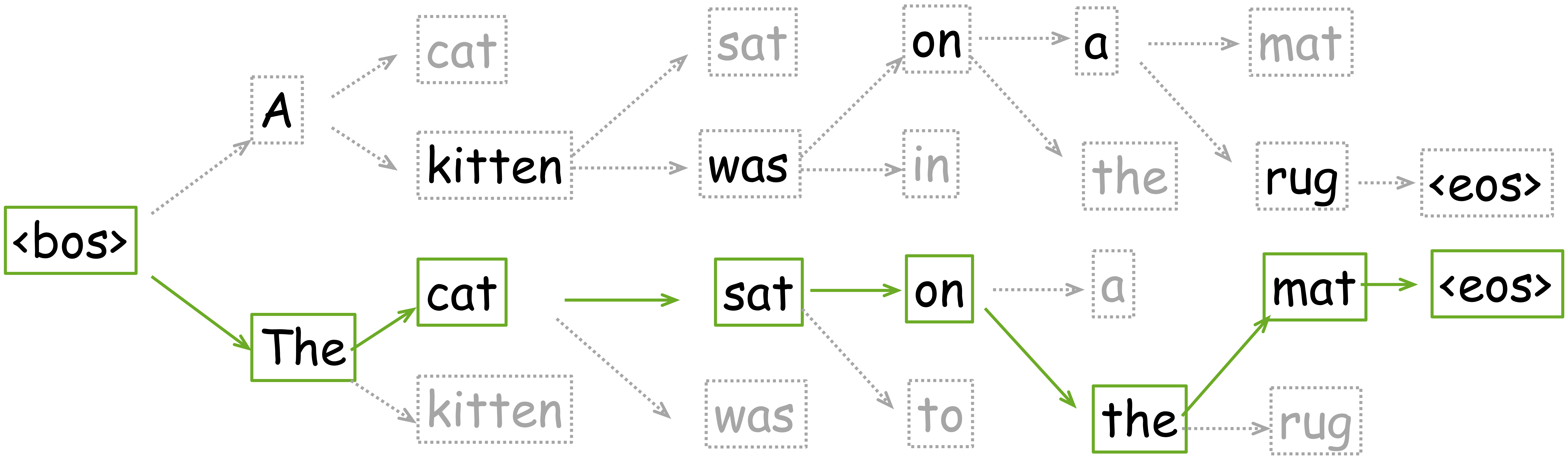
- At each step, keep several best hypotheses



Pick the hypothesis with the highest probability

Beam Search

- At each step, keep several best hypotheses



Result: **The cat sat on the mat**

What is going to happen:

- Seq2seq Basics
- Attention
- Transformer
- Subword Segmentation: BPE
- Analysis and Interpretability

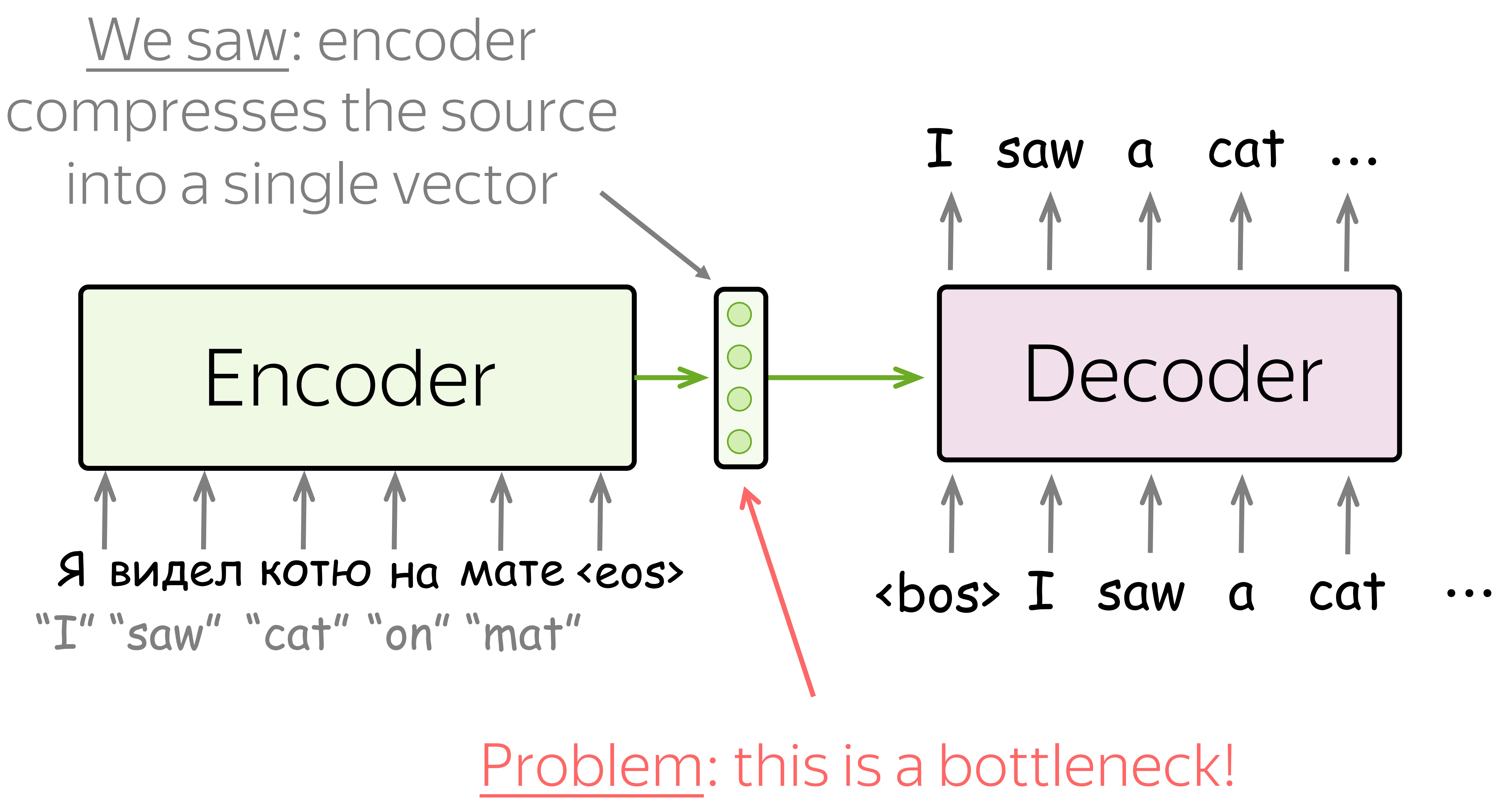
What is going to happen:

- Seq2seq Basics
- Attention
  - Why do we need it?
  - Attention: High-Level
  - Attention Score Functions
  - Models: Bahdanau vs Luong
- Transformer
- Subword Segmentation: BPE
- Analysis and Interpretability

The Problem of Fixed Encoder Representation

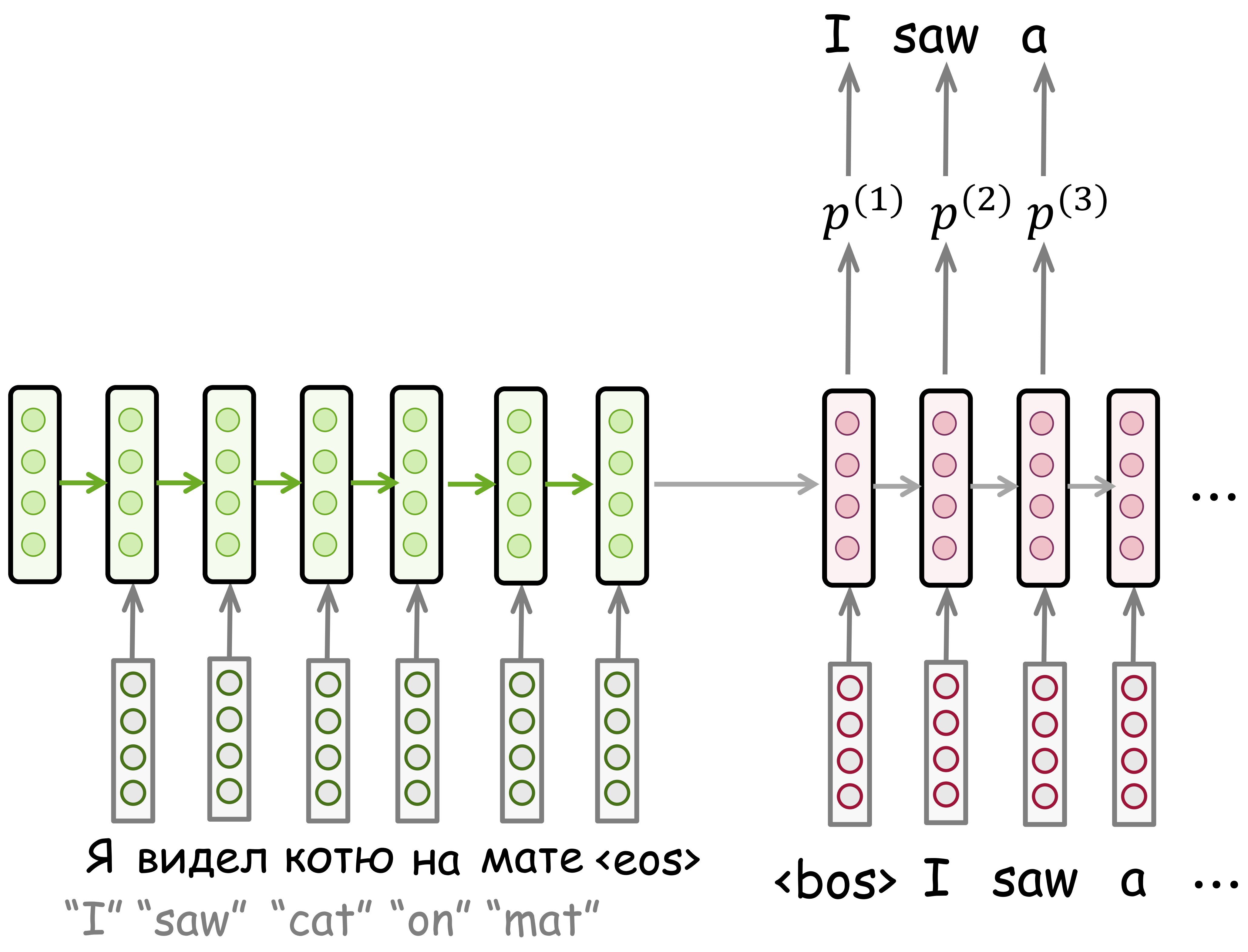
Fixed source representation is bad:

- for encoder, it is hard to compress a sentence
- for decoder, different information may be needed at different steps



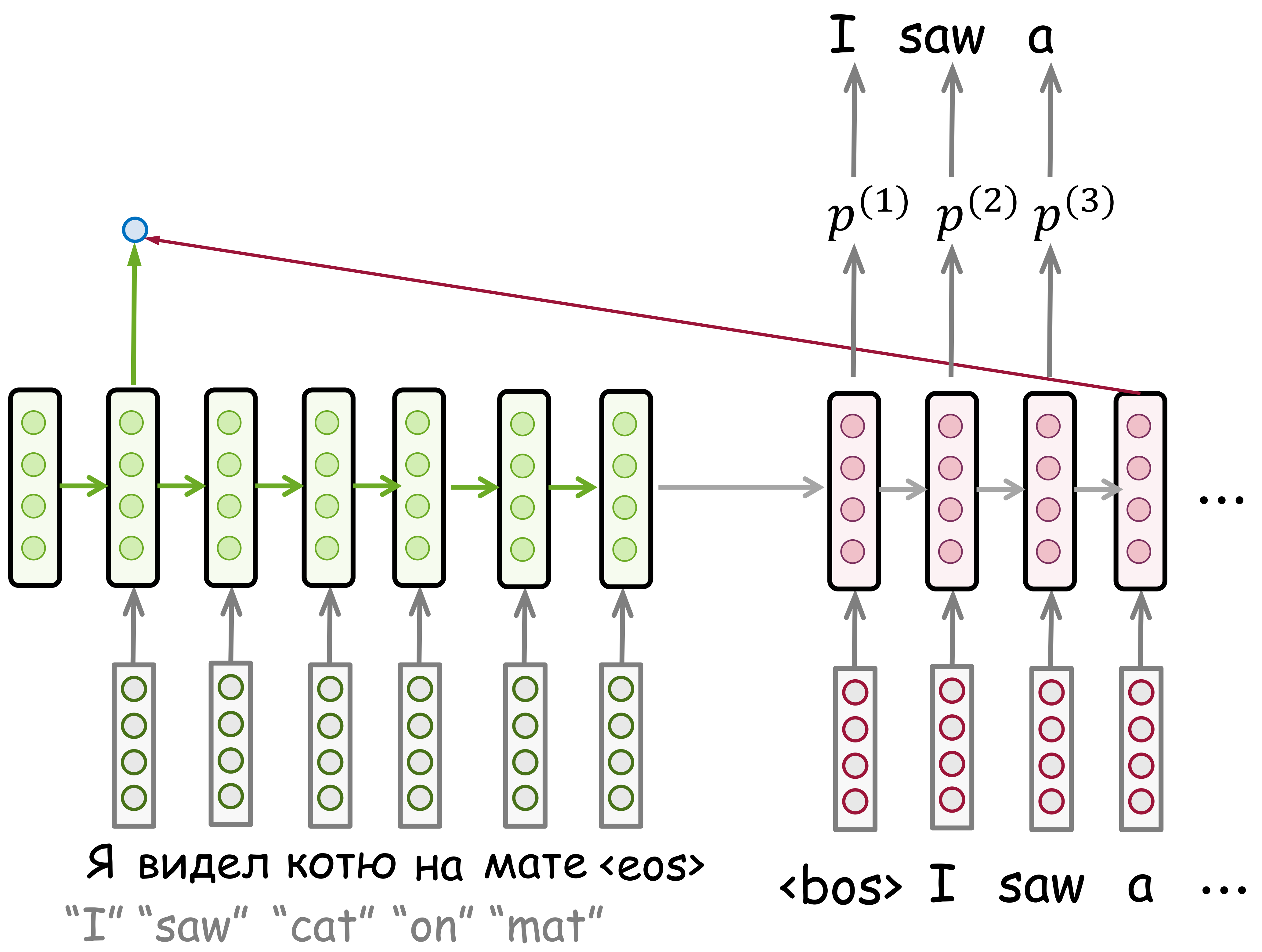
Attention: High-Level View

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



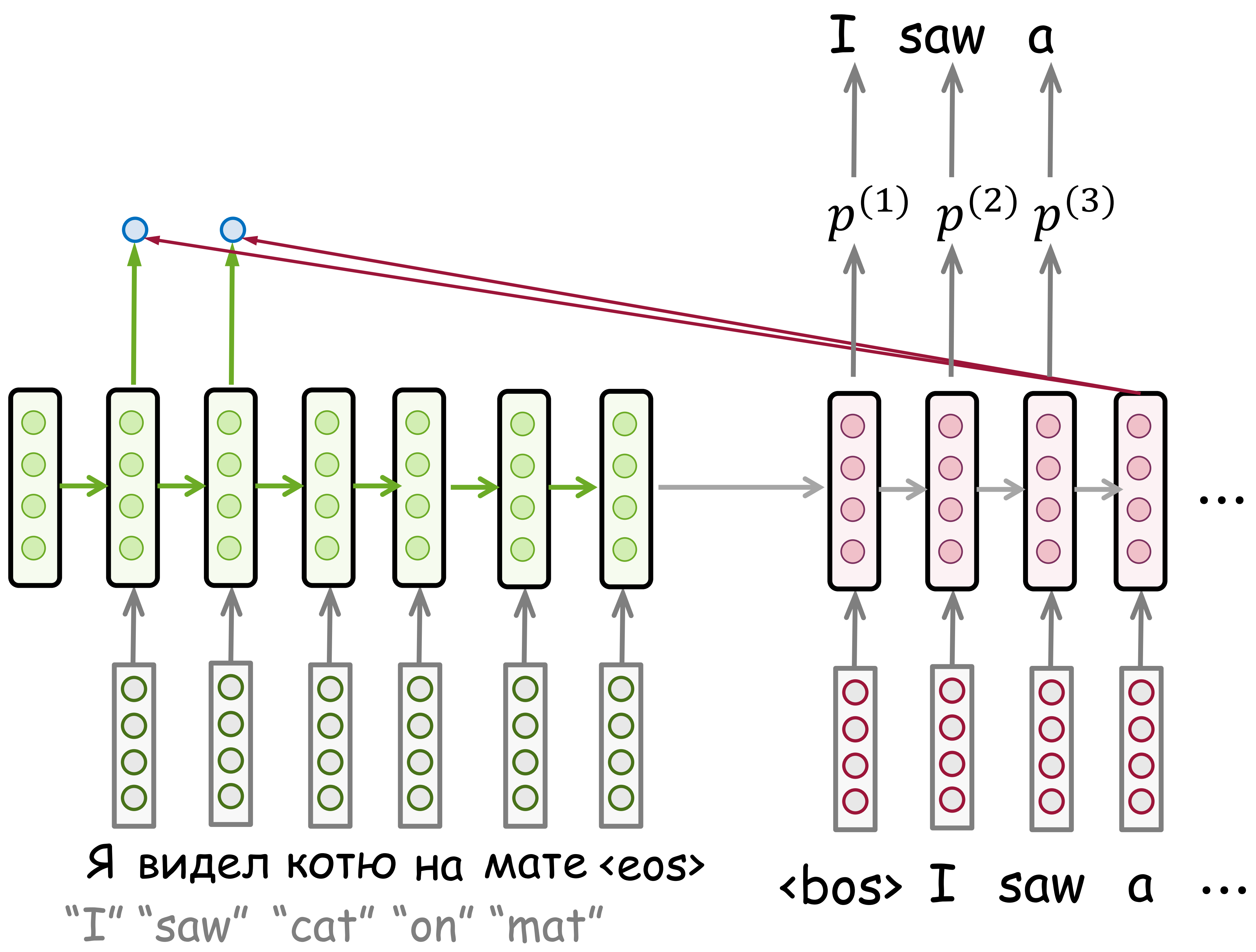
Attention: High-Level View

> <u>Attention</u>: At different steps, let a model "focus" on different parts of the input.



