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

Today, we will

- see if can figure what RNN states captures
- see how to generate text from a neural language model
  (we will use RNNs but applicable to any other NN model)

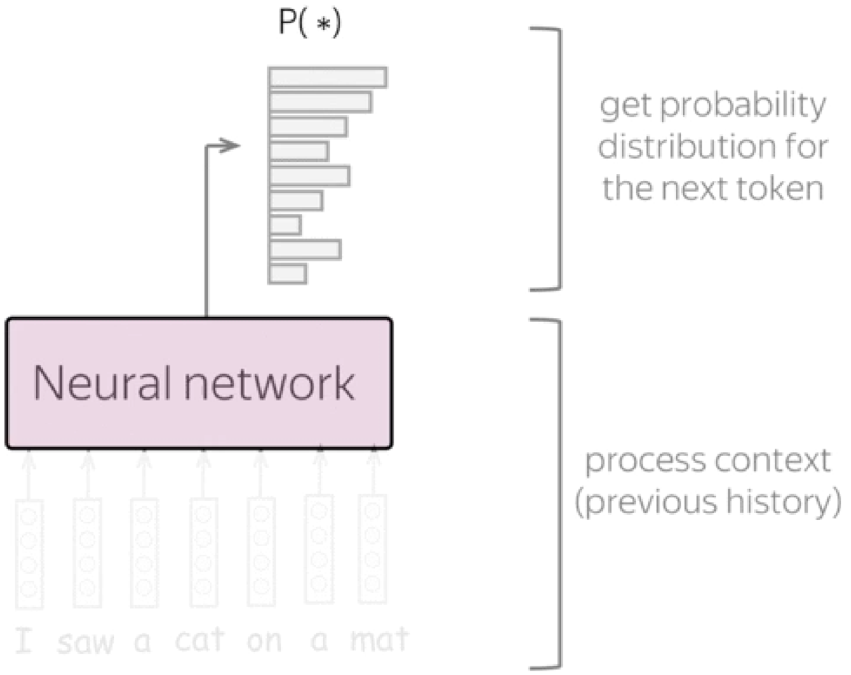
- consider sequence-to-sequence tasks (e.g., machine translation)

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

Recap: Neural Language Modeling

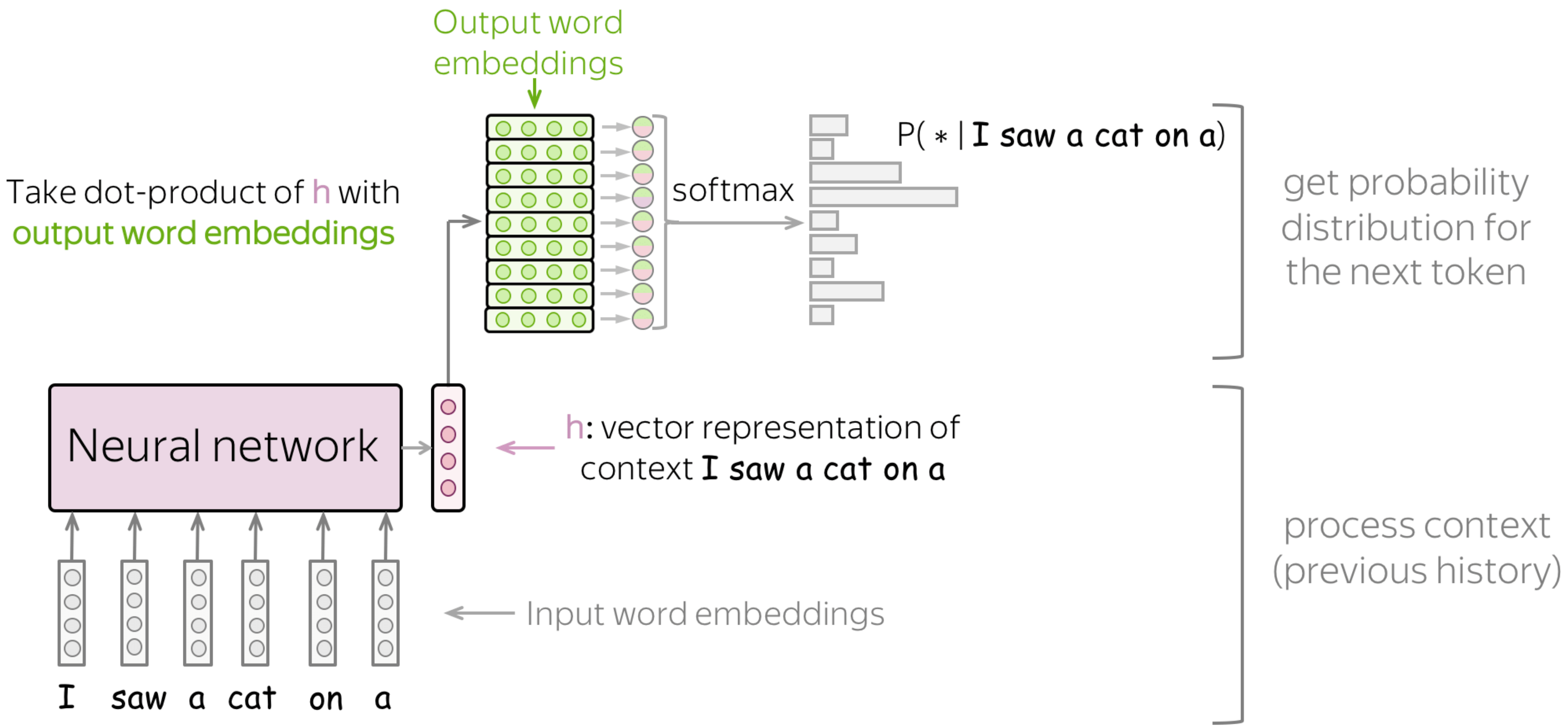
Neural language models has to:

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


P(*)
get probability distribution for the next token
Neural network
process context (previous history)
I saw a cat on a mat

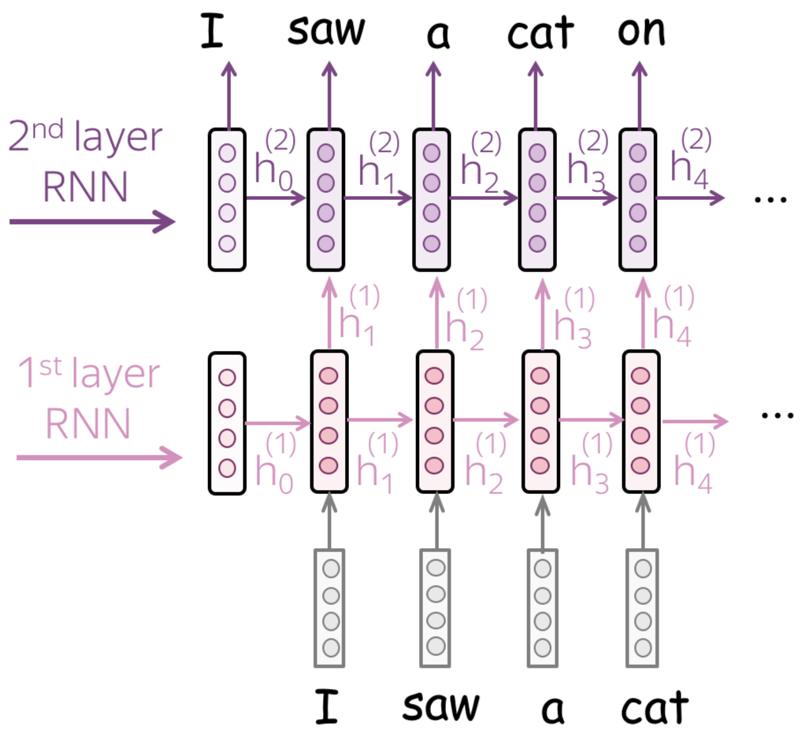
Predicting a word, given a prefix, it is just a classification problem!

Recap: High-level intuition for a language model


Output word embeddings
Take dot-product of h with output word embeddings
softmax
P(* | I saw a cat on a)
get probability distribution for the next token
Neural network
h: vector representation of context I saw a cat on a
Input word embeddings
process context (previous history)
I
saw
a
cat
on
a

