LMs and Seq2Seq

Deep Learning

Aziz Temirkhanov
Lambda, HSE

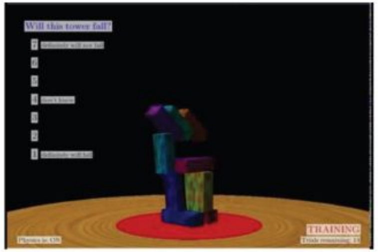
Train Models

- have some properties of trains (look like ones)
- can behave similarly to trains
- good models have more of the above

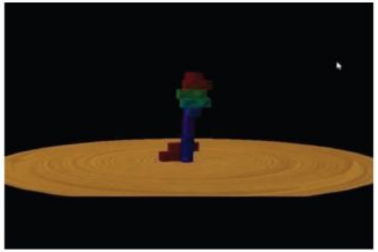
Models of Physical World

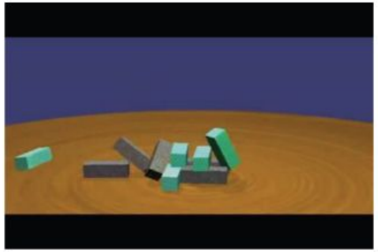
- understand which events are in better agreement with the world, which are more likely

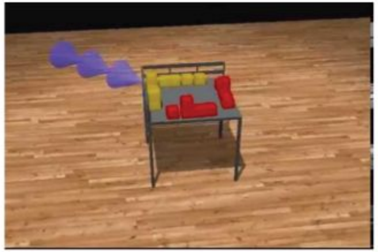
- can predict what happens given some “context”



Will it fall?


In which direction?

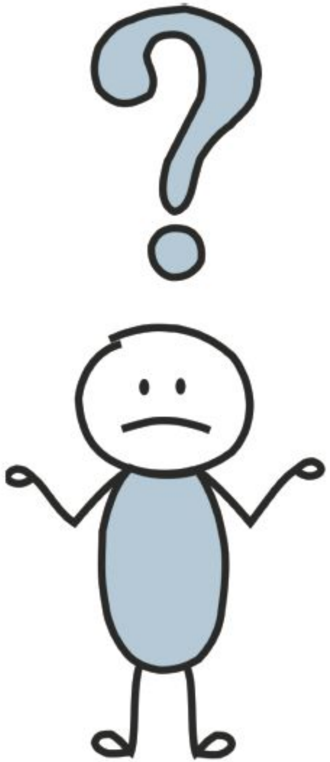

Different masses


Complex scenes


Infer the mass


Predict fluids

Language Models



Language Models

The intuition is exactly the same!

What is different, is the notion of an event: for language, an event is a linguistic unit (text, sentence, token, symbol).

> Language Models (LMs) estimate the probability of different linguistic units: symbols, tokens, token sequences.

We do we need it?

We deal with Language Models every day!

Web search engine / ...

I saw a cat|

I saw a cat on the chair
I saw a cat running after a dog
I saw a cat in my dream
I saw a cat book

We do we need it?

We deal with Language
Models every day!

Translation service / mail agent / ...

We do we need it?

We deal with Language Models every day!

Translation service / mail agent / ...

I saw a catt

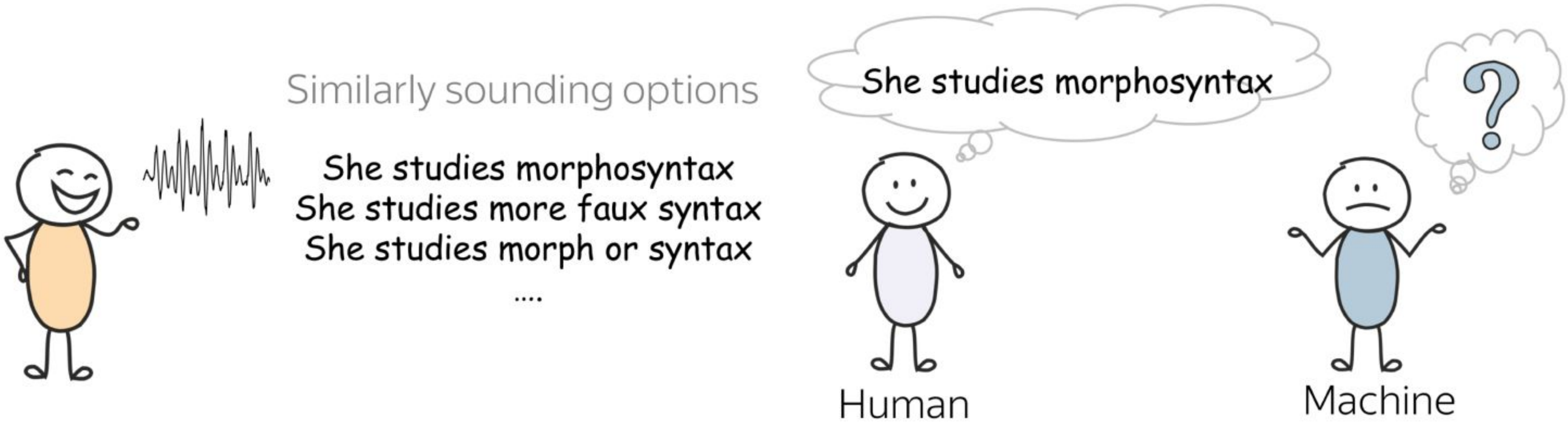
Probably you meant I saw a cat

We do we need it?