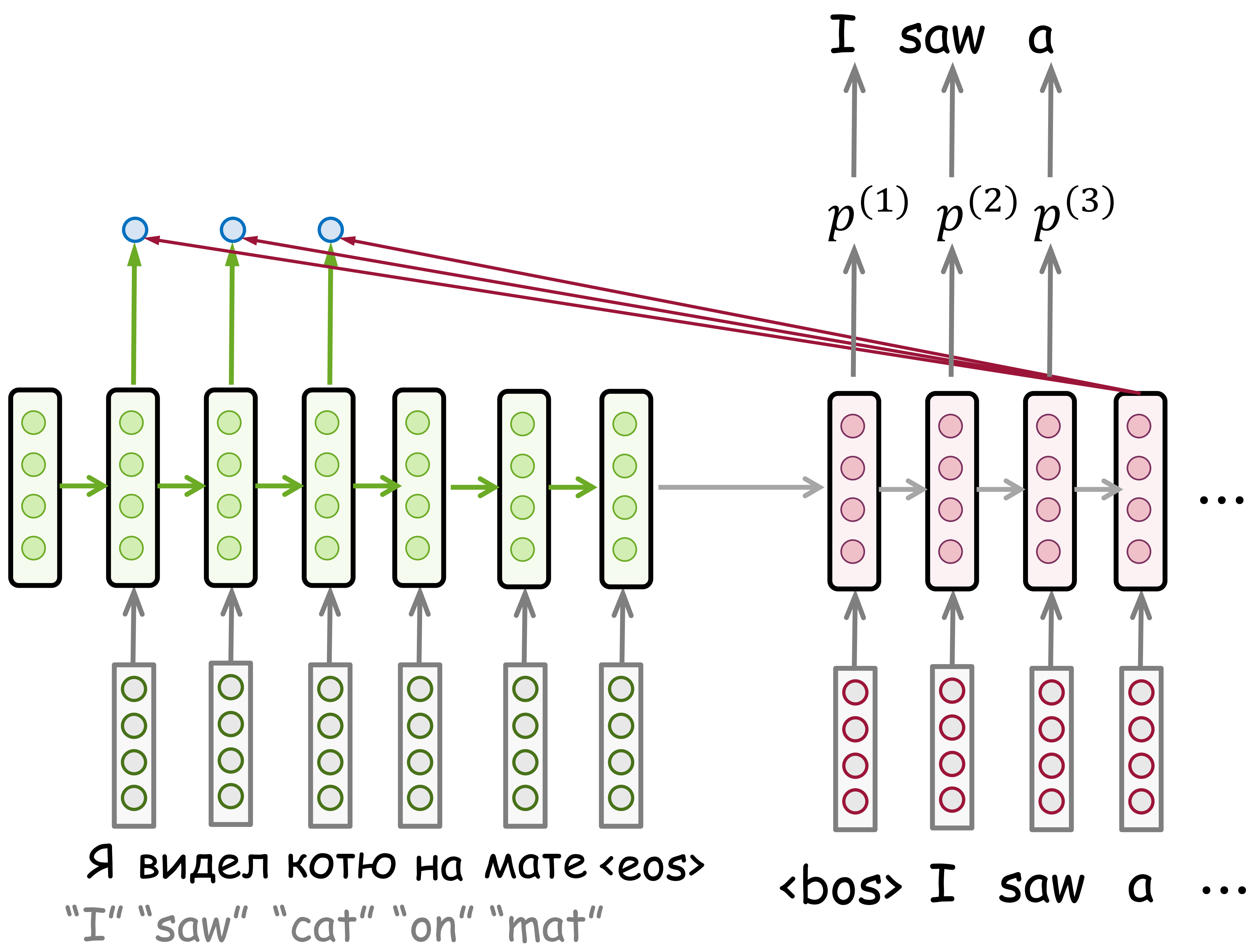
Attention: High-Level View

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



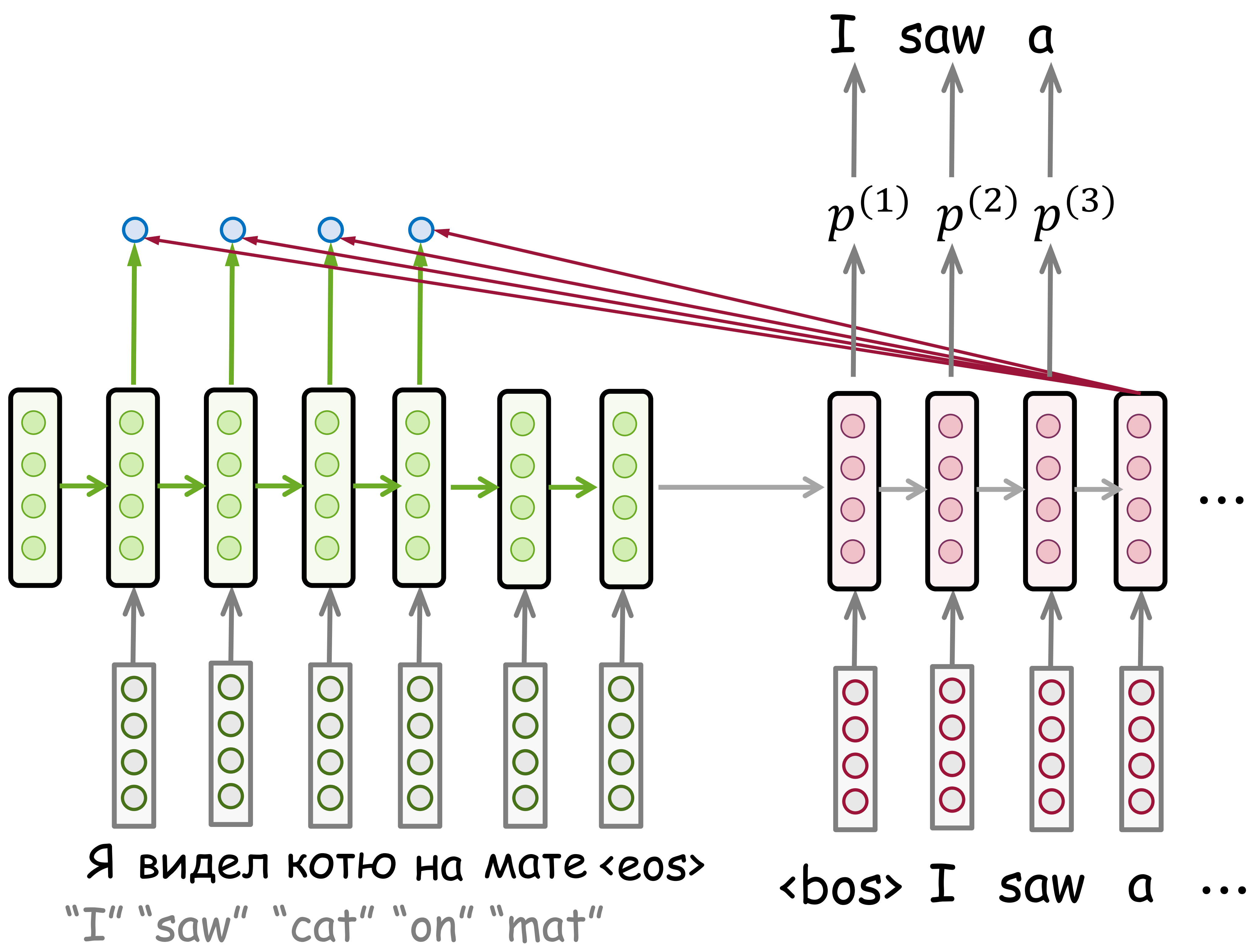
Attention: High-Level View

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



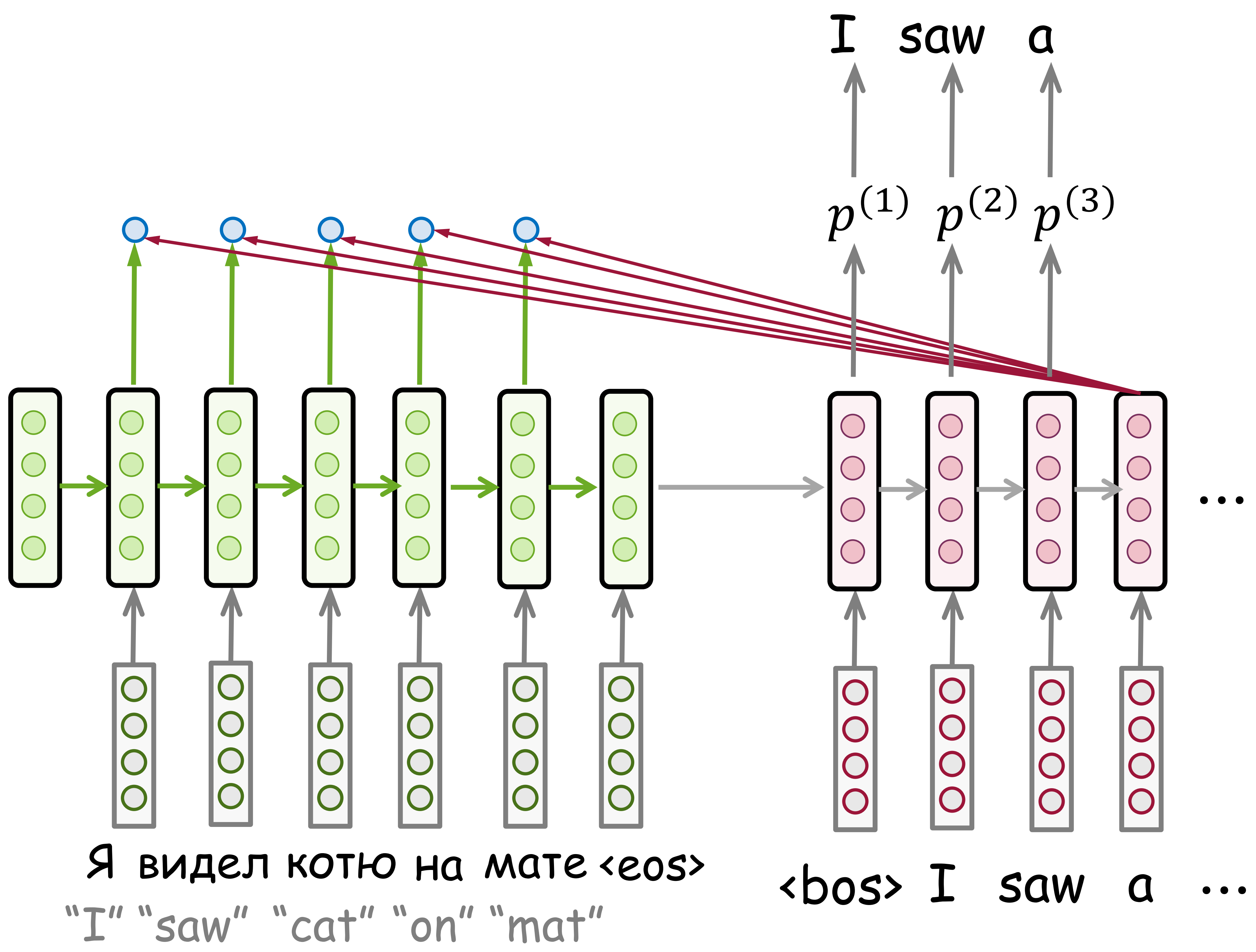
Attention: High-Level View

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



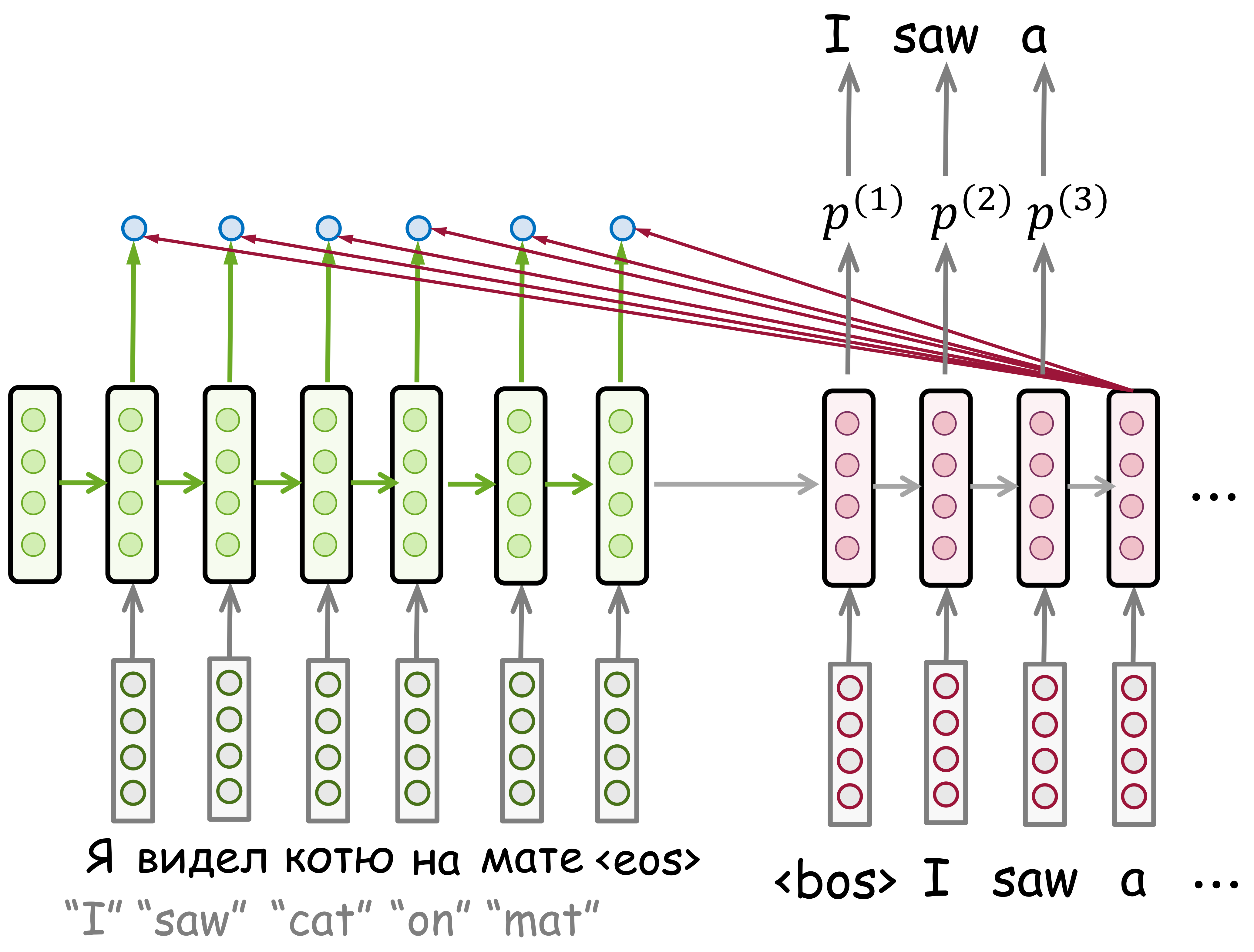
Attention: High-Level View

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



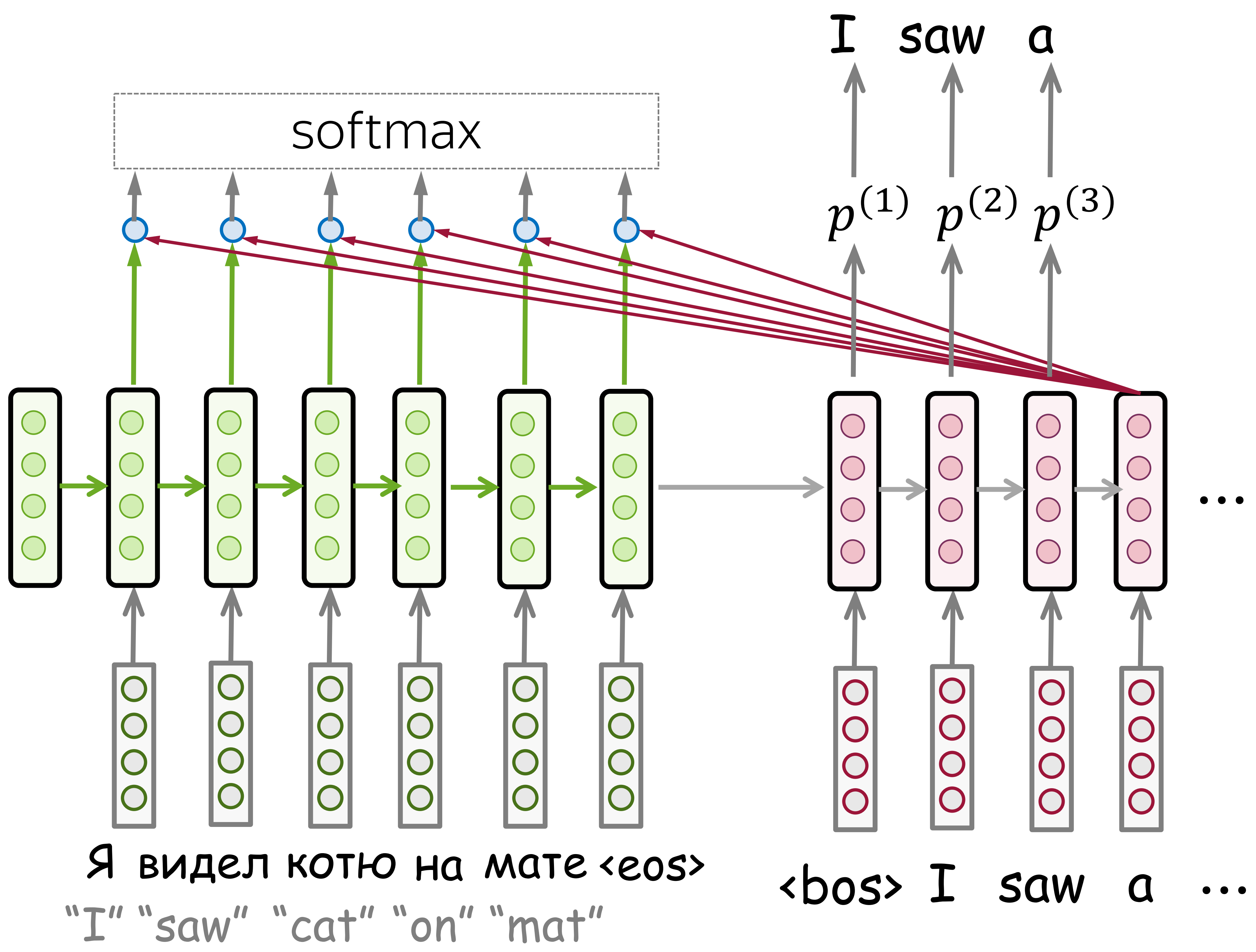
Attention: High-Level View

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



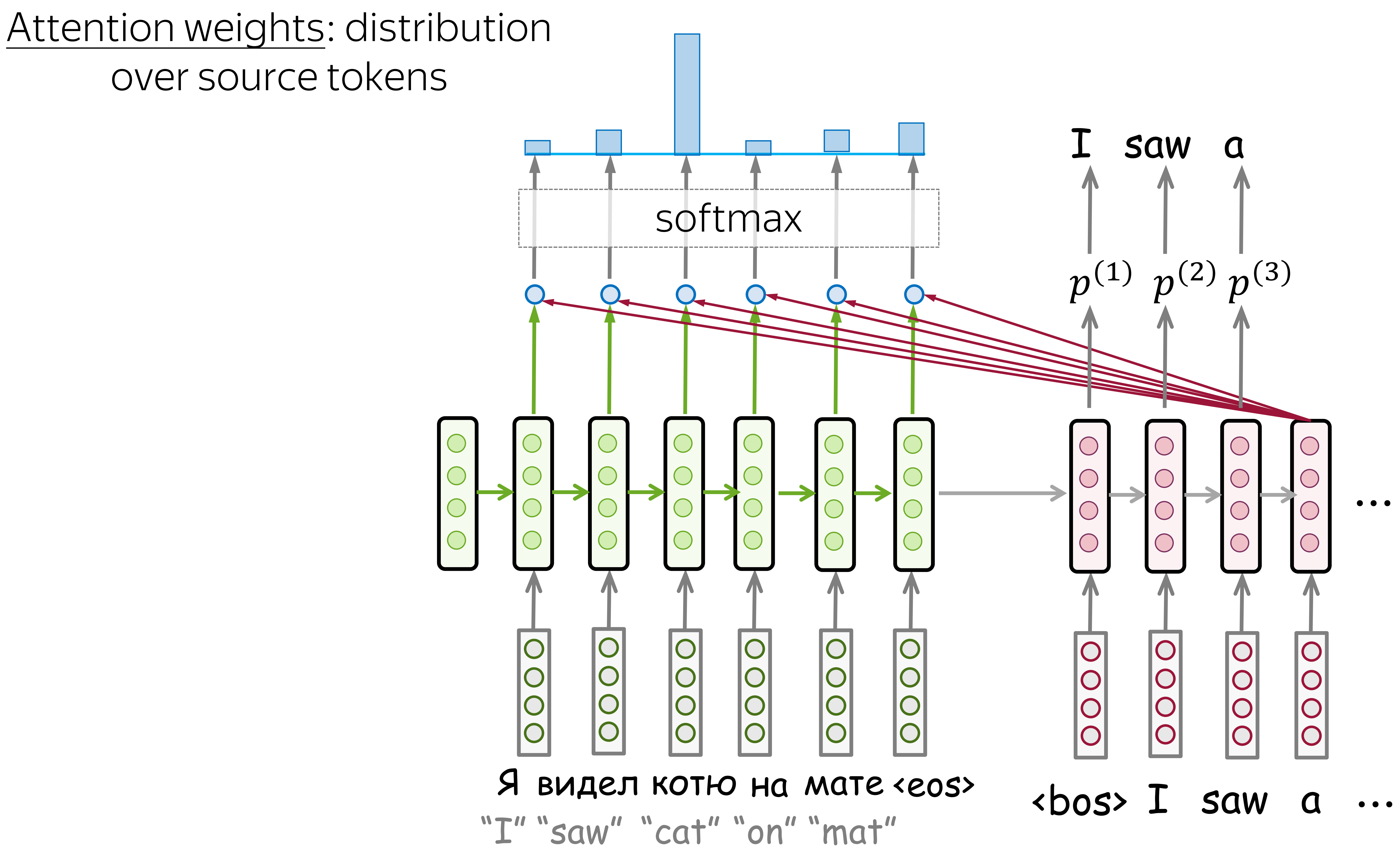
Attention: High-Level View

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



Attention: High-Level View

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

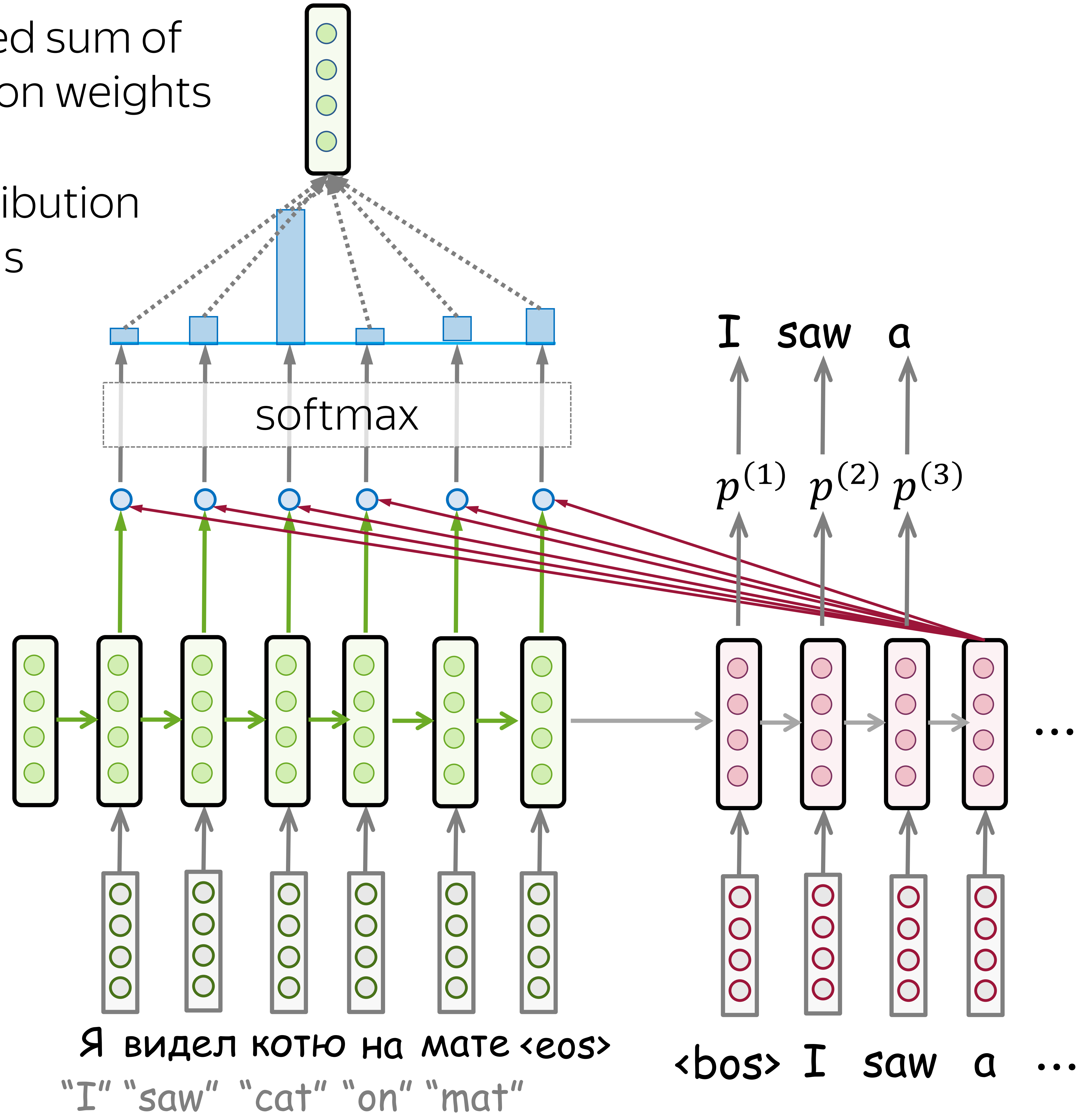


Attention: High-Level View

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

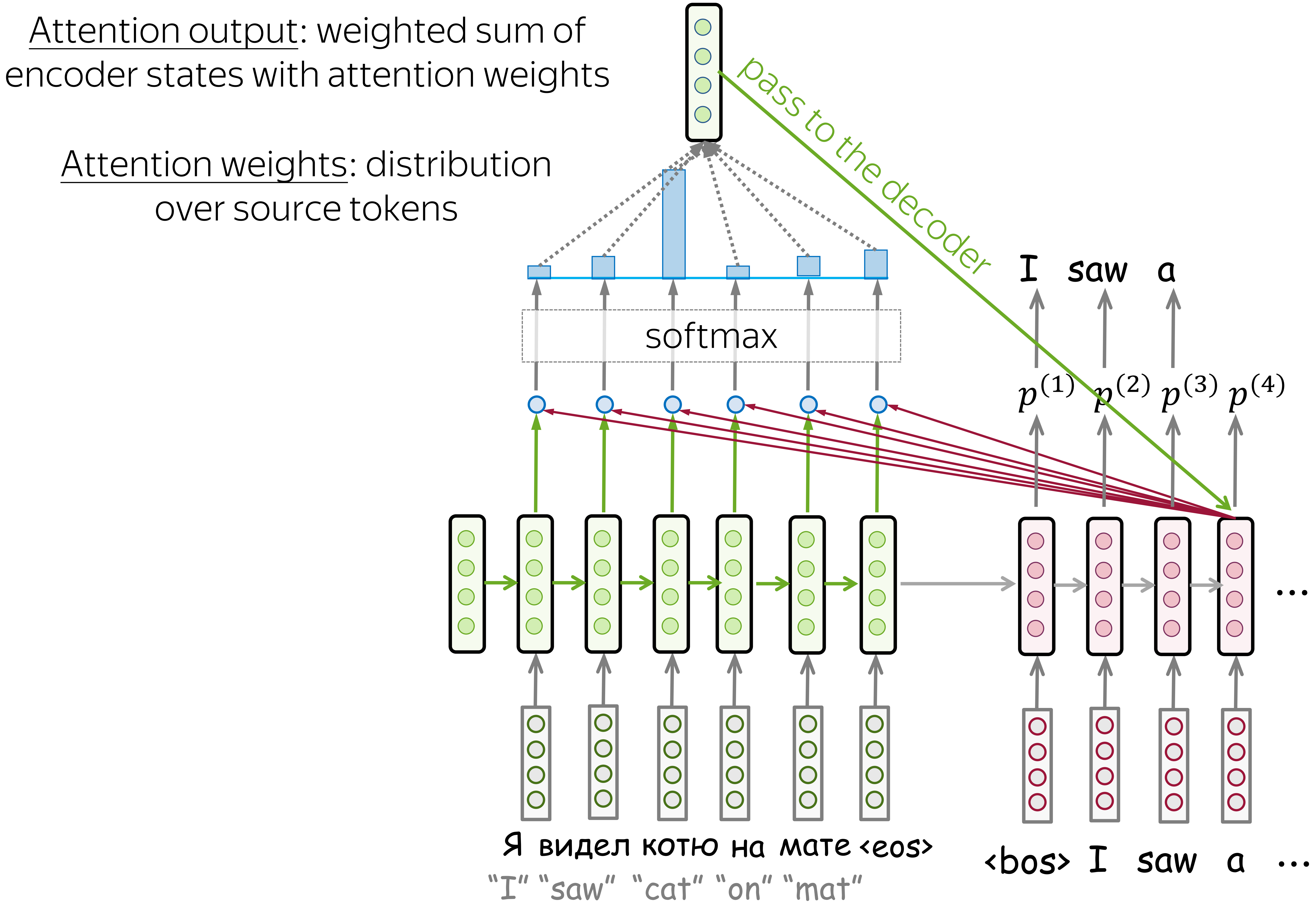
Attention output: weighted sum of encoder states with attention weights

Attention weights: distribution over source tokens



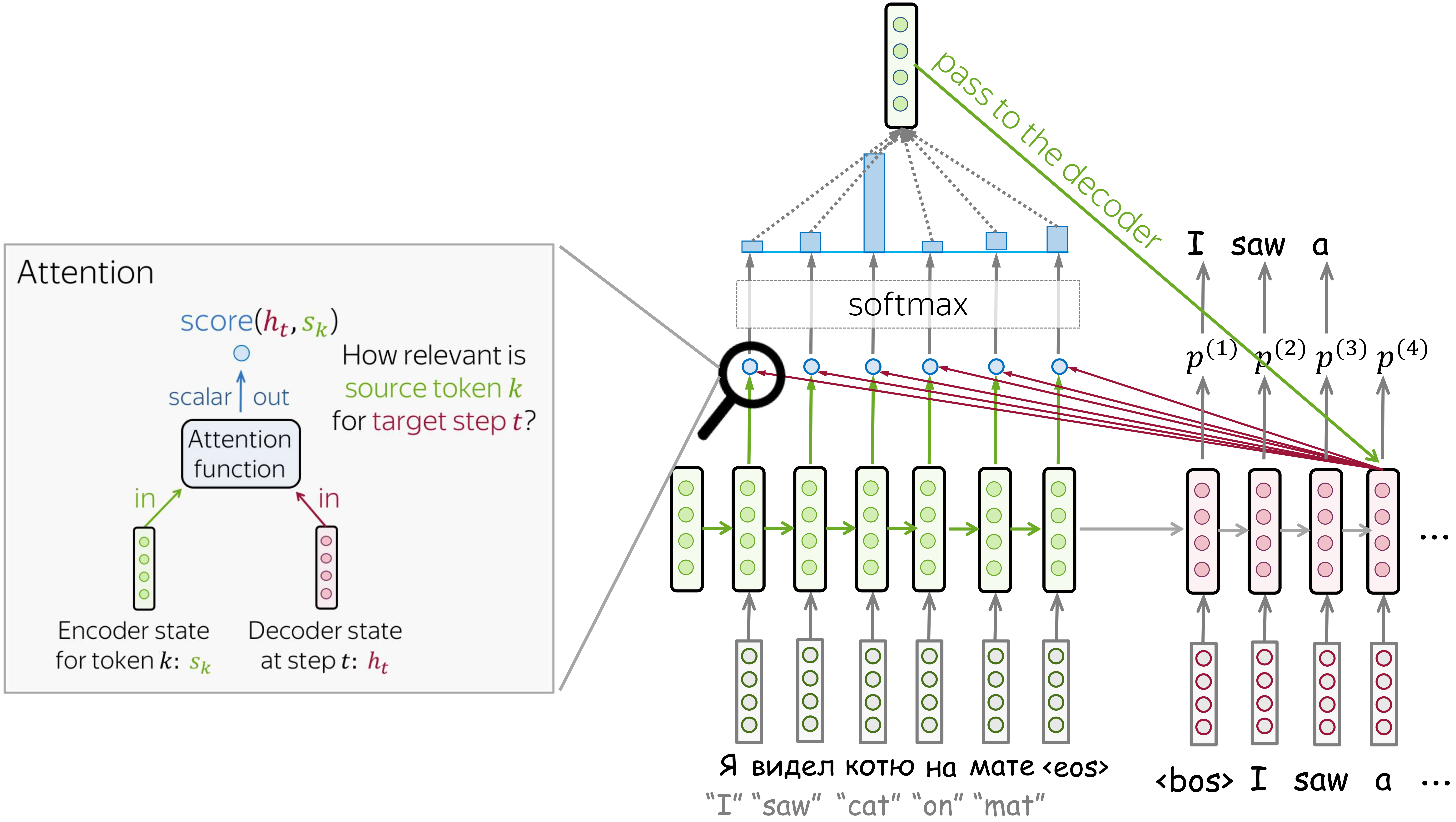
Attention: High-Level View

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



Attention: High-Level View

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



Computation Pipeline

Attention input $s_1, s_2, \ldots, s_m$ h_t

all encoder states one decoder state

Computation Pipeline

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

↑

Attention input $s_1, s_2, \dots, s_m$ h_t

all encoder states one decoder state

Computation Pipeline

Attention scores

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

"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

Computation Pipeline

Attention weights

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

(softmax)

Attention scores

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

"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

Computation Pipeline

Attention weights

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

“attention weight for source token k at decoder step t”

(softmax)

Attention scores

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

“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

Computation Pipeline

Attention output

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

(weighted sum)

Attention weights

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

“attention weight for source token k at decoder step t”

(softmax)

Attention scores

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

“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

Computation Pipeline

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

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

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

(softmax)