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

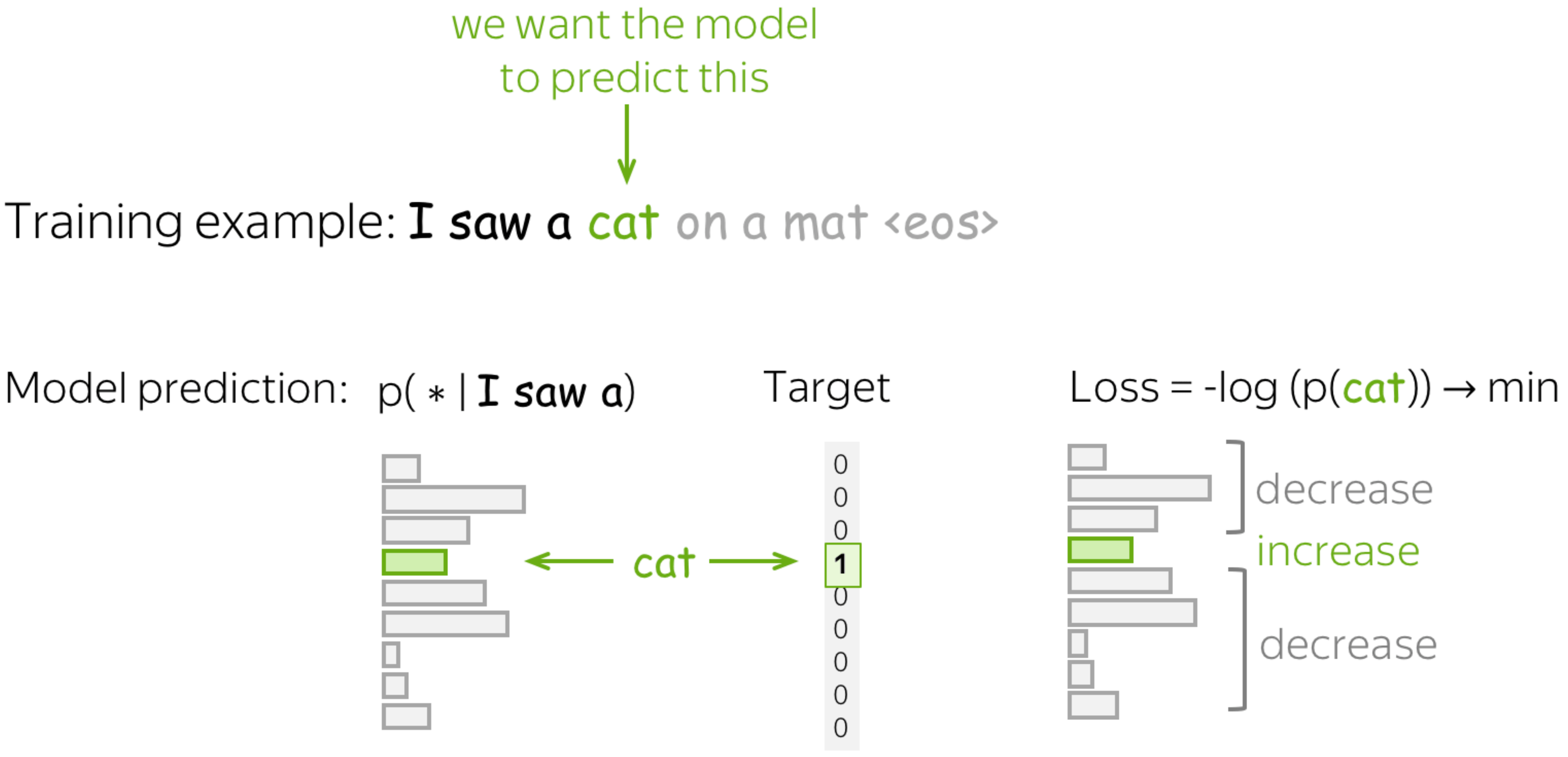
Recap: Multi-layer RNN language model



Recap: Training the language model

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

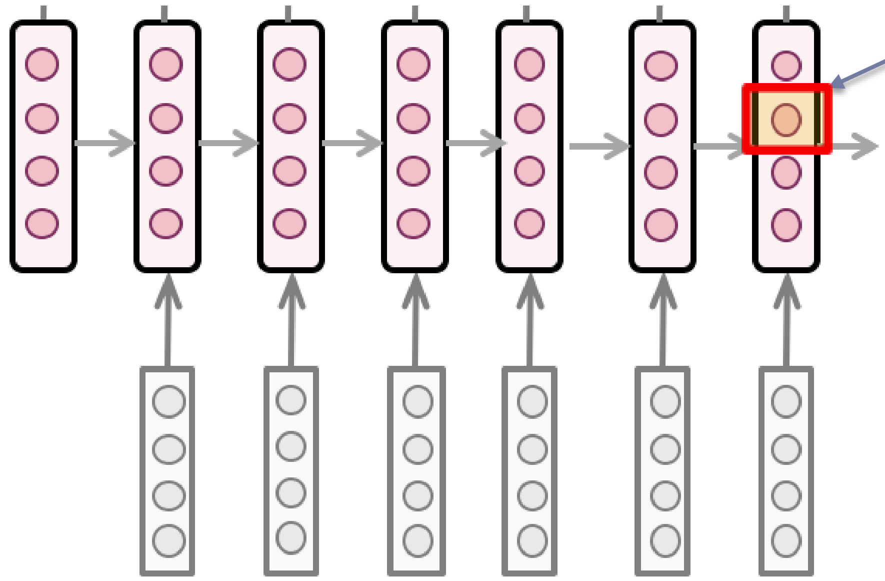
$$Loss = -\log(p(y_t|y_{<t}))$$


we want the model to predict this
Training example: I saw a cat on a mat <eos>
Model prediction: p(* | I saw a)
Target
Loss = -log (p(cat)) → min
cat
0
0
0
1
0
0
0
0
0
decrease
increase
decrease

Interpreting RNNs

What an RNN does capture in its state?

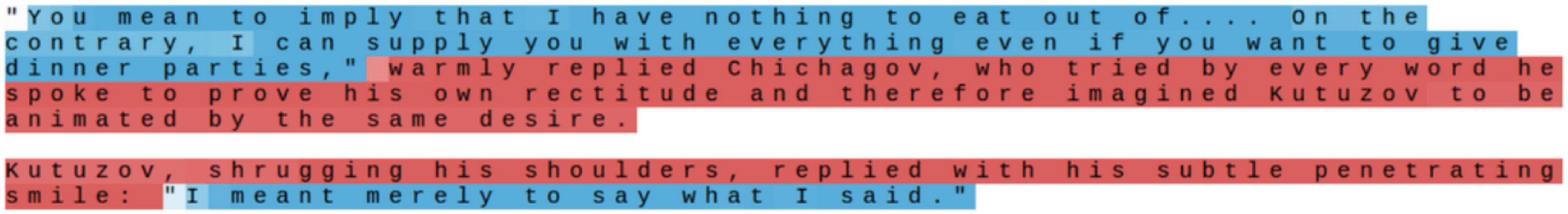
we will look in a specific neuron, and track its activation across a text



It is a character-level LM, i.e. models a sequence of characters (rather than words)
Trained on Tolstoy's War and Peace and the source code of Linux Kernel (in C)

Karpathy et el., 2015
https://arxiv.org/abs/1506.02078

What an RNN does capture in its state?

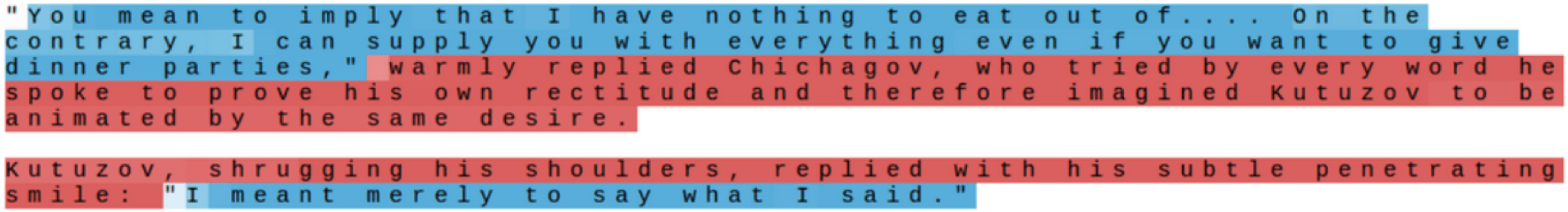

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

Any hypothesis what this neuron is doing?

Karpathy et el., 2015
https://arxiv.org/abs/1506.02078

What an RNN does capture in its state?



"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

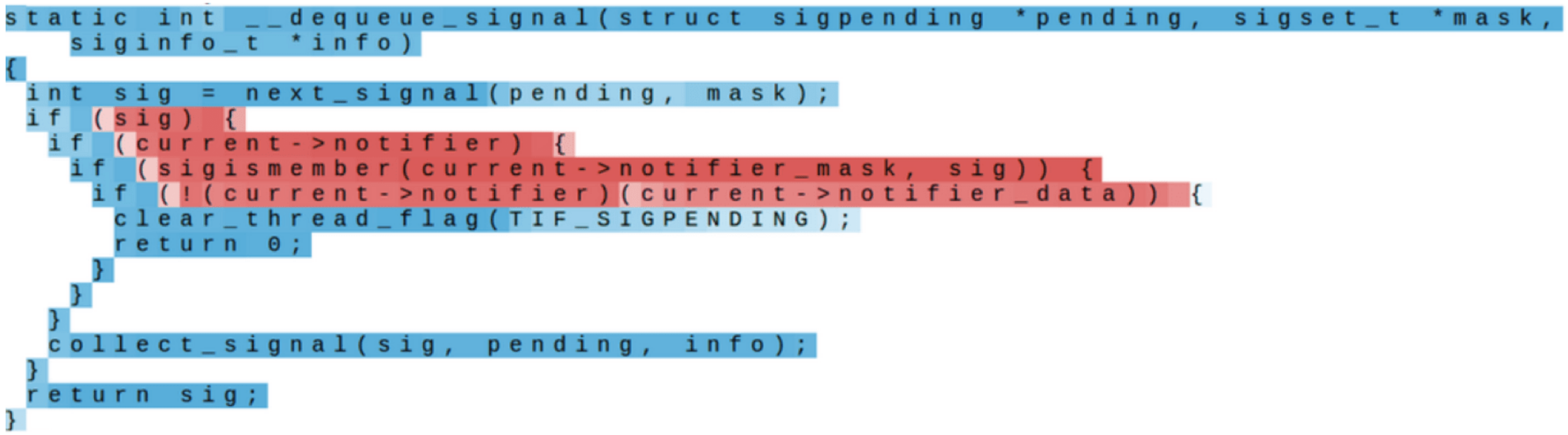
Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

Any hypothesis what this neuron is doing?

It activates within the quotes (“ .. ”)

Karpathy et el., 2015
https://arxiv.org/abs/1506.02078

What an RNN does capture in its state?