We deal with Language Models every day!

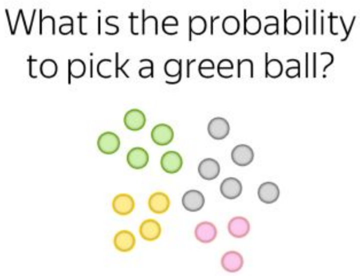
Keyboard / mail agent / ...

Ambiguity



The **morphosyntax** example is from the slides by Alex Lascarides and Sharon Goldwater, Foundations of Natural Language Processing course at the University of Edinburgh.

Modeling



Modeling

What is the probability to pick a green ball?

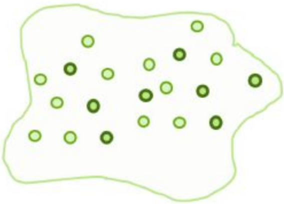
$$\frac{5}{5+6+4+3} = \frac{5}{18}$$

Modeling

What is the probability to pick a green ball?

$$\frac{5}{5+6+4+3} = \frac{5}{18}$$

Can we do the same for sentences?



Text corpus

$$P(\text{the mut is tinming the tebn}) = \frac{0}{|\text{corpus}|} = 0$$

$$P(\text{mut the tinming tebn is the}) = \frac{0}{|\text{corpus}|} = 0$$

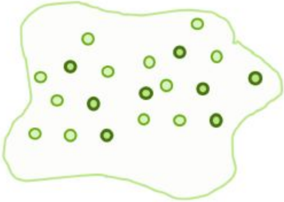
With this approach, sentences that never occurred in the corpus will receive zero probability

Modeling

What is the probability to pick a green ball?

$$\frac{5}{5+6+4+3} = \frac{5}{18}$$

Can we do the same for sentences?



Text corpus

$$P(\text{the mut is tinming the tebn}) = \frac{0}{|\text{corpus}|} = 0$$

$$P(\text{mut the tinming tebn is the}) = \frac{0}{|\text{corpus}|} = 0$$

With this approach, sentences that never occurred in the corpus will receive zero probability

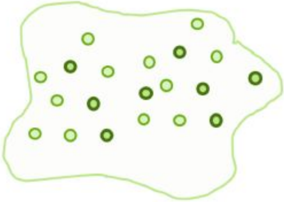
But the first sentence is "more likely" than the second! This method is not good!

Modeling

What is the probability to pick a green ball?

$$\frac{5}{5+6+4+3} = \frac{5}{18}$$

Can we do the same for sentences?



Text corpus

$$P(\text{the mut is tinming the tebn}) = \frac{0}{|\text{corpus}|} = 0$$

$$P(\text{mut the tinming tebn is the}) = \frac{0}{|\text{corpus}|} = 0$$

With this approach, sentences that never occurred in the corpus will receive zero probability

But the first sentence is "more likely" than the second! This method is not good!

We can not estimate sentence probabilities reliably if we treat them as atomic units!

Modeling

Image we

- read the sentence **I saw a cat on a mat** word by word,
- update probability every time we see a new token

