Language Modeling

Lena Voita

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

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

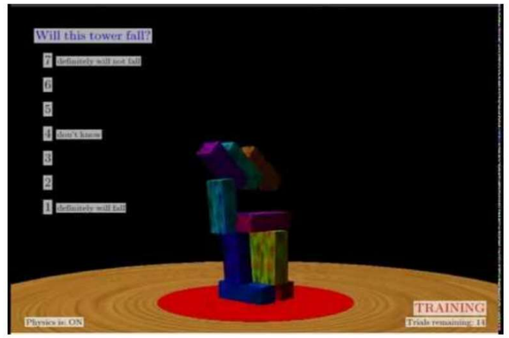
Models of a Train

- have some properties of trains (look like ones)
- can behave similarly to trains

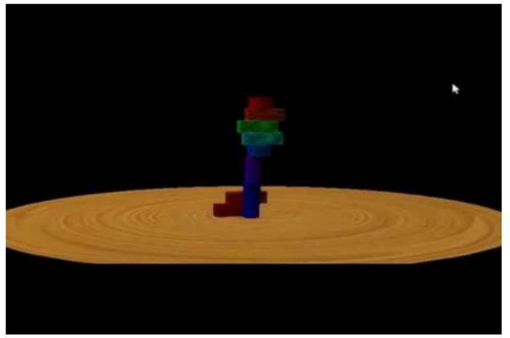
Models of a Train

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

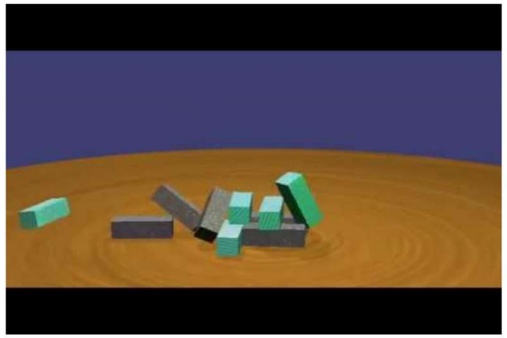
Models of a Physical World



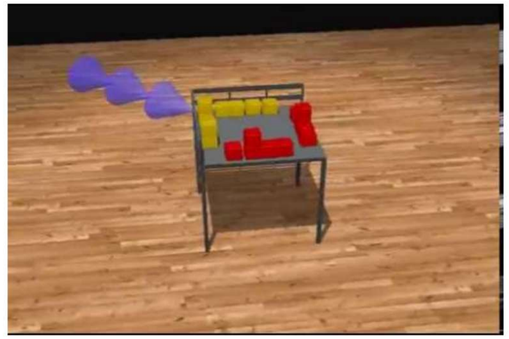
Will it fall?



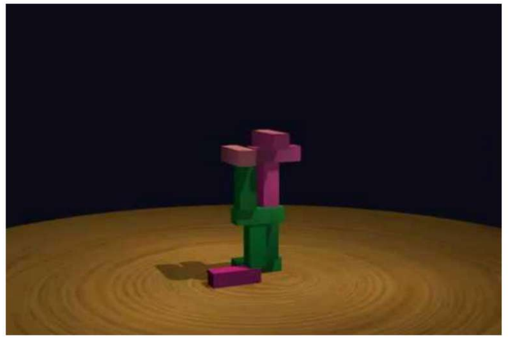
In which direction?



Different masses



Complex scenes



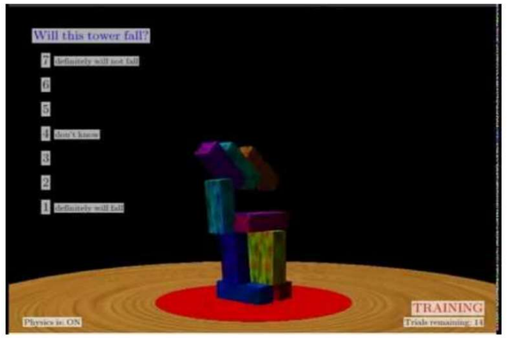
Infer the mass

Predict fluids

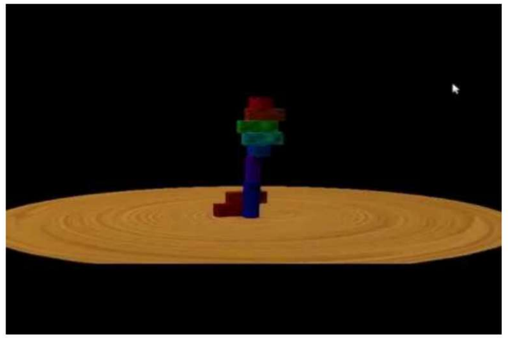
The picture is from the presentation by Peter Battaglia: http://phys.csail.mit.edu/talks/battaglia.pdf

Models of a Physical World

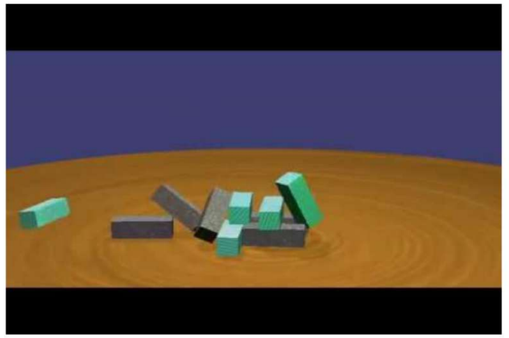
- understand which events are in better agreement with the world, which are more likely
- can predict what happens given some “context”



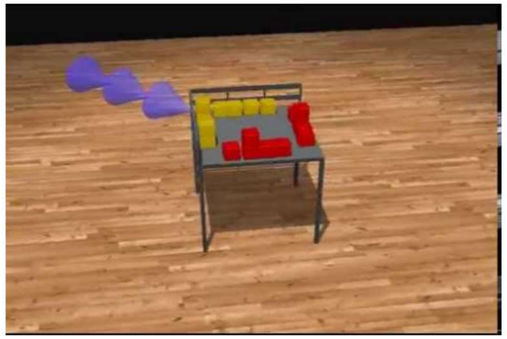
Will it fall?



In which direction?



Different masses



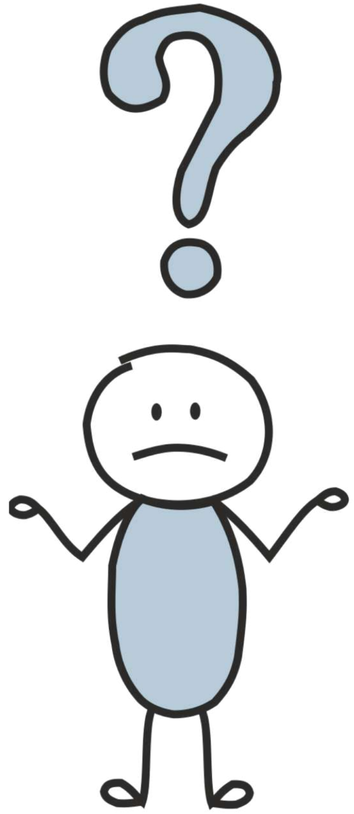
Complex scenes

Infer the mass

Predict fluids

The picture is from the presentation by Peter Battaglia: http://phys.csail.mit.edu/talks/battaglia.pdf

Model of a Language?..



Model of a Language?..

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.

How can this be useful?

We deal with Language Models every day!

Web search engine / ...

How can this be useful?

We deal with Language Models every day!

Translation service / mail agent / ...

How can this be useful?

We deal with Language Models every day!

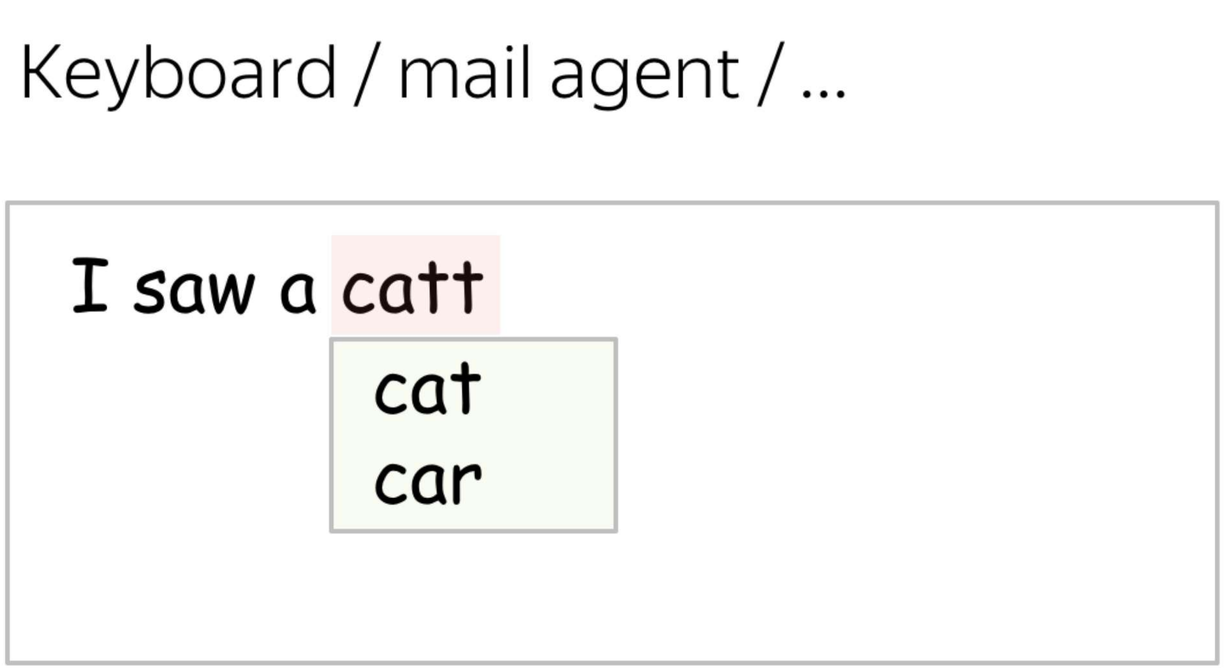
Translation service / mail agent / ...

I saw a catt

Probably you meant I saw a cat

How can this be useful?

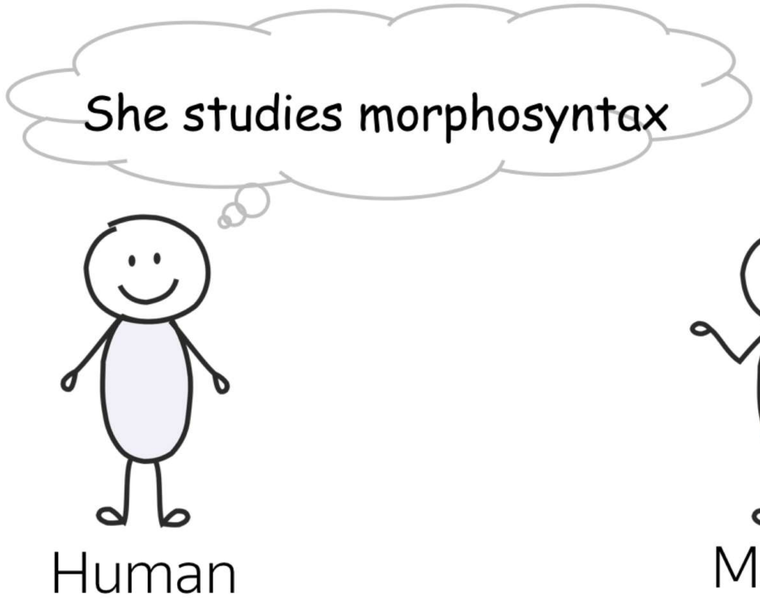
We deal with Language Models every day!



What is easy for humans can be very hard for machines

Similarly sounding options

She studies morphosyntax
She studies more faux syntax
She studies morph or syntax
....



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

What is going to happen:

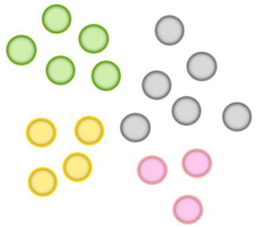
- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

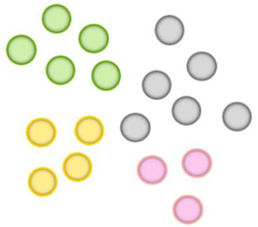
How likely is a sentence to appear in a language?

What is the probability
to pick a green ball?



How likely is a sentence to appear in a language?

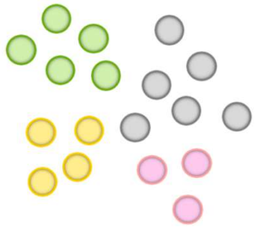
What is the probability to pick a green ball?



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

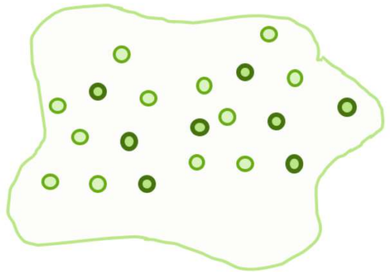
How likely is a sentence to appear in a language?

