Foundations for Natural Language Processing

Text Generation and Encoder-Decoder Models

Ivan Titov
(with graphics/materials from Elena Voita)

School of informatics

Plan for today

Last time:

- Defined RNNs
- Used them for classification and language modeling
- Tried to understand what they capture

Today, we will

- see how to generate text from a neural language model
  (we will use RNNs but applicable to any other NN model)

- consider sequence-to-sequence tasks (e.g., machine translation)

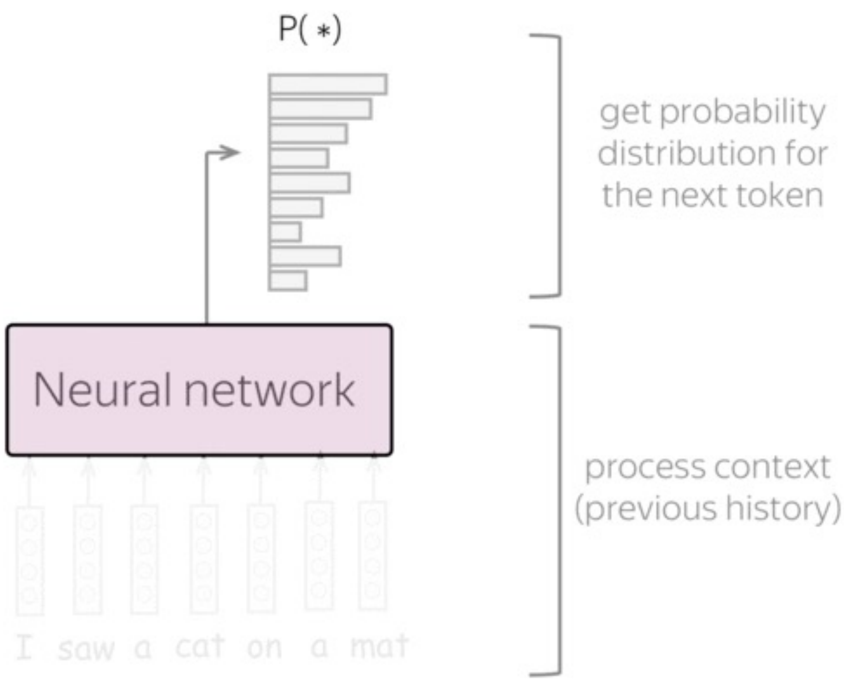
- introduce a basic form of *encoder-decoder models* for seq2seq

- discuss how to evaluate text generation systems

Recap: Neural Language Modeling

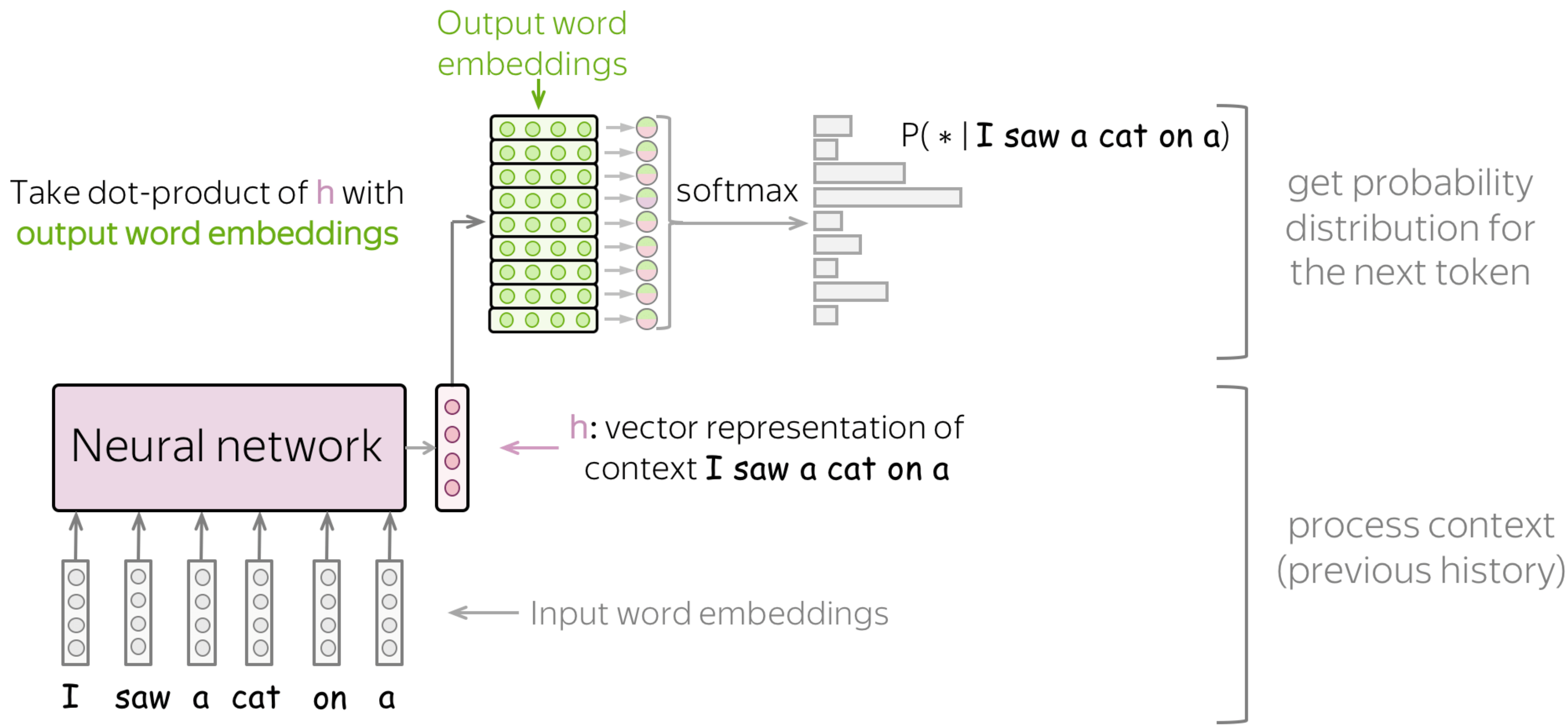
Neural language models has to:

1. *Produce a representation of the prefix*
2. *Generate a probability distribution over the next token*



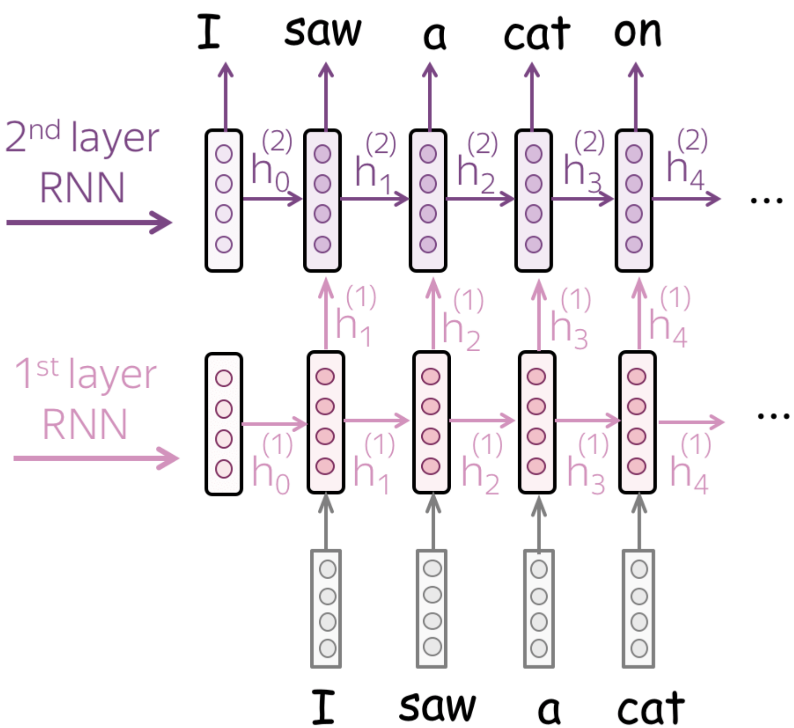
Predicting a word, given a prefix, it is just a classification problem!