$$P(\mathbf{I}) =$$

$$\underbrace{P(\mathbf{I})}_{\text{Probability of } \mathbf{I}}$$

Modeling

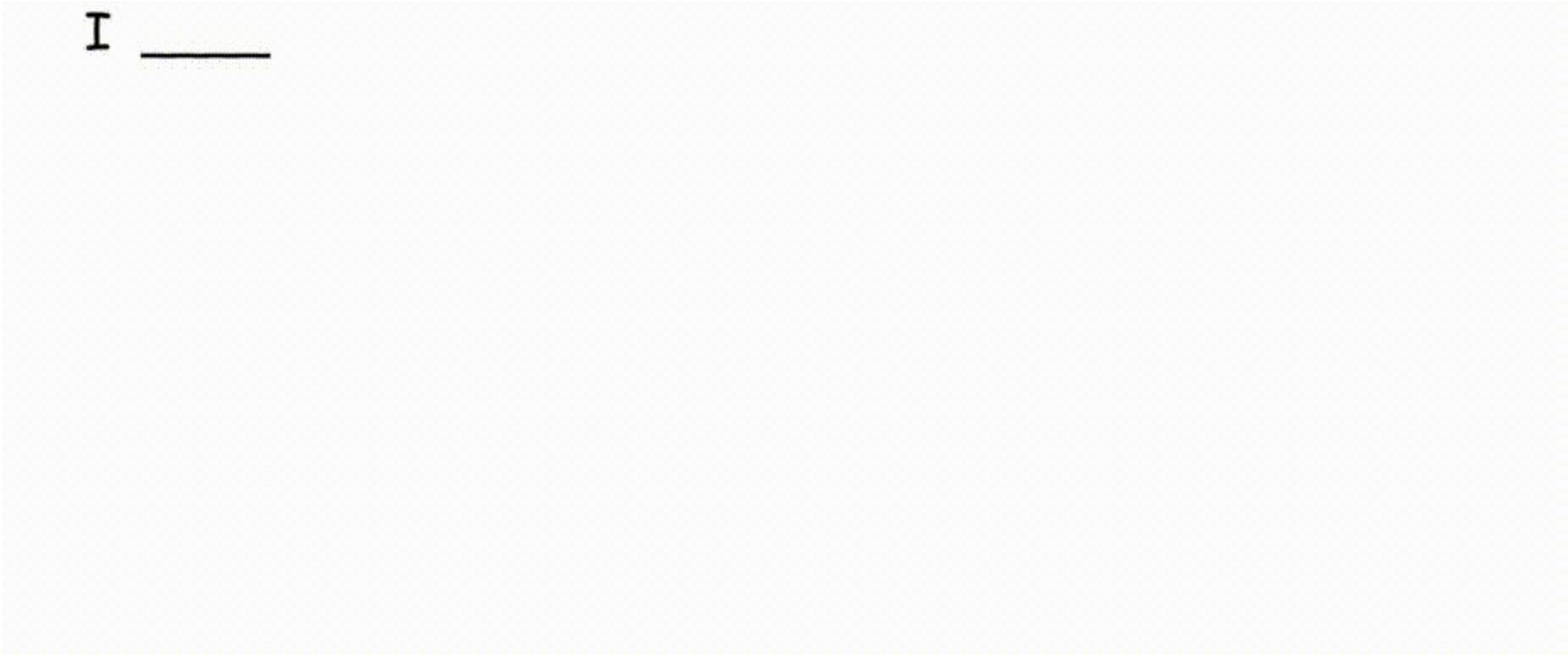
Formally,

- $(y_1, y_2, \ldots, y_n)$ is a sequence of tokens,
- $P(y_1, y_2, \ldots, y_n)$ - probability to see these tokens (in this order)

Using the product rule of probability (aka "chain rule"), we get:

$$P(y_1, y_2, \ldots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \cdots \cdot P(y_n|y_1, \ldots, y_{n-1}) = \prod_{t=1}^{n} P(y_t|y_{<t})$$

Modeling



Machine Translation

- Translation between natural languages

- More generally, translation between any sequences

Machine 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

Machine 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

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

Machine 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?

Machine 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?

Machine 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?

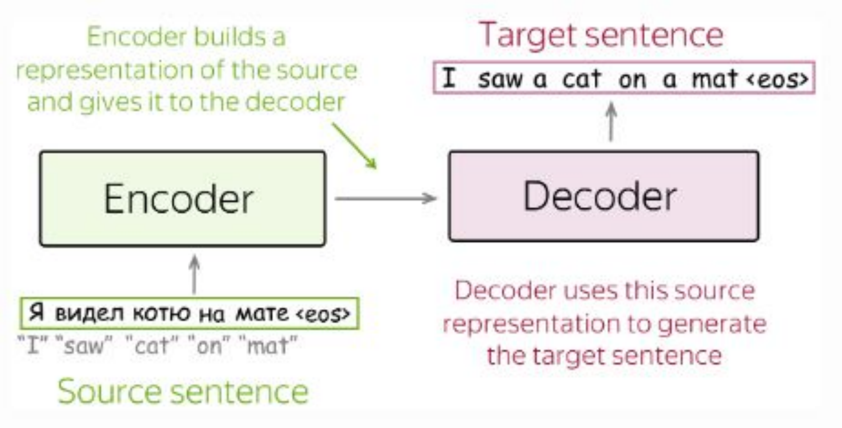
- search

How to find the argmax?

Encoder-Decoder Framework

The standard modeling paradigm:

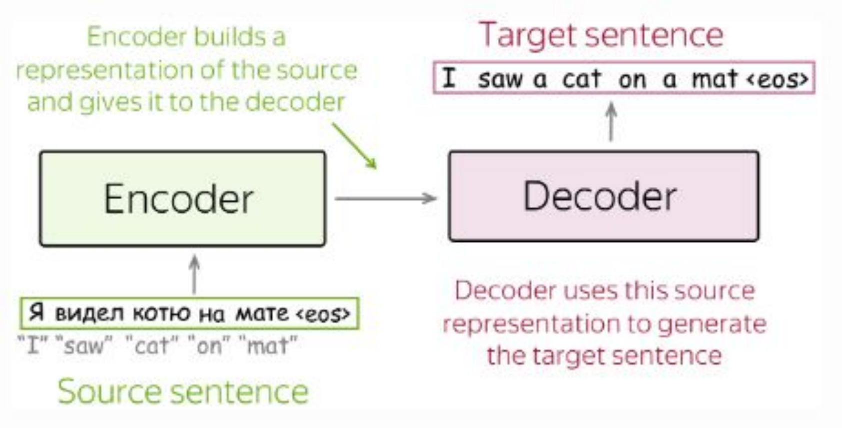
- Encoder – reads the source sentence and produces its representation