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

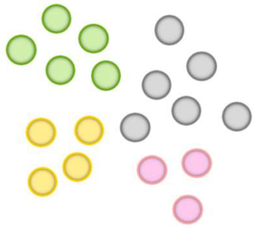
P(**the mut is tinming the tebn**)=$\frac{0}{|\text{corpus}|} = 0$

P(**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

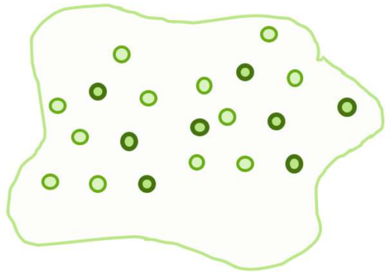
How likely is a sentence to appear in a language?

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?



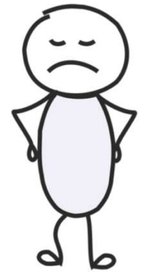
Text corpus

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

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

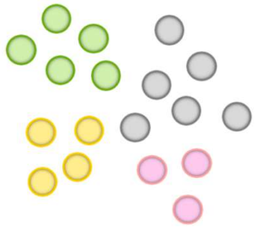
With this approach, sentences that never occurred in the corpus will receive zero probability

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



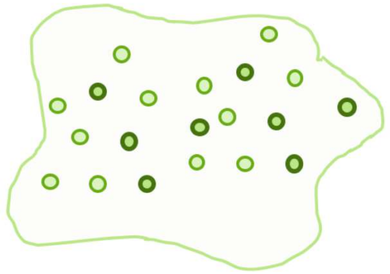
How likely is a sentence to appear in a language?

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?



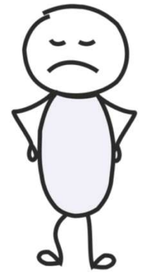
Text corpus

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

P(**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!

Sentence Probability: Decompose into Smaller Parts

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:

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

We got: Left-to-Right Language Modeling Framework

- This is the standard framework
- Models differ in a way they evaluate probabilities
- Later in the course, we'll see other language models (e.g., masked language models)

We got: Left-to-Right Language Modeling Framework

- This is the standard framework
- Models differ in a way they evaluate probabilities
- Later in the course, we'll see other language models (e.g., masked language models)

→ Need to define:

- how to compute $P(y_t|y_1, y_2, \dots, y_{t-1})$

→ N-gram models

→ Neural models

Generate a Text Using a Language Model

I ____

Alternatively, you can use **greedy** decoding: pick the token with the highest probability.
We'll see some examples a bit later.

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

Recap: What We Need

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

We need to: define how to compute the conditional probabilities $P(y_t|y_1, \ldots, y_{t-1})$.

Recap: What We Need

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

> We need to: define how to compute the conditional probabilities $P(y_t|y_1, \dots, y_{t-1})$.

This is a classical approach, so the solution is clear:

> How: estimate based on global statistics from a text corpora, i.e., **count**.

I.e., we do it **almost** as we counted green balls in the basket.

Recap: What We Need

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

> We need to: define how to compute the conditional probabilities $P(y_t|y_1, \dots, y_{t-1})$.

This is a classical approach, so the solution is clear:

> How: estimate based on global statistics from a text corpora, i.e., **count**.

I.e., we do it **almost** as we counted green balls in the basket.

This "almost" contains the key components of N-gram models: Markov property and smoothings

How to evaluate conditional probabilities?

A straightforward way: $P(y_t \mid y_1, y_2, \ldots, y_{t-1}) = \frac{N(y_1, y_2, \ldots, y_t)}{N(y_1, y_2, \ldots, y_{t-1})}$,

where $N(y_1, y_2, \ldots, y_t)$ is the number of times we met $(y_1, y_2, \ldots, y_t)$ in the training corpus.

How to evaluate conditional probabilities?

A straightforward way: $$P(y_t \mid y_1, y_2, \ldots, y_{t-1}) = \frac{N(y_1, y_2, \ldots, y_t)}{N(y_1, y_2, \ldots, y_{t-1})},$$

where $N(y_1, y_2, \ldots, y_t)$ is the number of times we met $(y_1, y_2, \ldots, y_t)$ in the training corpus.

These are often long sequences, and many of the counts will be zero!

Markov Assumption

> The probability of a word only depends on a **fixed** number of previous words.

For an n-gram model,

$$P(y_t \mid \underbrace{y_1, y_2, \cdots, y_{t-1}}) = P(y_t \mid \underbrace{y_{t-n+1}, \cdots, y_{t-1}})$$

all previous tokens

n-1 previous token

Markov Assumption

> The probability of a word only depends on a **fixed** number of previous words.

For an n-gram model,

$$P(y_t \mid \underbrace{y_1, y_2, \dots, y_{t-1}}) = P(y_t \mid \underbrace{y_{t-n+1}, \dots, y_{t-1}})$$

all previous tokens — n-1 previous token

For example,

- n=3 (trigram model): $P(y_t \mid y_1, y_2, \dots, y_{t-1}) = P(y_t \mid y_{t-2}, y_{t-1})$
- n=2 (bigram model): $P(y_t \mid y_1, y_2, \dots, y_{t-1}) = P(y_t \mid y_{t-1})$
- n=1 (unigram model): $P(y_t \mid y_1, y_2, \dots, y_{t-1}) = P(y_t)$

Markov Assumption: Example

P(I saw a cat on a mat) =?

Before

P(I saw a cat on a mat) =

P(I)

- P(saw | I)
- P(a | I saw)
- P(cat | I saw a)
- P(on | I saw a cat)
- P(a | I saw a cat on)
- P(mat | I saw a cat on a)

Markov Assumption: Example

P(I saw a cat on a mat) =?

Before

P(I saw a cat on a mat) =

P(I)
· P(saw | I)
· P(a | I saw)
· P(cat | I saw a)
· P(on | I saw a cat)
· P(a | I saw a cat on)
· P(mat | I saw a cat on a)

After (3-gram)

P(I saw a cat on a mat) =

P(I) → P(I)
· P(saw | I) → · P(saw | I)
· P(a | I saw) → · P(a | I saw)
· P(cat | ~~I~~ saw a) → · P(cat | saw a)
· P(on | ~~I saw~~ a cat) → · P(on | a cat)
· P(a | ~~I saw a~~ cat on) → · P(a | cat on)
· P(mat | ~~I saw a cat~~ on a) → · P(mat | on a)

ignore use

Houston, we have a problem

$$P(\text{mat} \mid \text{I saw a cat on a}) = P(\text{mat} \mid \text{cat on a}) = \frac{N(\text{cat on a mat})}{N(\text{cat on a})}$$

$$P(\text{mat} \mid \text{cat on a}) = \frac{N(\text{cat on a mat})}{\boxed{N(\text{cat on a})}} = ?$$

zero

not good: can not compute the probability

$$P(\text{mat} \mid \text{cat on a}) = \frac{\boxed{N(\text{cat on a mat})}}{N(\text{cat on a})} = ?$$

zero

not good: zeros out probability of the sentence

Backoff (aka Stupid Backoff)

$$P(\text{mat} \mid \text{cat on a}) = \frac{N(\text{cat on a mat})}{N(\text{cat on a})} = ?$$

N(cat on a) → zero

not good: can not compute the probability

Use less context for contexts we don't know much about:

- if you can, use trigram,
- if not, bigram,
- if not, unigram

Backoff (aka Stupid Backoff)

$$P(\text{mat} \mid \text{cat on a}) = \frac{N(\text{cat on a mat})}{N(\text{cat on a})} = ?$$

$N(\text{cat on a})$: zero

not good: can not compute the probability

$N(\text{cat on a}) = 0 \longrightarrow$ try "on a"

$P(\text{mat} \mid \text{cat on a}) \approx P(\text{mat} \mid \text{on a})$

$N(\text{on a}) = 0 \longrightarrow$ try "a"

$P(\text{mat} \mid \text{on a}) \approx P(\text{mat} \mid \text{a})$

$N(\text{a}) = 0 \longrightarrow$ try unigram

$P(\text{mat} \mid \text{a}) \approx P(\text{mat})$

Use less context for contexts we don't know much about:

- if you can, use trigram,
- if not, bigram,
- if not, unigram

More Clever: Linear Interpolation

$$P(\text{mat} \mid \text{cat on a}) = \frac{N(\text{cat on a mat})}{N(\text{cat on a})} = ?$$

zero

not good: can not compute the probability

$$\hat{P}(\text{mat} \mid \text{cat on a}) \approx \lambda_3 P(\text{mat} \mid \text{cat on a}) + \lambda_2 P(\text{mat} \mid \text{on a}) + \lambda_1 P(\text{mat} \mid \text{a}) + \lambda_0 P(\text{mat})$$

$$\sum_i \lambda_i = 1$$

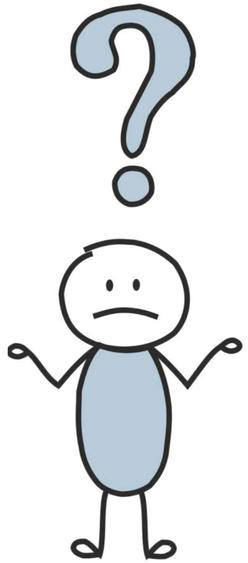
More Clever: Linear Interpolation

$$P(\text{mat} \mid \text{cat on a}) = \frac{N(\text{cat on a mat})}{N(\text{cat on a})} = ?$$

zero

not good: can not compute the probability

How to define good λ_i?



$$\hat{P}(\text{mat} \mid \text{cat on a}) \approx \lambda_3 P(\text{mat} \mid \text{cat on a}) + \lambda_2 P(\text{mat} \mid \text{on a}) + \lambda_1 P(\text{mat} \mid \text{a}) + \lambda_0 P(\text{mat})$$

$$\sum_i \lambda_i = 1$$

More Clever: Linear Interpolation

$$P(\text{mat} \mid \text{cat on a}) = \frac{N(\text{cat on a mat})}{N(\text{cat on a})} = ?$$

zero

not good: can not compute the probability

How to define good λ_i?

Cross-validation!

$$\widehat{P}(\text{mat} \mid \text{cat on a}) \approx \lambda_3 P(\text{mat} \mid \text{cat on a}) + \lambda_2 P(\text{mat} \mid \text{on a}) + \lambda_1 P(\text{mat} \mid \text{a}) + \lambda_0 P(\text{mat})$$

$$\sum_i \lambda_i = 1$$

Laplace (add-one) smoothing

$$P(\text{mat} \mid \text{cat on a}) = \frac{\overbrace{N(\text{cat on a mat})}^{\text{zero}}}{N(\text{cat on a})} = ?$$

not good: zeros out probability of the sentence

Pretend we saw each n-gram at least one time (or δ times):

$$\widehat{P}(\text{mat} \mid \text{cat on a}) = \frac{\delta + N(\text{cat on a mat})}{\delta \cdot |V| + N(\text{cat on a})}$$

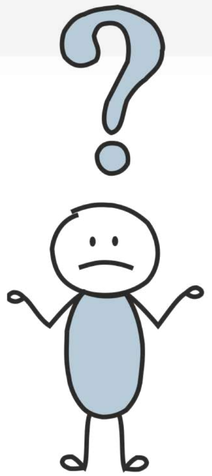
Examples of Generated Texts

```
when this option may be the worst day of amnesty
international delegations visited israel , and felt that
his sisters , that they are reserved for zyryanovsk
concentrating factory there is a member of the shire ,"
given as to damage the expansion of a meeting over a large
health maintenance organization , smoking ,
airconditioning , designated smoking area . _eos_
```

Examples of Generated Texts

```
so even when i talk a bit short , there was no easy thing
to do different buffer flushing strategies in the future ,
due to huge list of number - one just has started
production of frits in the process and has free wi - fi ”
operation .... _eos_
```

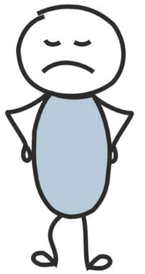
What is wrong with these texts? Why?



Examples of Generated Texts

```
so even when i talk a bit short , there was no easy thing
to do different buffer flushing strategies in the future ,
due to huge list of number - one just has started
production of frits in the process and has free wi - fi ”
operation .... _eos_
```

Texts are incoherent because
fixed contexts are bad!



What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
  - Idea and High-Level Pipeline
  - Training: Cross-Entropy
  - Models: Recurrent
  - Models: Convolutional →
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

Recap: What We Need

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

We need to: define how to compute the conditional probabilities $P(y_t|y_1, \ldots, y_{t-1})$.

Classical approaches:

How: estimate based on global statistics from a text corpora, i.e., **count**.

Recap: What We Need

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

> We need to: define how to compute the conditional probabilities $P(y_t|y_1, \ldots, y_{t-1})$.

In neural networks, we do as usually:

> How: Train a neural network to **predict them**.