Recap: High-level intuition for a language model



$$p(y_t|y_{<t}) = \frac{exp(h_t^T e_{y_t})}{\sum_{w \in V} exp(h_t^T e_w)}$$

Recap: Multi-layer RNN language model



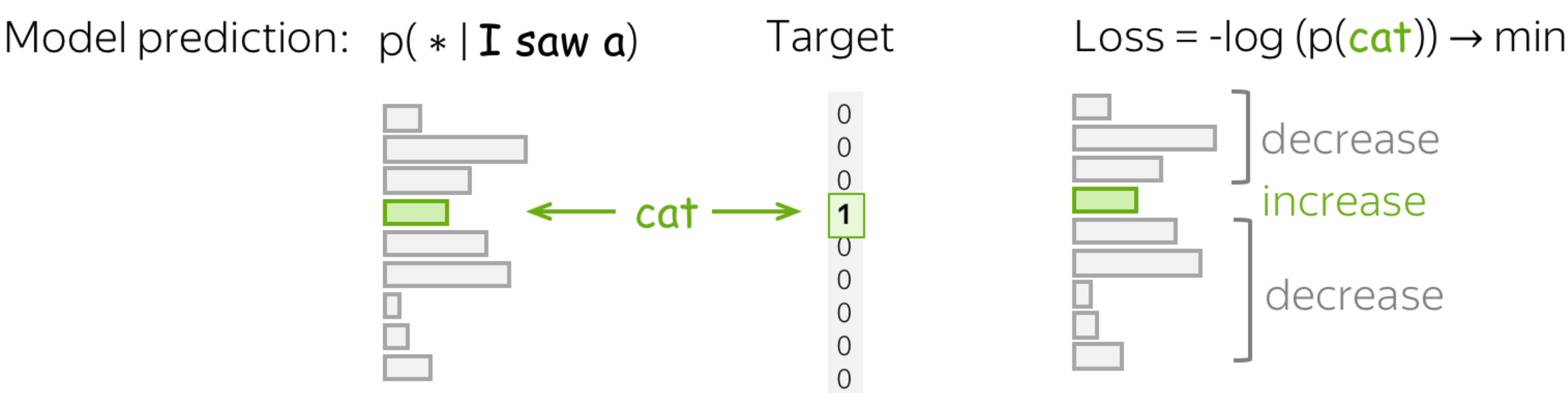
Recap: Training the language model

Training is done in a very much the same way as we train a classifier!

$$Loss = -\log(p(y_t|y_{<t}))$$

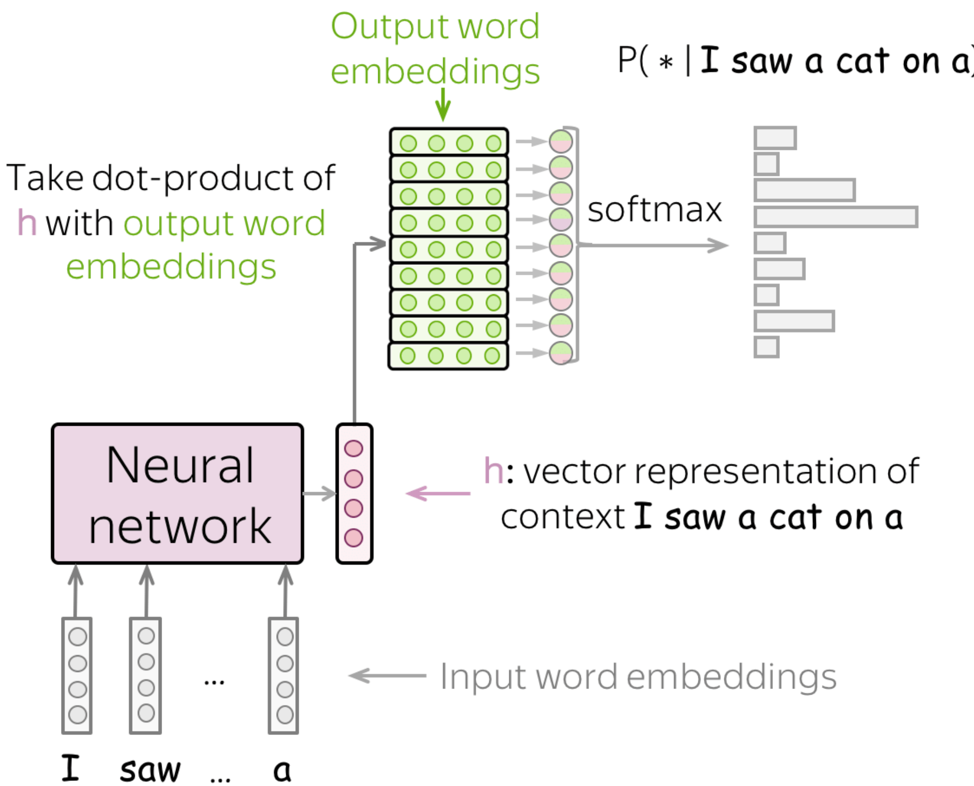
we want the model
to predict this

Training example: **I saw a** cat on a mat <eos>



Generating text

To generate text using a language model, you could just *sample* tokens from the probability distribution predicted by a model



Generating text

To generate text using a language model. you could just sample tokens from the probability distribution predicted by a model

I _____

Generating text: greedy decoding

An alternative to sampling from the distribution is selecting the most probable word at every step (called greedy decoding)

```
so even if the us , and the united states , the
hotel is located in the list of songs , you can
add them in our collection by this form . _eos_
```

```
alas , the hotel is located in the list of songs ,
you can add them in our collection by this form .
_eos_
```

Anything you notice about these samples?

Generating text: greedy decoding

An alternative to sampling from the distribution is selecting the most probable word at every step (called greedy decoding)

```
so even if the us , and the united states , the
hotel is located in the list of songs , you can
add them in our collection by this form . _eos_
```

```
alas , the hotel is located in the list of songs ,
you can add them in our collection by this form .
_eos_
```

Anything you notice about these samples?

They contain only frequent words and are boring!