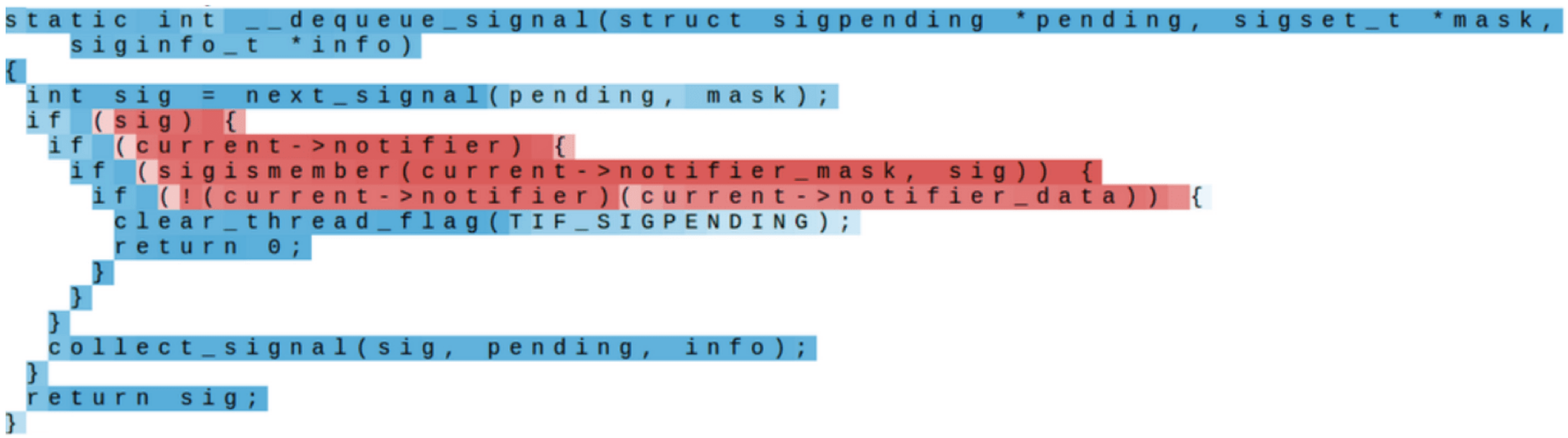


```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
   siginfo_t *info)
{
 int sig = next_signal(pending, mask);
 if (sig) {
  if (current->notifier) {
   if (sigismember(current->notifier_mask, sig)) {
    if (!(current->notifier)(current->notifier_data)) {
     clear_thread_flag(TIF_SIGPENDING);
     return 0;
    }
   }
  }
  collect_signal(sig, pending, info);
 }
 return sig;
}
```

Any hypothesis what this neuron is doing?

Karpathy et el., 2015
https://arxiv.org/abs/1506.02078

What an RNN does capture in its state?



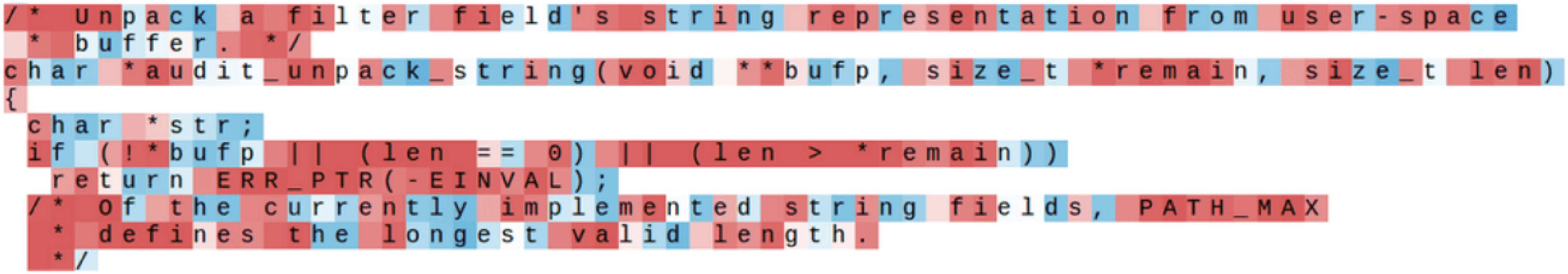
```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
   siginfo_t *info)
{
 int sig = next_signal(pending, mask);
 if (sig) {
  if (current->notifier) {
   if (sigismember(current->notifier_mask, sig)) {
    if (!(current->notifier)(current->notifier_data)) {
     clear_thread_flag(TIF_SIGPENDING);
     return 0;
    }
   }
  }
  collect_signal(sig, pending, info);
 }
 return sig;
}
```

Any hypothesis what this neuron is doing?

Activates within an if statement

Karpathy et el., 2015
https://arxiv.org/abs/1506.02078

What an RNN does capture in its state?



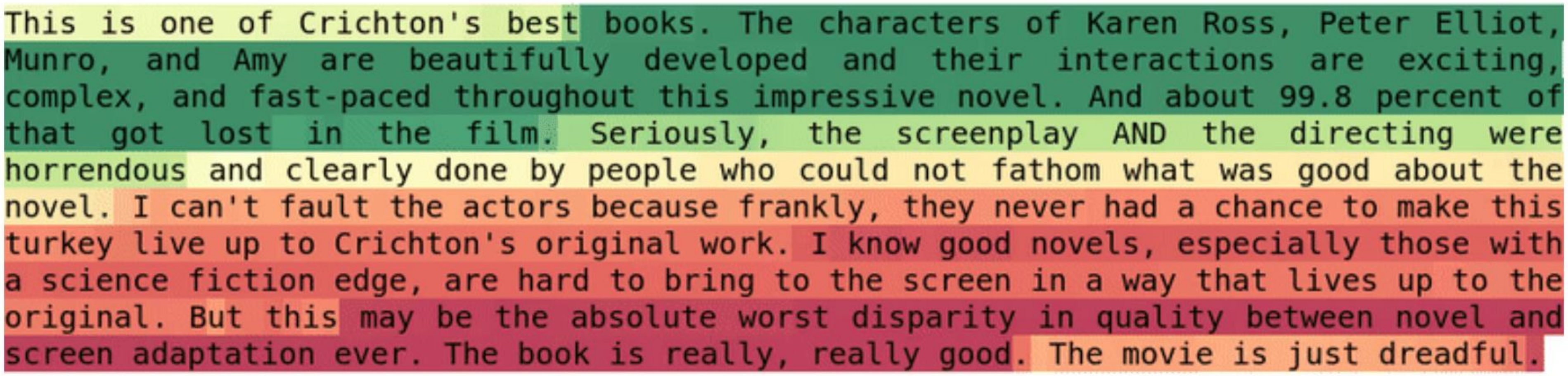
```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
```

Many neurons are not so easily interpretable

Karpathy et el., 2015
https://arxiv.org/abs/1506.02078

Sentiment neuron

This is from a much bigger LSTM model trained by OpenAI on Amazon reviews



This is one of Crichton's best books. The characters of Karen Ross, Peter Elliot, Munro, and Amy are beautifully developed and their interactions are exciting, complex, and fast-paced throughout this impressive novel. And about 99.8 percent of that got lost in the film. Seriously, the screenplay AND the directing were horrendous and clearly done by people who could not fathom what was good about the novel. I can't fault the actors because frankly, they never had a chance to make this turkey live up to Crichton's original work. I know good novels, especially those with a science fiction edge, are hard to bring to the screen in a way that lives up to the original. But this may be the absolute worst disparity in quality between novel and screen adaptation ever. The book is really, really good. The movie is just dreadful.

They could also switch the sentiment at generation time to change the sentiment of an utterance (call *interventions*)

https://openai.com/research/unsupervised-sentiment-neuron

Do RNNs learn syntax?

Ngrams are not able to capture syntactic agreement, but can RNNs?

Sam/Dogs sleeps/sleep soundly
Sam, who is my cousin, sleeps soundly
Dogs often stay at my house and sleep soundly
Sam, the man with red hair who is my cousin, sleeps soundly

The roses in the vase by the door __?__   Competing answers: is, are

P(The roses in the vase by the door are)

P(The roses in the vase by the door is)

Is the correct answer ranked higher?

P(...are) > P(...is)?

Contrastive evaluation

is/are

The roses _?_ — Simple: no attractors

The roses in the <u>vase</u> _?_ — Harder: 1 attractor

The roses in the <u>vase</u> by the <u>door</u> _?_ — Harder: 2 attractors

Attractors: nouns with different number than the subject

Short summary:

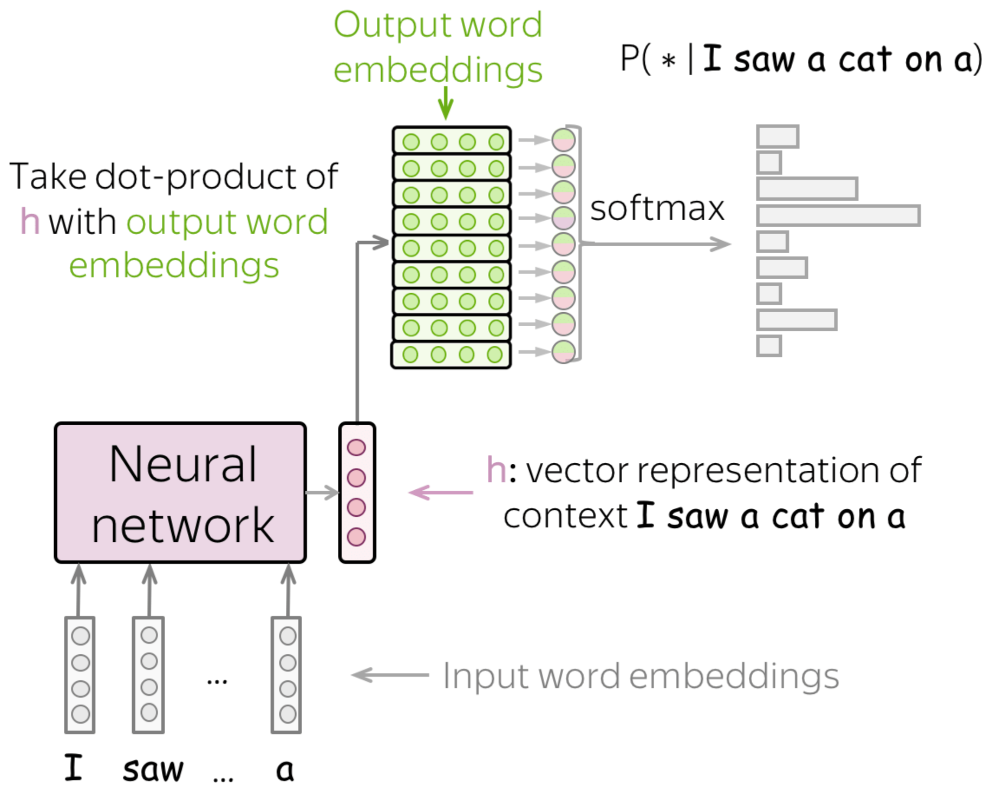
- need to be careful to prevent the model from relying on non-syntactic 'shortcuts'
- LSTMs models trained for language modeling were not as strong in that evaluation (but more powerful models will be)