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

Conditional Language Models

Language Models:
(left-to-right)

$$P(y_1, y_2, \ldots, y_n) = \prod_{t=1}^{n} p(y_t | y_{<t})$$

Conditional
Language Models:

$$P(y_1, y_2, \ldots, y_n, | x) = \prod_{t=1}^{n} p(y_t | y_{<t}, x)$$

condition on source x

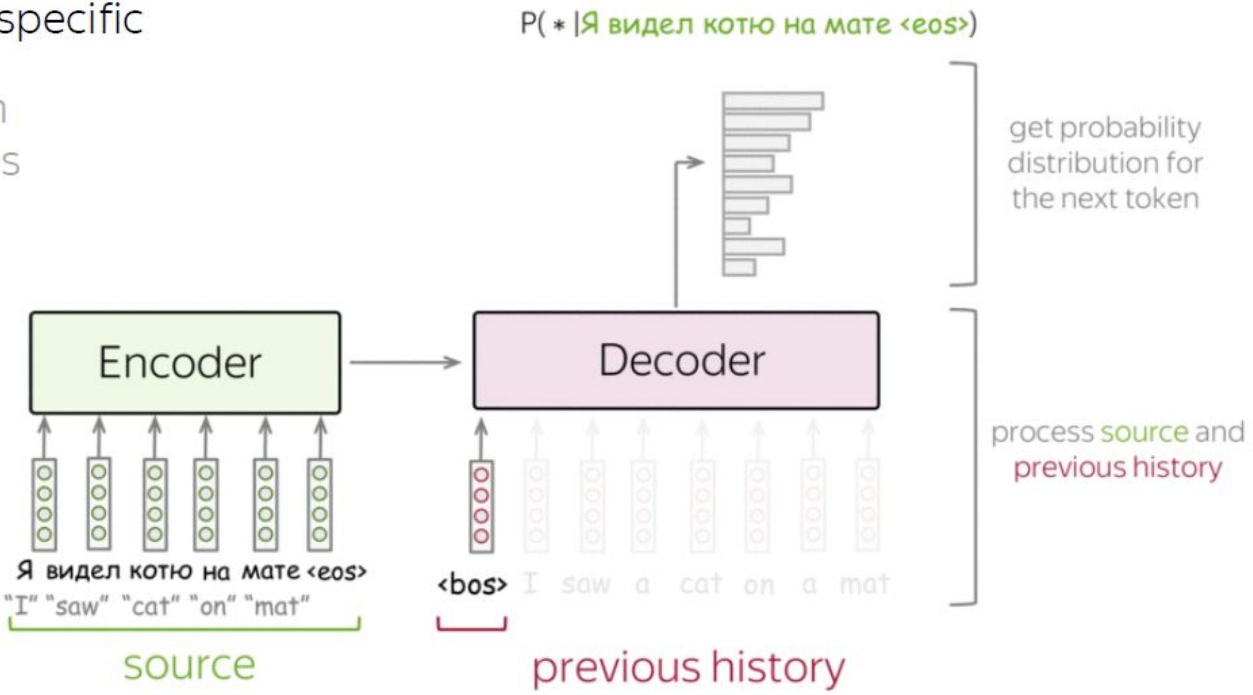