Generating text: greedy decoding

An alternative to sampling from the distribution is selecting the most probable word at every step (called greedy decoding)

```
so even if the us , and the united states , the
hotel is located in the list of songs , you can
add them in our collection by this form . _eos_
```

```
alas , the hotel is located in the list of songs ,
you can add them in our collection by this form .
_eos_
```

Anything you notice about these samples?

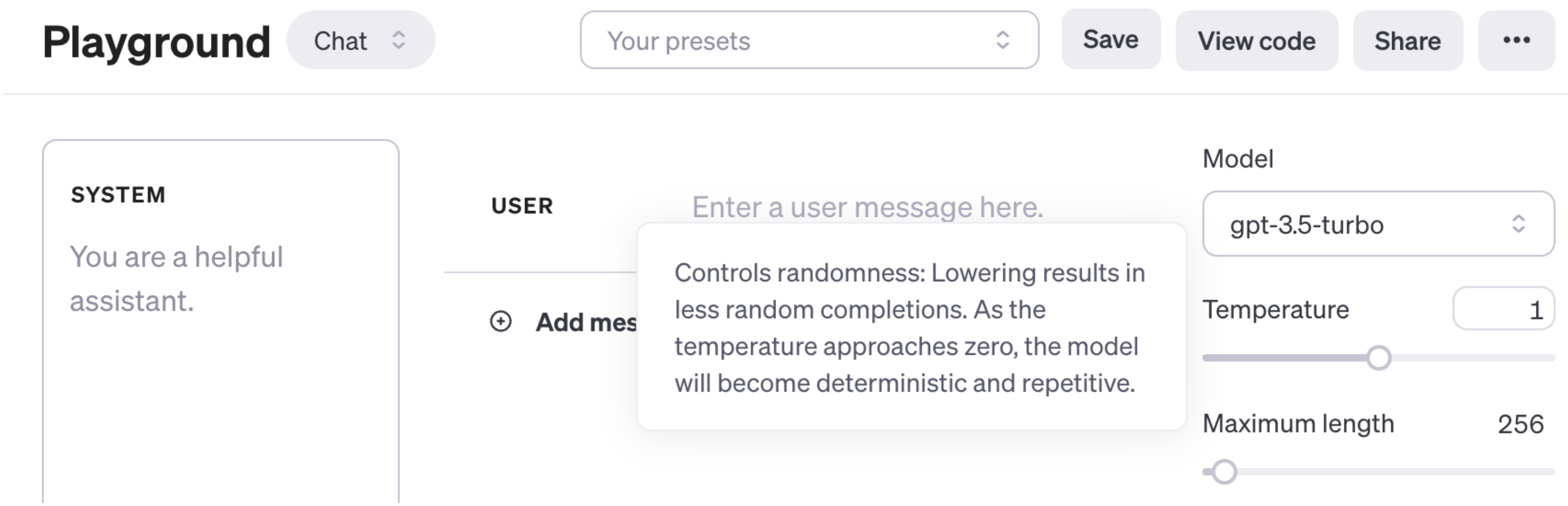
They contain only frequent words and are boring!

Greedy decoding is not generally a good way of producing text from a LM (but is a viable strategy when the output is more constrained, as in machine translation but we will talk later about it)

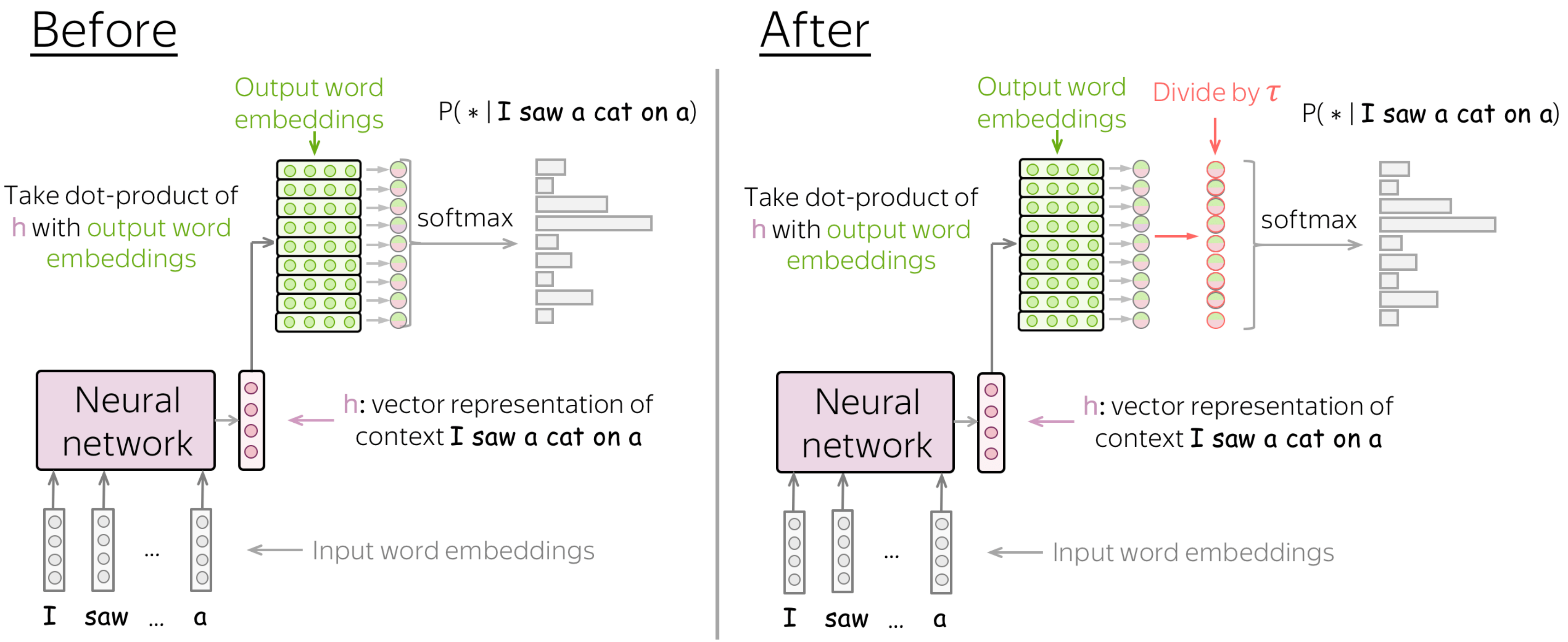
Controling diversity

We want the generated text to be coherent (or fluent) but also diverse (or interesting)

The standard way of controlling the generation characteristics is the softmax temperature parameter