Linzen, Dupoux and Goldberg, 2016
https://aclanthology.org/Q16-1037.pdf

Generating text from a (R)NN language model

Generating text

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


Output word embeddings
P(∗ | I saw a cat on a)
Take dot-product of h with output word embeddings
softmax
Neural network
h: vector representation of context I saw a cat on a
Input word embeddings
I
saw
...
a

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_
```

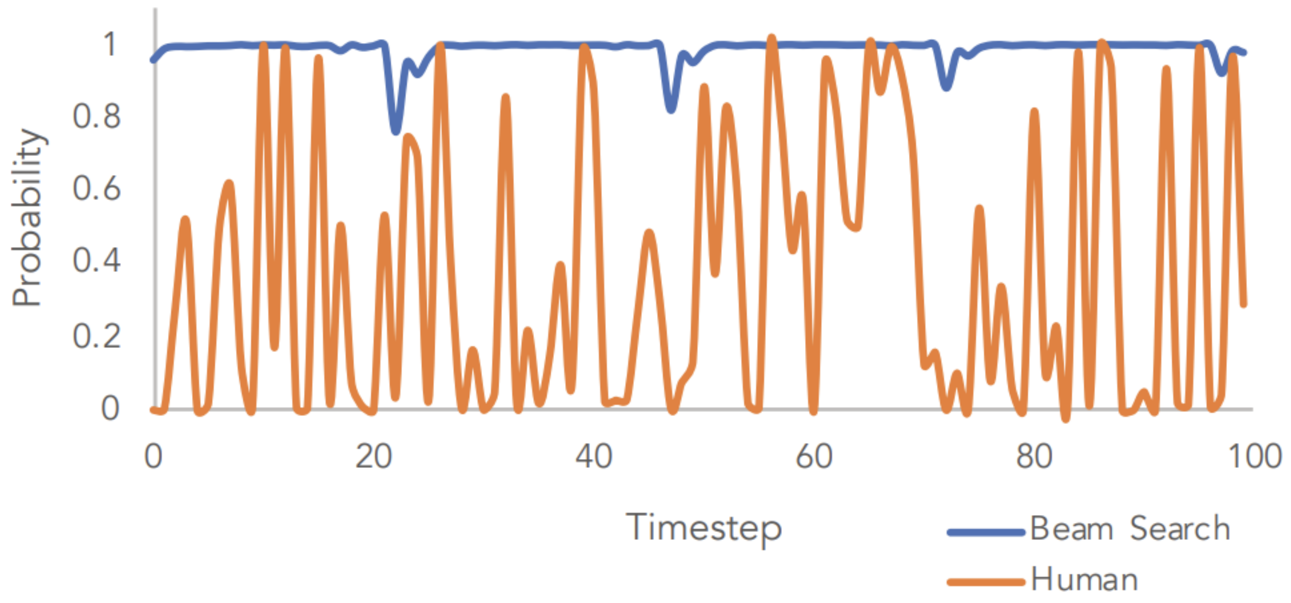
```
alas , the hotel is located in the list of songs ,
you can add them in our collection by this form .
_eos_
```

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)

Distributions in human written texts



Humans generate "surprising" token, though many tokens are fairly predictable

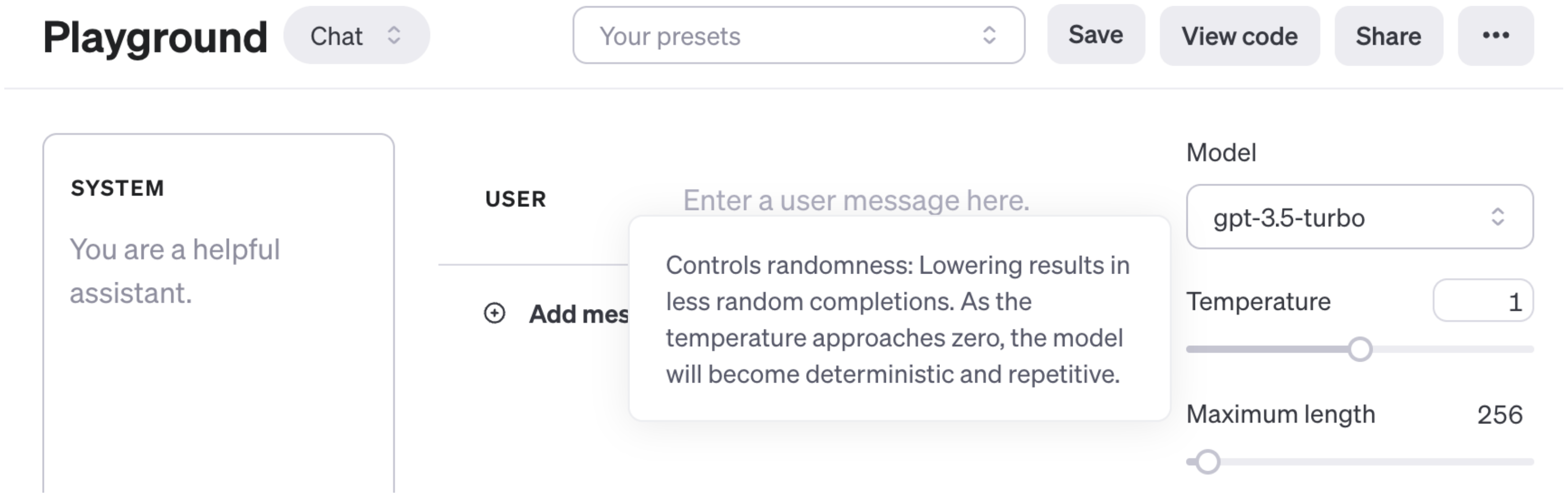
(We will discuss *beam search* later; think of it as of greedy decoding)

Image from Holtzman et al. (ICLR 2020)

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



You are a helpful assistant.

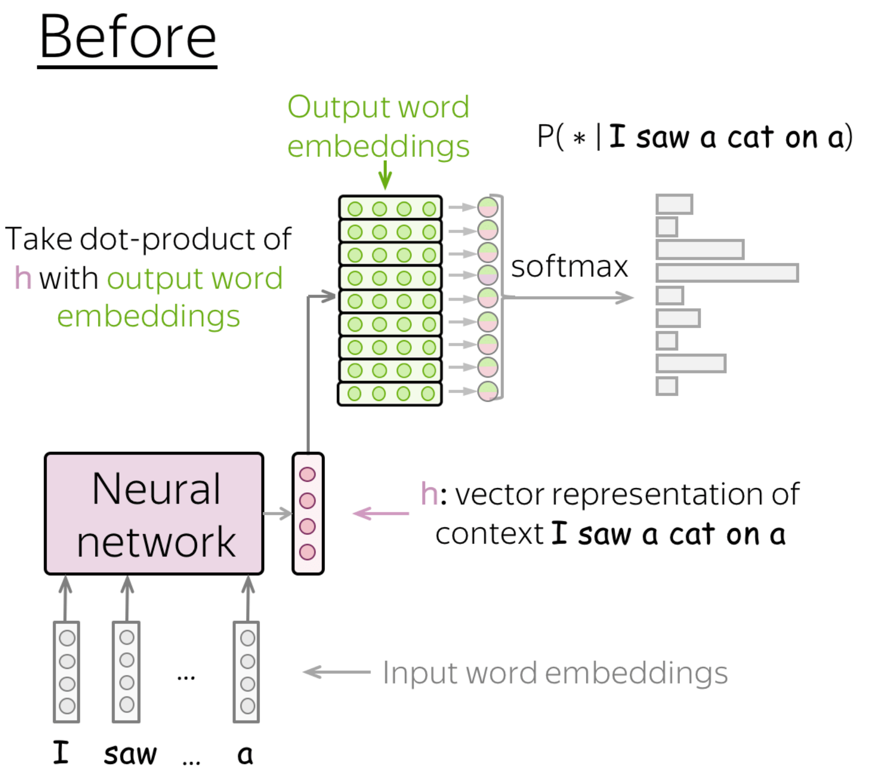
Controls randomness: Lowering results in less random completions. As the temperature approaches zero, the model will become deterministic and repetitive.

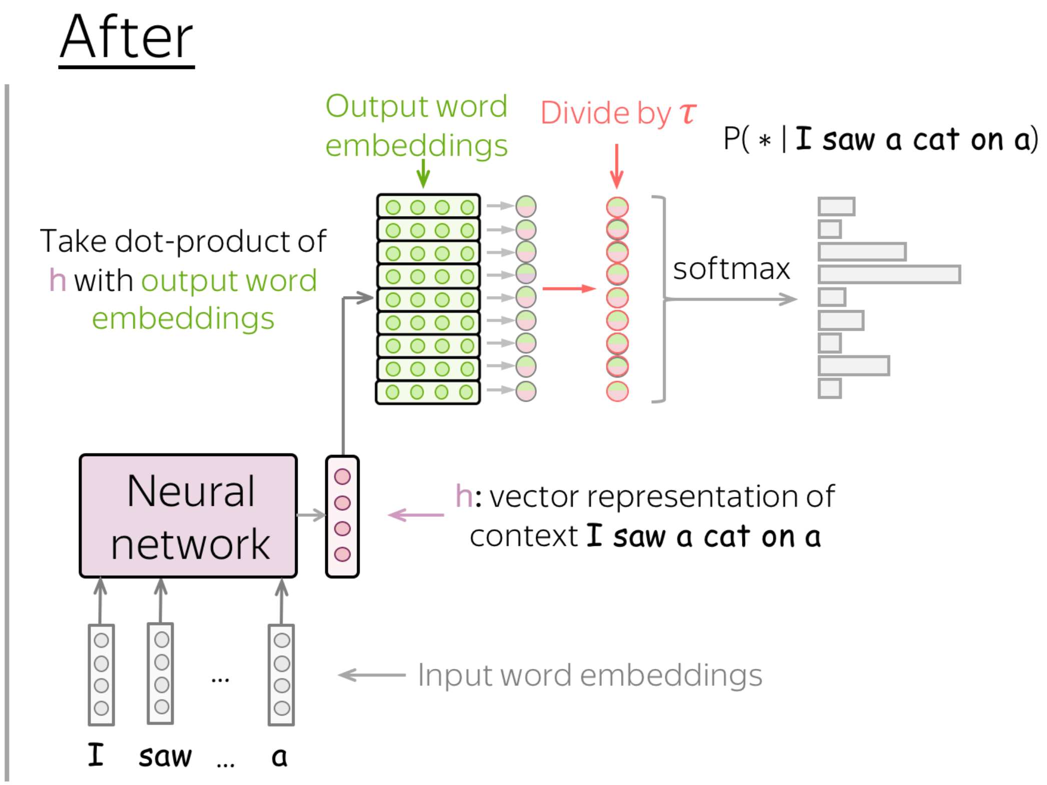
Model: gpt-3.5-turbo

Temperature: 1

Maximum length: 256

Temperature

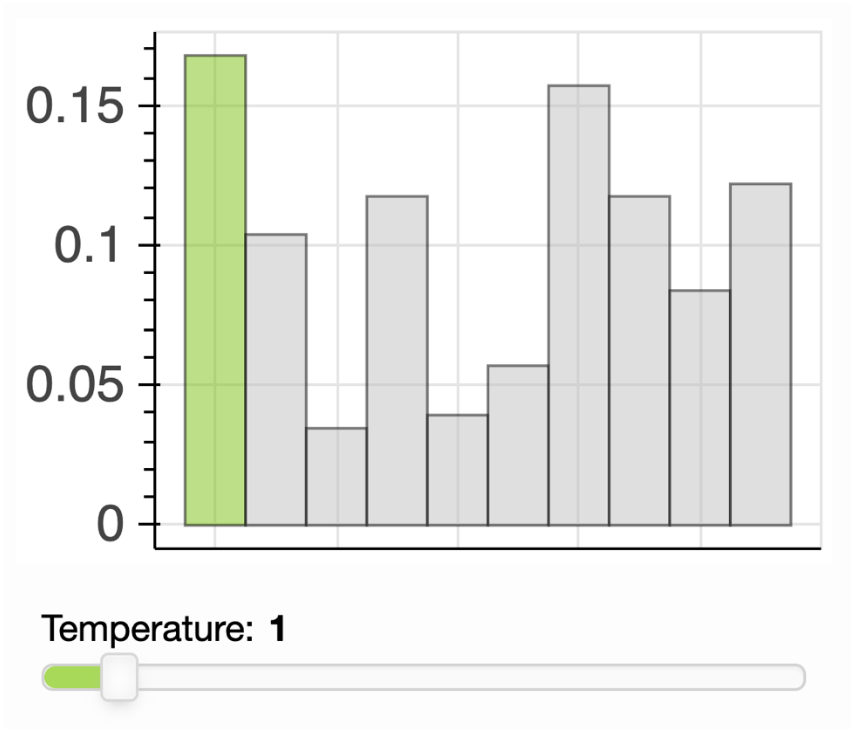

Before
Output word embeddings
P(∗ | I saw a cat on a)
Take dot-product of h with output word embeddings