Attention scores

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

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

Attention input

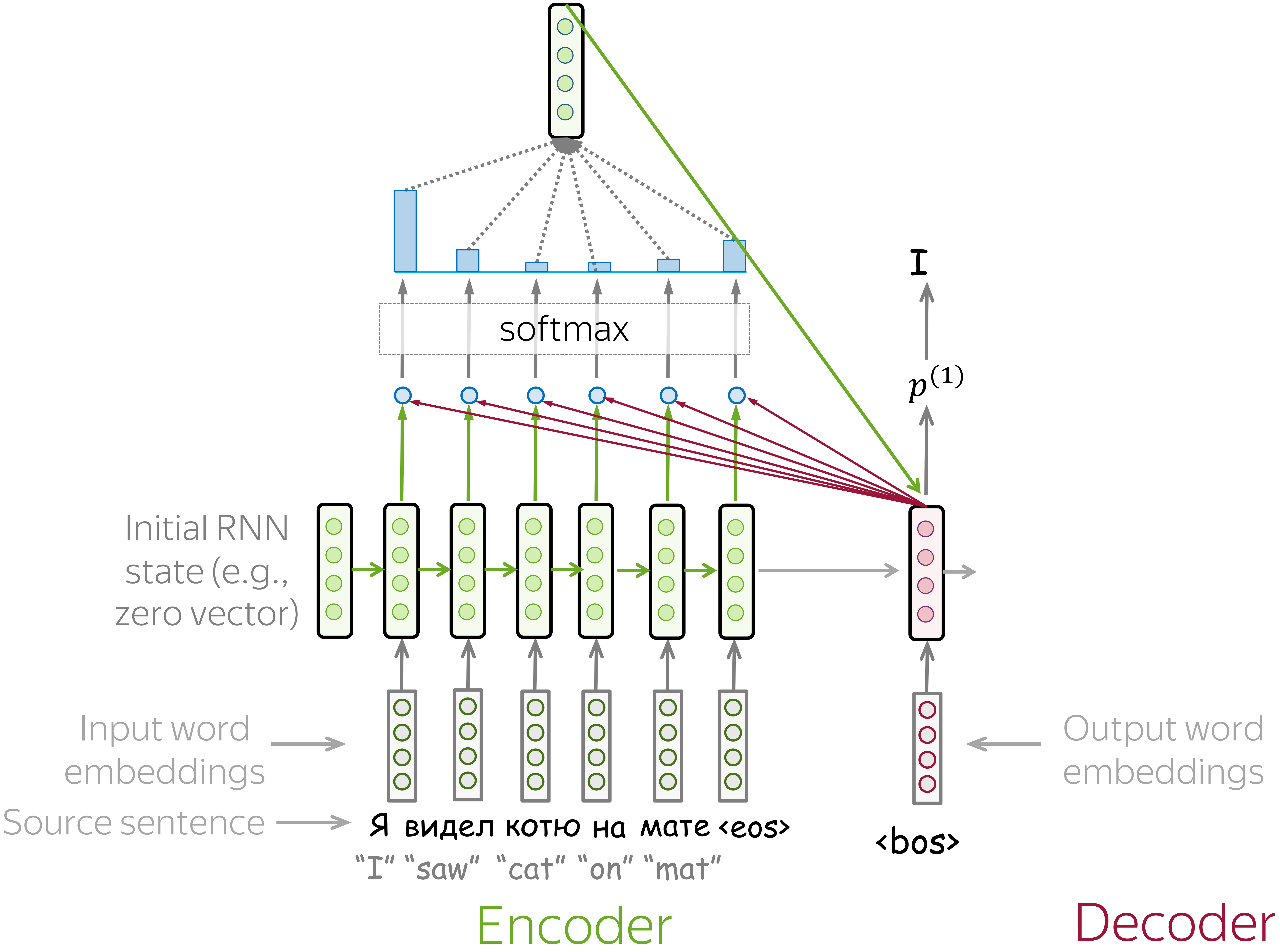
$s_1, s_2, \ldots, s_m$

all encoder states

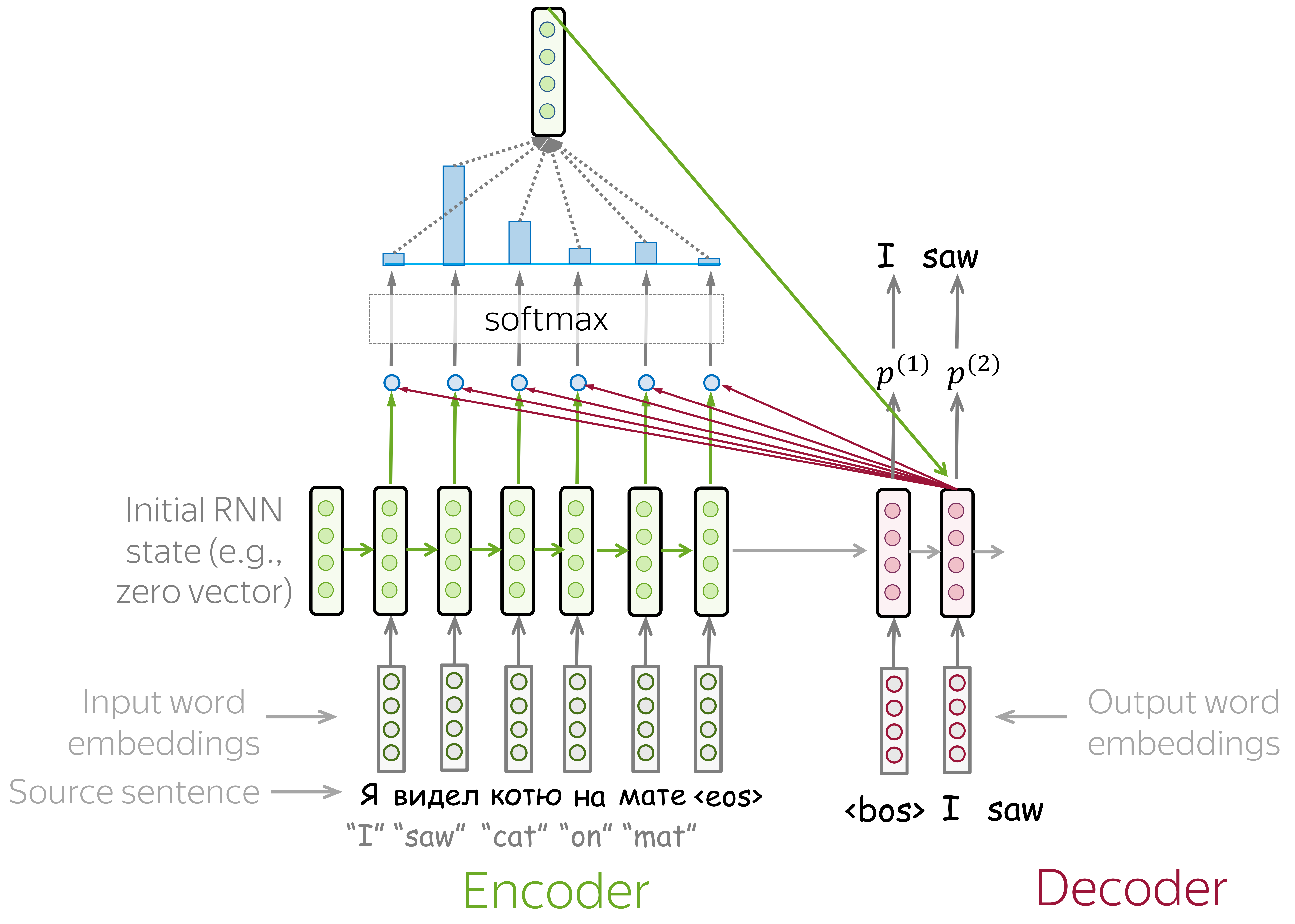
h_t

one decoder state

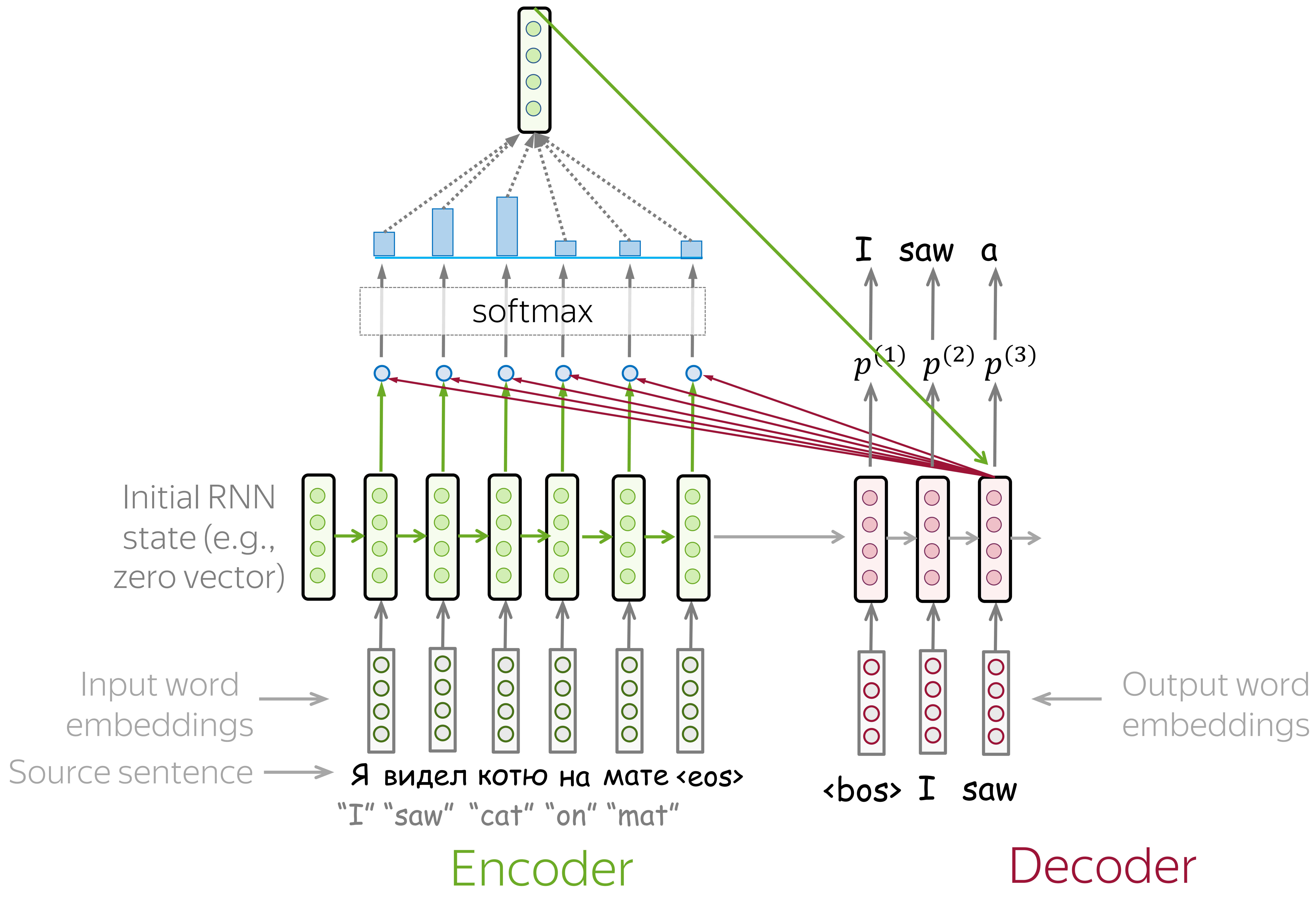
Model Learns to Pick Relevant Tokens



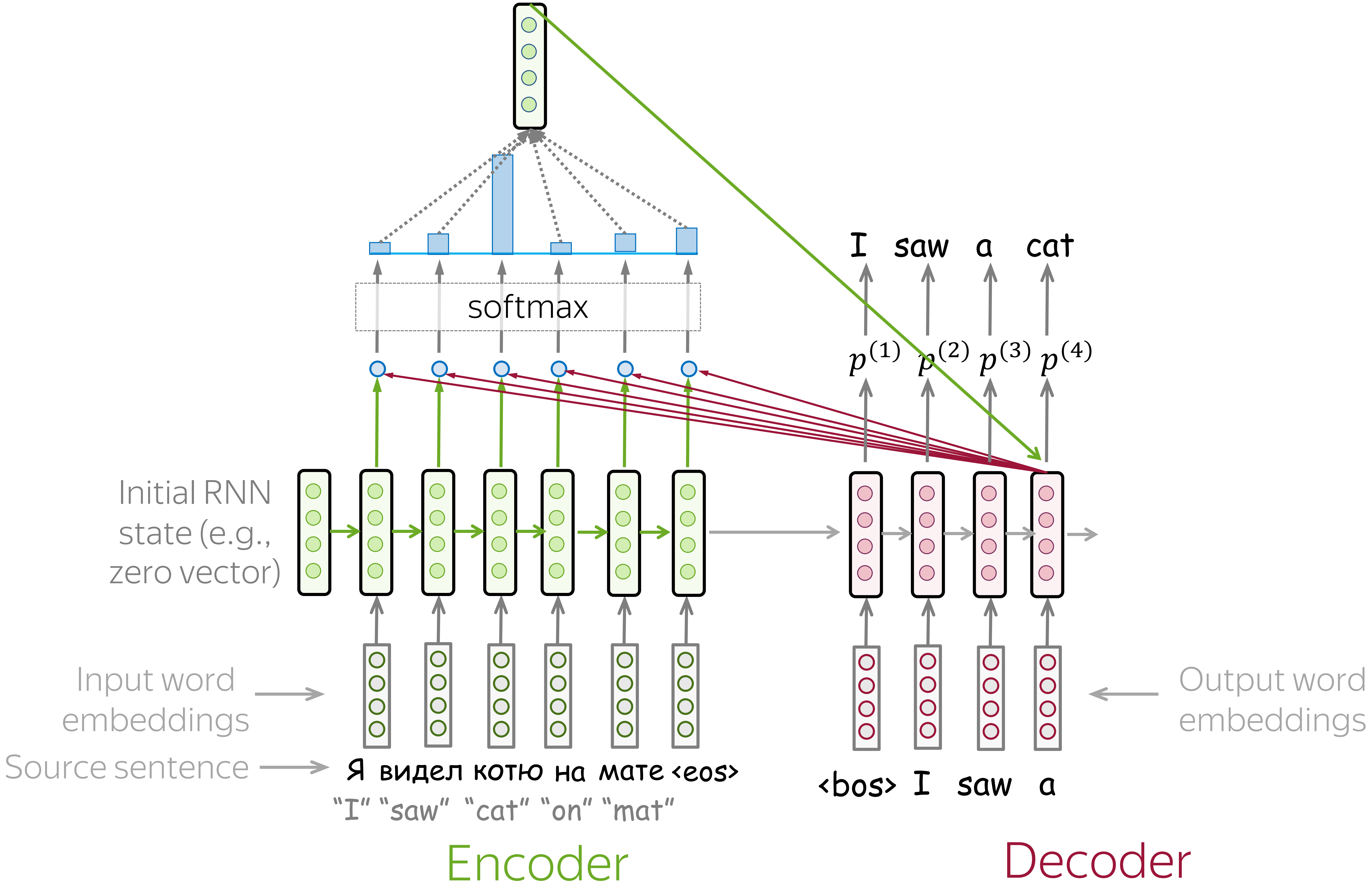
Model Learns to Pick Relevant Tokens



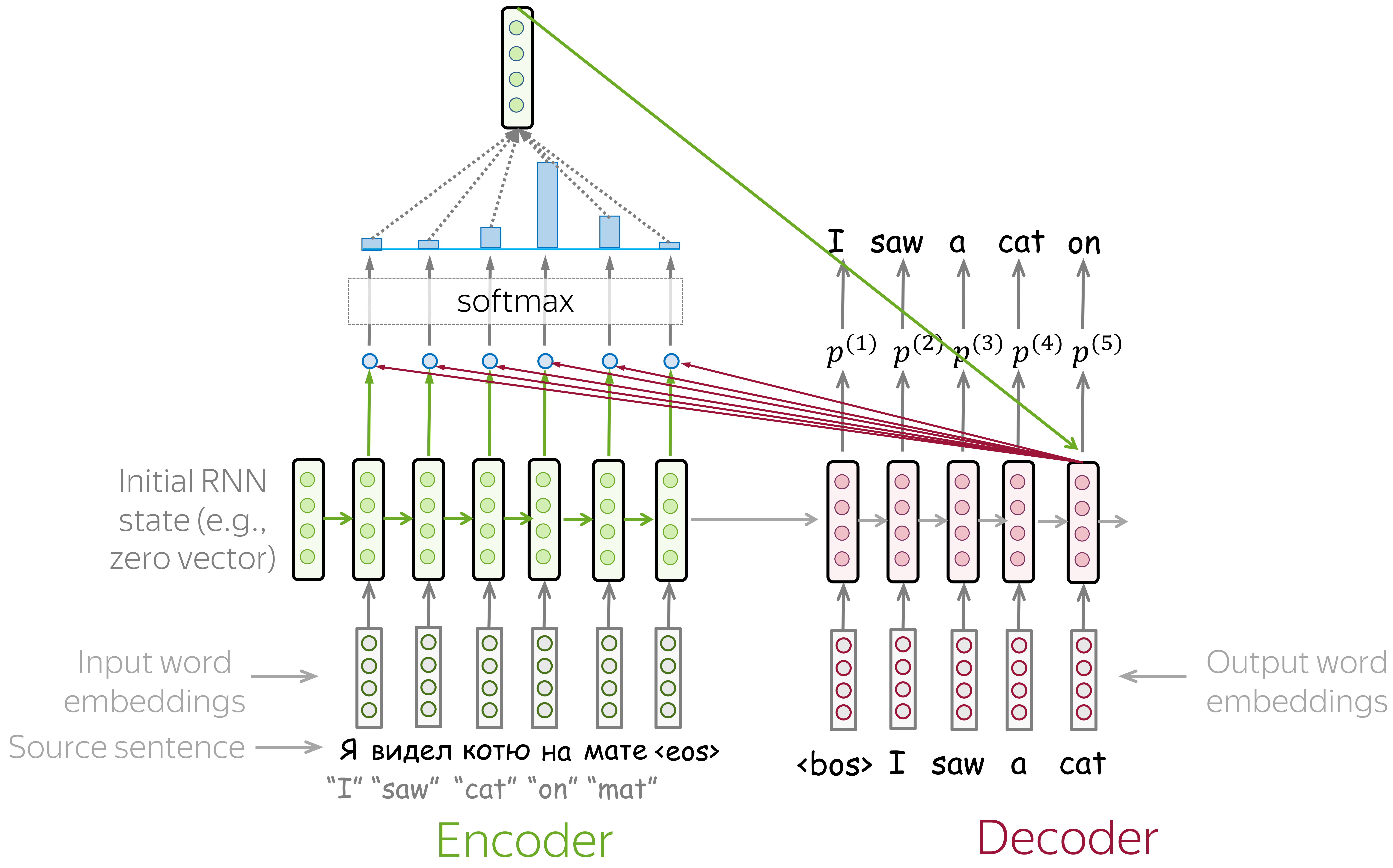
Model Learns to Pick Relevant Tokens



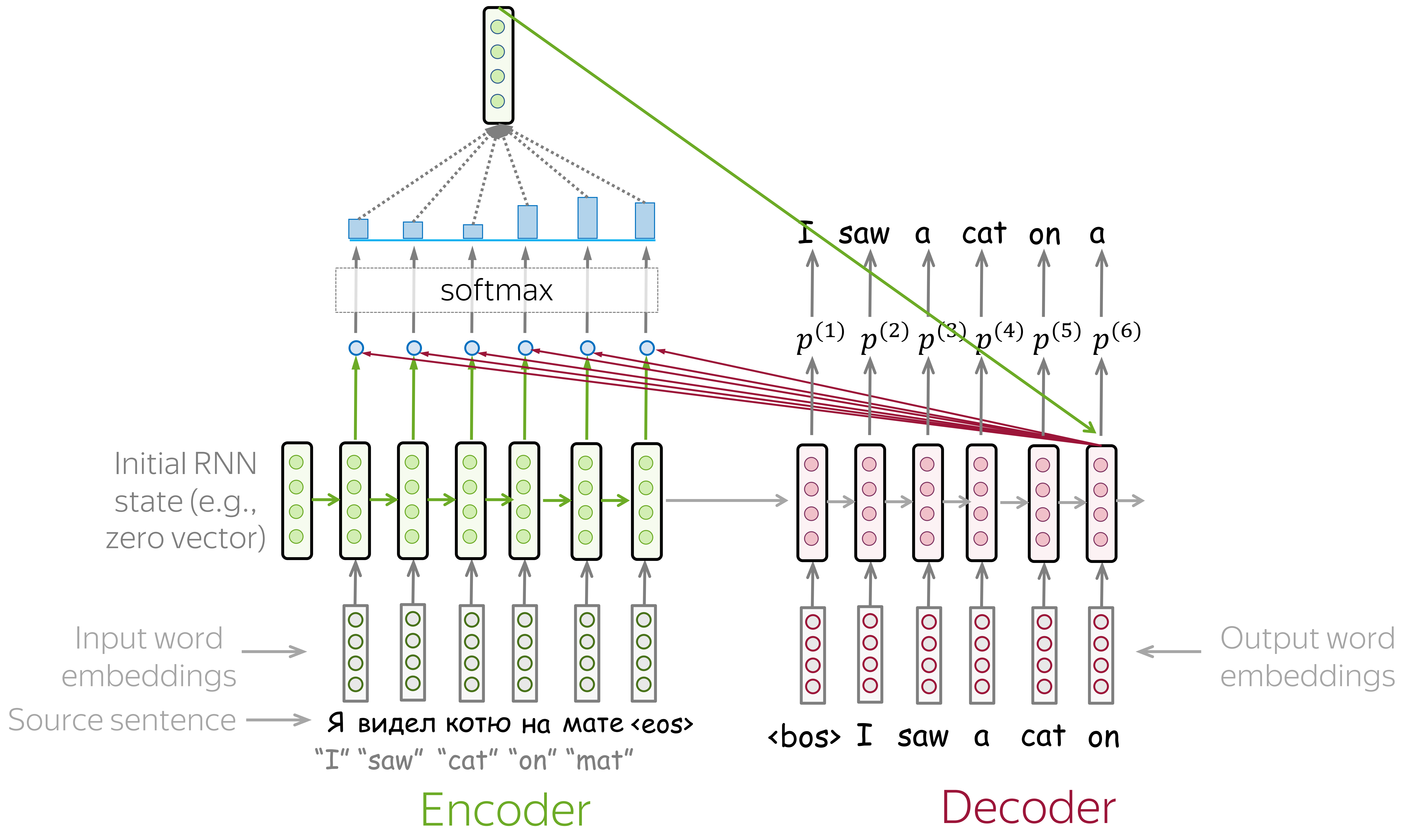
Model Learns to Pick Relevant Tokens



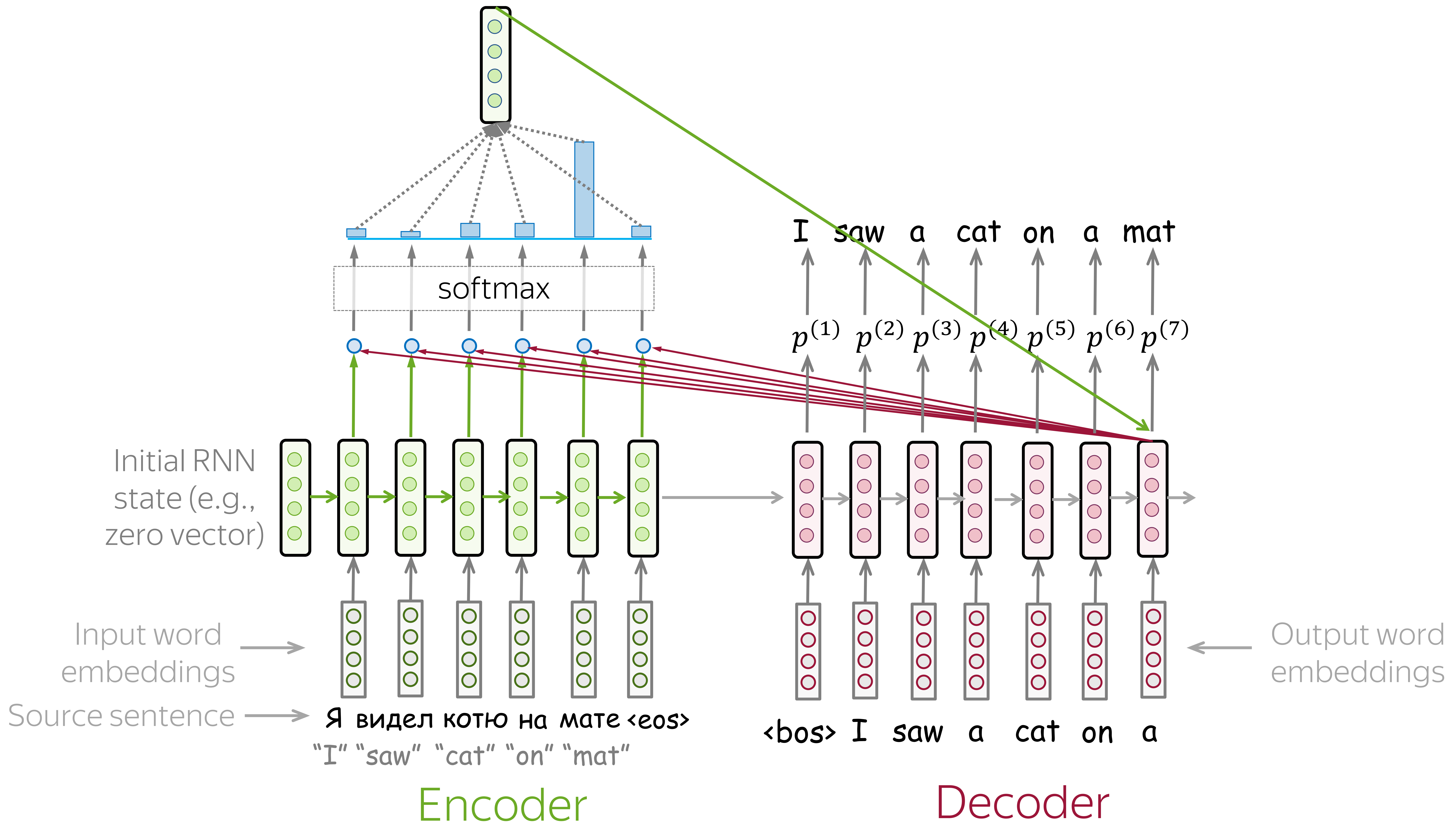