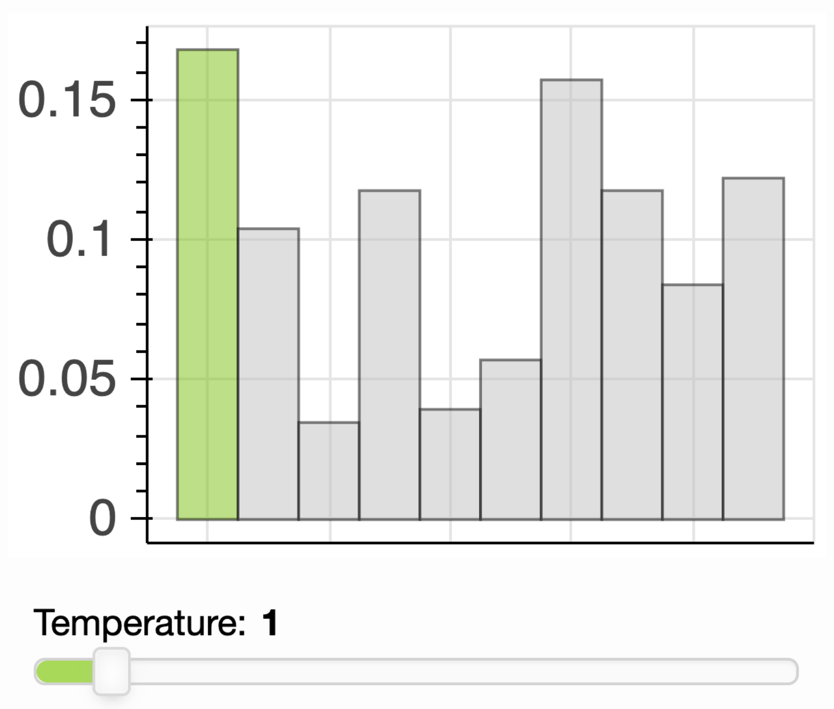
Temperature



Temperature – more formally

$$\frac{\exp(h^T w)}{\sum_{w_i \in V} \exp(h^T w_i)} \rightarrow \frac{\exp\left(\frac{h^T w}{\tau}\right)}{\sum_{w_i \in V} \exp\left(\frac{h^T w_i}{\tau}\right)}$$

τ - softmax temperature



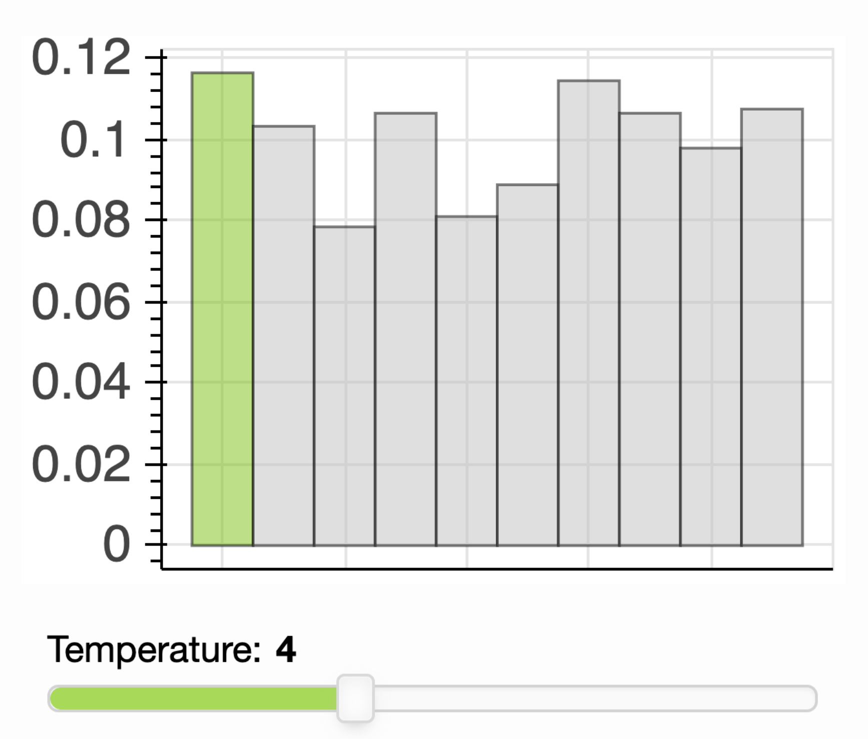
Temperature – more formally

$$\frac{\exp(h^T w)}{\sum_{w_i \in V} \exp(h^T w_i)} \rightarrow \frac{\exp\left(\frac{h^T w}{\tau}\right)}{\sum_{w_i \in V} \exp\left(\frac{h^T w_i}{\tau}\right)}$$

τ - softmax temperature

The most probably choice does not change, but with high temperatures, all the probabilities become closer to each other

The samples will become more diverse



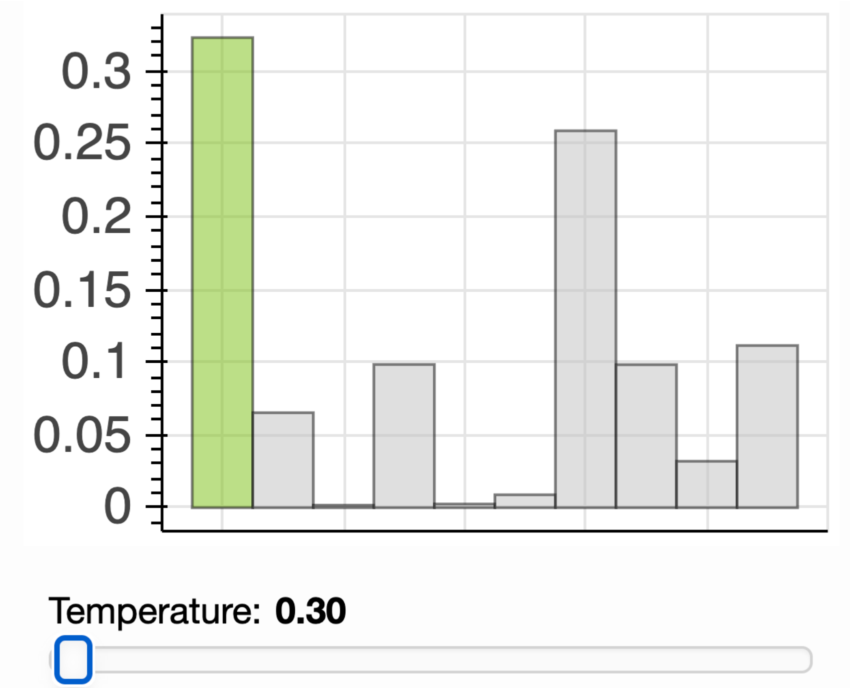
Temperature – more formally

$$\frac{\exp(h^T w)}{\sum_{w_i \in V} \exp(h^T w_i)} \rightarrow \frac{\exp\left(\frac{h^T w}{\tau}\right)}{\sum_{w_i \in V} \exp\left(\frac{h^T w_i}{\tau}\right)}$$