softmax
Neural network
h: vector representation of context I saw a cat on a
Input word embeddings
I
saw
…
a



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: **1**

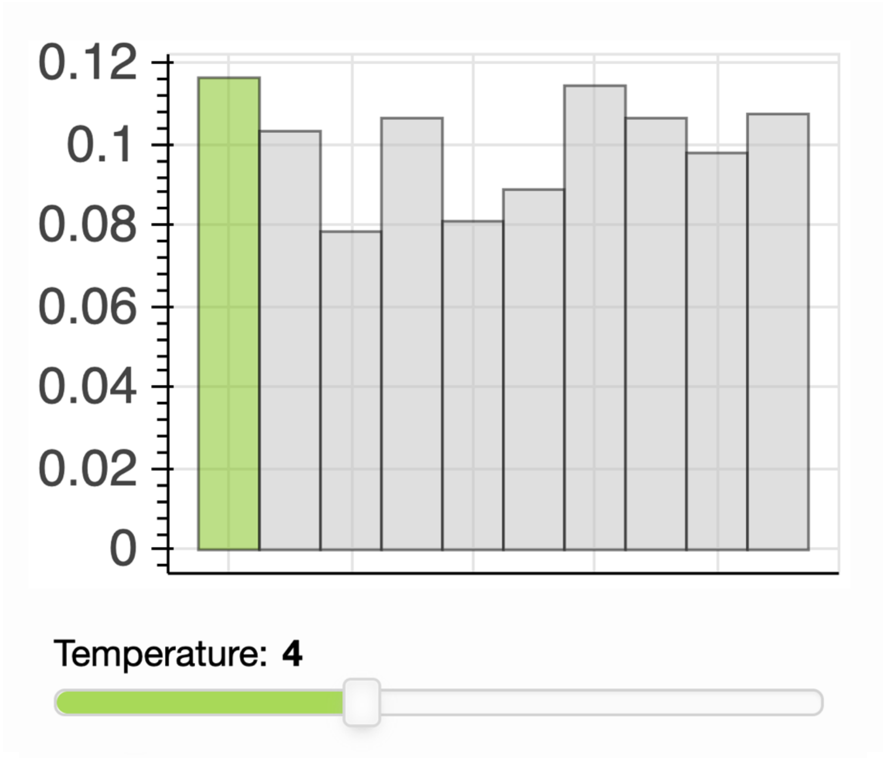
Temperature – more formally

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

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



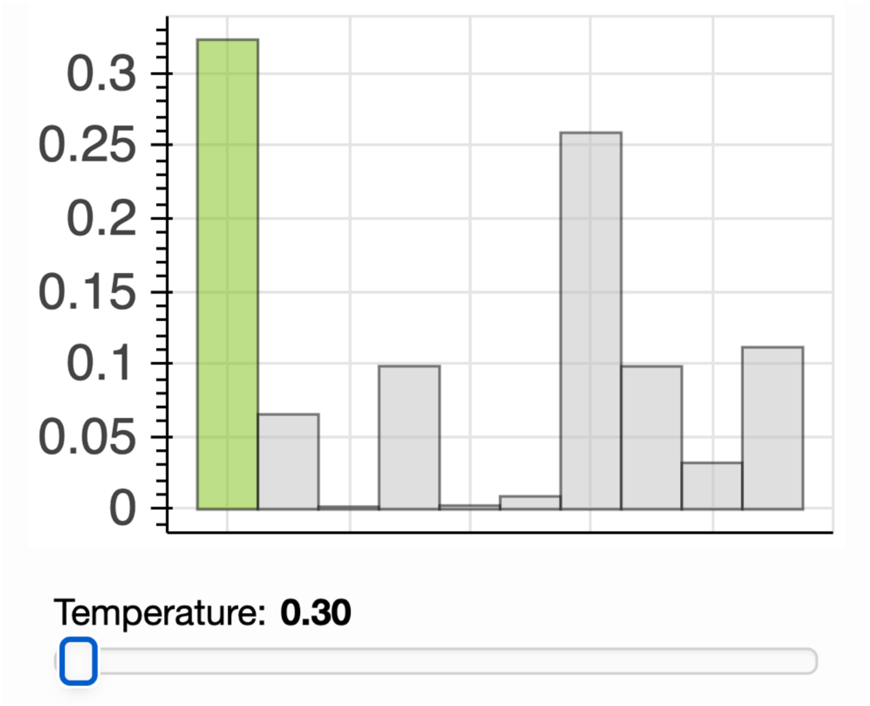
Temperature – more formally

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

τ - softmax temperature

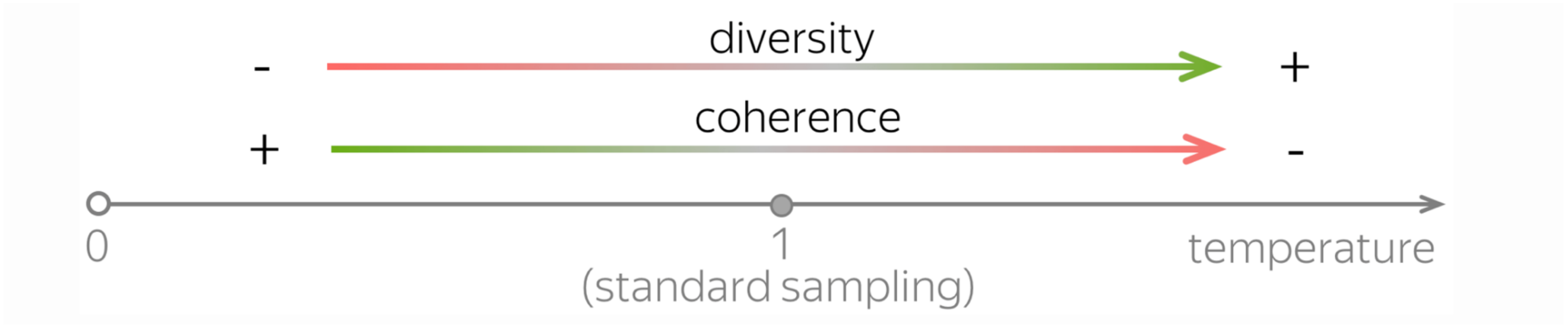
The most probably choice does not change, but with low temperatures, all the probabilities become further away from each other

The samples will become more similar



Temperature: **0.30**

Trade-off: coherence vs diversity



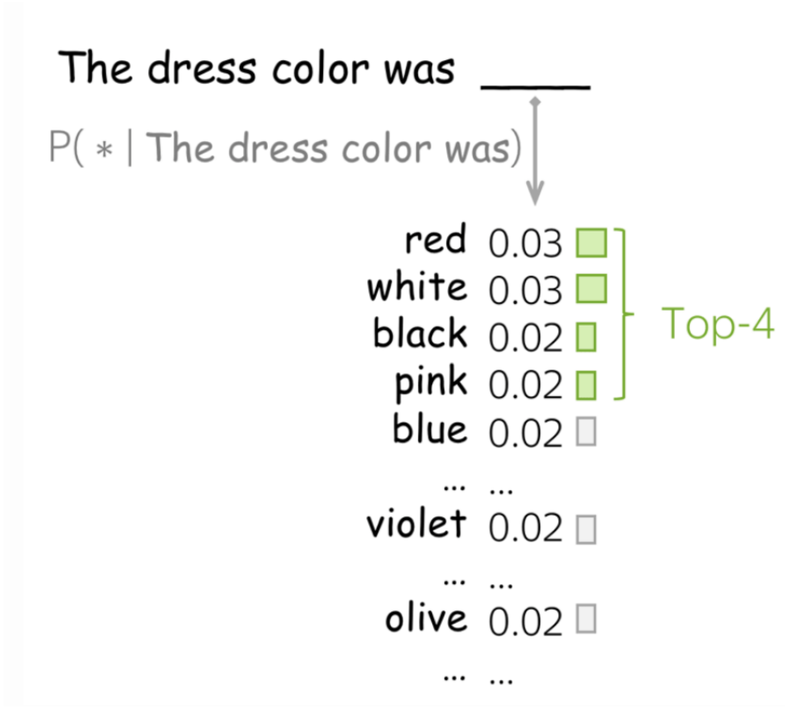
The choice of temperature parameters is dependent on your goal / situation

Varying temperature is tricky: if the temperature is too low, then almost all tokens receive very low probability; if the temperature is too high, plenty of tokens (not very good) will receive high probability.

Top-k sampling

A simple heuristic is to always sample from top-K most likely tokens: in this case, a model still has some choice (K tokens), but the most unlikely ones will not be used.

Formally, you take top K predicted tokens, *truncate* the distribution assigning 0 probability to the rest, and *renormalize* among the top K



Example samples (k = 10)