General View

- process context – model-specific

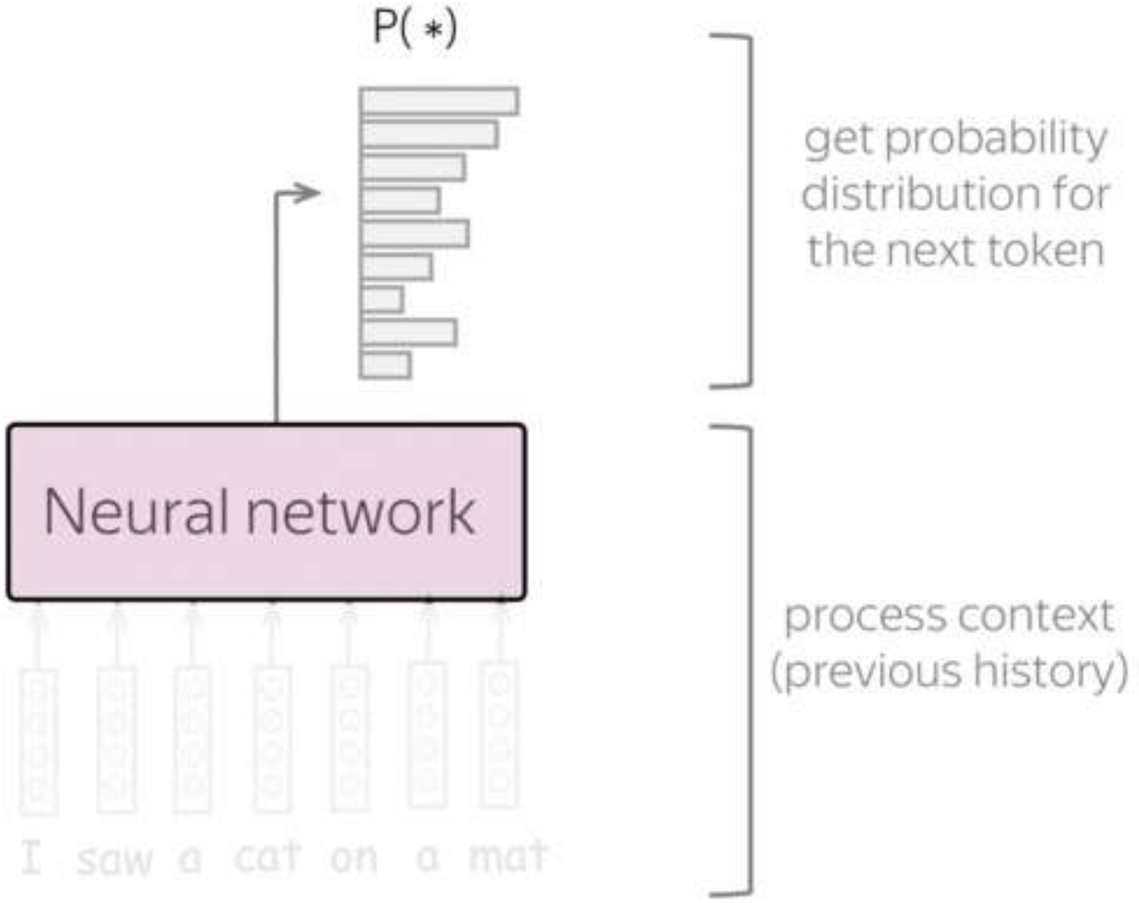
  Get vector representation of the previous context

- evaluate probabilities – model-agnostic

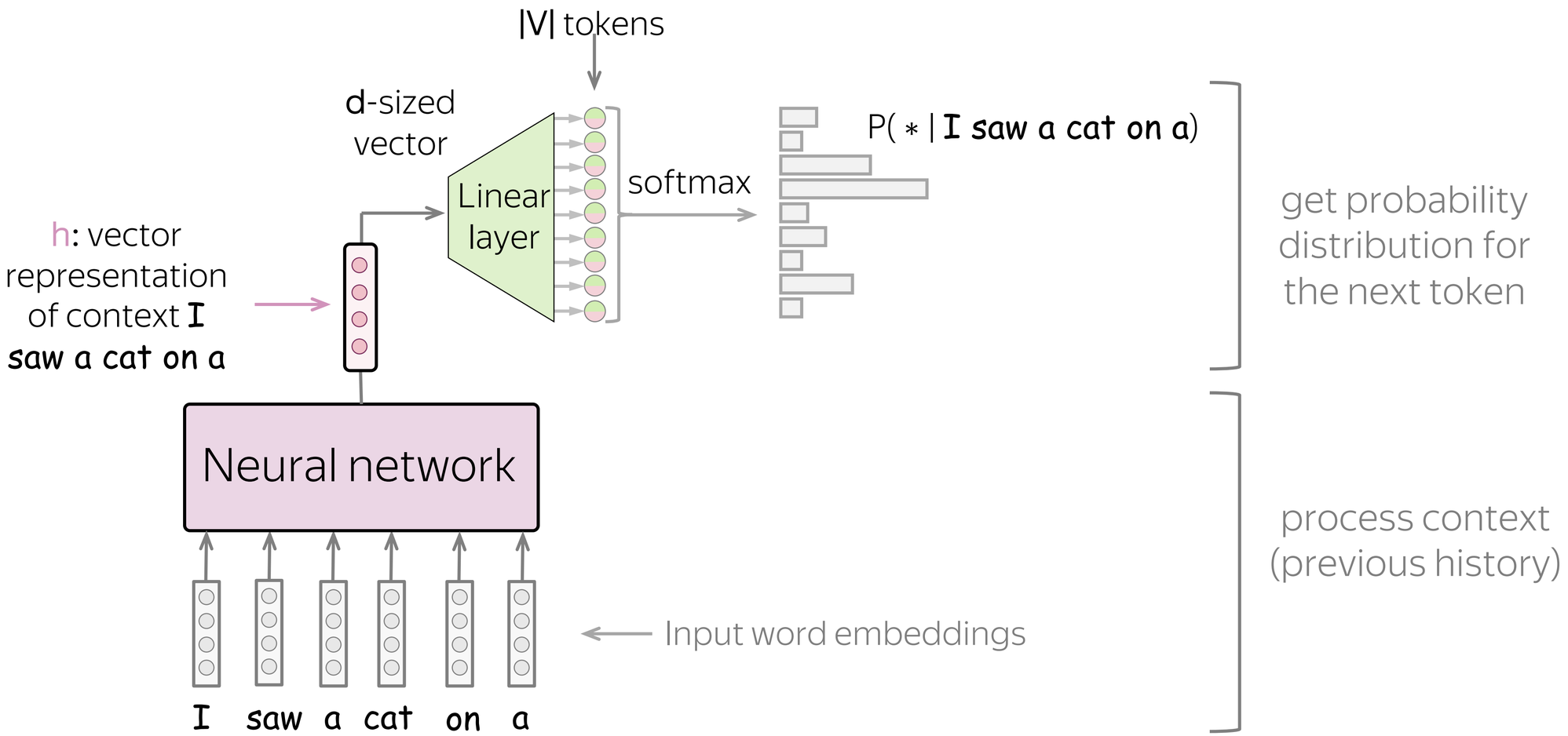
  Predict probability distribution for the next token