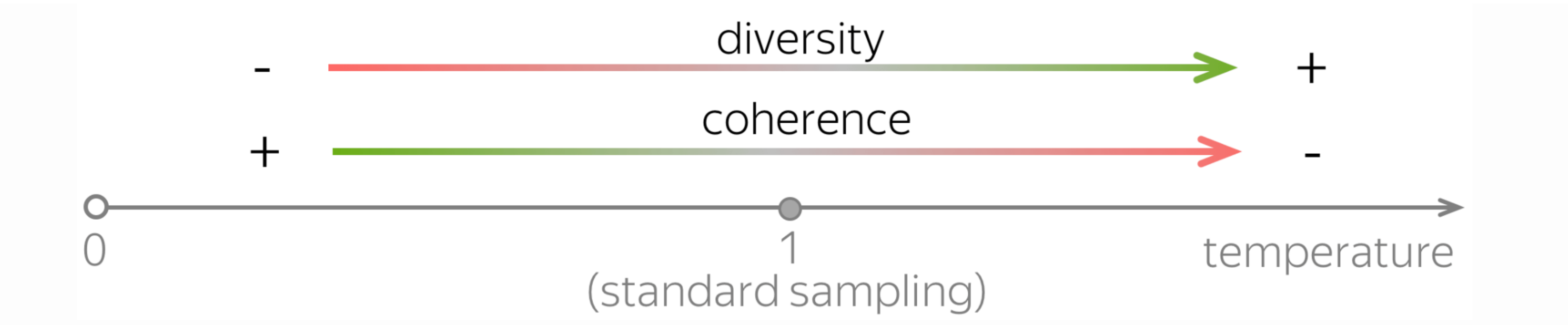
τ - softmax temperature

The most probably choice does not change, but with low temperatures, all the probabilities become further away from each other

The samples will become more similar



Trade-off: coherence vs diversity



The choice of temperature parameters is dependent on your goal / situation

There are smarter ways to sample from LMs (e.g., top-k sampling, / nucleus sampling)

Summary so far

We now know how to

- build a neural network for language modeling
- train it on a corpus
- generate text from a neural language model

.. but how do we use these ideas if we want to solve a task?

- generate a translation of an English sentence into Chinese
- produce a summary of a document
- generate an answer to a question

Sequence-to-Sequence modeling

x – input sentence,
y - its translation

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

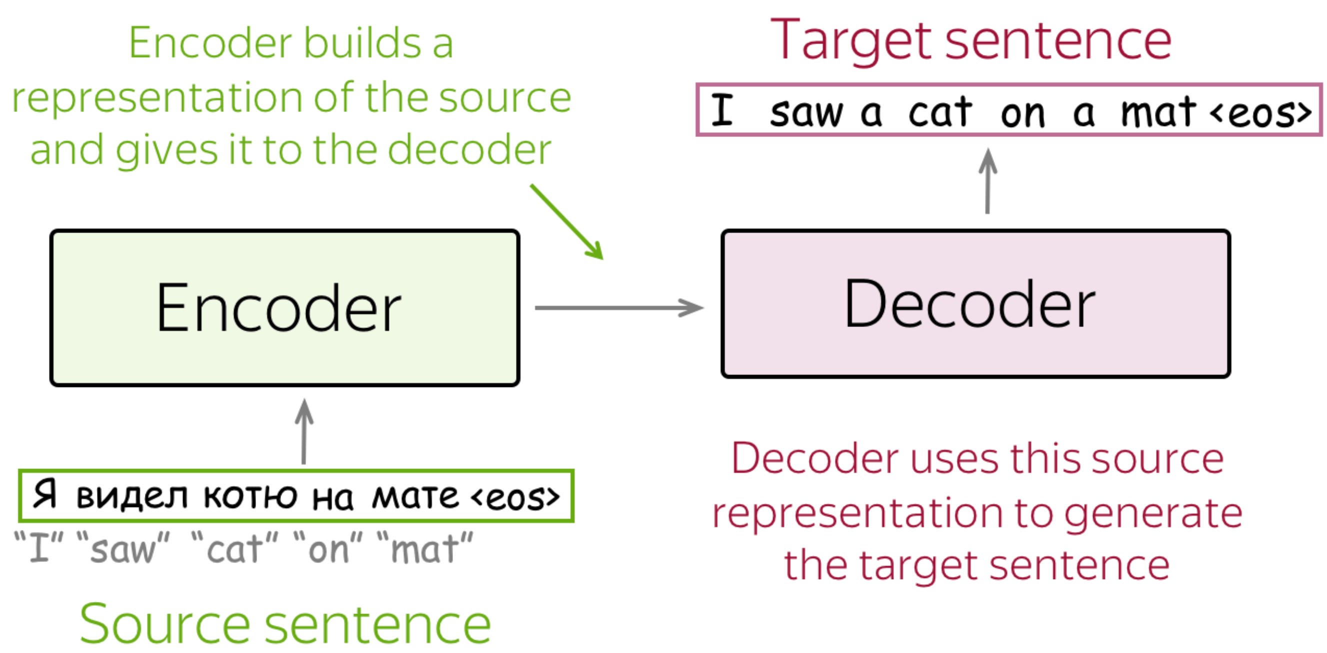
  How to find θ?

- search

  How to find the argmax?

Encoder-decoder framework

- encoder - reads source sequence and produces its representation;
- decoder - uses source representation from the encoder to generate the target sequence.