```
the guest reviews are submitted by our customers after
their stay at the hotel . _eos_
```

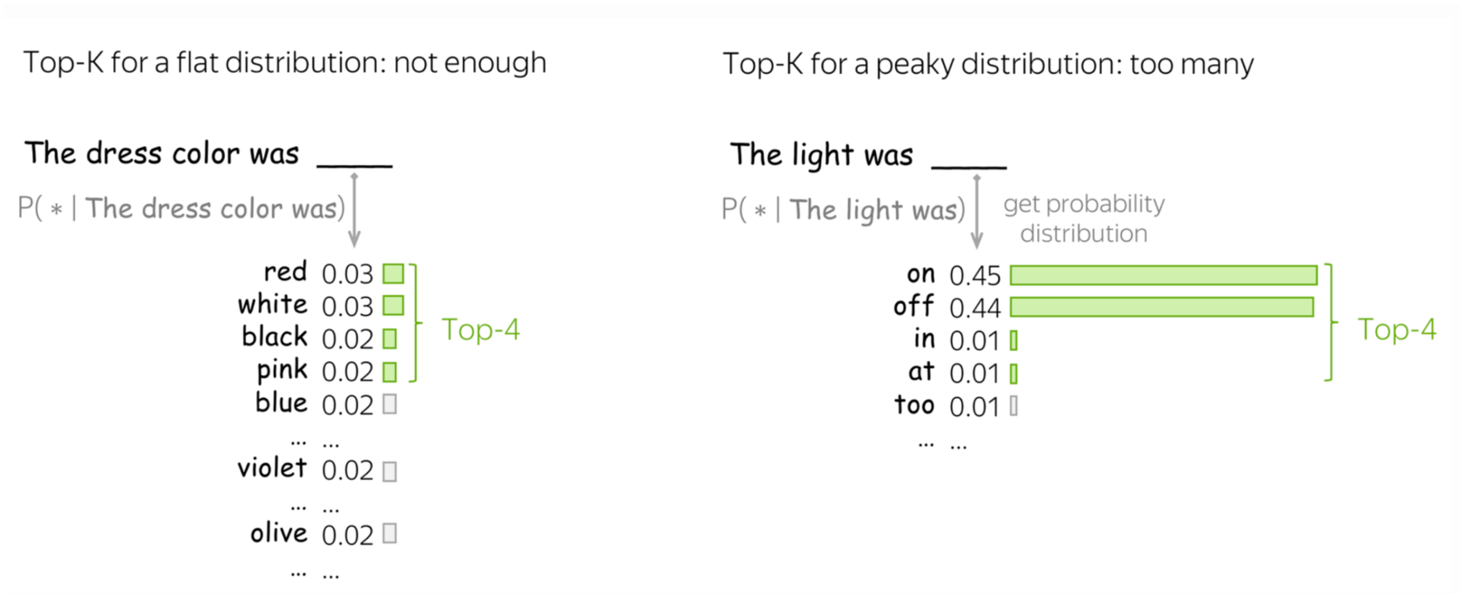
```
if a new government will have to pay for more or more
than 10 days , in the case of the company or to be the
right to cancel your account . _eos_
```

```
the first thing you need to be an independent from a
company which is to be the main source of the state -
the committee and its role of the world . _eos_
```

Issues with having a fixed k

But how do you choose an appropriate k?

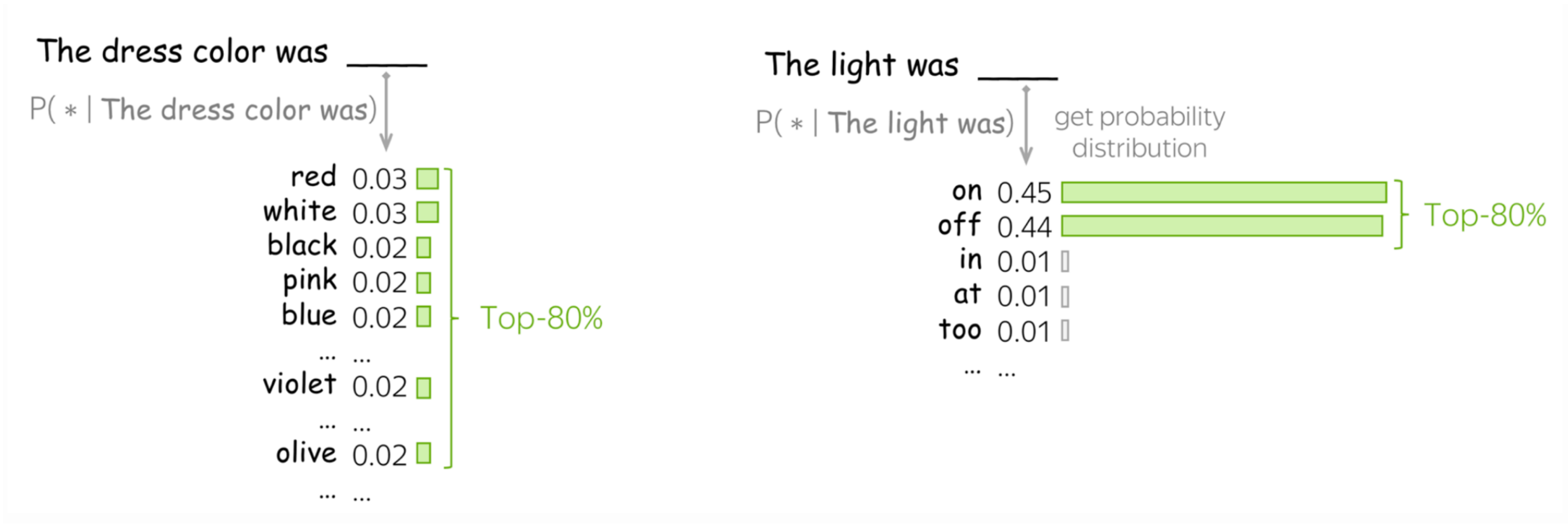
The same k can be too low when the distribution is too flat (high entropy) or too high if the distribution is too peaky (low entropy)



Nucleus sampling (top-p) – adaptive k

A more reasonable strategy is to consider not top-K most probable tokens, but **top-p%** of the probability mass *(nucleus sampling)*

Algorithmically it is similar to top-k (i.e. truncate and renormalize), but now you truncate to top choices which together constitute at least p%.



Nucleus sampling (top-p) – adaptive k

This makes the generation of 'surprising' tokens possible (unlike top-k for a reasonable k)

```
the third party runs when the us federal reserve the
house of 300 pieces of raw materials in the game , by
accident - and never - ending such clashes . _eos_
```