Model Learns to Pick Relevant Tokens



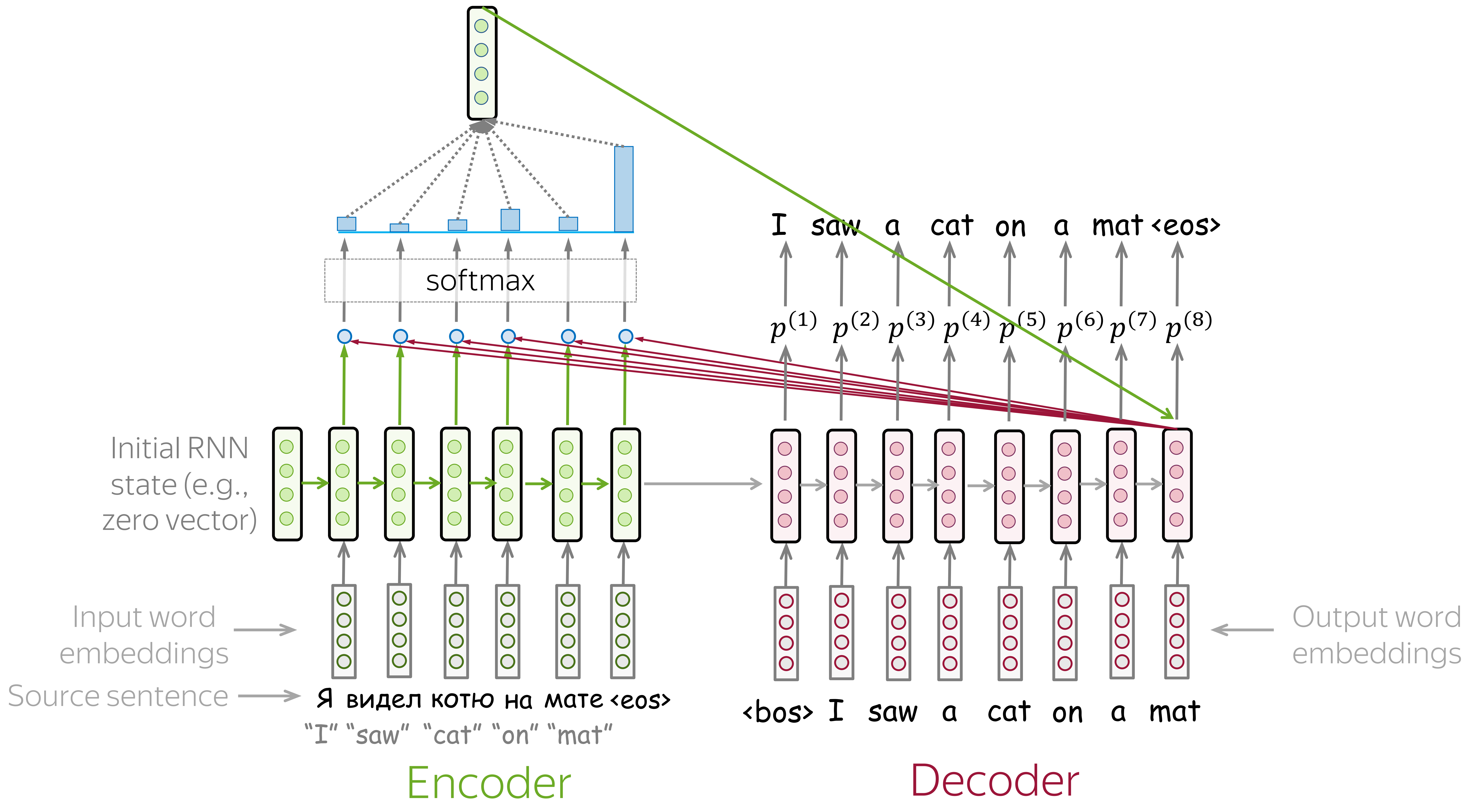
Model Learns to Pick Relevant Tokens



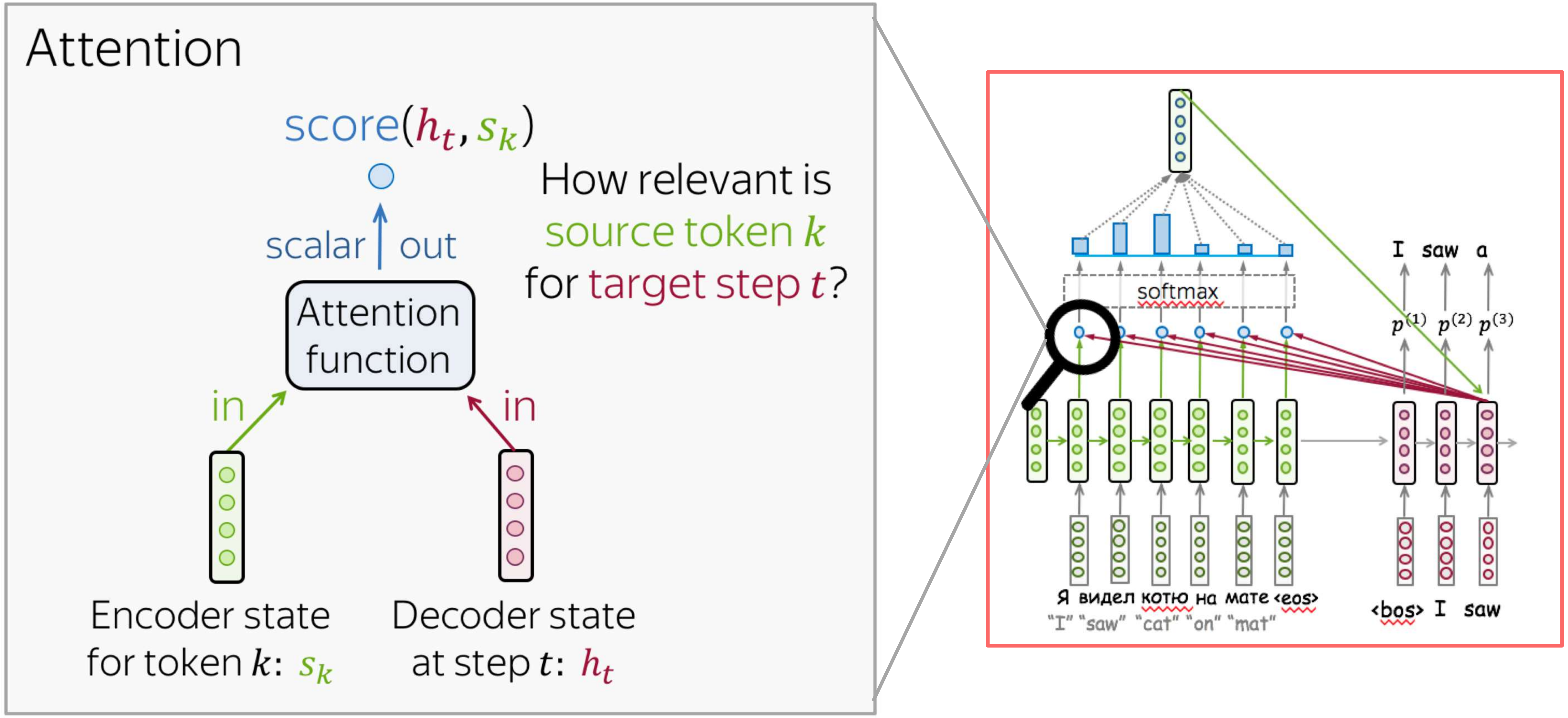
Model Learns to Pick Relevant Tokens



Model Learns to Pick Relevant Tokens

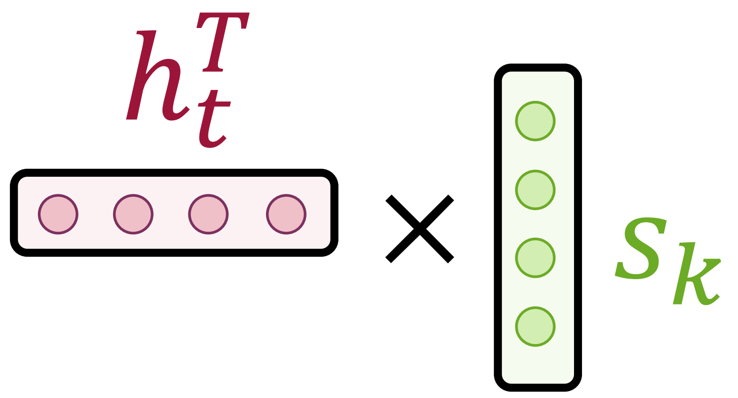


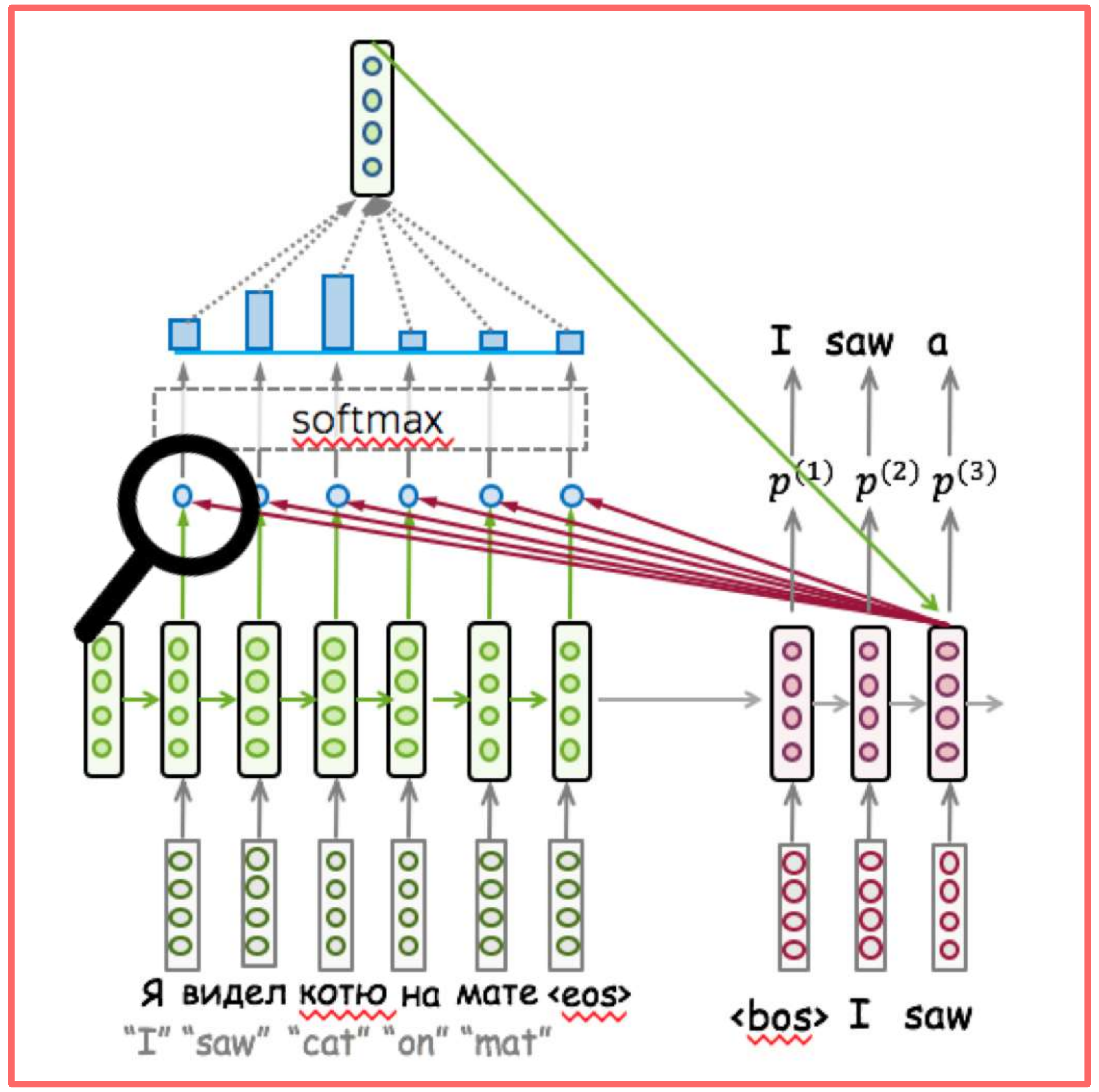
Attention Score Functions



Attention Score Functions

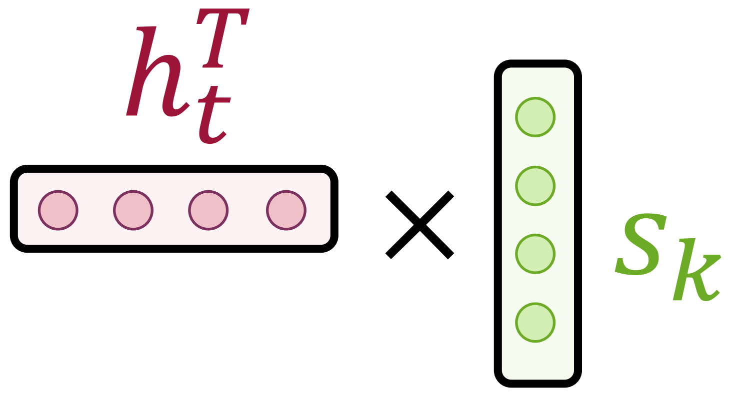
- Dot-product: $\text{score}(h_t, s_k) = h_t^T\ s_k$



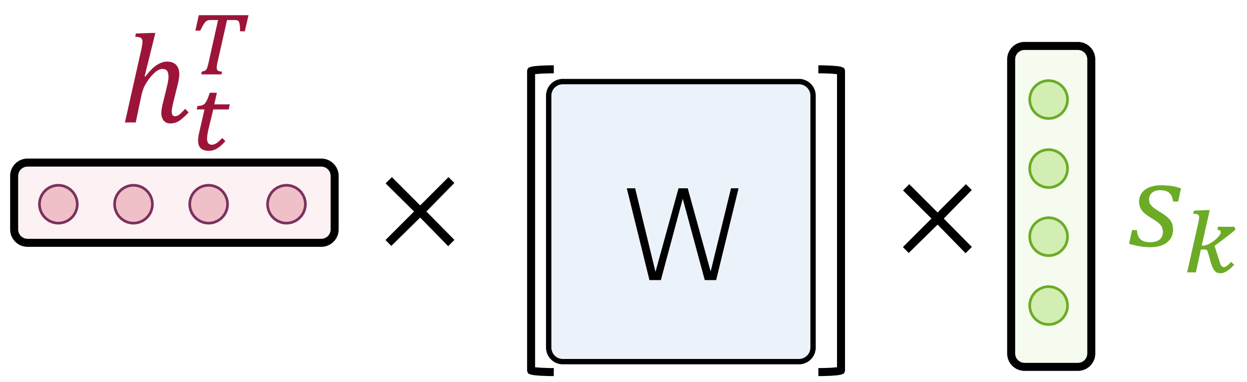


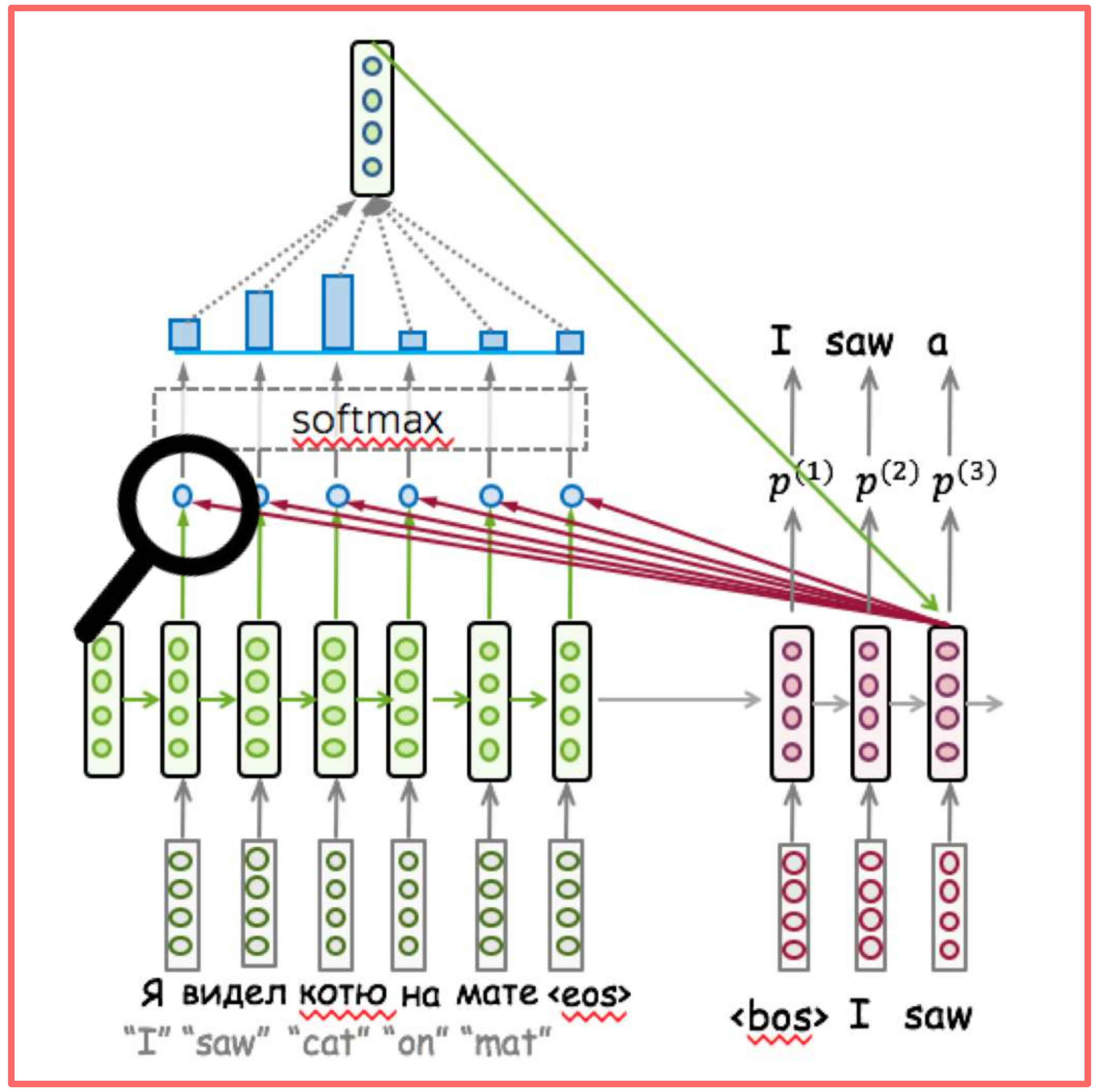
Attention Score Functions

- Dot-product: $\text{score}(h_t, s_k) = h_t^T s_k$



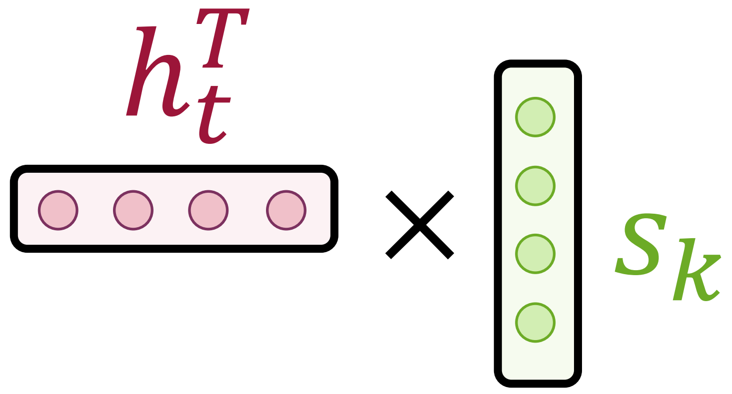