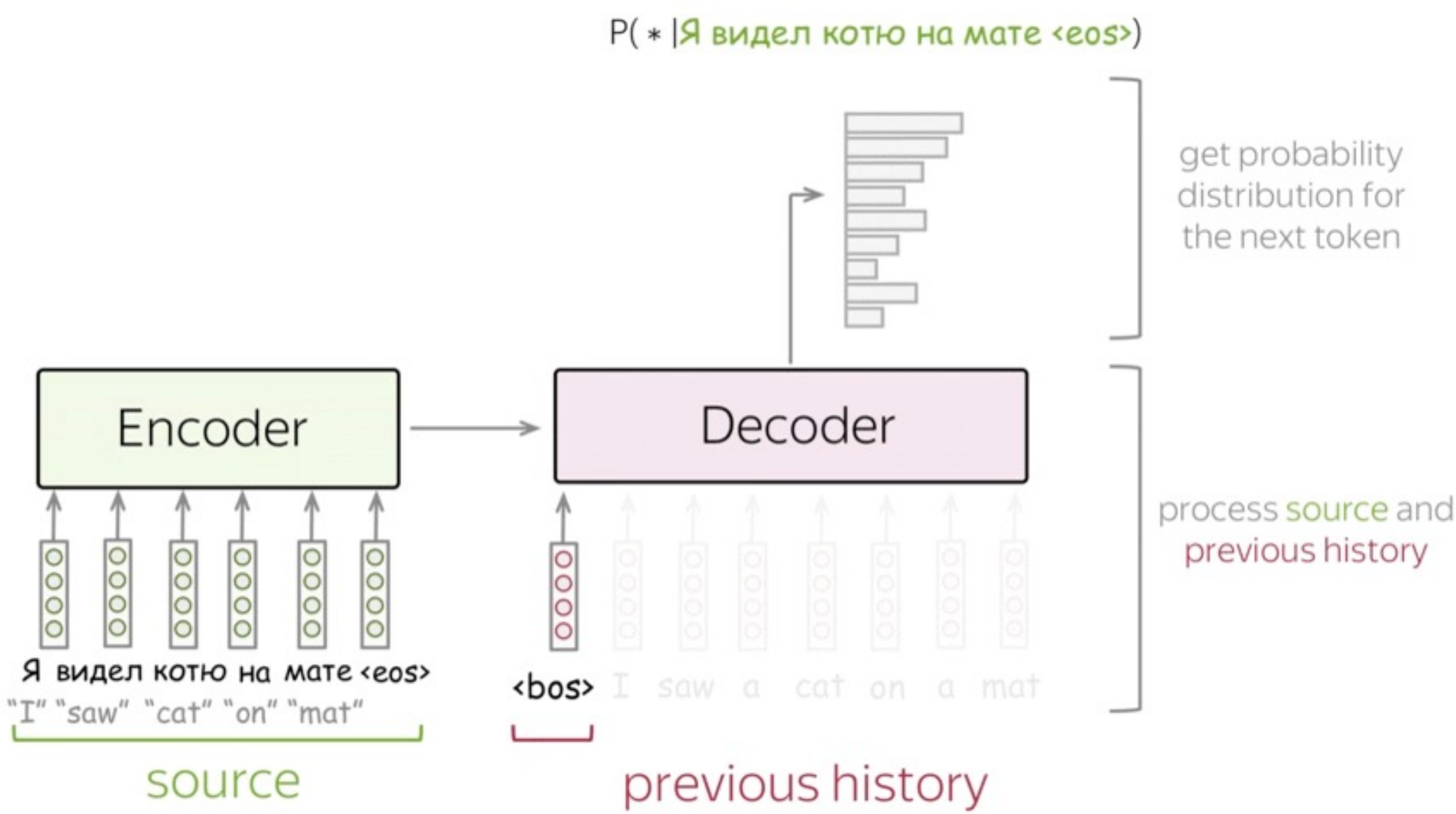
Language modeling perspective

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

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

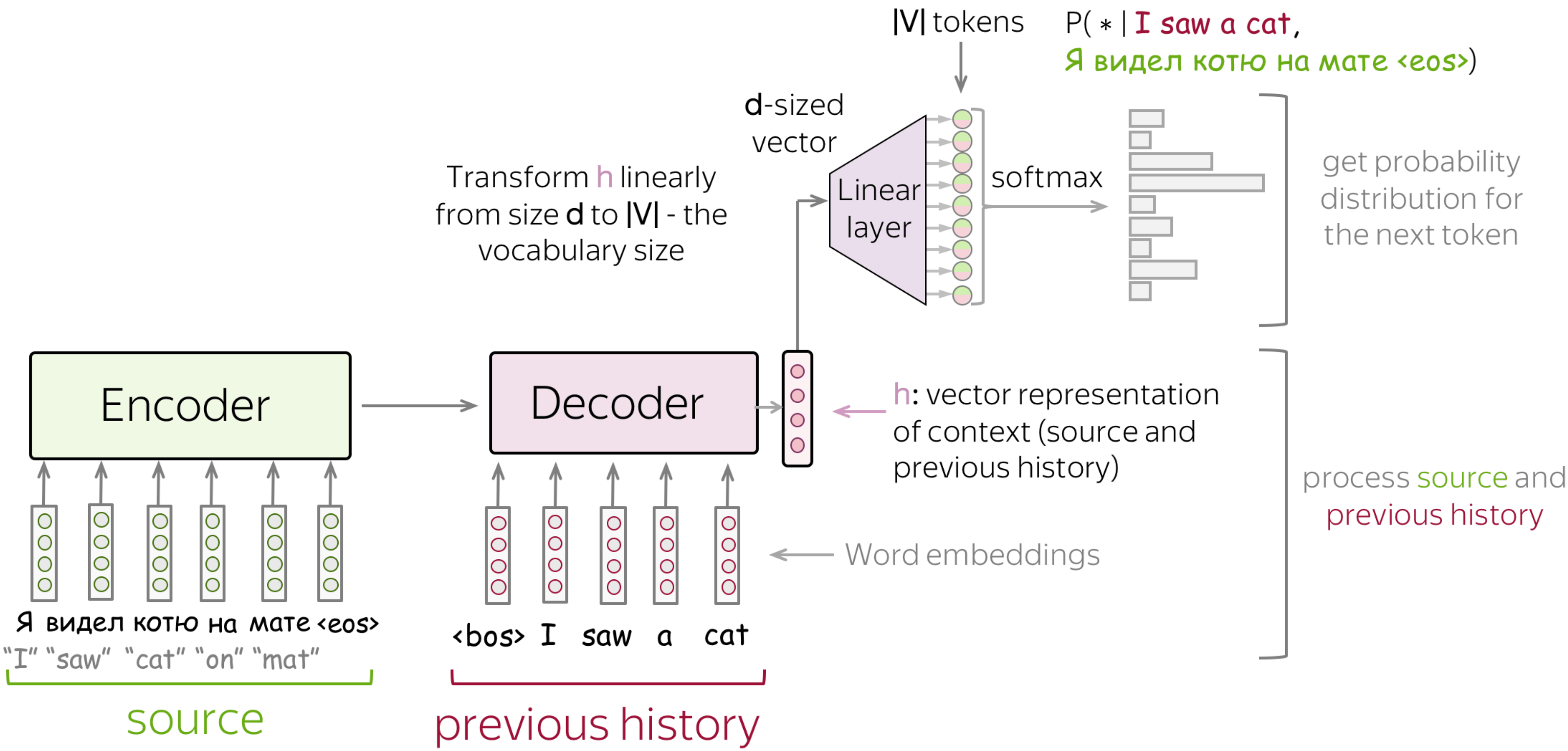
condition on source x

Encoder-decoder in action



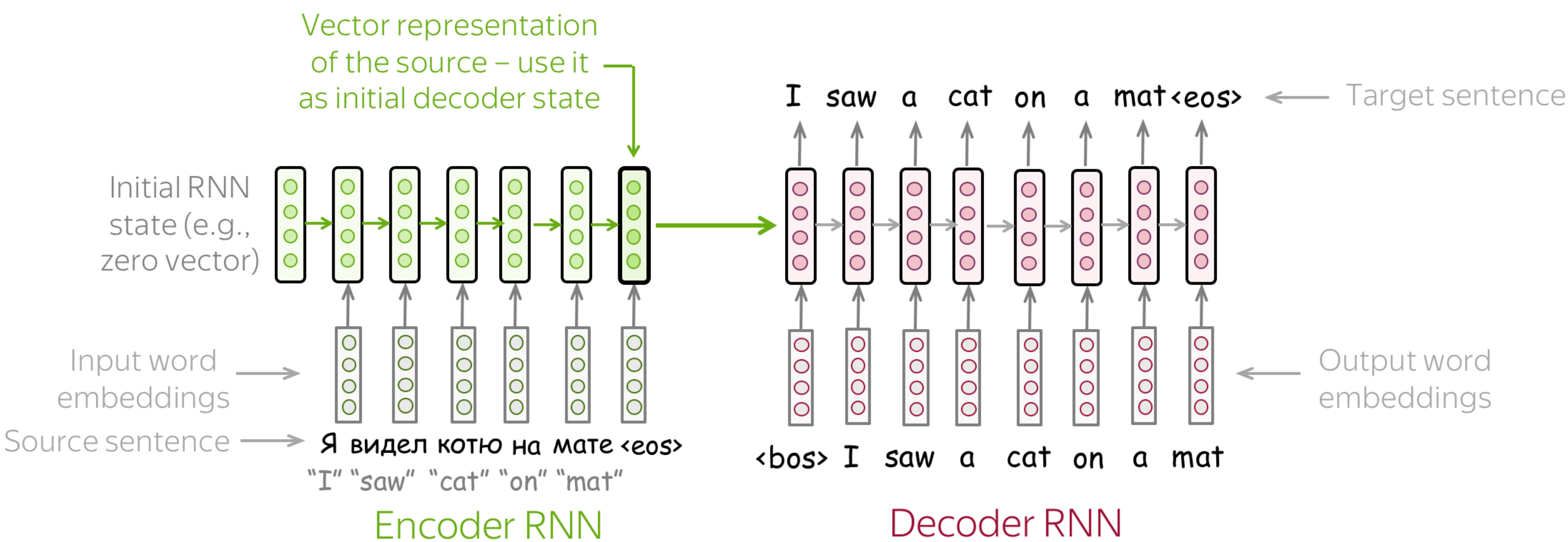
(video, not visible in pdf)