  ↕

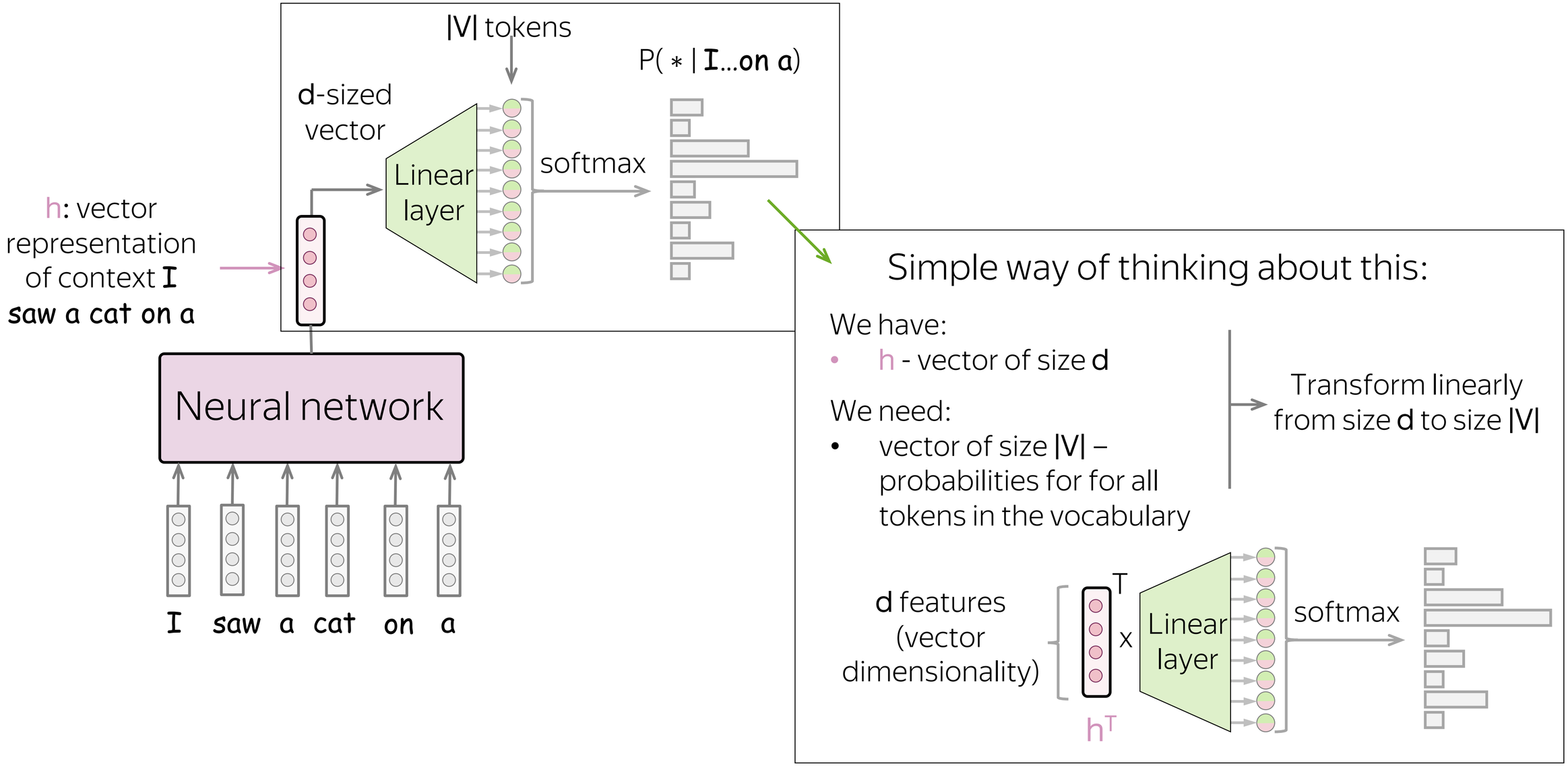
  Classify into |V| classes



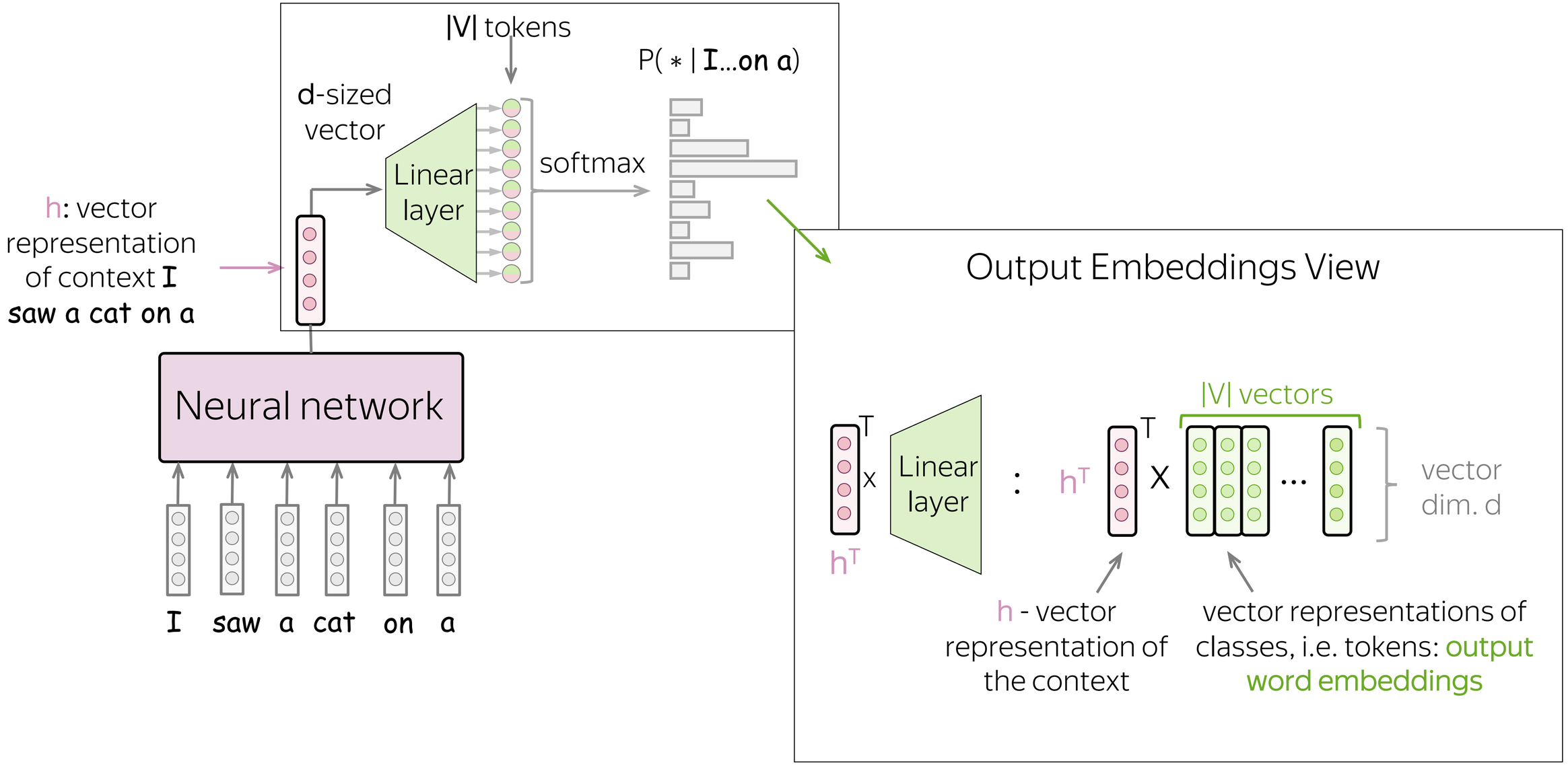
High-Level Pipeline



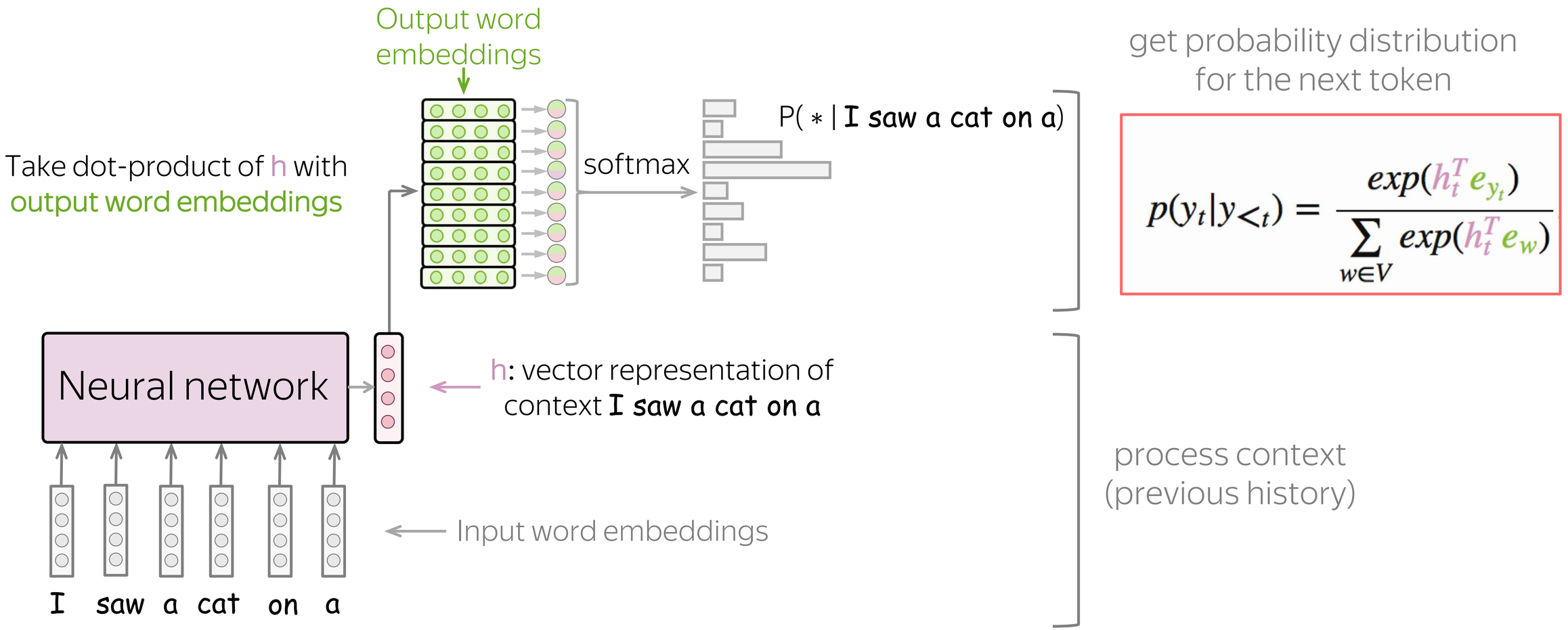
High-Level Pipeline



High-Level Pipeline



High-Level Pipeline: Output Embeddings View



What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
  - Idea and High-Level Pipeline
  - Training: Cross-Entropy
  - Models: Recurrent
  - Models: Convolutional →
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
  - Idea and High-Level Pipeline
  - Training: Cross-Entropy
  - Models: Recurrent
  - Models: Convolutional →
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

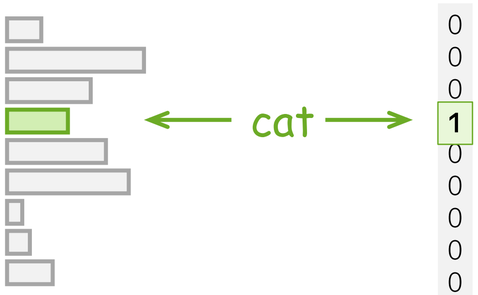
Training: Cross-Entropy

Target at this step

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

Model prediction: p(∗ | I saw a)

Target: p^*



Training: Cross-Entropy

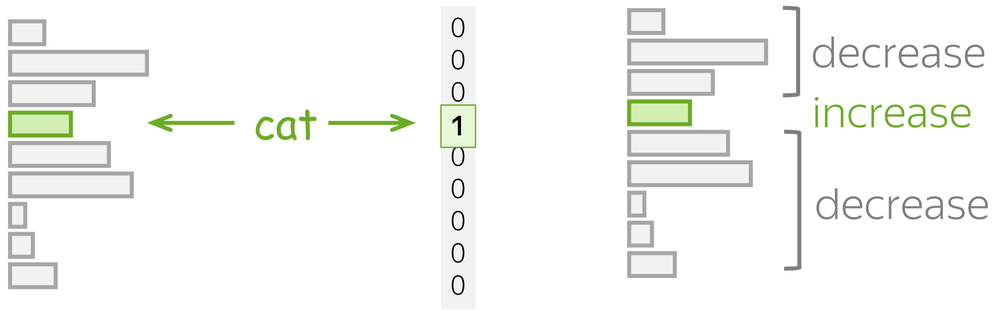
Target at this step

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

Model prediction: p(∗ | I saw a)

Target: p^*

Loss = -log (p(cat)) → min



Cross-entropy loss:

$$-\sum_{i=1}^{|V|} p_i^* \cdot \log P(y_t = i|x) \rightarrow min \ (p_k^* = 1, p_i^* = 0, i \neq k)$$

For one-hot targets, this is equivalent to

$$-\log P(y_t = cat|x) \rightarrow min$$

What is going to happen:

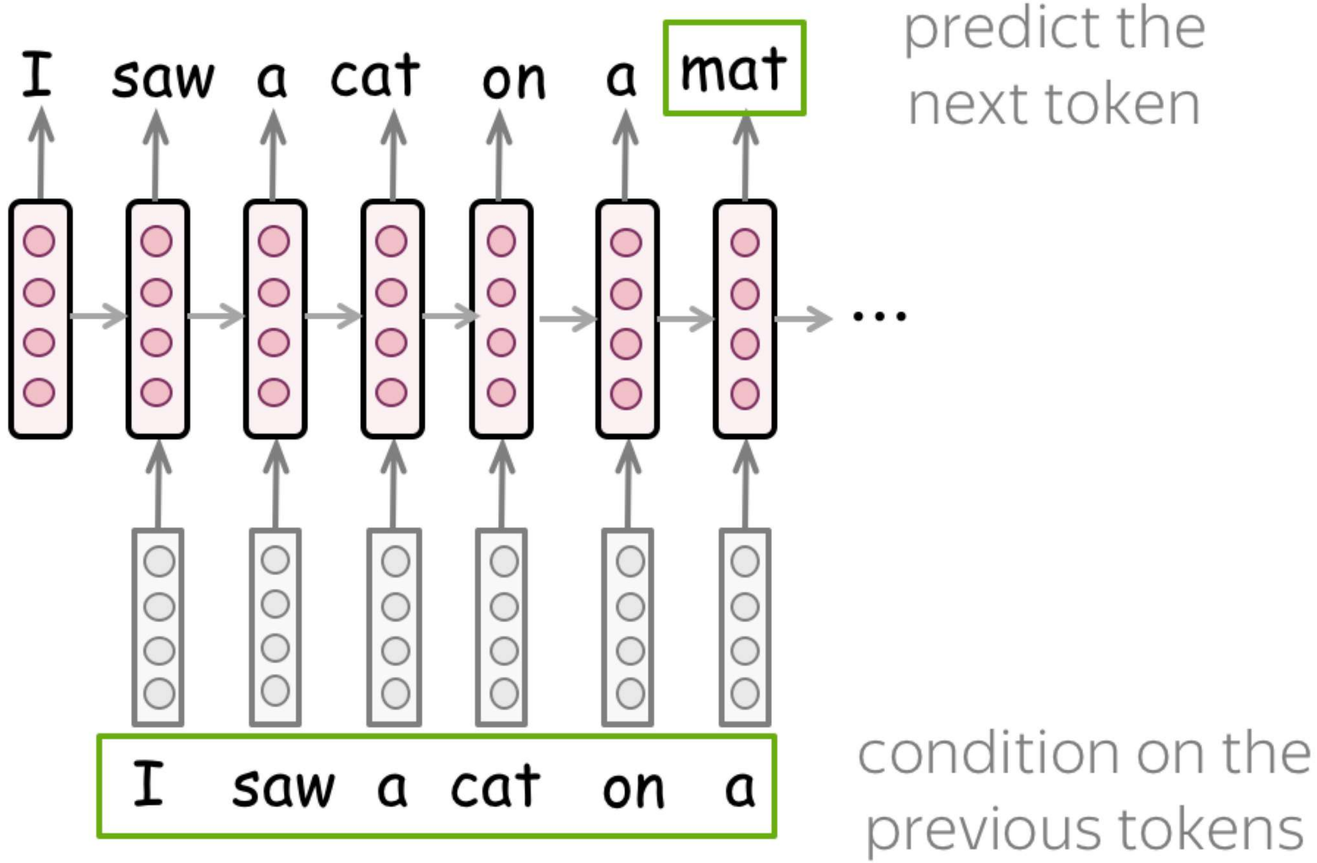
- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
  - Idea and High-Level Pipeline
  - Training: Cross-Entropy
  - Models: Recurrent
  - Models: Convolutional →
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
  - Idea and High-Level Pipeline
  - Training: Cross-Entropy
  - Models: Recurrent
  - Models: Convolutional →
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

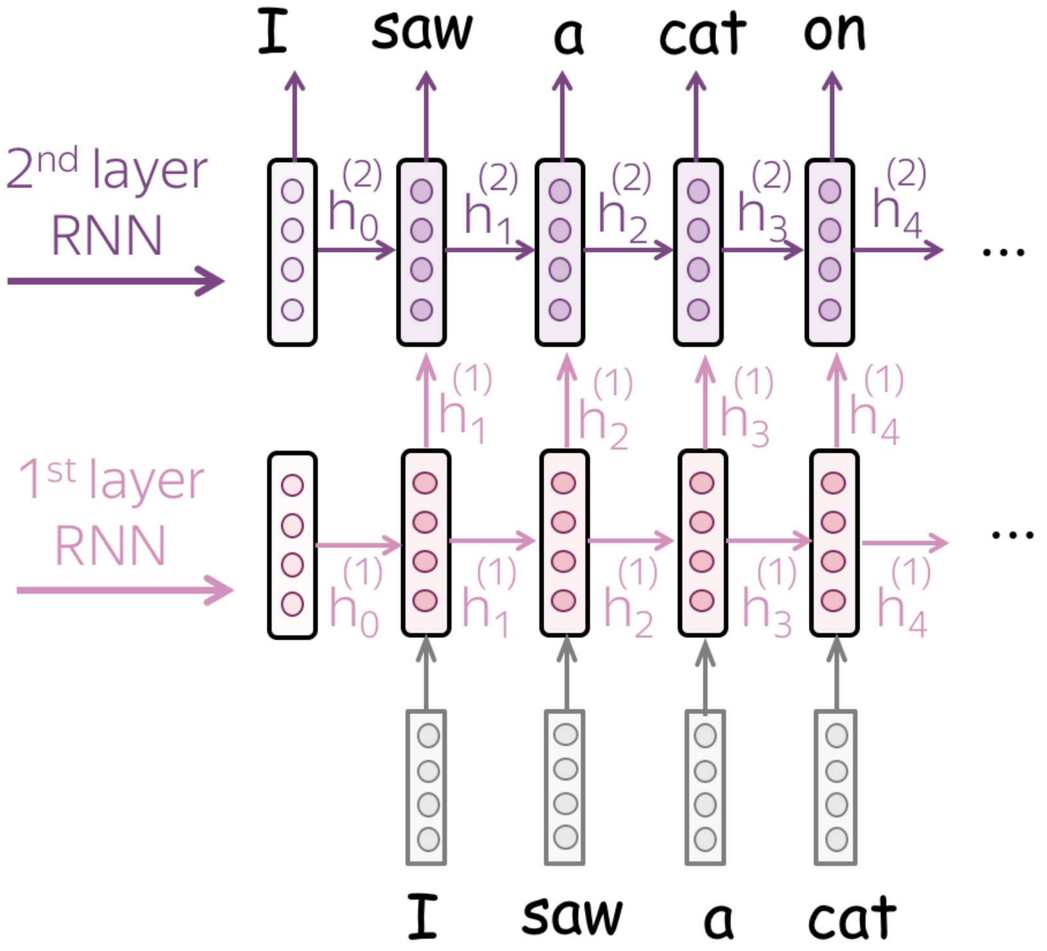
Recurrent Models for Language Modeling

- **simple**: read a text, predict next token at each step



Recurrent Models for Language Modeling

- **Multi-layer**: feed the states from one RNN to the next



What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
  - Idea and High-Level Pipeline
  - Training: Cross-Entropy
  - Models: Recurrent
  - Models: Convolutional →
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
  - Idea and High-Level Pipeline
  - Training: Cross-Entropy
  - Models: Recurrent
  - Models: Convolutional
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

NLP Course For You:
if you want, read **here**

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
  - Goal: Coherence and Diversity
  - Sampling
  - Sampling with Temperature
  - Top-K Sampling
  - Top-p (Nucleus) Sampling
- Evaluation
- Practical Tips
-  Analysis and Interpretability

We Need: Coherence and Diversity

Usually, we need generated texts to be:

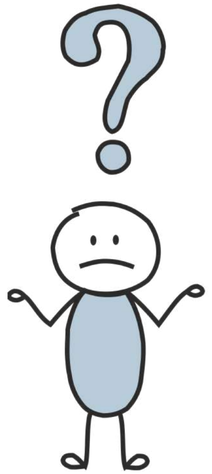
- coherent - the generated text has to make sense;
- diverse - the model has to be able to produce very different samples.