General View

- process context – model-specific

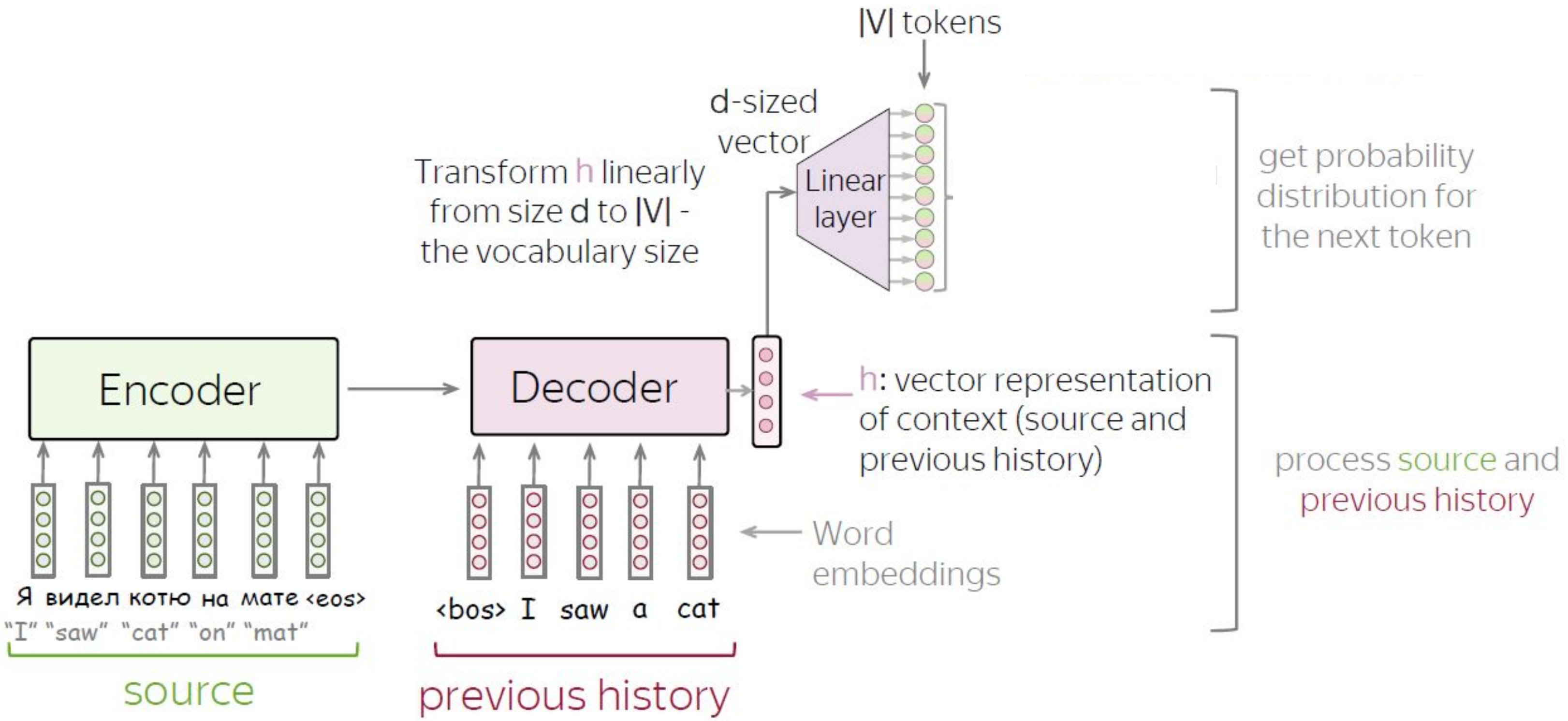
  Get vector representation of the source and previous target tokens

- evaluate probabilities – model-agnostic

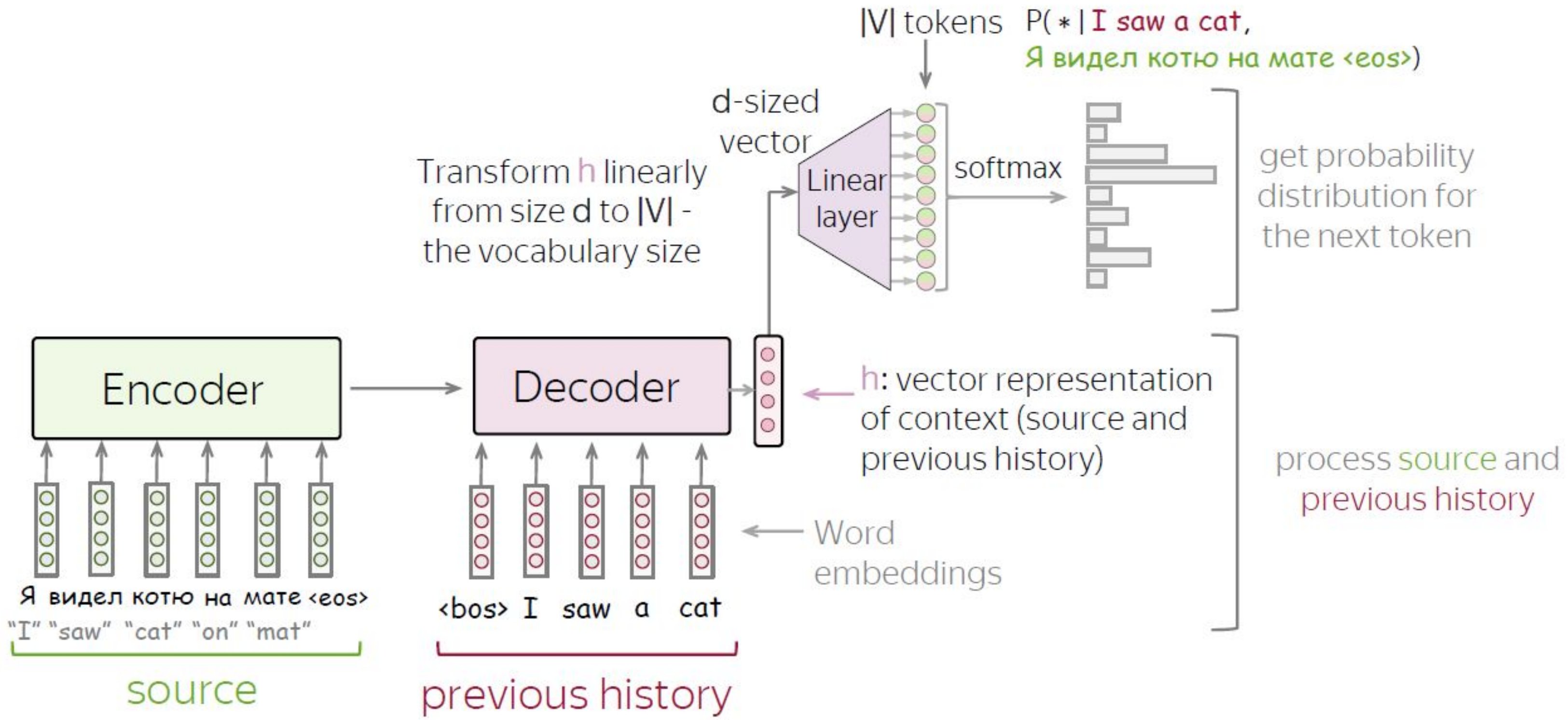
  Predict probability distribution for the next target token