Encoder-decoder: under the hood



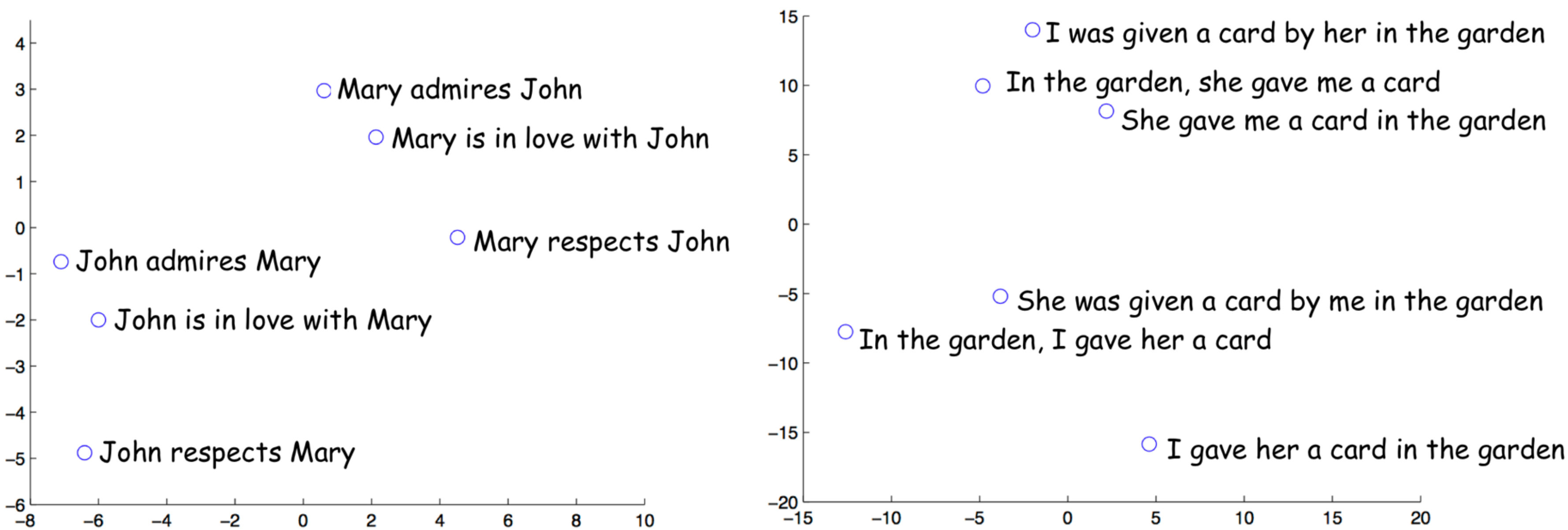
A lot like in language modeling, which was a lot like in text classification!

Simplest RNN-based Model:



Simplest RNN-based Model:

Last encoder states: near-paraphrases seem close in the space!



Sutskever et al. (2014)

Training

Source sequence:

Я видел котю на мате <eos>

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

Target sequence: ← one training example

$\underbrace{\text{I saw a}}_{\text{previous tokens}}$ cat on a mat <eos> ← one step for this example

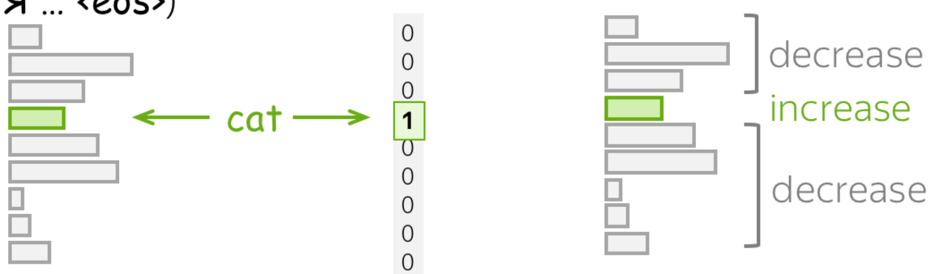
previous tokens

we want the model to predict this

Model prediction: p($*$ |I saw a, Я ... <eos>)

Target

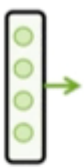
Loss = -log (p(cat)) → min



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

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>

(video, not visible in pdf)

Inference (aka decoding)

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

How to find the argmax?

The simplest idea – greedy decoding, at each step, pick the most likely token, but note:

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

Can we do better?

Beam search

Maintaining top hypotheses as you go

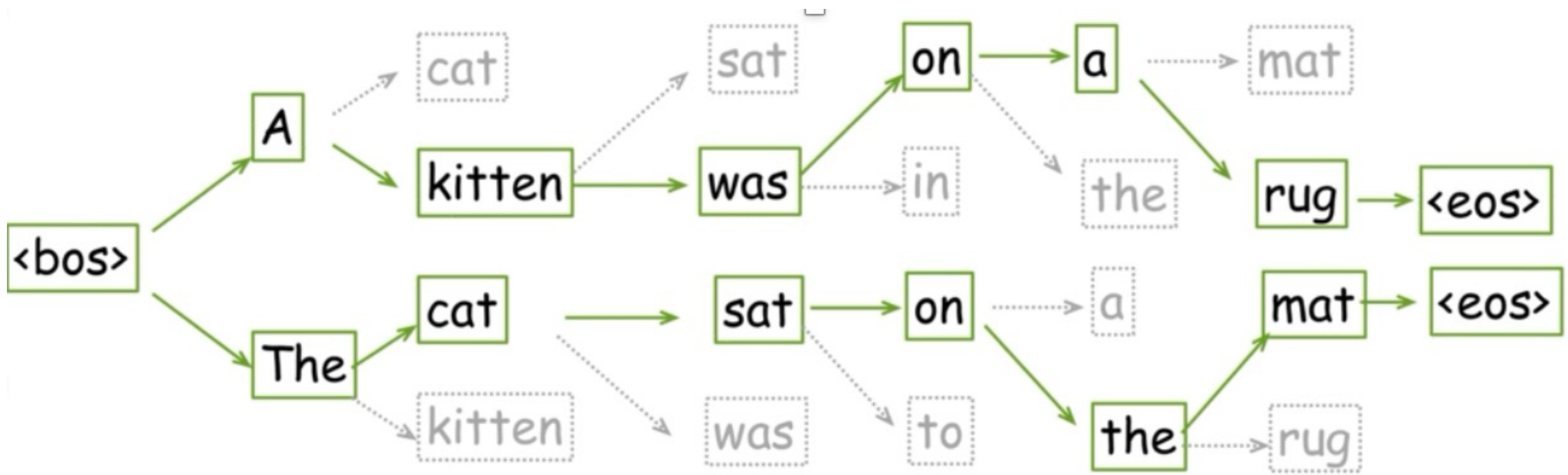
<bos>

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

(video, not visible in pdf)

Beam search

Maintaining top hypotheses as you go



All hypotheses are complete - generation ended

Why not sampling?