We Need: Coherence and Diversity

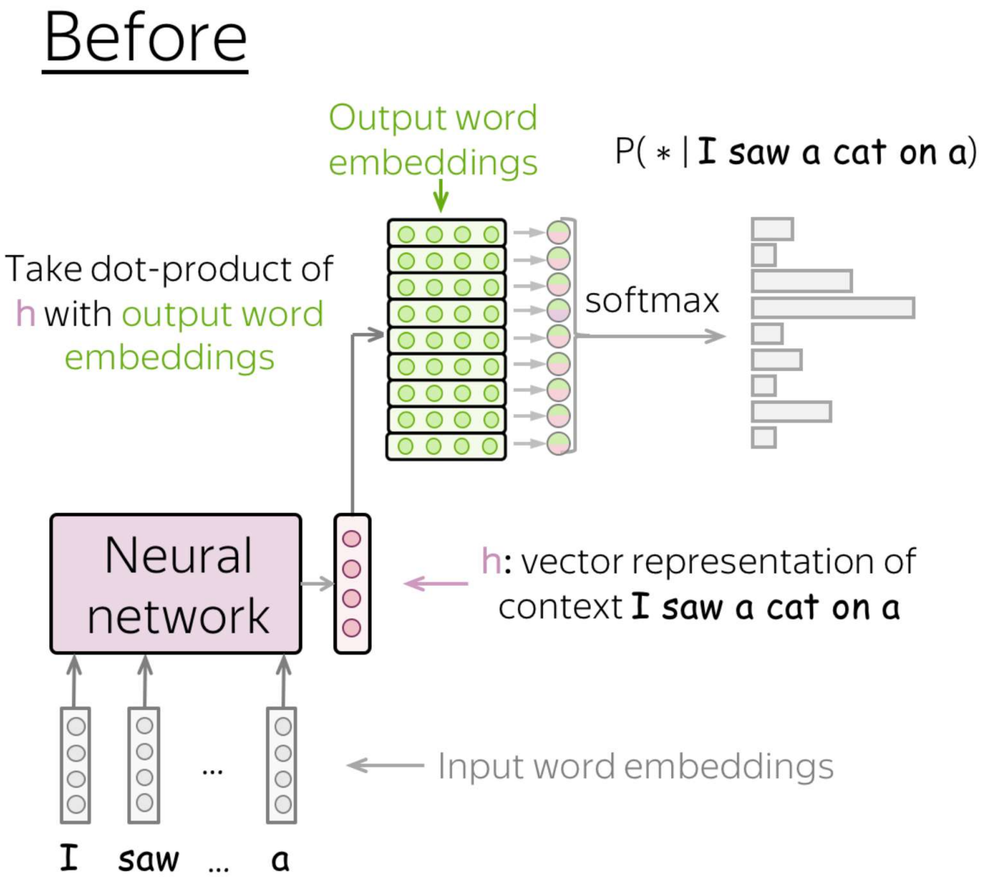
Usually, we need generated texts to be:

- **coherent** - the generated text has to make sense;
- **diverse** - the model has to be able to produce very different samples.

How can we control this?



Sampling with Temperature

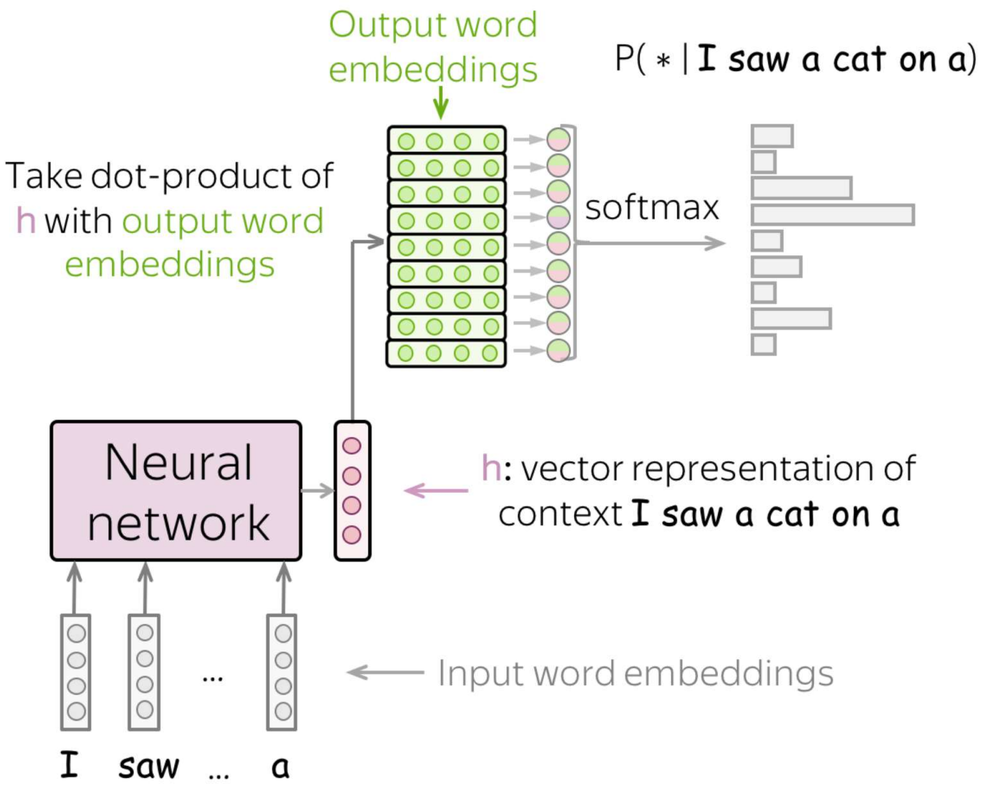


Sampling with Temperature

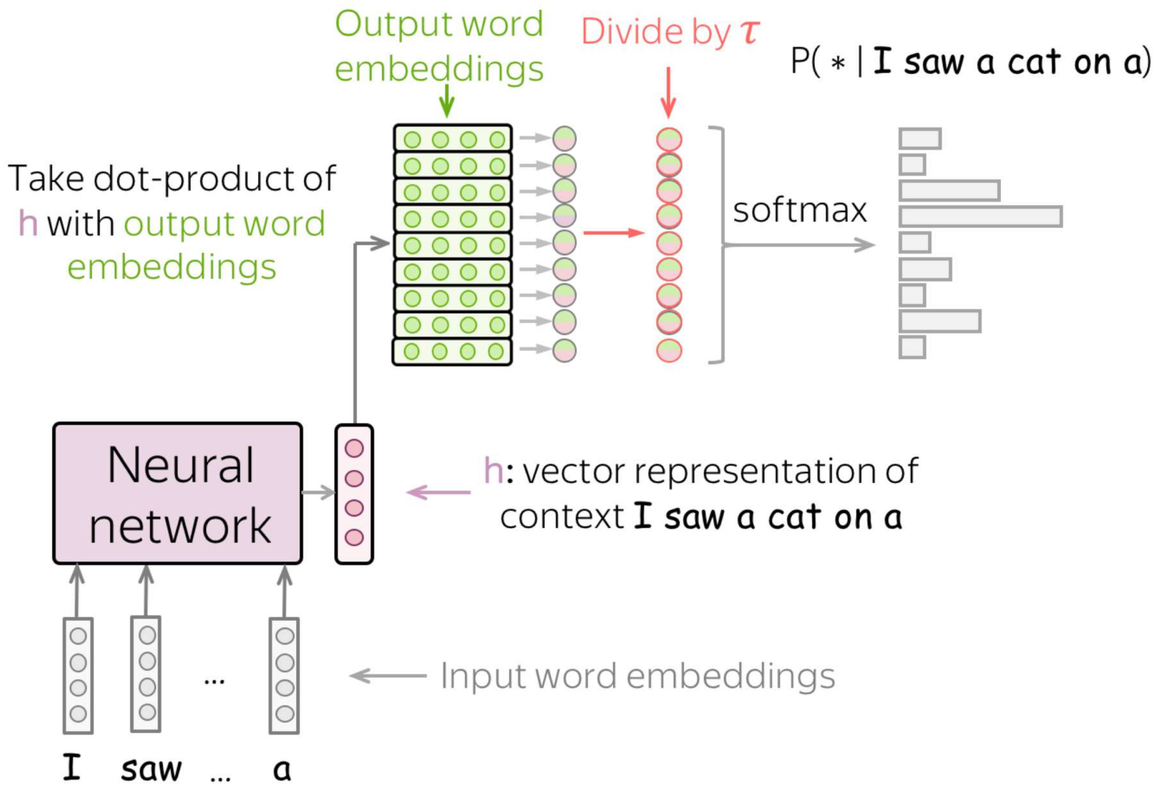
$$\frac{\exp(h^T w)}{\sum_{w_i \in V} \exp(h^T w_i)} \rightarrow \frac{\exp\left(\frac{h^T w}{\tau}\right)}{\sum_{w_i \in V} \exp\left(\frac{h^T w_i}{\tau}\right)}$$

τ - softmax temperature

Before



After

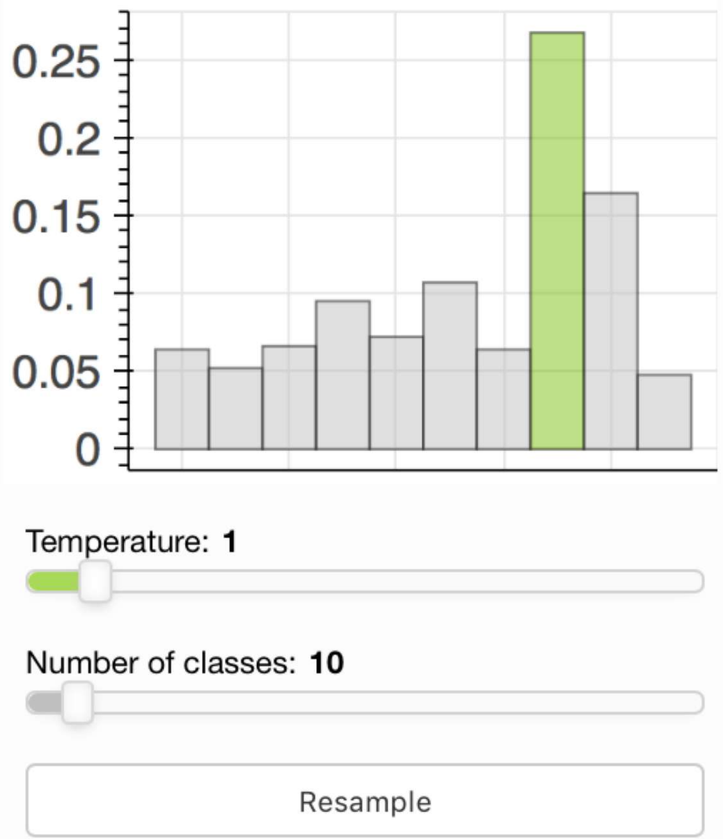


Sampling with Temperature

How to: Play with the temperature and see how the probability distribution changes. Note how changes the difference between the probability of the most likely class (green) and others.

- What happens when the temperature is close to zero?
- What happens when the temperature is high?
- Sampling with which temperature corresponds to greedy decoding?