```
you ' re on - day to use a symbol of the « mystery »
of ukrainian chamber choir . _eos_
```

The distribution *profiles* also starts to resemble those for human-written texts

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

<u>Machine Translation</u>

model parameters

$$y' = \underset{y}{\arg\max}\, p(y|x,\theta)$$

Questions we need to answer

- modeling

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

- learning

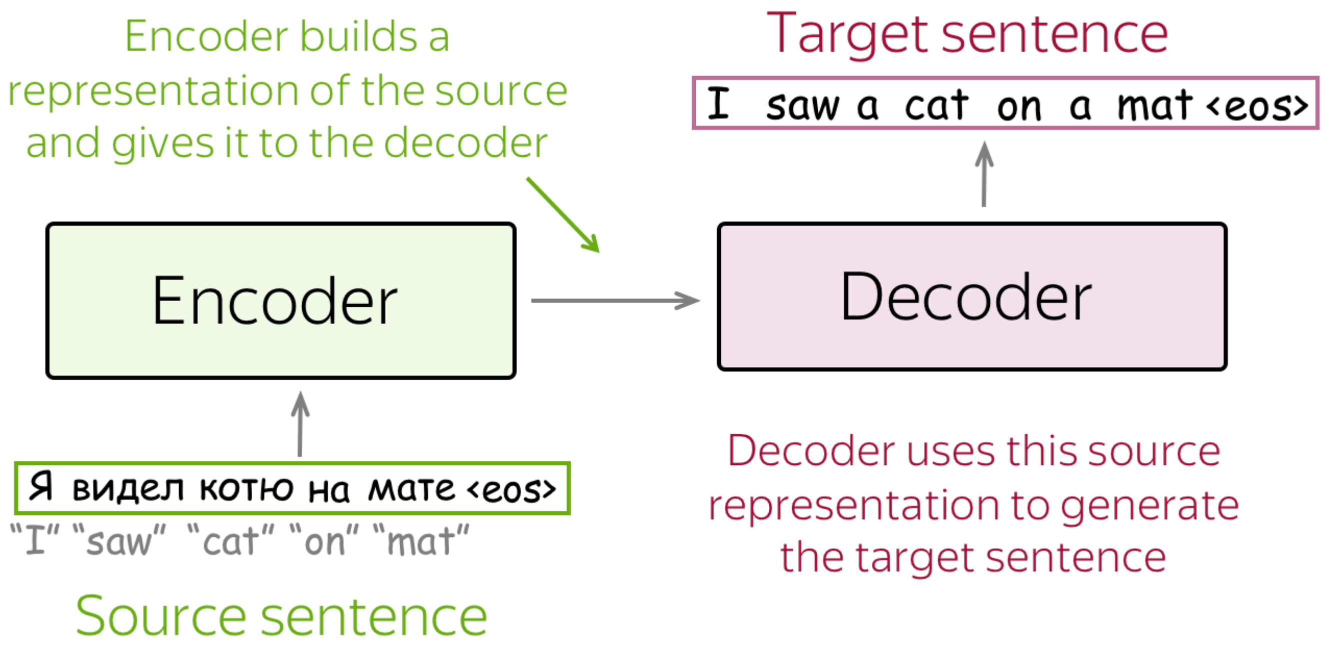
  How to find θ?

- 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.


Encoder builds a representation of the source and gives it to the decoder
Target sentence
I saw a cat on a mat <eos>
Encoder
Decoder
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
Source sentence
Decoder uses this source representation to generate the target sentence

Language modeling perspective

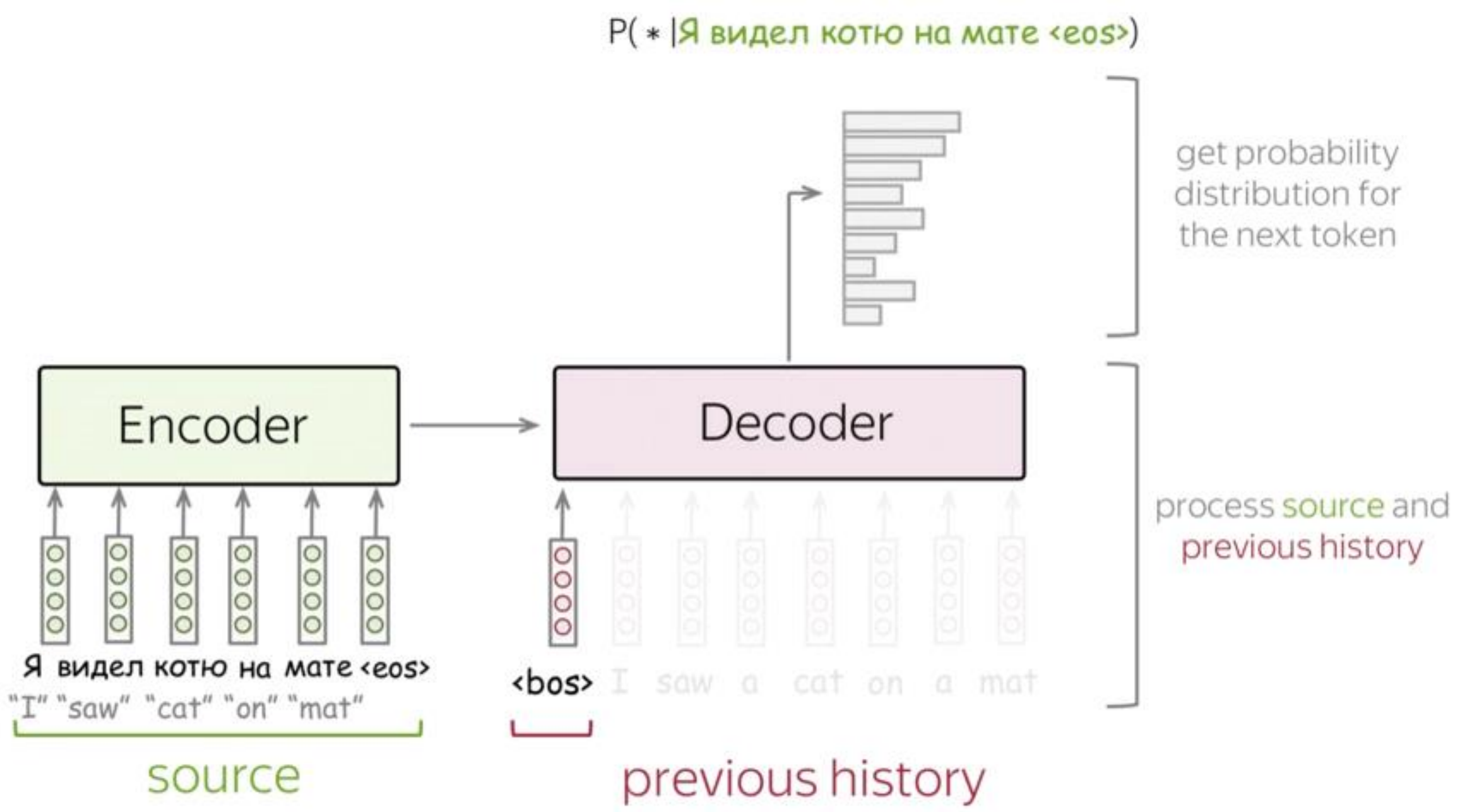
Language Models: $P\left(y_{1,}y_2, \ldots, y_n\right) = \prod_{t=1}^{n} p(y_t | y_{<t})$

Conditional

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

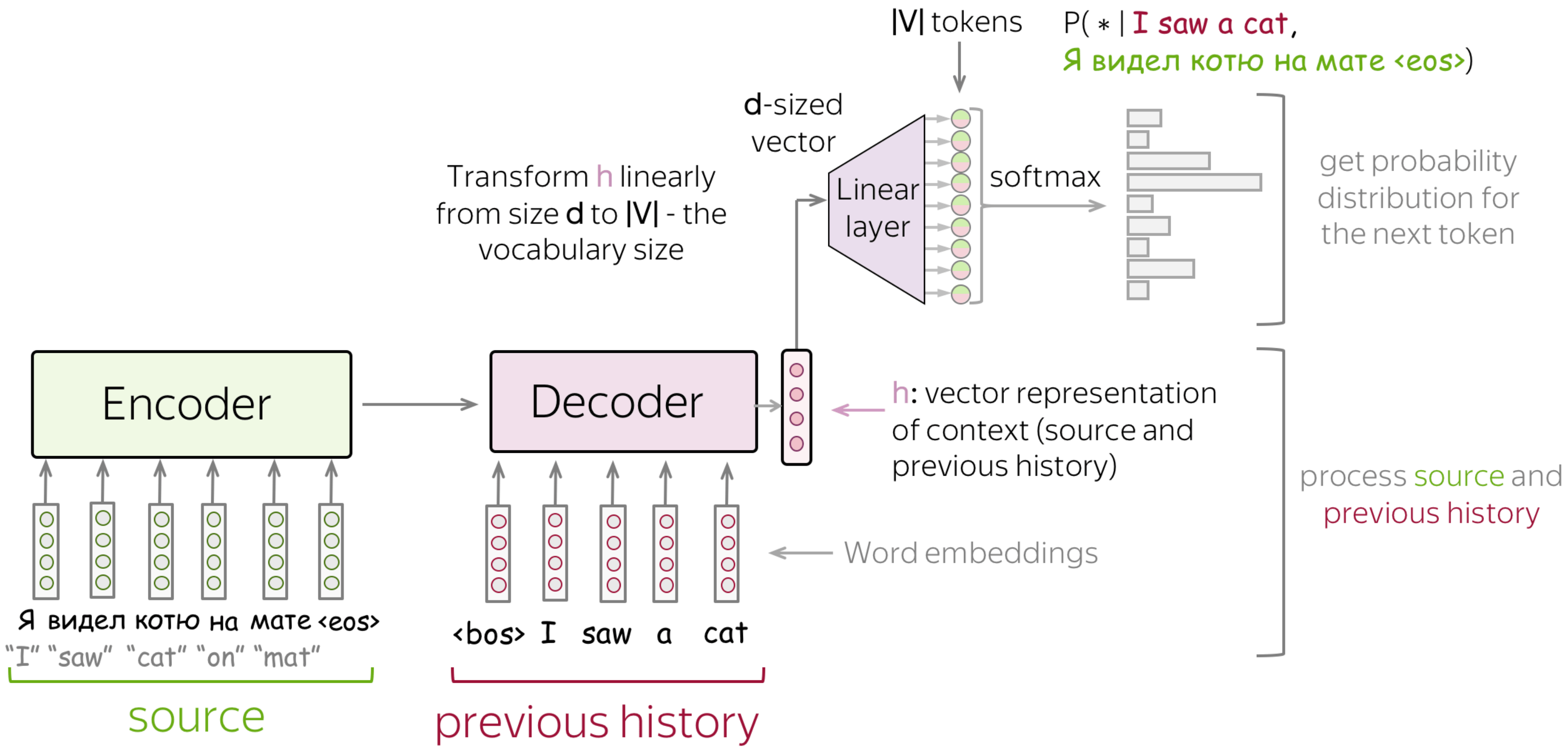
condition on source x

Encoder-decoder in action


P(* |Я видел котю на мате <eos>)
get probability distribution for the next token
Encoder
Decoder
process source and previous history
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
source
<bos> I saw a cat on a mat
previous history

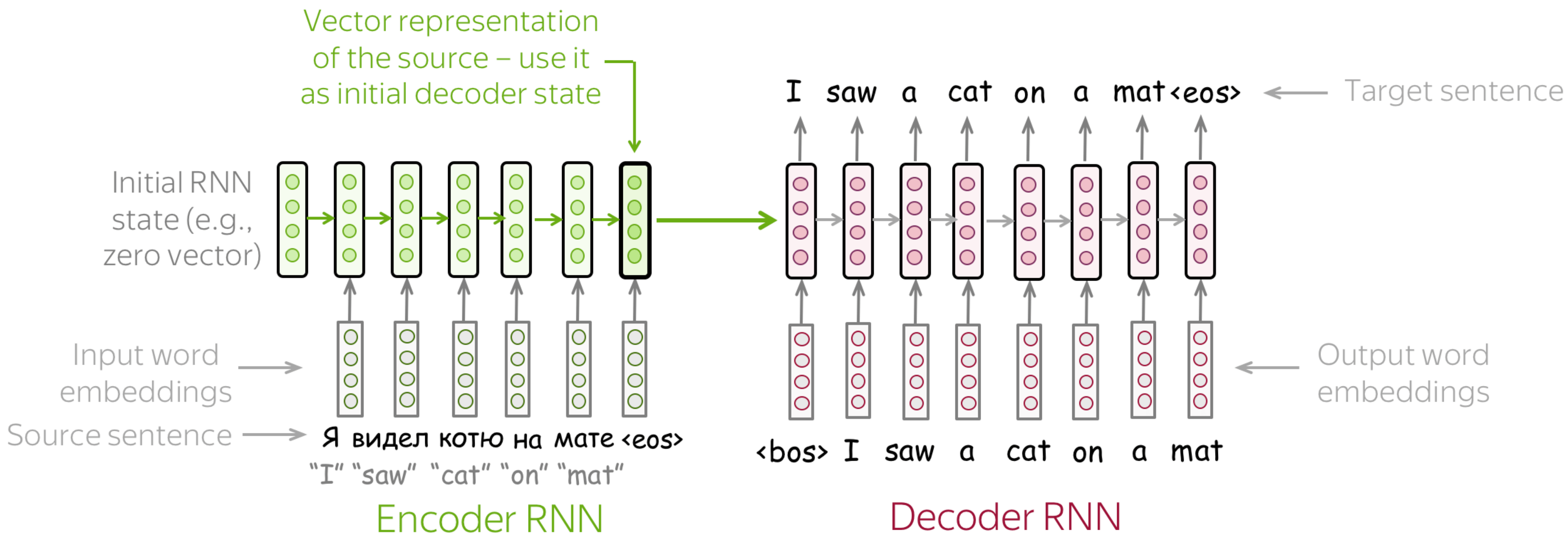
(video, not visible in pdf)

Encoder-decoder: under the hood



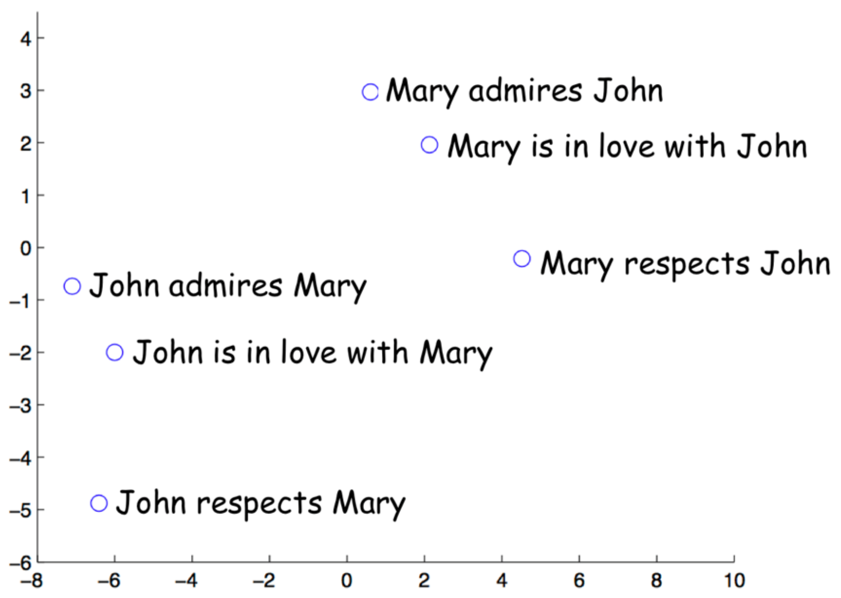