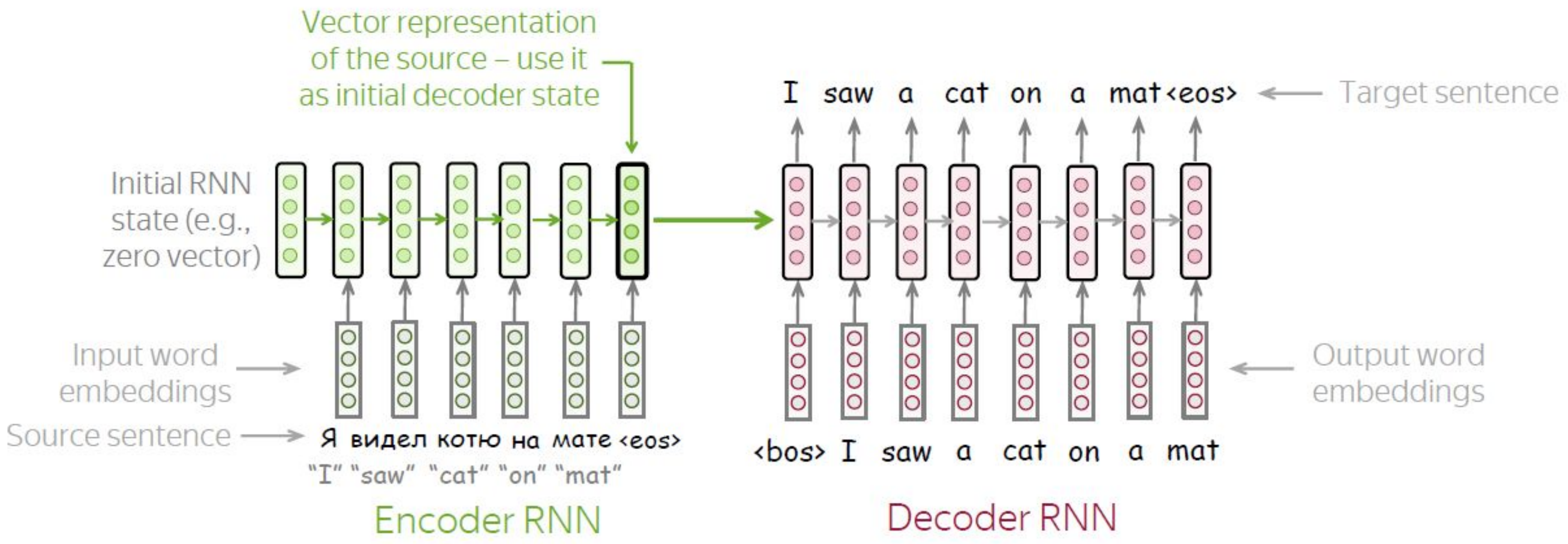
High-Level Pipeline



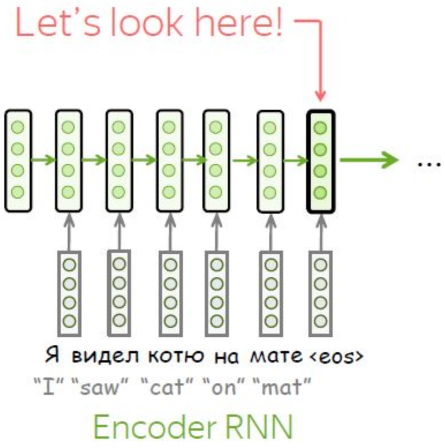
High-Level Pipeline