Note that you can also change the number of classes and generate another probability distribution.



http://lena-voita.github.io/nlp_course/language_modeling.html#generation_strategies_temperature

Examples: Temperature=2

```
paradise sits farms started paint hollow almost
unprecedented decisions, care using withdrawal from
rebel cis ( , saying graphics mongolia official line,
greeted agenda victor is exploring anger :) draw
testify liberalization decay productive 2 went
exchanges of marketing drawing enabling challenging
systematic crisis influencing the executive arrangement
performs designs
```

http://lena-voita.github.io/nlp_course/language_modeling.html#generation_strategies_temperature

Examples: Temperature=2

```
believes transactions article remained considered
britain holding presidency which had fled profit like
first directly immediately authoritative scheme
bluetooth as mas series _eos_
```

http://lena-voita.github.io/nlp_course/language_modeling.html#generation_strategies_temperature

Examples: Temperature=2

```
on 25 allegations may vary utilizing sweet
organizations excluding commissions gas approaching
security metal — pro was growing for foreign primary
education on as kyrgyz manufacturers lining , sd or 100
from the tin _eos_
```

http://lena-voita.github.io/nlp_course/language_modeling.html#generation_strategies_temperature

Examples: Temperature=0.2

```
the first time the two - year - old - old girl with a
new version of the new version of the new version of
the new version of the new version of the new version
of the new version of the new version of the new
version of the
```

http://lena-voita.github.io/nlp_course/language_modeling.html#generation_strategies_temperature

Examples: Temperature=0.2

```
the first step is to be used in the first time . _eos_
```

(this sample appeared several times)

http://lena-voita.github.io/nlp_course/language_modeling.html#generation_strategies_temperature

Examples: Temperature=0.2

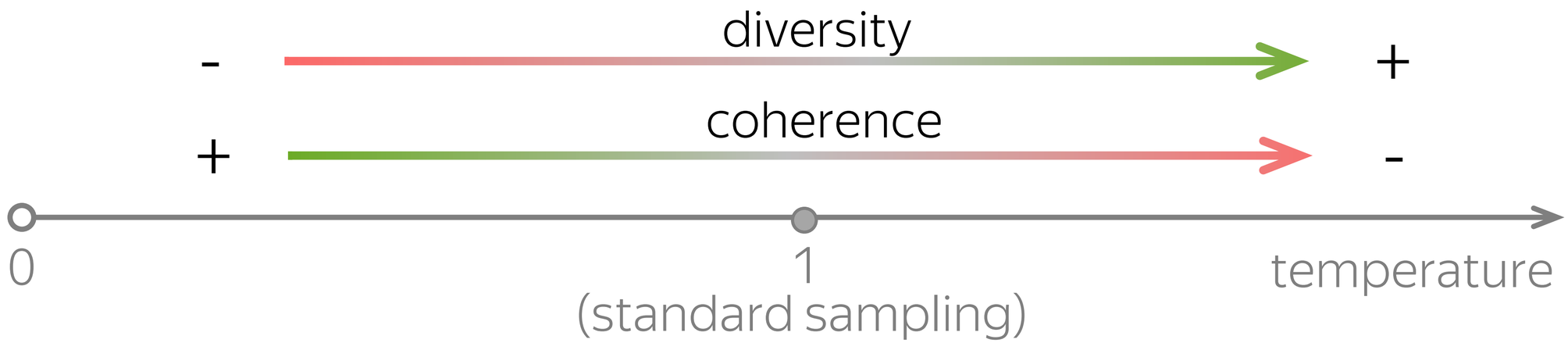
```
the hotel is located in the heart of the city . _eos_
```

(this sample appeared many times)

http://lena-voita.github.io/nlp_course/language_modeling.html#generation_strategies_temperature

Sampling with Temperature

- Temperature can help a bit
- It’s not ideal: both increasing and decreasing improve one of coherence and diversity, but hurt the other



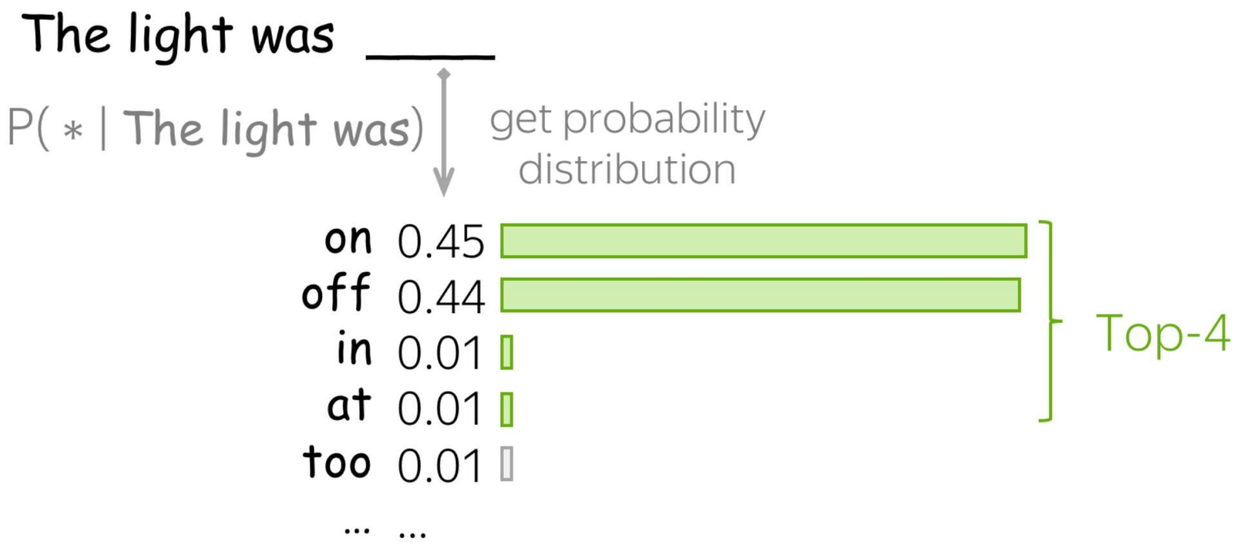
Top-K Sampling: top K tokens

- always sample from top-K most probable tokens (usually, K is about 10-50)

Problems of the Top-K Sampling: Fixed K is Not Good

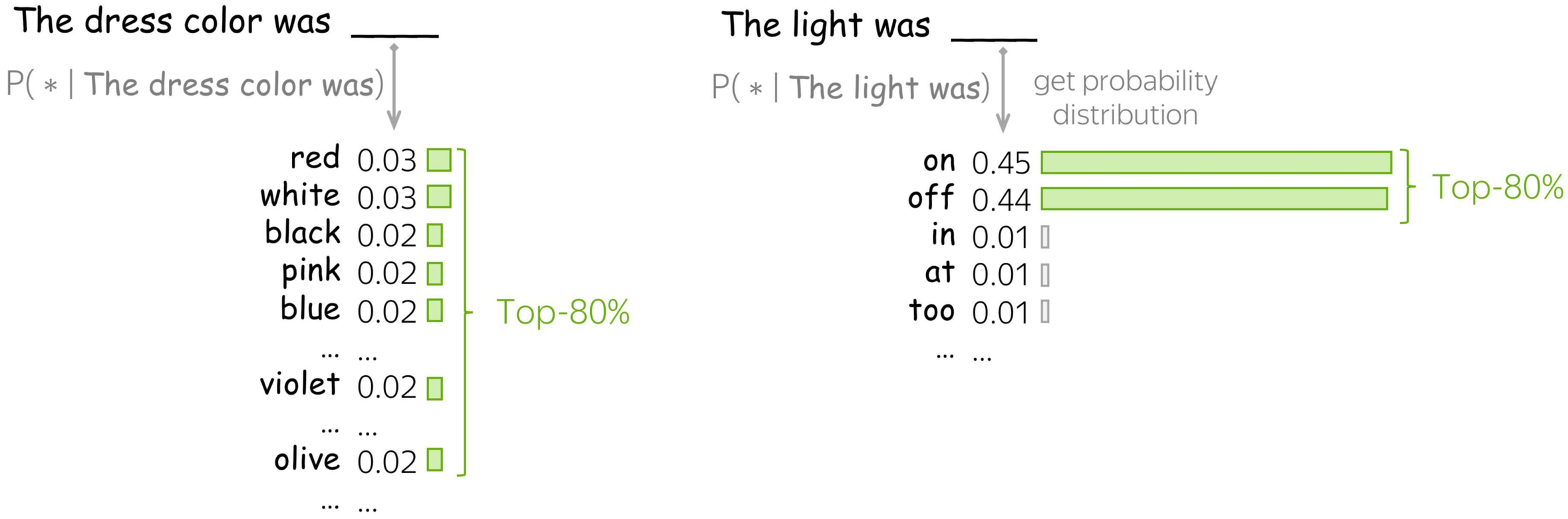
Top-K for a flat distribution: not enough

Top-K for a peaky distribution: too many



Top-p (Nucleus) Sampling: p% of the probability mass

- at each step, pick so many top tokens to cover p% of the probability mass



What is going to happen:

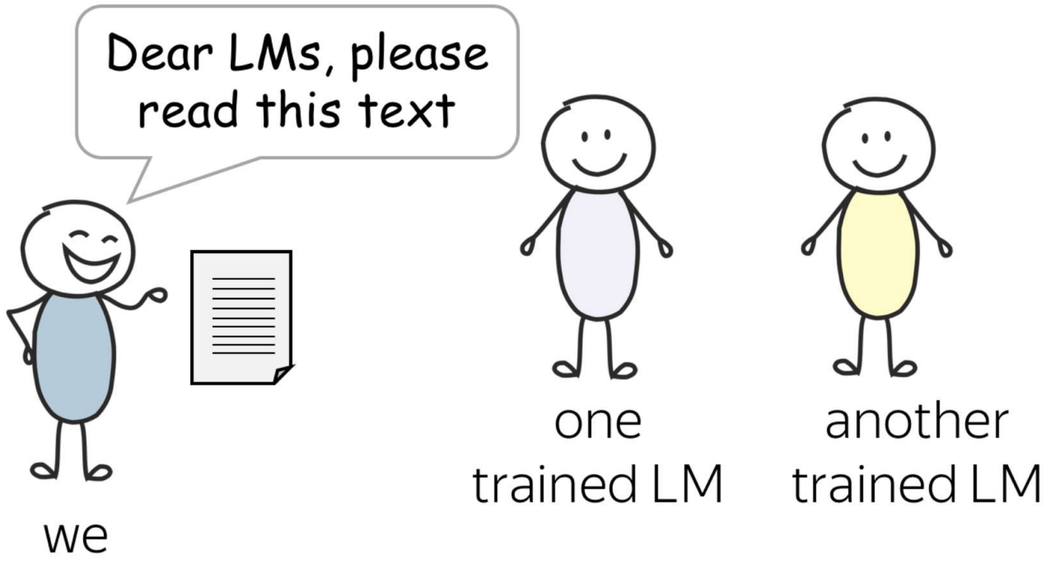
- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

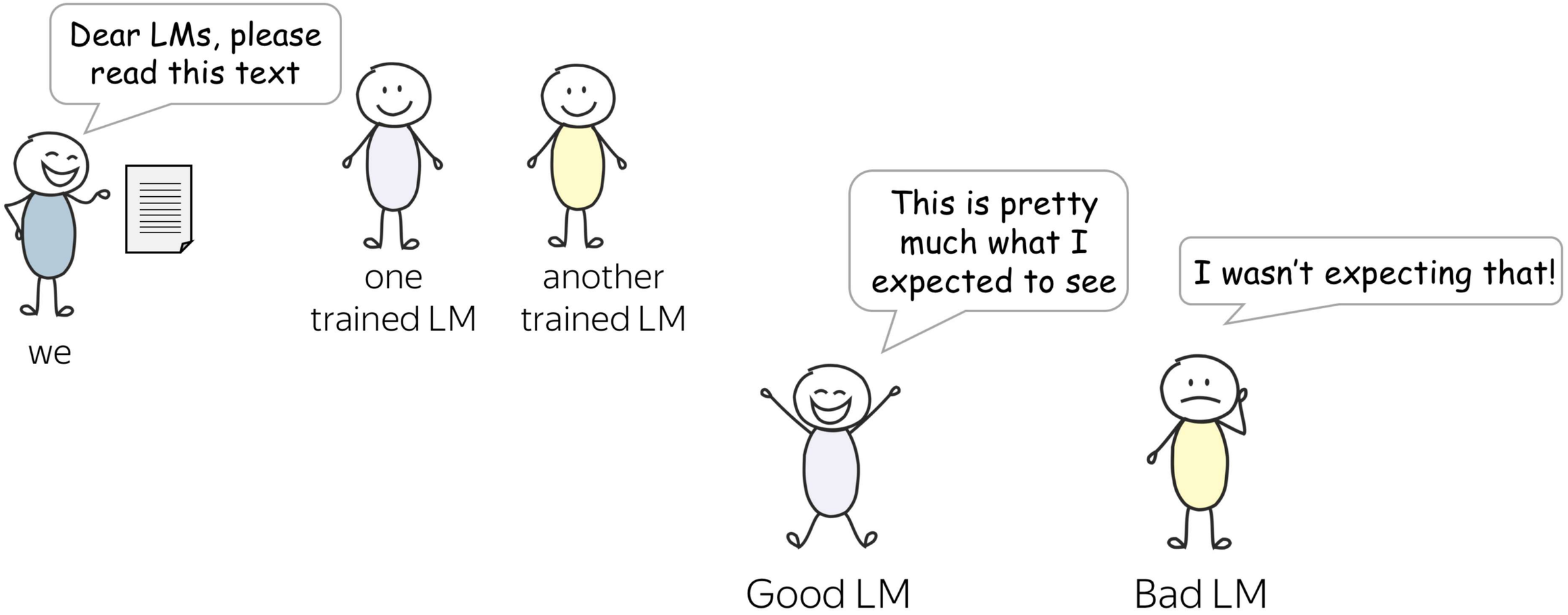
Evaluation Intuition

> TL;DR When reading a new text, how much is a model "surprised"?