A lot like in language modeling, which was a lot like in text classification!

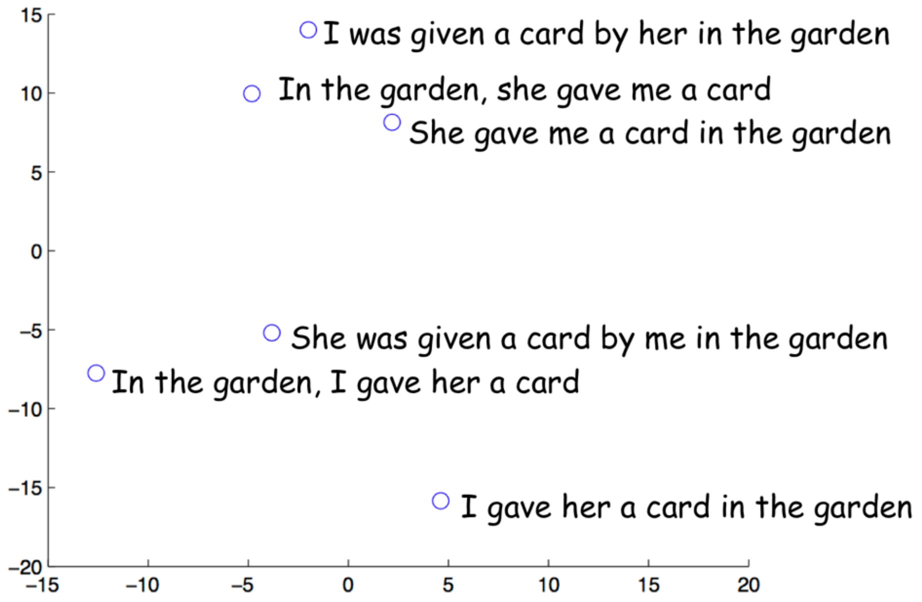
Simplest RNN-based Model:


Vector representation of the source – use it as initial decoder state
I saw a cat on a mat <eos>
Target sentence
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 RNN
Decoder RNN

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:

I saw a cat on a mat <eos>

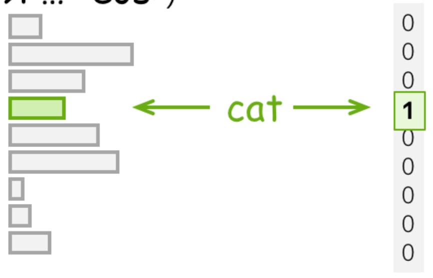
previous tokens

we want the model to predict this

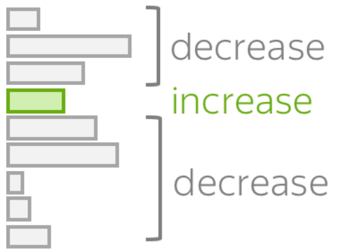
← one training example

← one step for this example

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

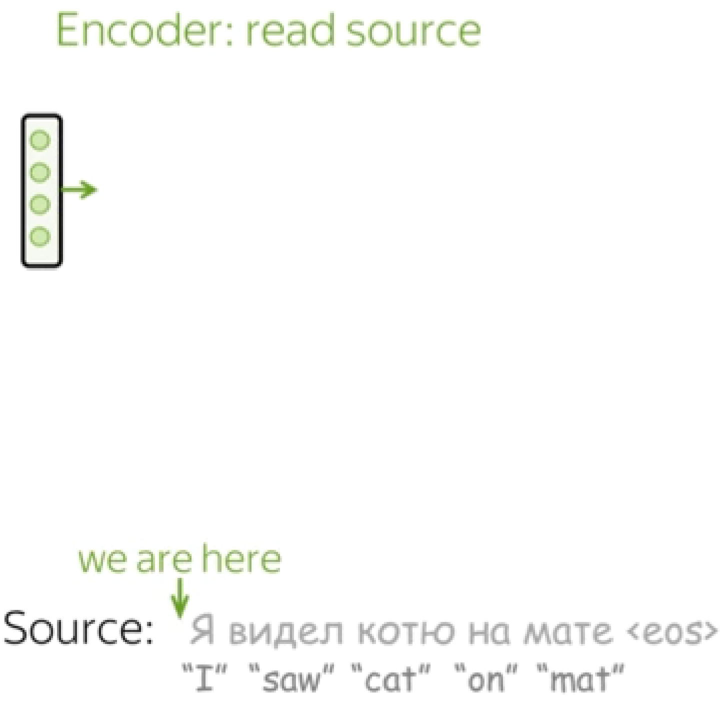


Loss = -log (p(cat)) → min



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

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:

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

We can also do sampling, but do we necessarily want surprising tokens in machine translation?

But what if we use a sequence-to-sequence model to generate a summary? To generate a report given a table? Generate a book given a plot?

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:

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

We can also do sampling, but do we necessarily want surprising tokens in machine translation?

Summary

- Encoder-Decoder architecture for Seq2Seq
- Inference algorithms (greedy, sampling, temperature,…)
    - We will add *beam search* next time as it is a natural choice for inference in machine translation