- Bilinear: $\text{score}(h_t, s_k) = h_t^T W s_k$



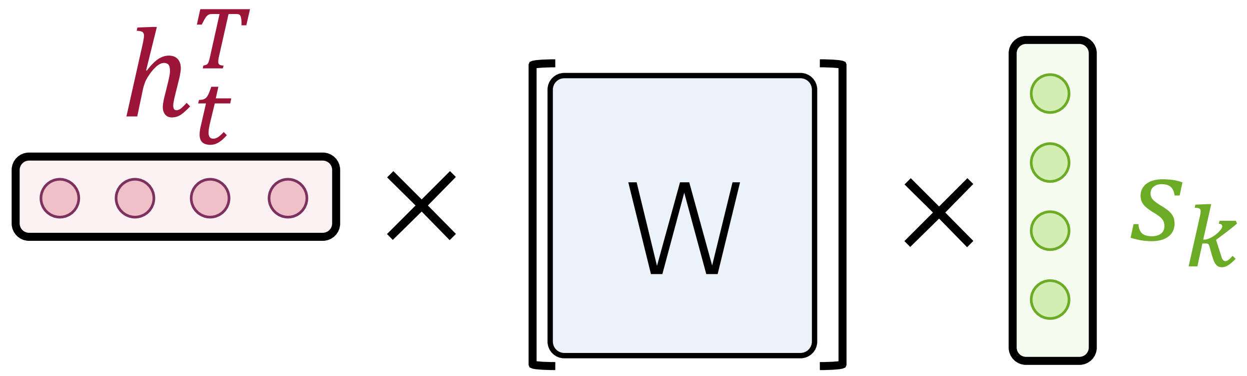


Attention Score Functions

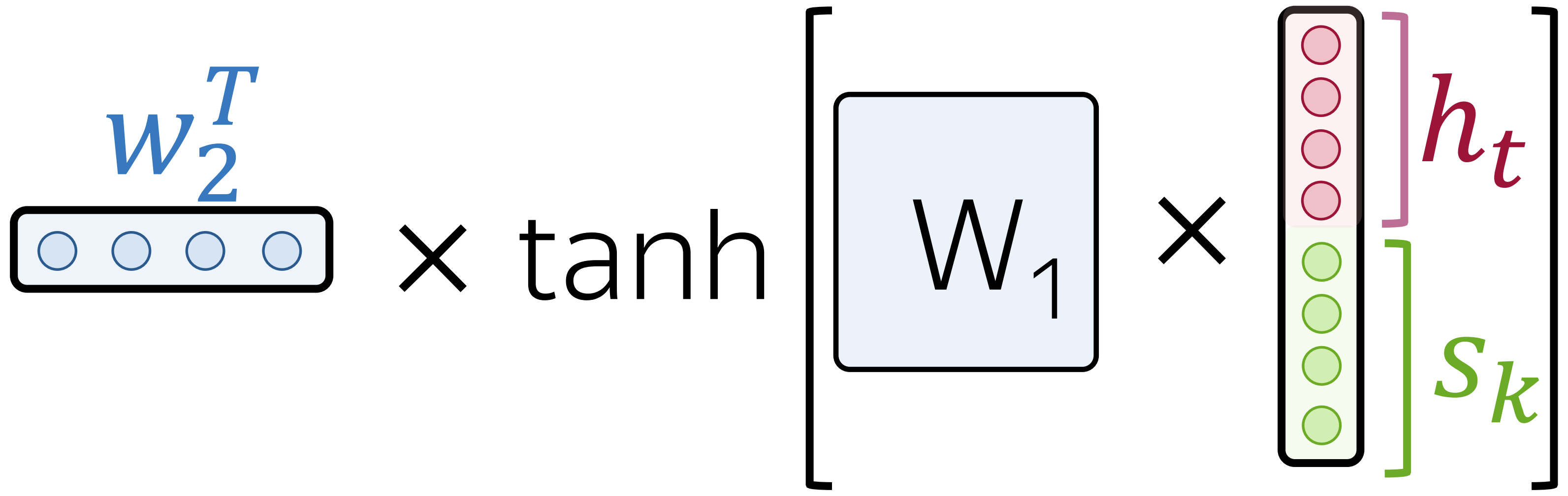
- Dot-product: $\text{score}(h_t, s_k) = h_t^T \; s_k$

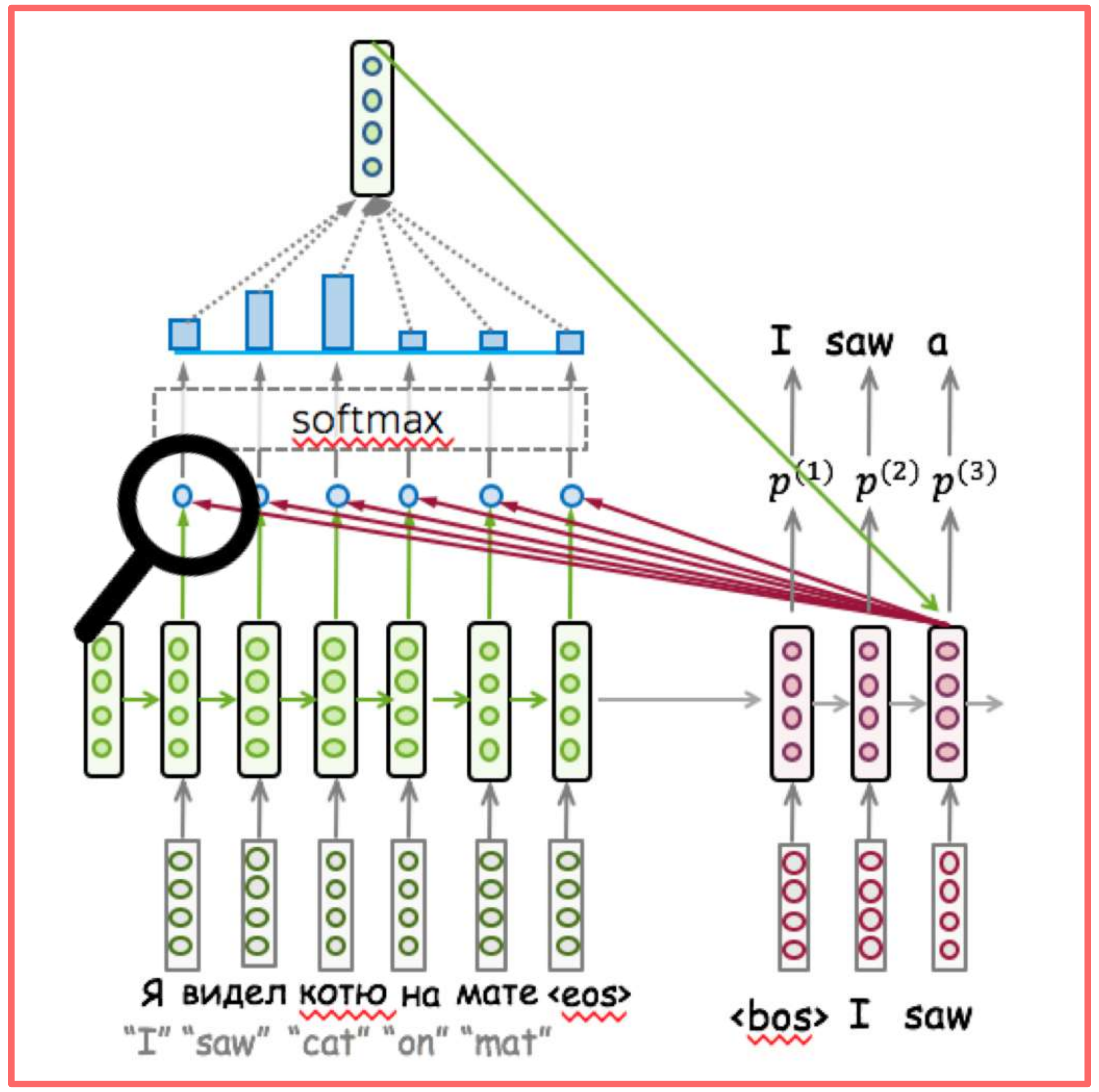


- Bilinear: $\text{score}(h_t, s_k) = h_t^T \; W s_k$



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





What is going to happen:

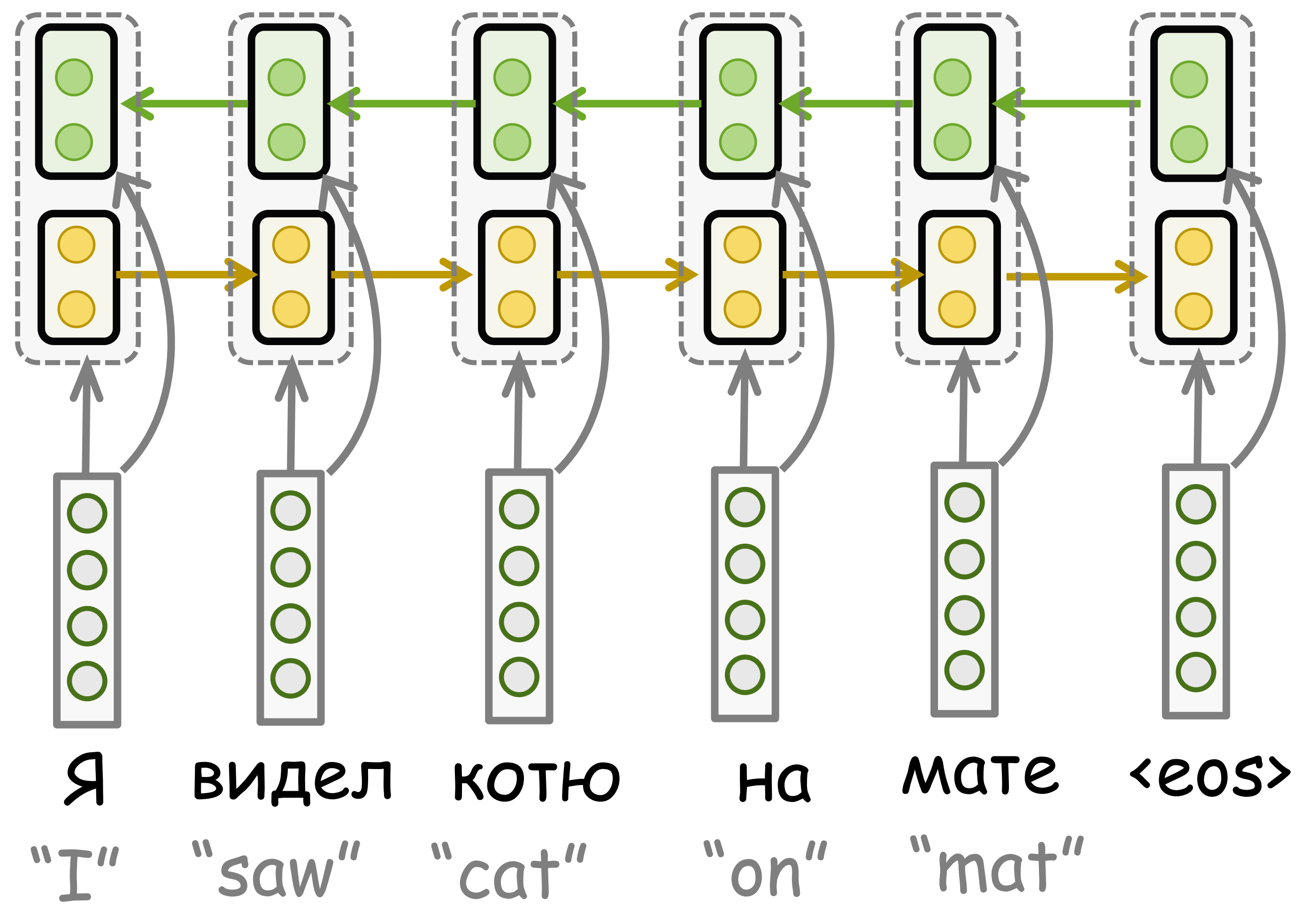
- Seq2seq Basics
- Attention
  - Why do we need it?
  - Attention: High-Level
  - Attention Score Functions
  - Models: Bahdanau vs Luong
- Transformer
- Subword Segmentation: BPE
- Analysis and Interpretability

What is going to happen:

- Seq2seq Basics
- Attention
    - Why do we need it?
    - Attention: High-Level
    - Attention Score Functions
    - Models: Bahdanau vs Luong
- Transformer
- Subword Segmentation: BPE
- Analysis and Interpretability

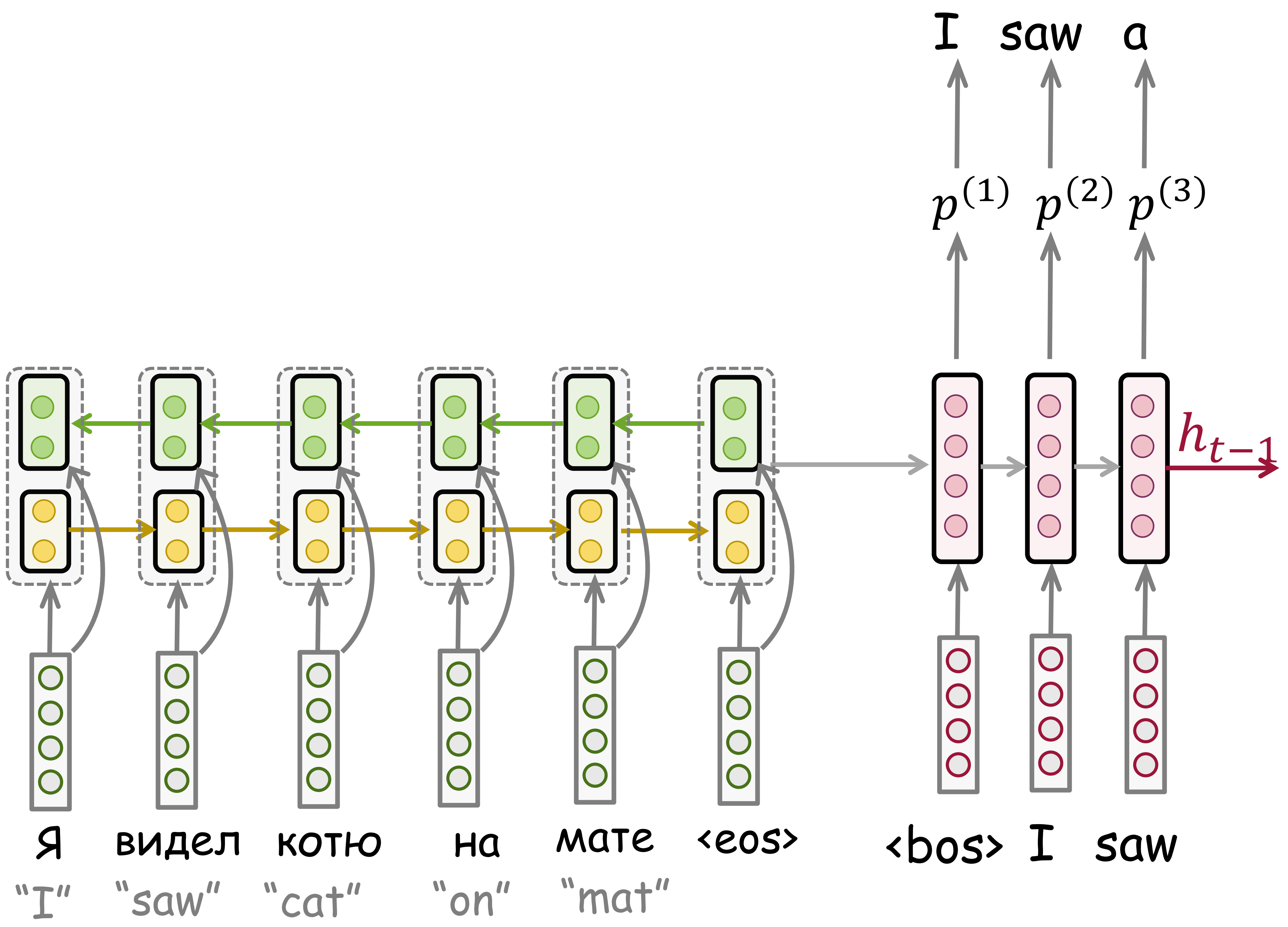
Bahdanau Model (the original attention model)

Bidirectional encoder
Concatenate states from
forward and backward RNNs



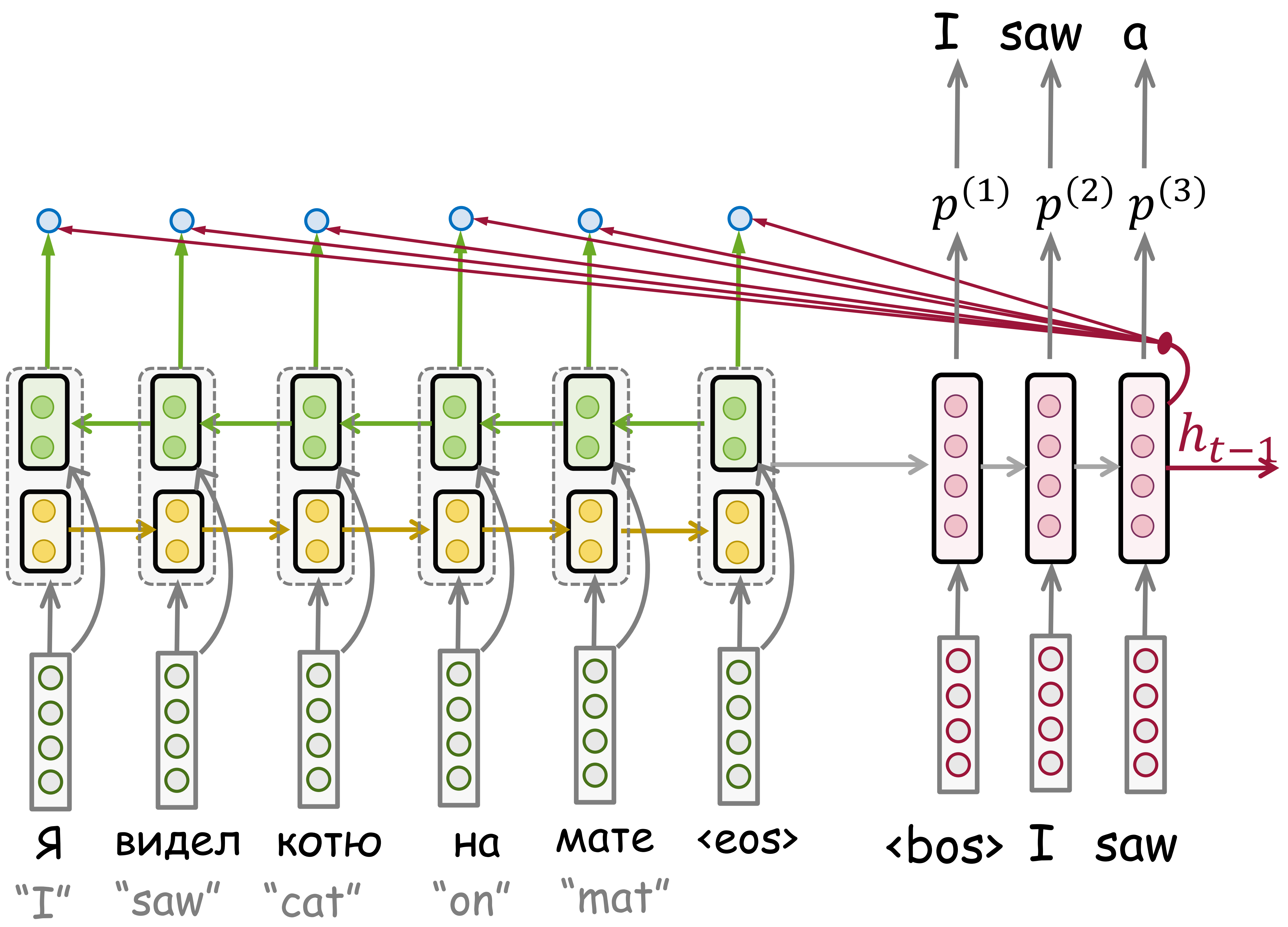
Bahdanau Model (the original attention model)

Bidirectional encoder
Concatenate states from
forward and backward RNNs



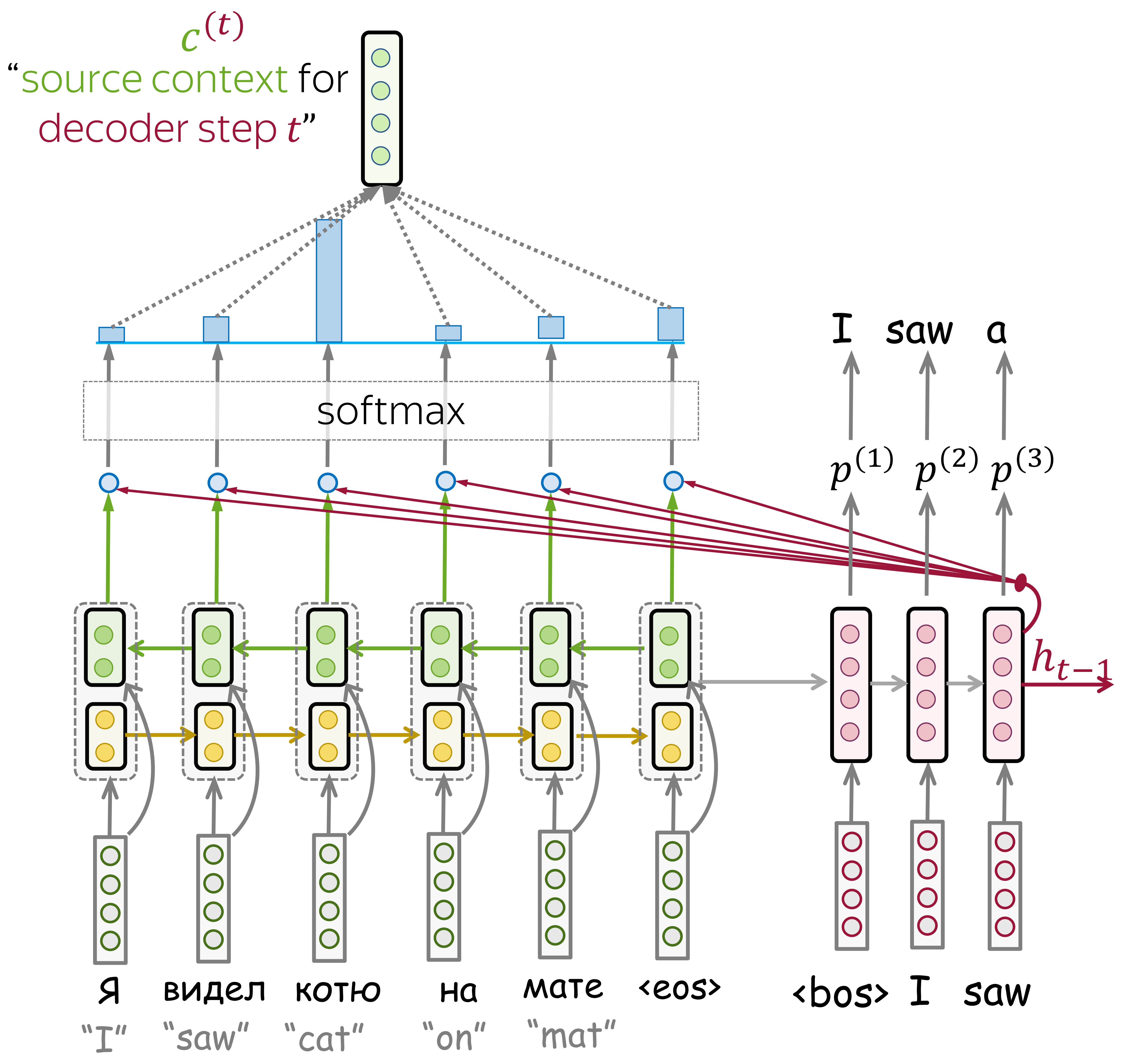
Bahdanau Model (the original attention model)

Bidirectional encoder
Concatenate states from
forward and backward RNNs

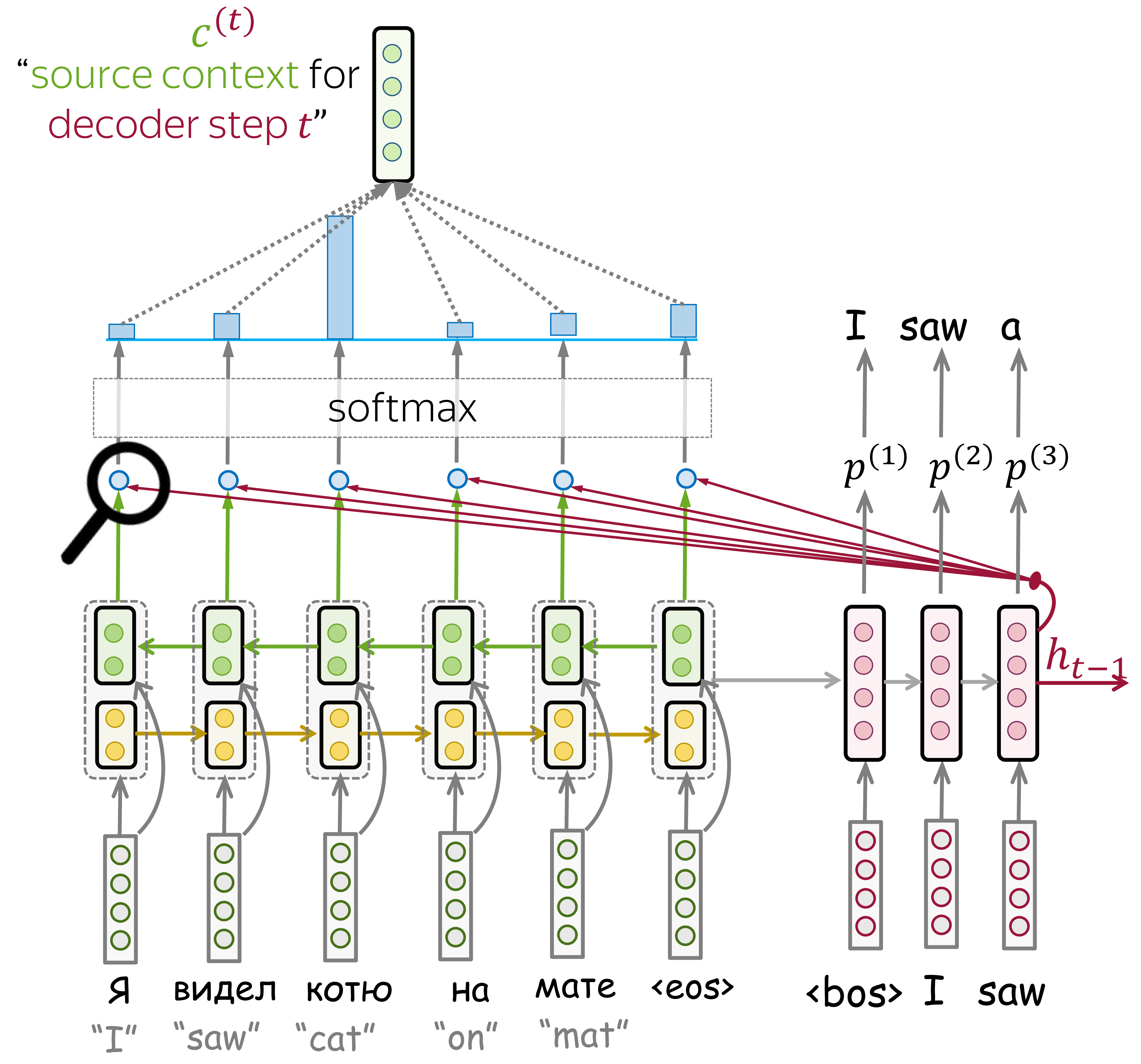


Bahdanau Model (the original attention model)

Bidirectional encoder
Concatenate states from
forward and backward RNNs

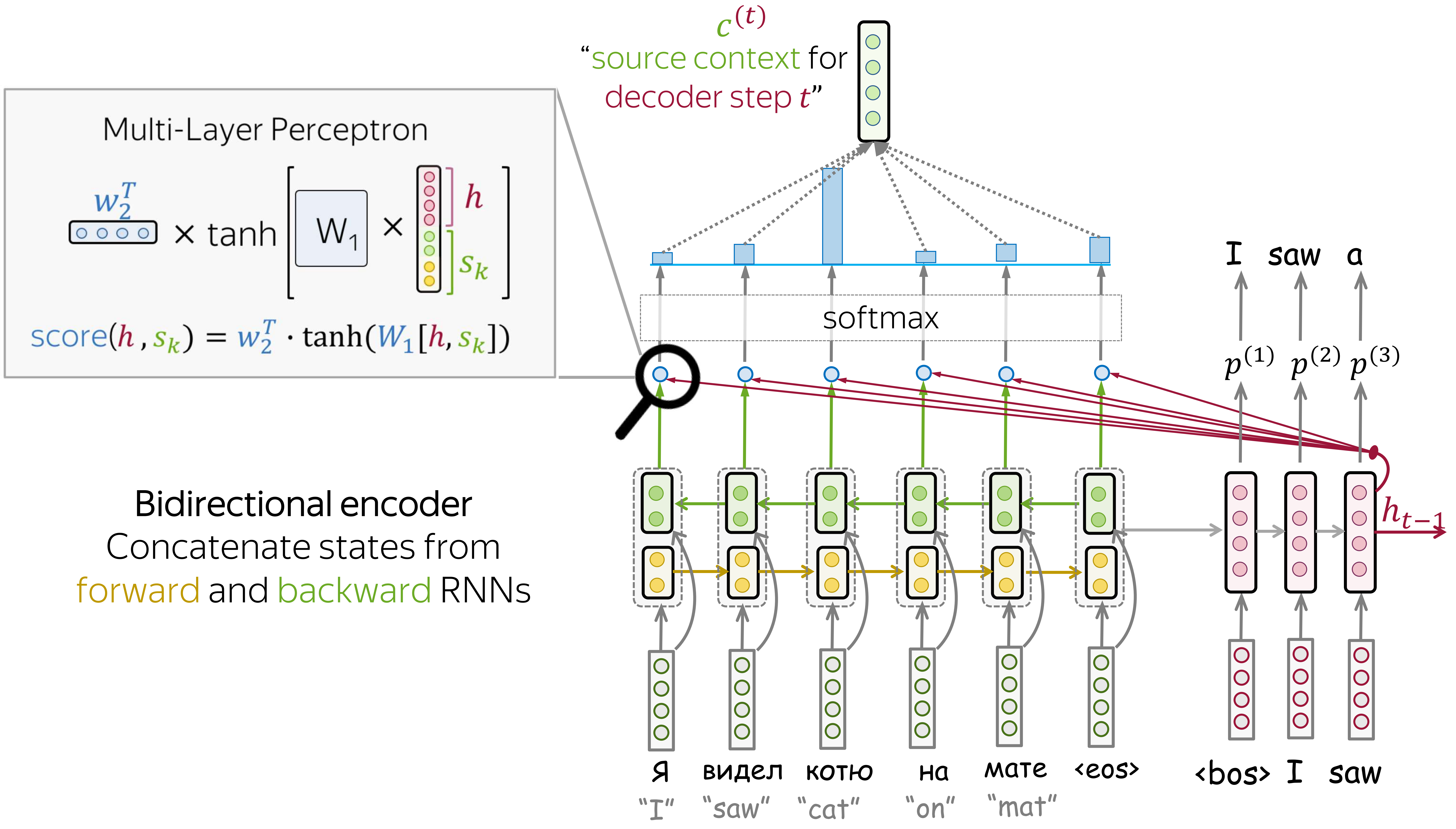


Bahdanau Model (the original attention model)



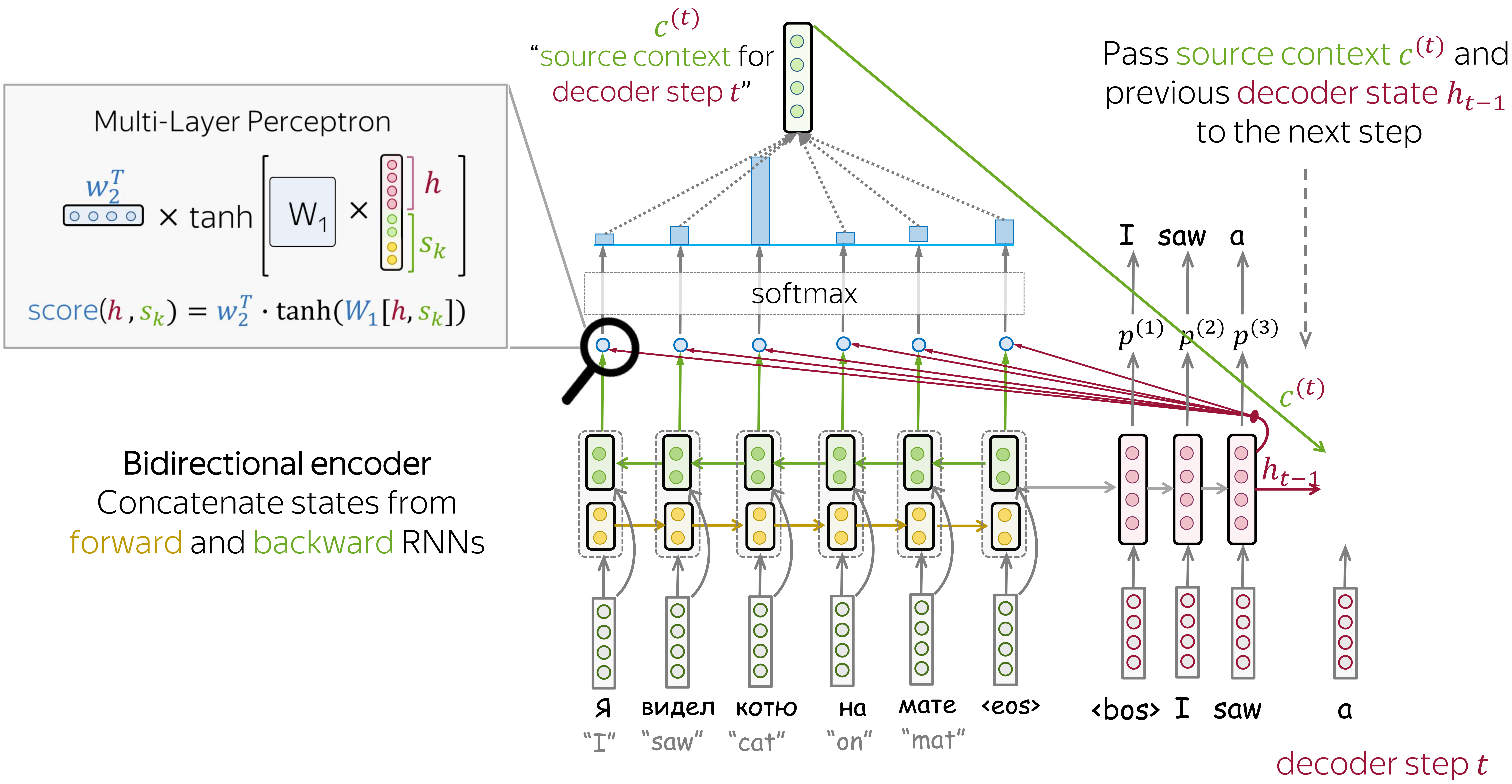
Bidirectional encoder
Concatenate states from
forward and backward RNNs