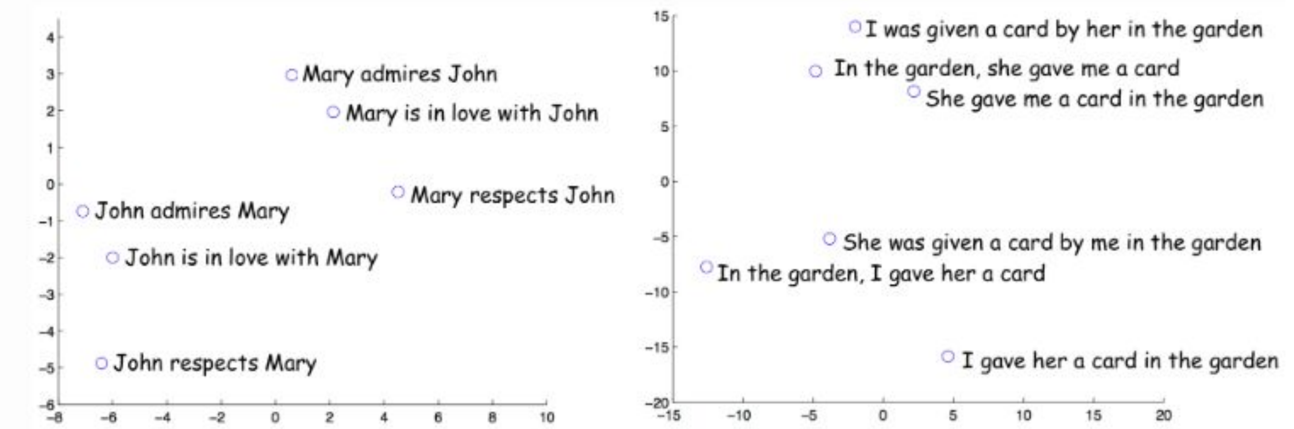
Two RNN Model



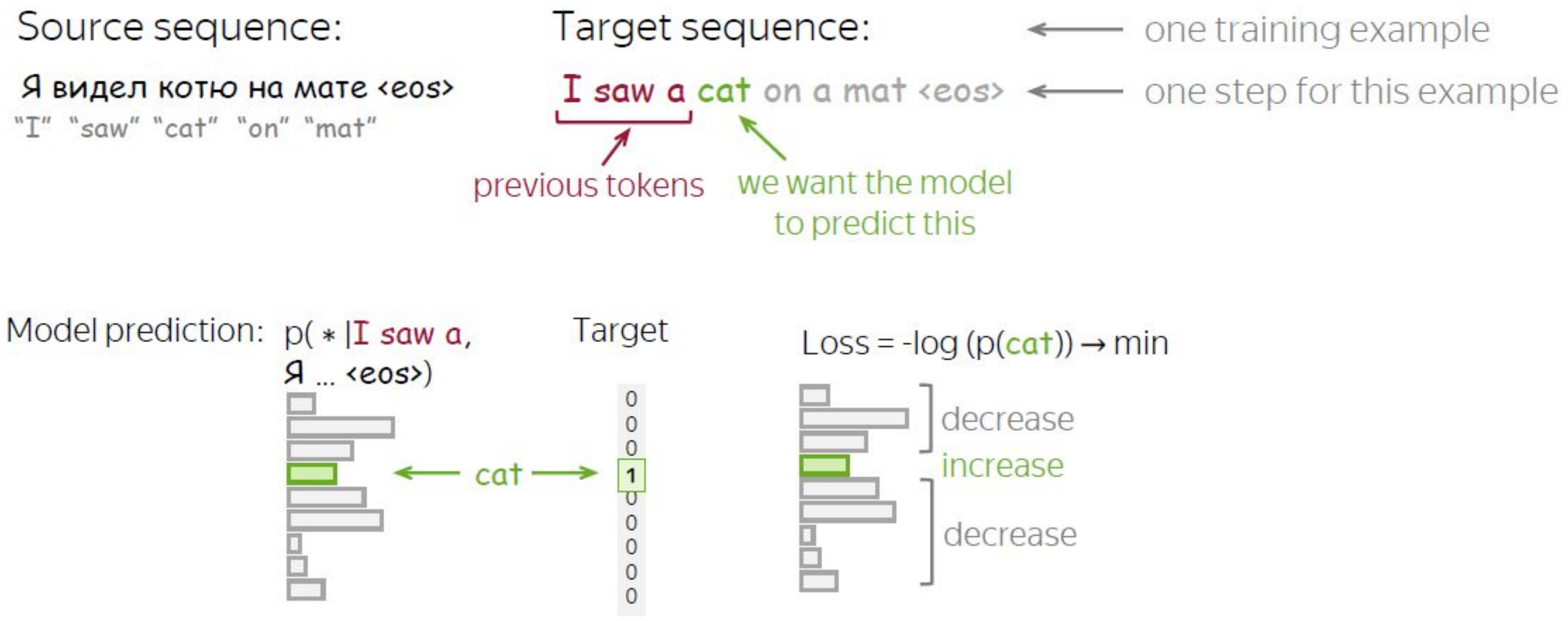
What does final state represents?