Evaluation Intuition

> TL;DR When reading a new text, how much is a model "surprised"?



Perplexity and Text Log- Likelihood

$$L(y_{1:M}) = L(y_1, y_2, \dots, y_M) = \sum_{t=1}^{M} \log_2 p(y_t | y_{<t})$$

Log-likelihood of the text

$$Loss(y_{1:M}) = -\sum_{t=1}^{M} \log p(y_t | y_{<t})$$

Note: cross-entropy (our loss) is negative log-likelihood

Perplexity and Text Log- Likelihood

$$L(y_{1:M}) = L(y_1, y_2, \dots, y_M) = \sum_{t=1}^{M} \log_2 p(y_t|y_{<t})$$

Log-likelihood of the text

$$Loss(y_{1:M}) = -\sum_{t=1}^{M} \log p(y_t|y_{<t})$$

Note: cross-entropy (our loss) is negative log-likelihood

It is more common to report its transformation called perplexity:

$$Perplexity(y_{1:M}) = 2^{-\frac{1}{M}L(y_{1:M})}$$

Low perplexity -> high data likelihood -> good!

Perplexity: Which Values Does it Take?

$$Perplexity(y_{1:M}) = 2^{-\frac{1}{M}L(y_{1:M})}$$

- The **best** perplexity is 1

  If the model is perfect and assigns probability 1 to correct tokens, then the log-probabilities are zero

Perplexity: Which Values Does it Take?

$$Perplexity(y_{1:M}) = 2^{-\frac{1}{M}L(y_{1:M})}$$

- The **best** perplexity is 1

  If the model is perfect and assigns probability 1 to correct tokens, then the log-probabilities are zero

- The **worst** perplexity is |V|

  If the model knows nothing about the data, it assigns probability 1/|V| to all tokens, regardless of context. Then:

$$Perplexity(y_{1:M}) = 2^{-\frac{1}{M}L(y_{1:M})} = 2^{-\frac{1}{M}\sum_{t=1}^{M}\log_2 p(y_t|y_{1:t-1})} = 2^{-\frac{1}{M}\cdot M\cdot\log_2\frac{1}{|V|}} = 2^{\log_2|V|} = |V|$$

Perplexity: Which Values Does it Take?

$$Perplexity(y_{1:M}) = 2^{-\frac{1}{M}L(y_{1:M})}$$

- The **best** perplexity is 1

  If the model is perfect and assigns probability 1 to correct tokens, then the log-probabilities are zero

- The **worst** perplexity is |V|

  If the model knows nothing about the data, it assigns probability 1/|V| to all tokens, regardless of context. Then:

Perplexity is always between 1 and |V|

$$Perplexity(y_{1:M}) = 2^{-\frac{1}{M}L(y_{1:M})} = 2^{-\frac{1}{M}\sum_{t=1}^{M}\log_2 p(y_t|y_{1:t-1})} = 2^{-\frac{1}{M}\cdot M\cdot\log_2\frac{1}{|V|}} = 2^{\log_2|V|} = |V|$$

What is going to happen:

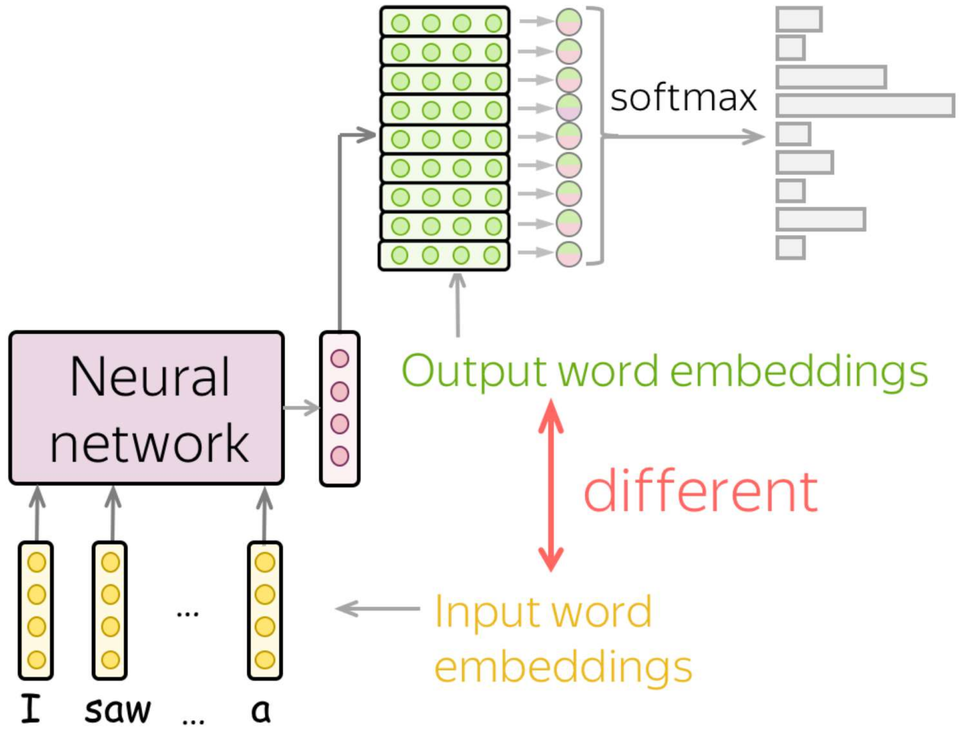
- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips → ○ Weight Tying
-  Analysis and Interpretability

Weight Tying (aka Parameter Sharing)

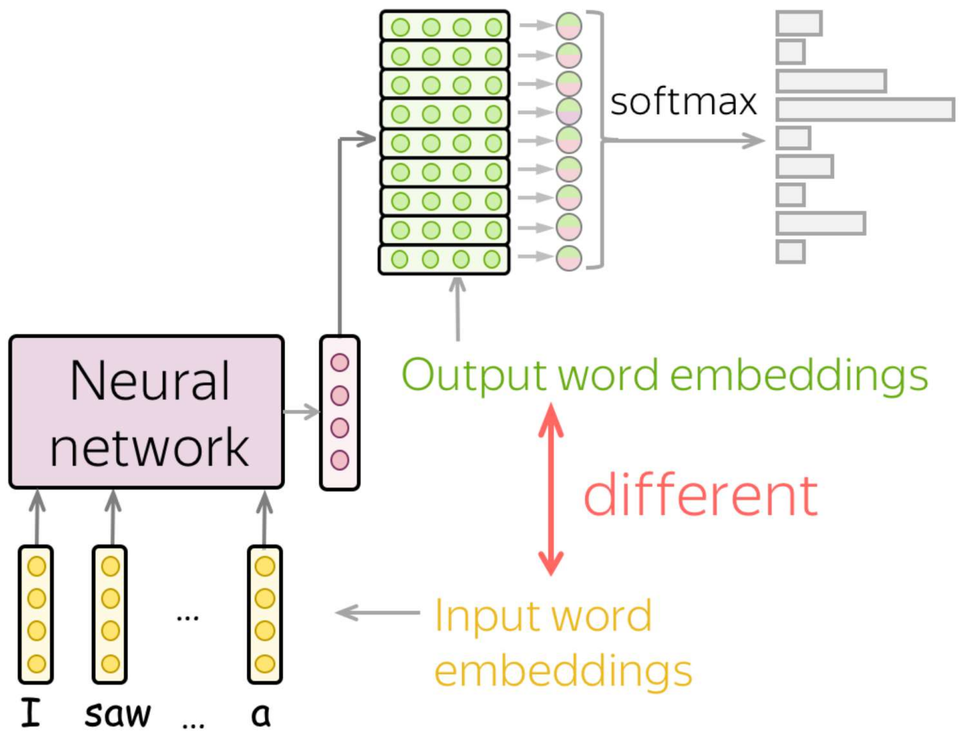
Default (no weight tying)



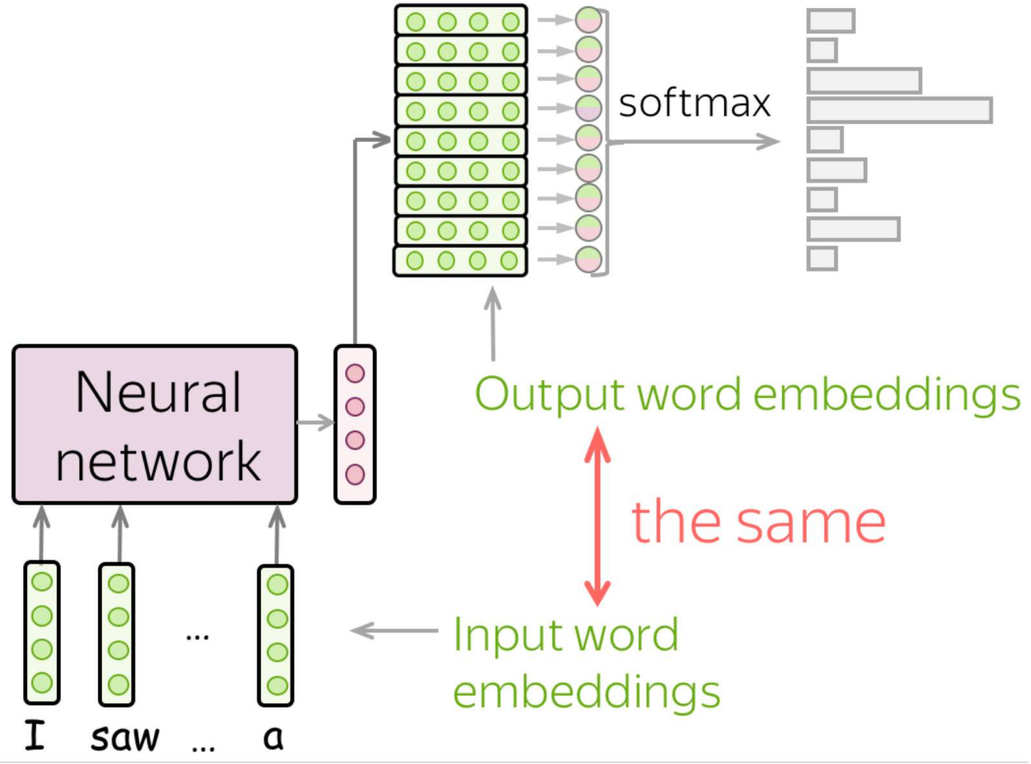
Weight Tying (aka Parameter Sharing)

- Large part of model parameters comes from these matrices - you can reduce model size!

Default (no weight tying)



Weight tying



What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

Examples of generated texts

- Char-level LSTM trained on Latex book on algebraic geometry

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{étale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

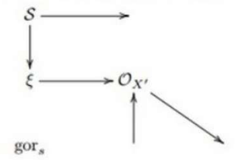
be a morphism of algebraic spaces over S and Y.

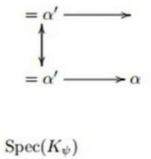
Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram





Spec(K_ψ) Mor$_{Sets}$ d($\mathcal{O}_{\mathcal{X}_{X/k}}, \mathcal{G}$)

is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{étale}}) \longrightarrow \mathcal{O}_{X_t}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{w}})$$

is an isomorphism of covering of $\mathcal{O}_{X_t}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

More hallucinated algebraic geometry. Nice try on the diagram (right).