Bahdanau Model (the original attention model)

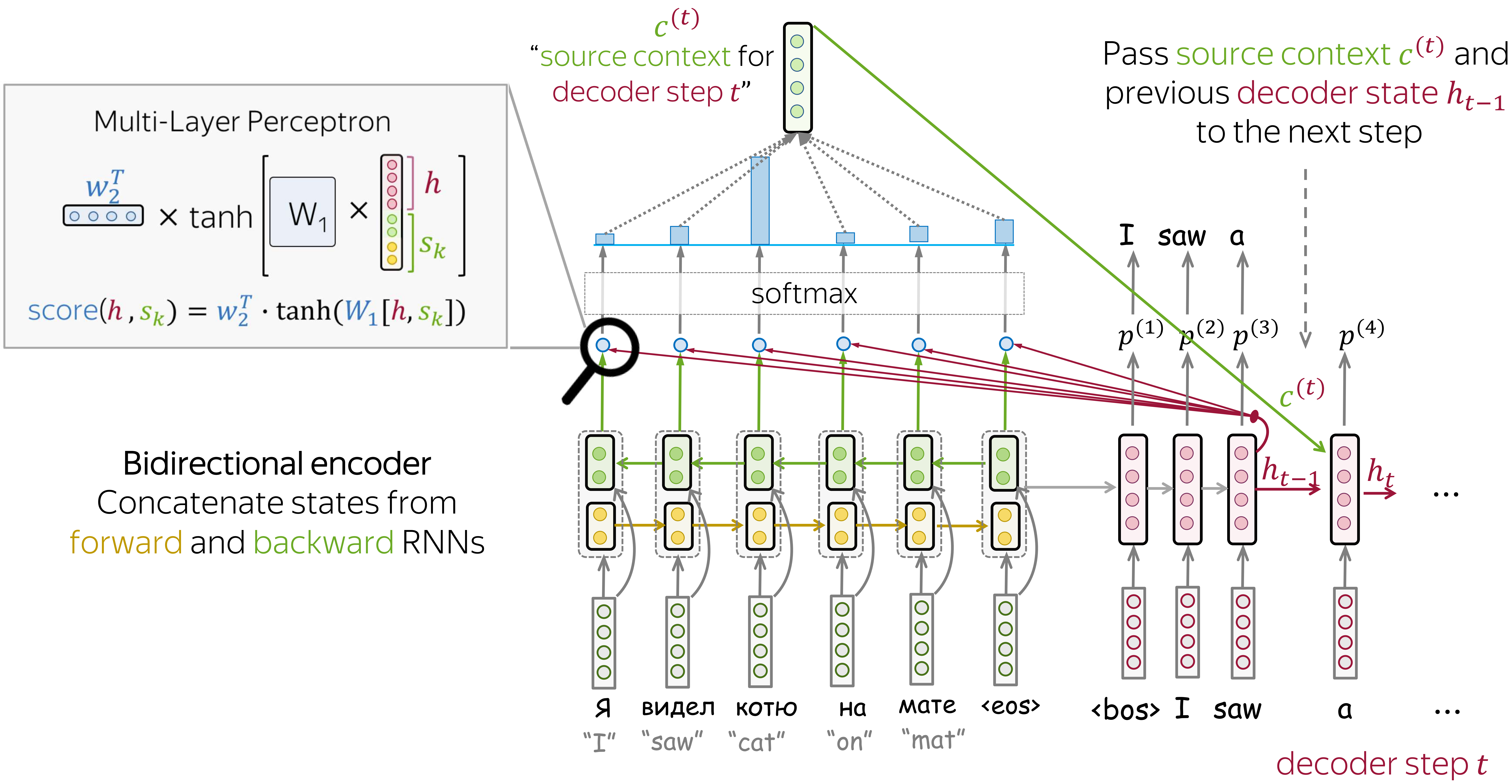


Bidirectional encoder
Concatenate states from
forward and backward RNNs

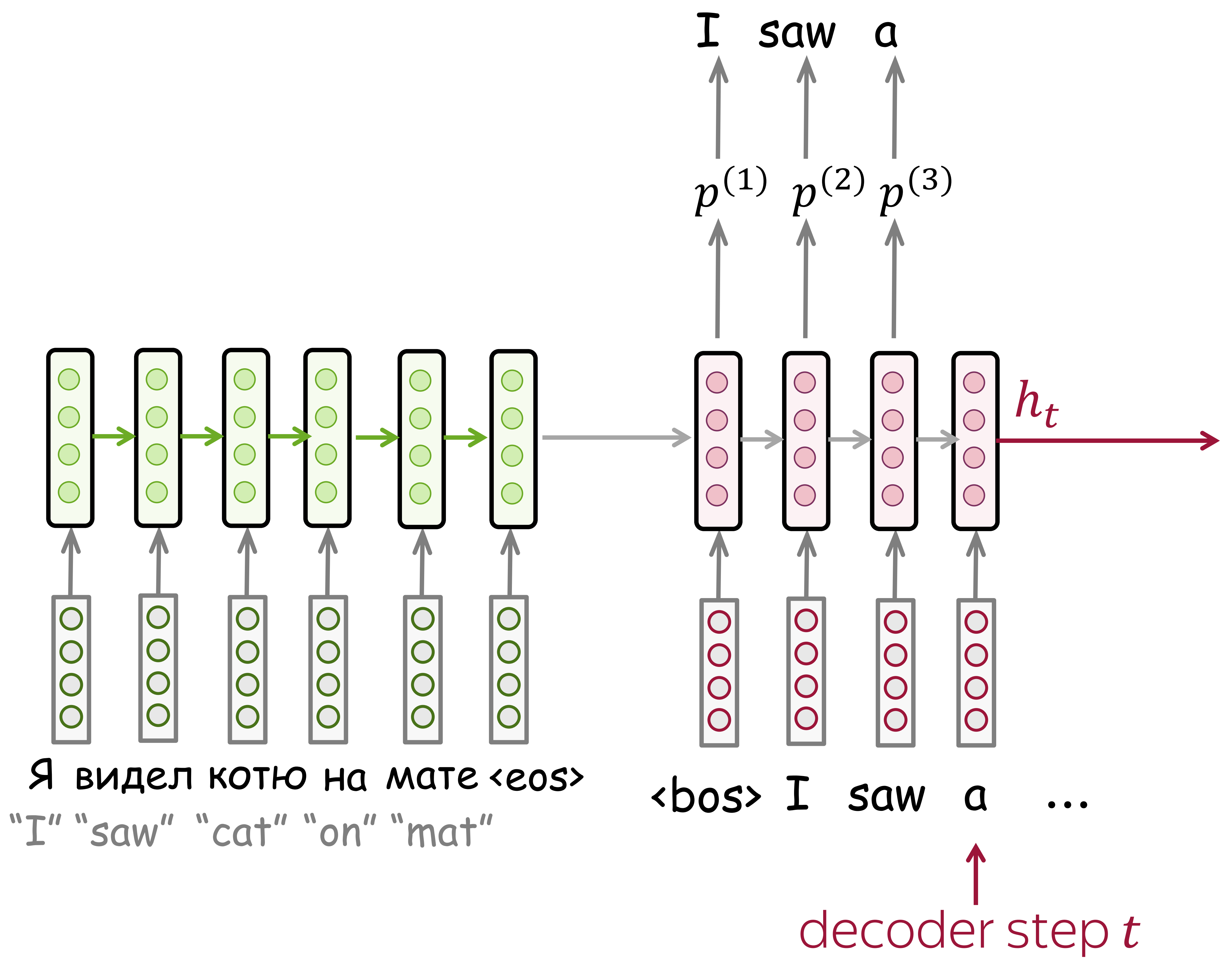
Bahdanau Model (the original attention model)



Bahdanau Model (the original attention model)

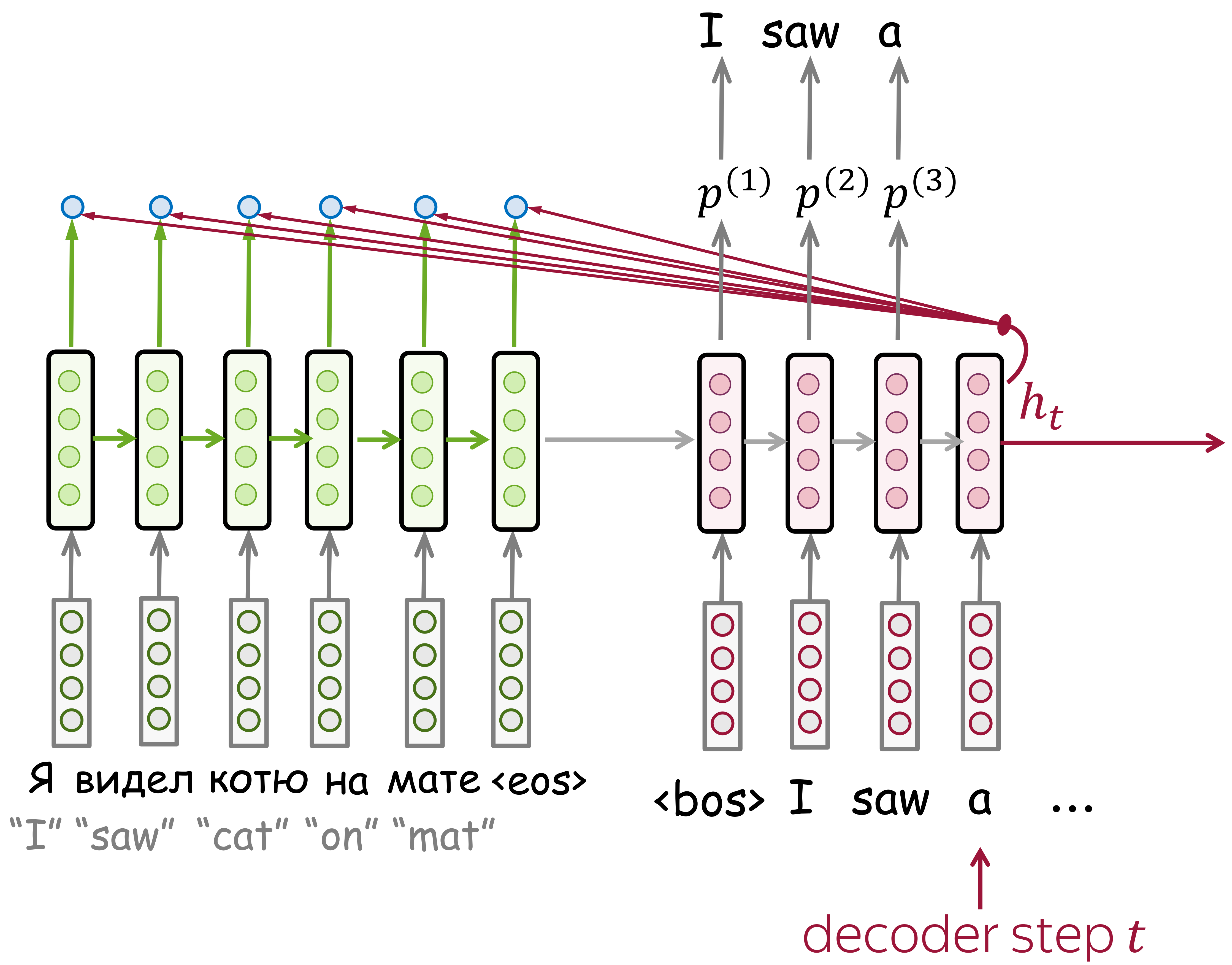


Luong Model

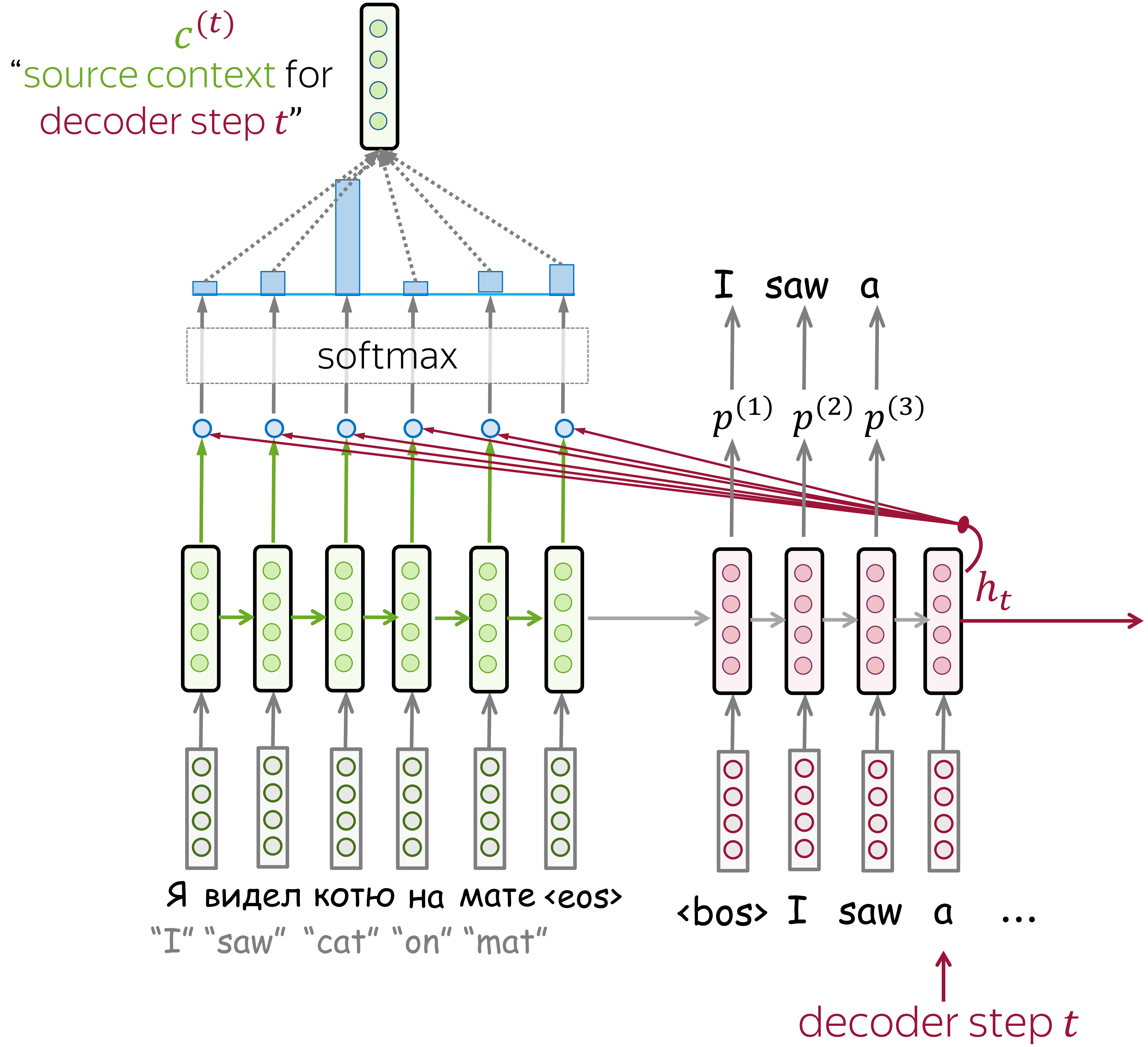


I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$
h_t
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
decoder step t

Luong Model

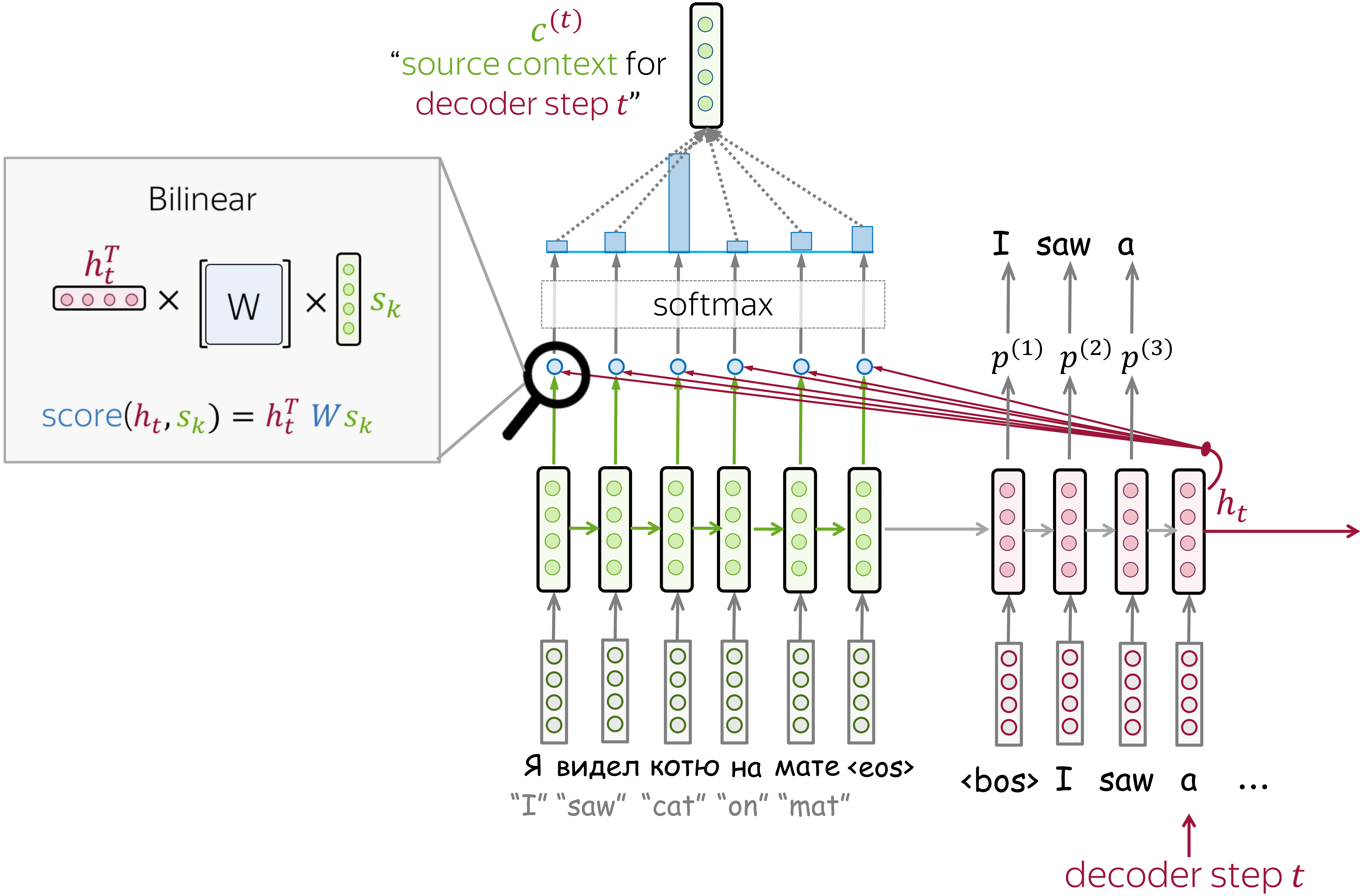

I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$
h_t
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
decoder step t

Luong Model



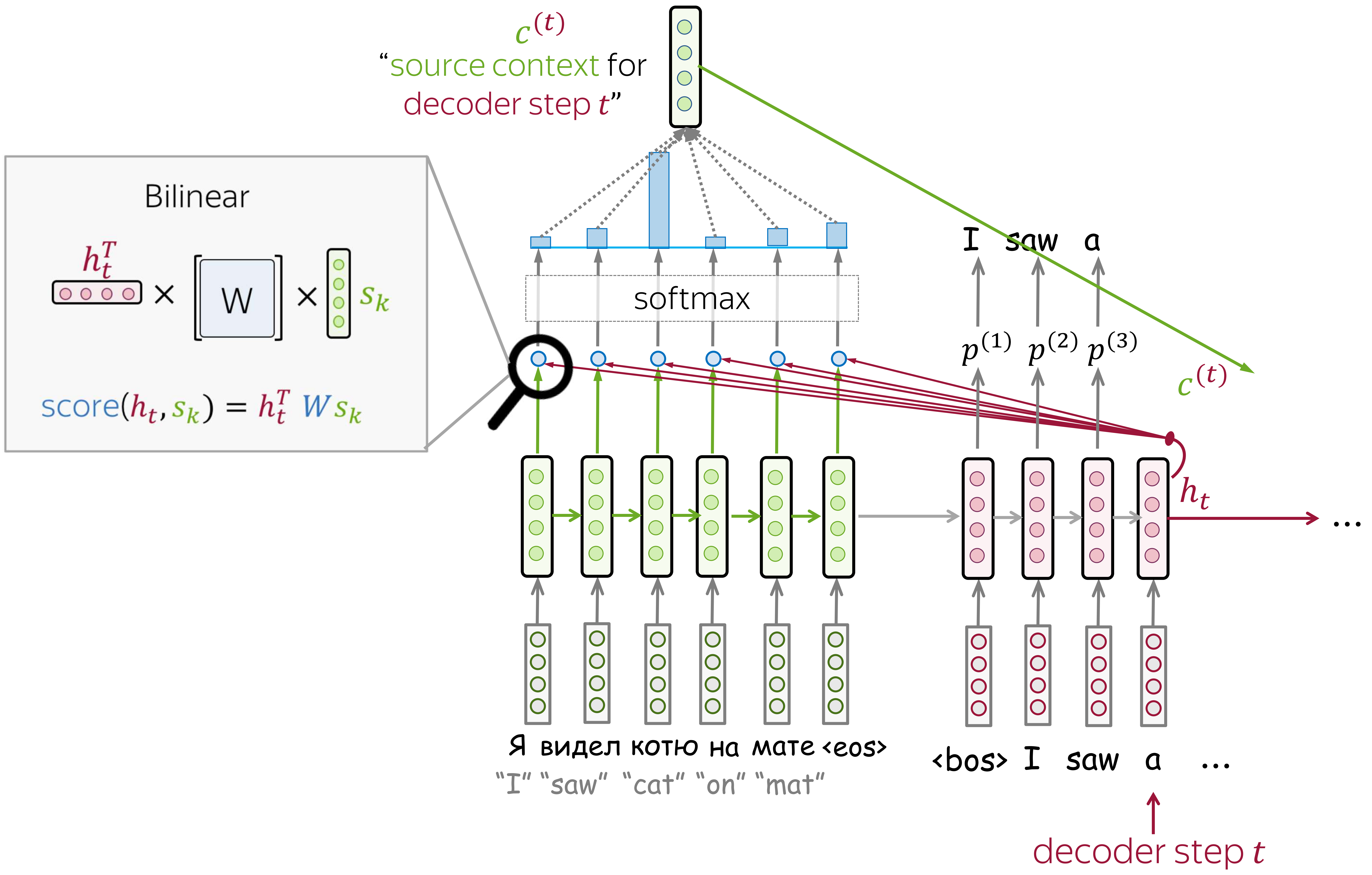
$c^{(t)}$
"source context for decoder step t"
softmax
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$
h_t
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
decoder step t