[Sequence to Sequence Learning with Neural Networks]



Training


Source sequence:
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Target sequence:
I saw a cat on a mat <eos>
one training example
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

Training

Formally, let's assume we have a training instance with the source $x = (x_1, \dots, x_m)$ and the target $y = (y_1, \dots, y_n)$. Then at the timestep t, a model predicts a probability distribution $p^{(t)} = p(*|y_1, \dots, y_{t-1}, x_1, \dots, x_m)$. The target at this step is $p^* = \mathrm{one\text{-}hot}(y_t)$, i.e., we want a model to assign probability 1 to the correct token, y_t, and zero to the rest.

The standard loss function is the cross-entropy loss. Cross-entropy loss for the target distribution p^* and the predicted distribution p is

$$Loss(p^*, p) = -p^* \log(p) = -\sum_{i=1}^{|V|} p_i^* \log(p_i).$$

Since only one of p_i^* is non-zero (for the correct token y_t), we will get

$$Loss(p^*, p) = -\log(p_{y_t}) = -\log(p(y_t|y_{<t}, x)).$$

Training


Encoder: read source
we are here
Source: Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Target: I saw a cat on a mat <eos>

Generating

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

Generating. Greed

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

Straightforward:

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

Generating. Greed Bad?

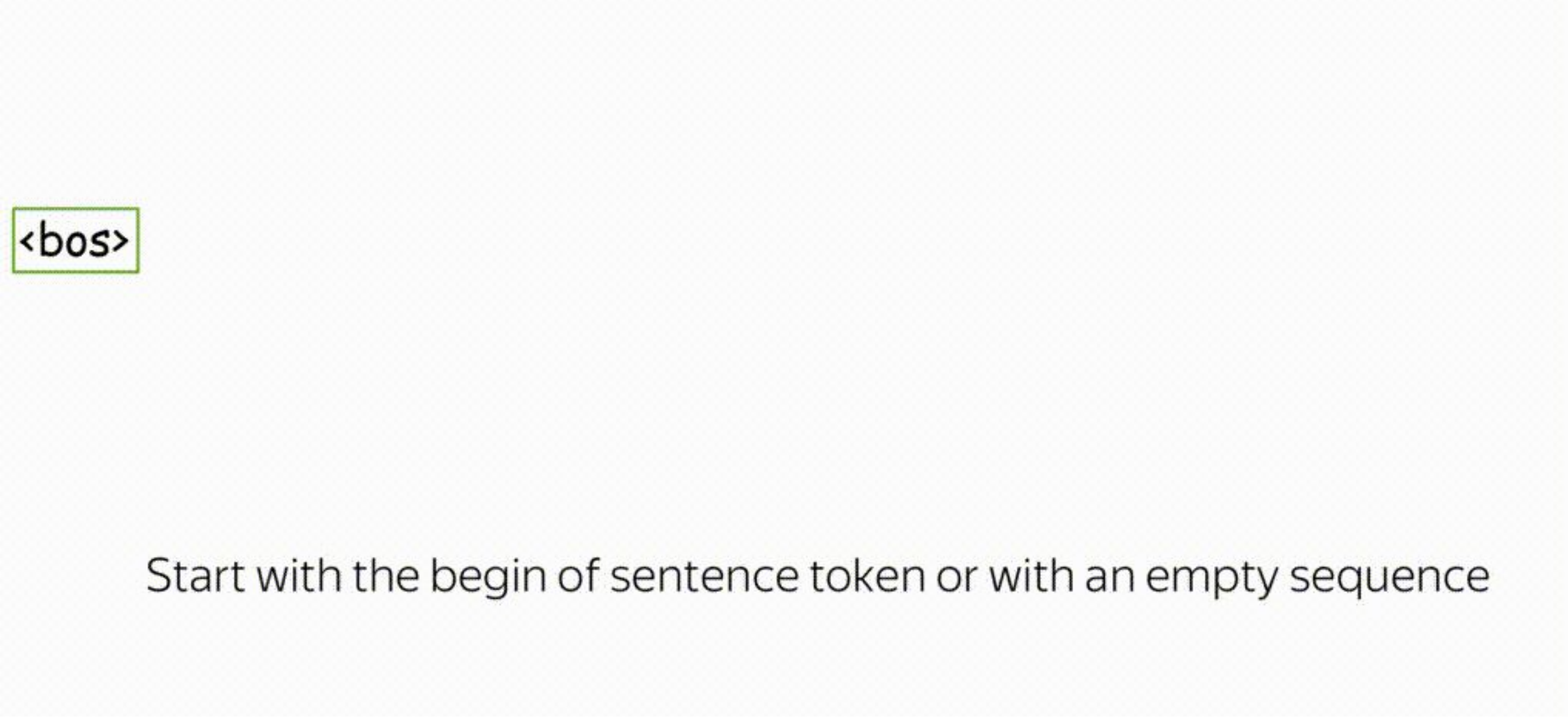
$$y' = \arg\max_y p(y|x) = \arg\max_y \prod_{t=1} p(y_t|y_{<t}, x)$$

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


<bos>
Start with the begin of sentence token or with an empty sequence