The examples are from the paper Visualizing and Understanding Recurrent Networks

Examples of generated texts

- Char-level LSTM trained on Linux source code

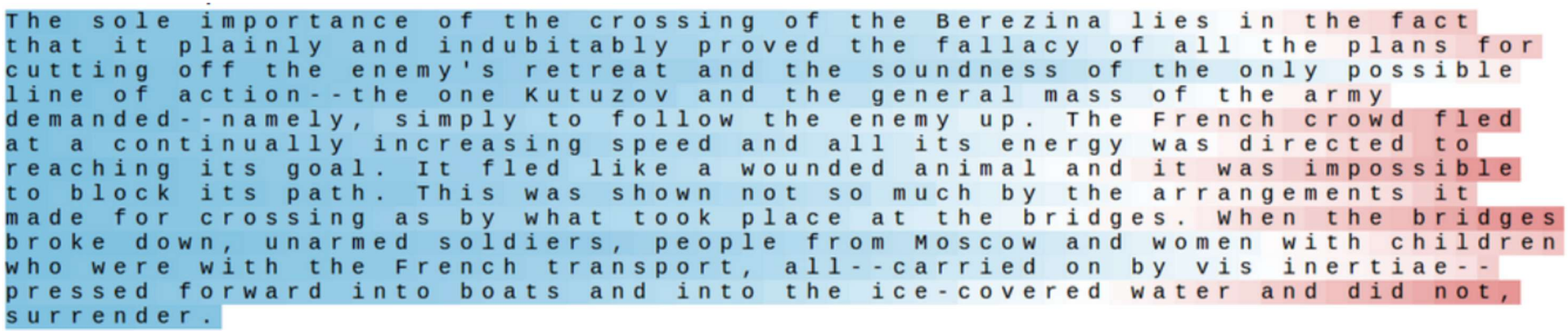
```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

The examples are from the paper Visualizing and Understanding Recurrent Networks

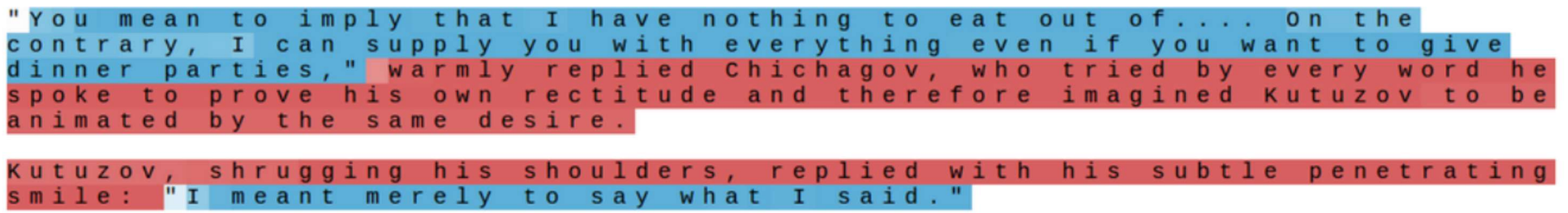
Visualize Neuron Activations: Some Are Interpretable!

- Char-level LSTMs trained on Linux Kernel and War and Peace

Cell sensitive to position in line



Cell that turns on inside quotes



The examples are from the paper Visualizing and Understanding Recurrent Networks

Visualize Neuron Activations: Some Are Interpretable!

- Char-level LSTMs trained on Linux Kernel and War and Peace

Cell that activates inside if statements

```
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;
}
```

The examples are from the paper Visualizing and Understanding Recurrent Networks

Visualize Neuron Activations: Some Are Interpretable!

- Char-level LSTMs trained on Linux Kernel and War and Peace

Cell that turns on inside comments and quotes

```
/* Duplicate LSM field information.  The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
                                       struct audit_field *sf)
{
  int ret = 0;
  char *lsm_str;
  /* our own copy of lsm_str */
  lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
  if (unlikely(!lsm_str))
    return -ENOMEM;
  df->lsm_str = lsm_str;
  /* our own (refreshed) copy of lsm_rule */
  ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                                 (void **)&df->lsm_rule);
  /* Keep currently invalid fields around in case they
   * become valid after a policy reload. */
  if (ret == -EINVAL) {
    pr_warn("audit rule for LSM \'%s\' is invalid\n",
      df->lsm_str);
    ret = 0;
  }
  return ret;
}
```

The examples are from the paper Visualizing and Understanding Recurrent Networks

Visualize Neuron Activations: Some Are Interpretable!

- Char-level LSTMs trained on Linux Kernel and War and Peace

Cell sensitive to the depth of an expression

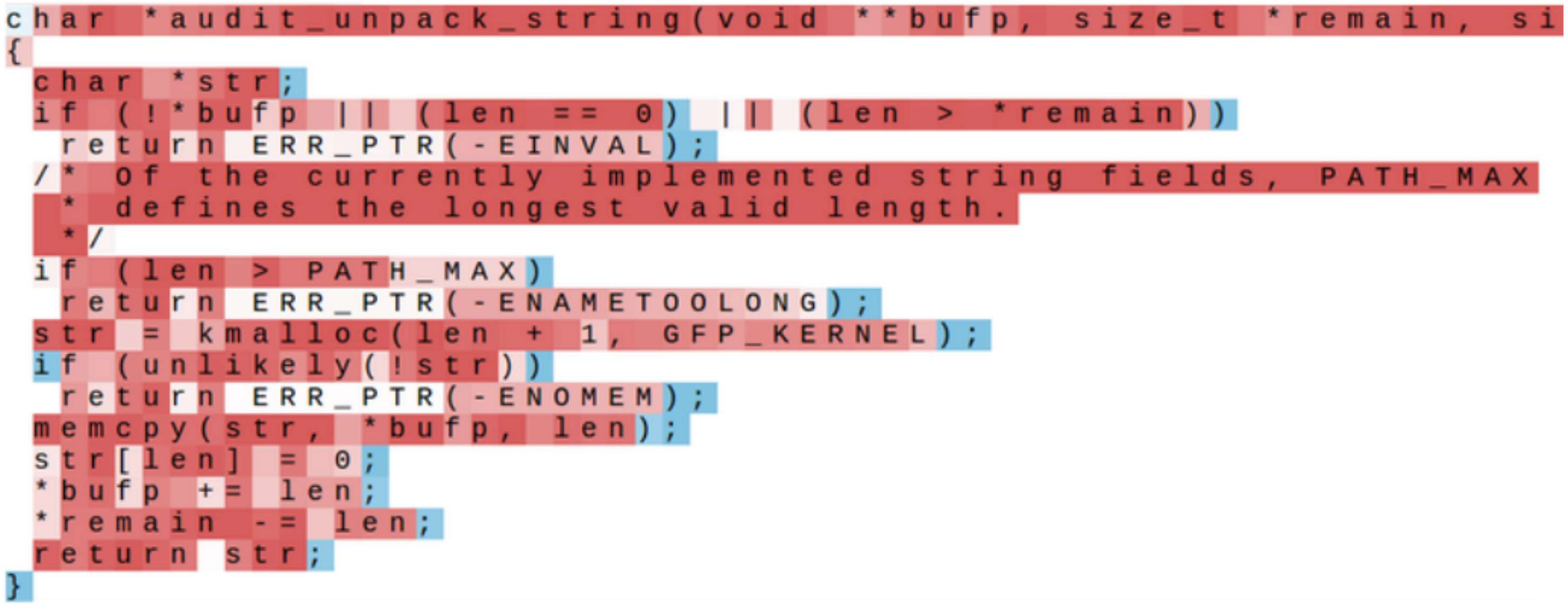
```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
  int i;
  if (classes[class]) {
    for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
      if (mask[i] & classes[class][i])
        return 0;
  }
  return 1;
}
```

The examples are from the paper Visualizing and Understanding Recurrent Networks

Visualize Neuron Activations: Some Are Interpretable!

- Char-level LSTMs trained on Linux Kernel and War and Peace

Cell that might be helpful in predicting new line

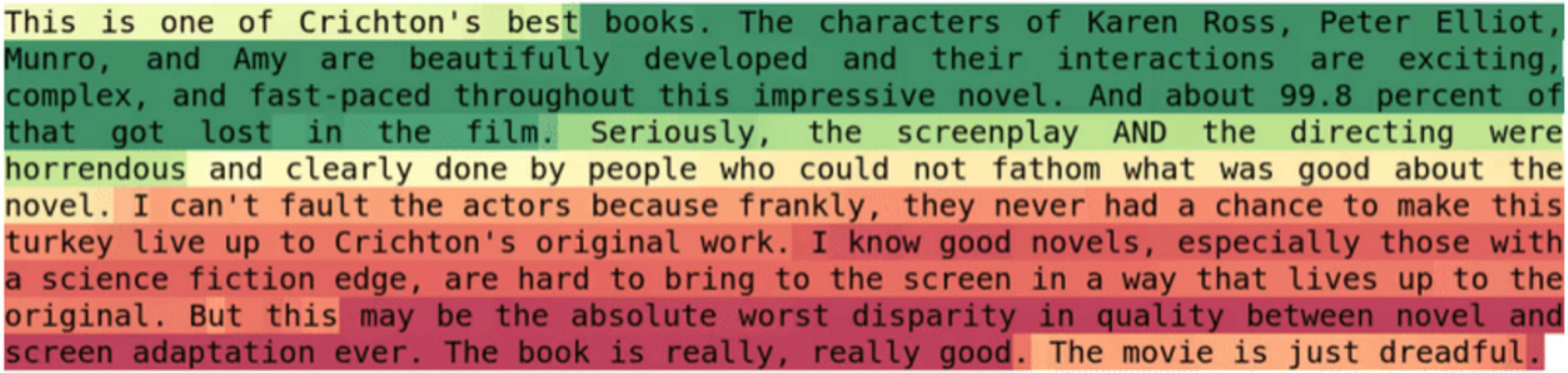


```
char *audit_unpack_string(void **bufp, size_t *remain, si
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
  if (len > PATH_MAX)
    return ERR_PTR(-ENAMETOOLONG);
  str = kmalloc(len + 1, GFP_KERNEL);
  if (unlikely(!str))
    return ERR_PTR(-ENOMEM);
  memcpy(str, *bufp, len);
  str[len] = 0;
  *bufp += len;
  *remain -= len;
  return str;
}
```

The examples are from the paper Visualizing and Understanding Recurrent Networks

More Recent: Sentiment Neuron

- Char-level LSTM trained on Amazon Reviews



The example is from the Open AI blog post

Use Interpretable Neurons to Control Generated Texts

- Fix neuron value, sample from the prefix **I couldn't figure out**

SENTIMENT FIXED TO POSITIVE	SENTIMENT FIXED TO NEGATIVE
I couldn't figure out the shape at first but it definitely does what it's meant to do. It's a great product and I recommend it highly	I couldn't figure out how to use the product. It did not work.At least there was no quality control; this tablet does not work. I would have given it zero stars, but that was not an option.
I couldn't figure out why this movie had been discontinued! Now I can enjoy it anytime I like. So glad to have found it again.	I couldn't figure out how to set it up being that there was no warning on the box. I wouldn't recommend this to anyone.
I couldn't figure out how to use the video or the book that goes along with it, but it is such a fantastic book on how to put it into practice!	I couldn't figure out how to use the gizmo. What a waste of time and money. Might as well through away this junk.
I couldn't figure out how to use just one and my favorite running app. I use it all the time. Good quality, You cant beat the price.	I couldn't figure out how to stop this drivel. At worst, it was going absolutely nowhere, no matter what I did.Needles to say, I skim-read the entire book. Don't waste your time.
I couldn't figure out how to attach these balls to my little portable drums, but these fit the bill and were well worth every penny.	I couldn't figure out how to play it.

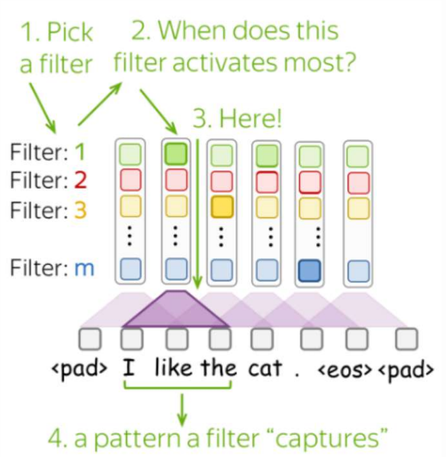
The example is from the Open AI blog post

What About Neurons in CNNs?

A Bit of Analysis

Which patterns a CNN learned? Check your intuition

As we saw in the previous lecture, filters of CNNs trained for sentiment analysis capture very interpretable and informative "clues" for the sentiment (e.g., **poorly designed junk**, **still working perfect**, **a mediocre product**, etc.). What do you think CNNs capture when trained as language models? What could be these "informative clues" for language models?



? Imagine you trained a simple CNN-LM on the data containing parliamentary debates and News Commentary data. How do you imagine the patterns your model might capture?

▶ Possible answers

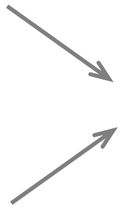
http://lena-voita.github.io/nlp_course/language_modeling.html#research_thinking

Contrastive Evaluation: Text Specific Phenomena

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)?

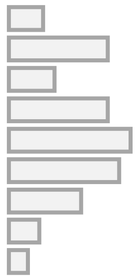
Contrastive Evaluation: Text Specific Phenomena

is/are

The roses in the vase by the door ___?___

P(∗ | The roses in the vase by the door)

get probability distribution from LM



pick tokens of interest

is

are

compare the scores: correct ranking or not?

Contrastive Evaluation: Text Specific Phenomena

- you can create examples with varying difficulty

is/are

The roses ___?___ — Simple: no attractors

The roses in the vase ___?___ — Harder: 1 attractor

The roses in the vase by the door ___?___ — Harder: 2 attractors

Attractors: nouns with different number than the subject

Contrastive Evaluation: Text Specific Phenomena

A Bit of Analysis

NAACL, 2018

Colorless green recurrent networks dream hierarchically

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, Marco Baroni

As we saw earlier, contrastive evaluation can be used to check whether a model is grammatical by testing e.g. subject-verb agreement. But how do we know if it learned syntax or just collocations/semantic? The authors suggest a really fun way to find out: let's take sentences that do not make sense and look if a model generates the correct inflection.

▶ More details

Original example:

It presents the case for marriage equality and states. . .

Generated nonce:

It stays the shuttle for honesty insurance and finds. . .

Can a model get this right?

http://lena-voita.github.io/nlp_course/language_modeling.html#related_papers

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

What is going to happen:

- What is Language Modeling?
- General Framework
- N-Gram Language Models
- Neural Language Models (NN-LM)
- Generation Strategies
- Evaluation
- Practical Tips
-  Analysis and Interpretability

 Learn more in the NLP Course For You → This is up to You!

Thank you!

Lena Voita

PhD student, Uni Edinburgh & Uni Amsterdam

Facebook PhD Fellow in NLP

lena-voita@hotmail.com

https://lena-voita.github.io

@lena_voita

lena-voita