Luong Model



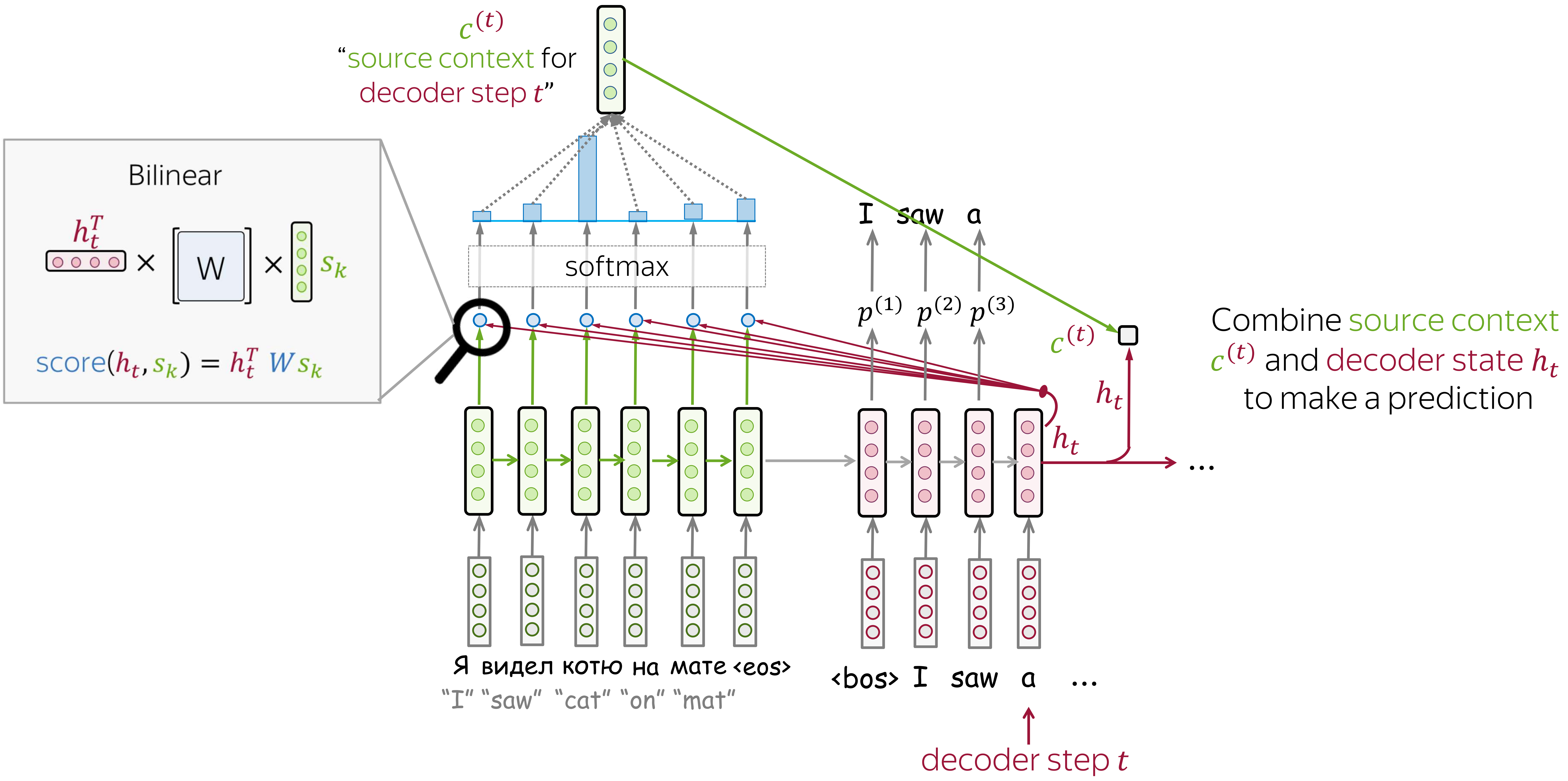
$c^{(t)}$
"source context for decoder step t"
Bilinear
h_t^T
W
s_k
$\text{score}(h_t, s_k) = h_t^T W s_k$
softmax
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$
h_t
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
decoder step t

Luong Model



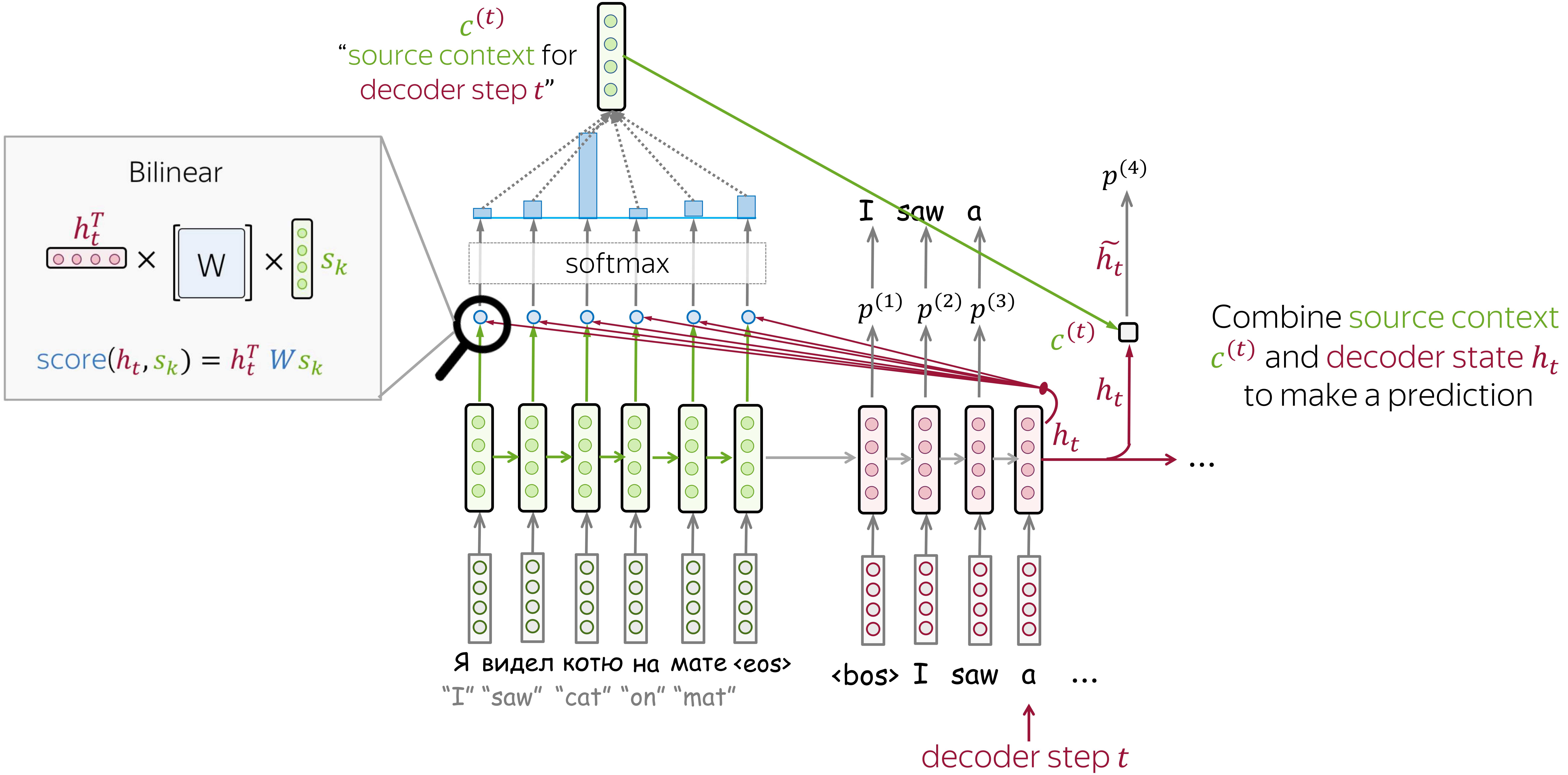
$c^{(t)}$
"source context for decoder step t"
Bilinear
h_t^T × W × s_k
score$(h_t, s_k) = h_t^T\, W s_k$
softmax
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$
$c^{(t)}$
h_t
...
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
decoder step t

Luong Model



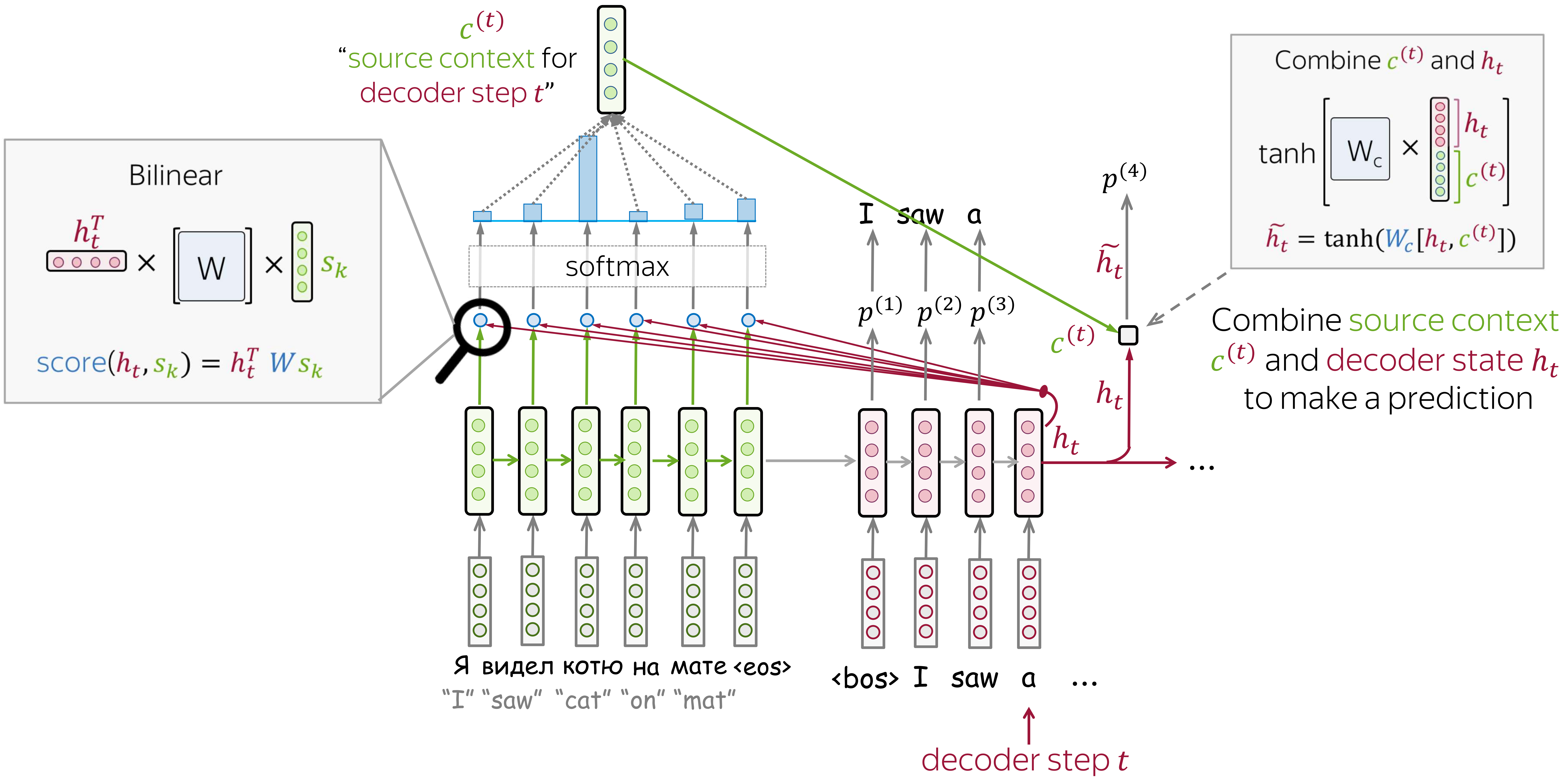
$c^{(t)}$
"source context for decoder step t"
Bilinear
h_t^T
$\times$ W $\times$ s_k
score$(h_t, s_k) = h_t^T W s_k$
softmax
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$
$c^{(t)}$
h_t
h_t
...
Combine source context $c^{(t)}$ and decoder state h_t to make a prediction
Я видел котю на мате <eos>
"I" "saw" "cat"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

Luong Model



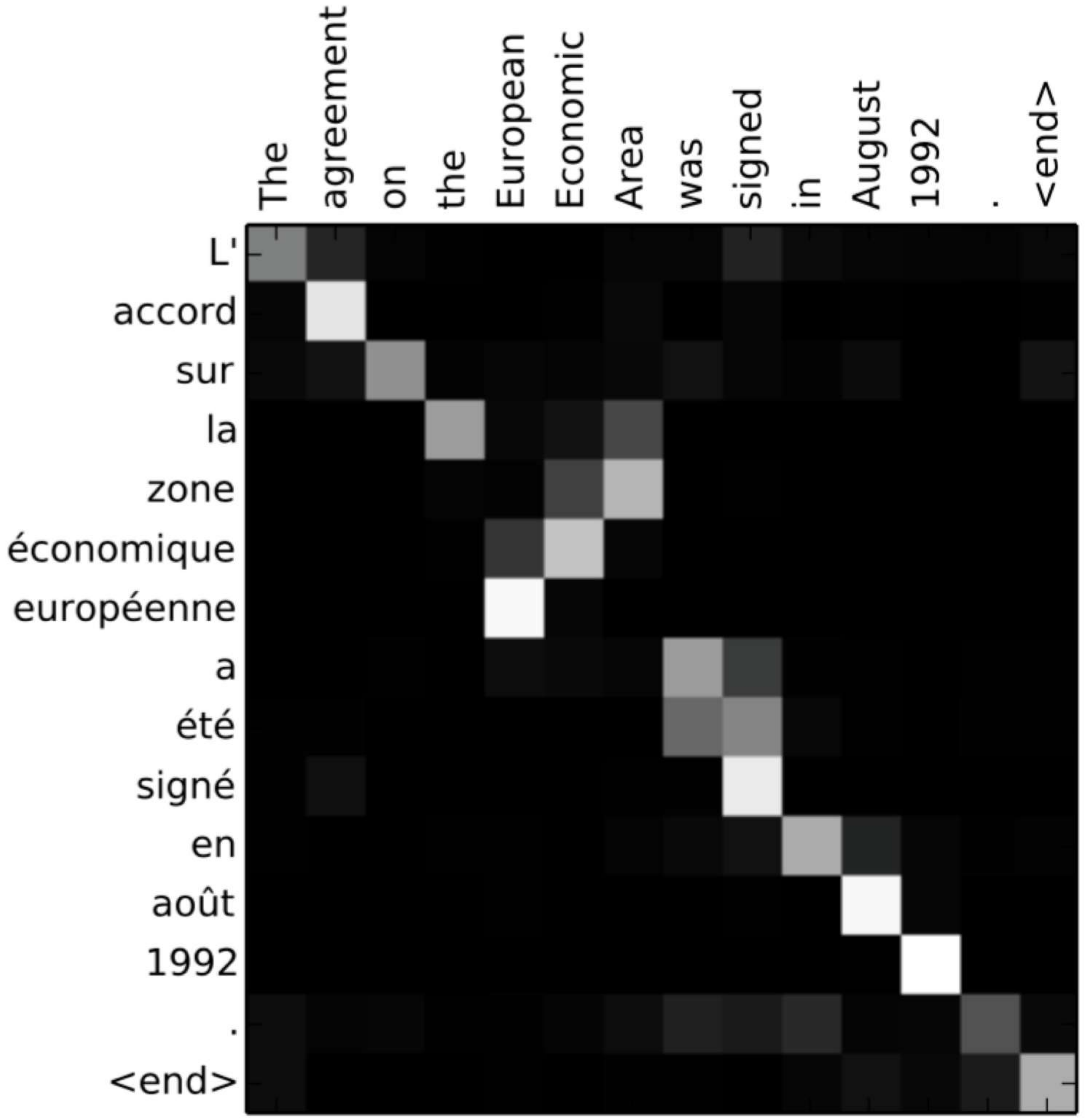
$c^{(t)}$
"source context for decoder step t"
Bilinear
h_t^T
$\times$ W $\times$ s_k
score(h_t, s_k) = h_t^T $W s_k$
softmax
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$
$p^{(4)}$
$\tilde{h}_t$
$c^{(t)}$
h_t
h_t
...
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
decoder step t
Combine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

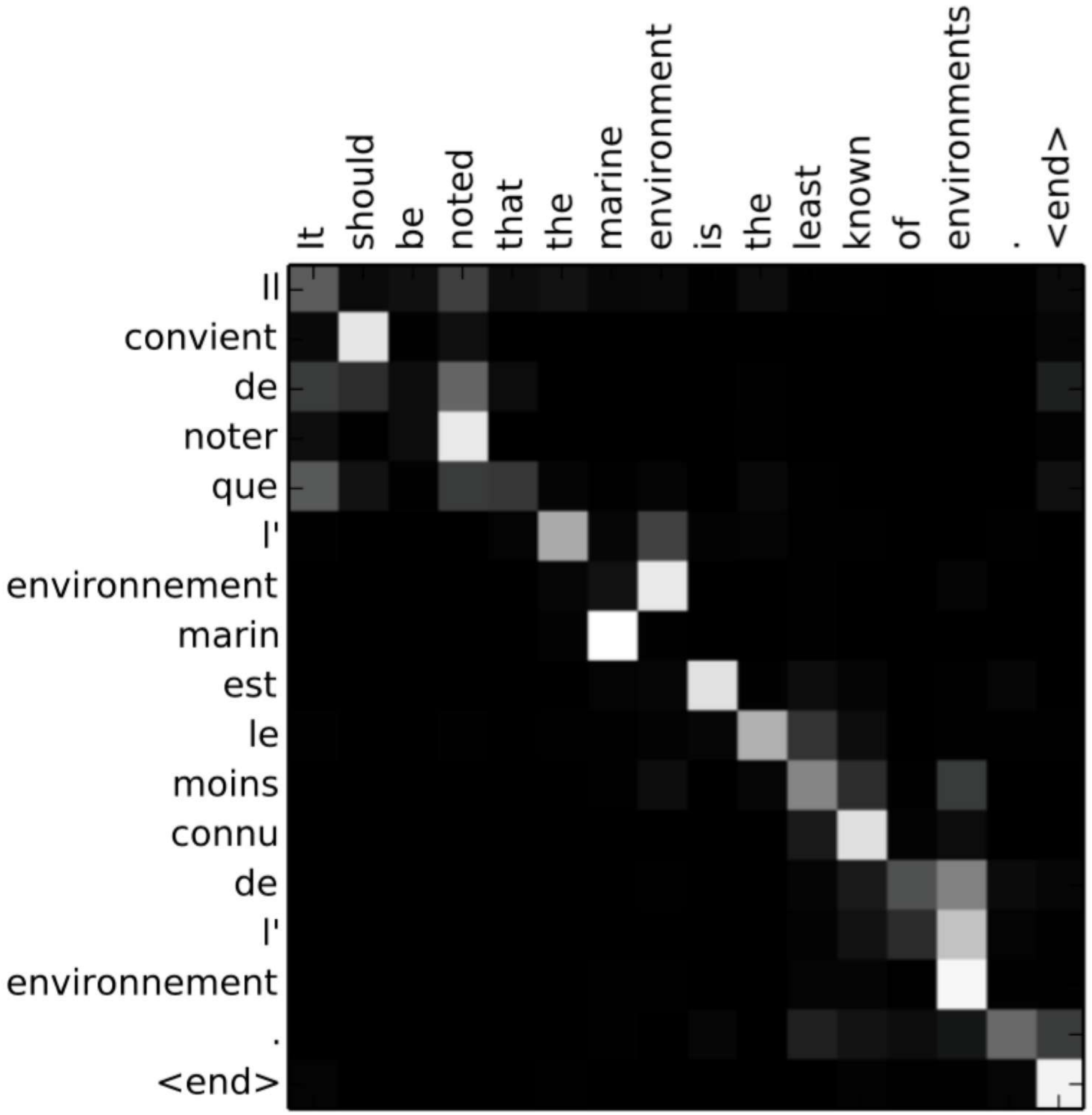
Luong Model



Bilinear
h_t^T
W
s_k
score$(h_t, s_k) = h_t^T\ W s_k$
$c^{(t)}$
"source context for decoder step t"
softmax
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
decoder step t
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$
$p^{(4)}$
$\tilde{h}_t$
$c^{(t)}$
h_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

Attention Learned (almost) Alignment





The examples are from the paper Neural Machine Translation by Jointly Learning to Align and Translate.