Actually, we can also sample in machine translation too (as with language modelling)

The risk is that a sample translation can deviate from the source sentence in meaning (i.e. hallucinate)

Evaluating text generation models

How to evaluate text generation?

Consider French to English machine translation

Source sentence: Le chat est assis sur le tapis

Human translation into English: The cat is on the carpet

How to evaluate text generation?

Consider French to English machine translation

Source sentence: Le chat est assis sur le tapis

Human translation into English: The cat is on the carpet

How can we design a metric which would score MT1 > MT2?

MT1: The cat is seated on the mat

MT2: The chat is assassinated on the tape

(We are looking into automatic extrinsic evaluation, recall Bracket F1 for parsing)

Idea: count overlapping ngrams - BLEU

Typically, we need more than 1 human (aka reference) translation per sentence to have reliable evaluation.

Let's focus on unigrams (individual tokens) for now

MT: The the the the the the the a

Reference 1: The cat is on the mat

Reference 2: There is a cat on the mat

(ignore capitalization for evaluation, i.e. treat 'The' and 'the' as the same word)

Idea: count overlapping ngrams - BLEU

MT: The the the the the the the a — 'the' appears 7 times

Reference 1: The cat is on the mat — 'the' appears 2 times

Reference 2: There is a cat on the mat — 'the' appears 1 time

Modified unigram precision: 2 / 7

Idea: count overlapping ngrams - BLEU

MT: The the the the the the the <u>a</u> — 'a' appears 1 time

Reference 1: The cat is on the mat — 'a' appears 0 times

Reference 2: There is <u>a</u> cat on the mat — 'a' appears 1 time

Modified unigram precision: (2 + 1) / (7 + 1) = 3 / 8

Aggregate over all unigrams in the MT ('candidate')

BLEU metric

Actual BLEU is considerably more complicated, as needs to

- aggregate over the entire test set
- aggregate over ngrams of different order (unigrams, bigrams, …)
- penalize short translation (remember from parsing: *precision* favors models producing short outputs)
- …

There are other ngram overlap metrics which can be more suitable for other text generation problems (e.g., ROUGE for summarization)

Ngram overlap metrics - weaknesses

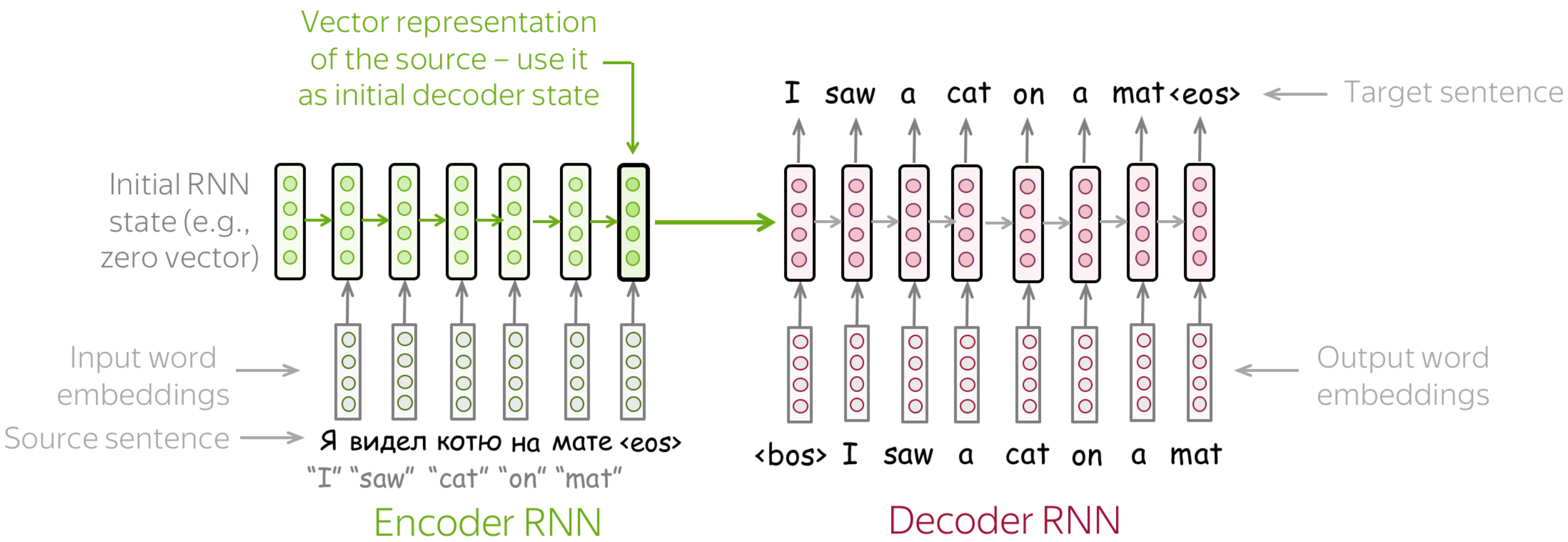
- do not account for lexical paraphrases (e.g., substituting words with their synonyms)
- even more problematic for long text generation (e.g., document machine translation)
- unreliable for tasks with less restricted outputs (e.g., generate “a scary novel about Edinburgh”)
- do not sufficiently penalize hallucinations

..

What can we do if they are so unreliable?

- Human evaluation (expensive, hard to relate to results of older experiments)
- Neural model-based metrics (e.g., BERT Score, GPTScore)
- Specialized metrics (e.g., FActScore for hallucinations)

Tokenization



We considered tokenization of sentences into 'words'
(whatever we mean by a 'word')

Tokenization

Tokenization

Word-level

- fixed vocabulary
- can process only a fixed number of words

Subword-level

- open vocabulary
- rare and unknown tokens are encoded as sequences of subword units

Instead of '*unrelated*', we get two tokens 'un@@' '*related*'

Subword segmentations reduces sparsity and results in a speeds-up (recall: softmax involves summation over all token types, few token typs -> faster computation)

Crucial for morphologically-rich languages

Standard segmentation algorithms rely on character ngram frequency (not on morphology (e.g., Byte-Pair Encoding)

Used in virtually any modern neural model

Summary

- Encoder-Decoder architecture for Seq2Seq
- Inference algorithms (greedy, beam-search, sampling, temperature,…)
- Evaluating text generation (e.g., BLEU)
- Subword tokenization