

Language Models

SEPTEMBER 26, 2019

Lena Voita

 lena-voita@yandex-team.ru

 <https://lena-voita.github.io>

 @lena_voita

 lena-voita

Plan

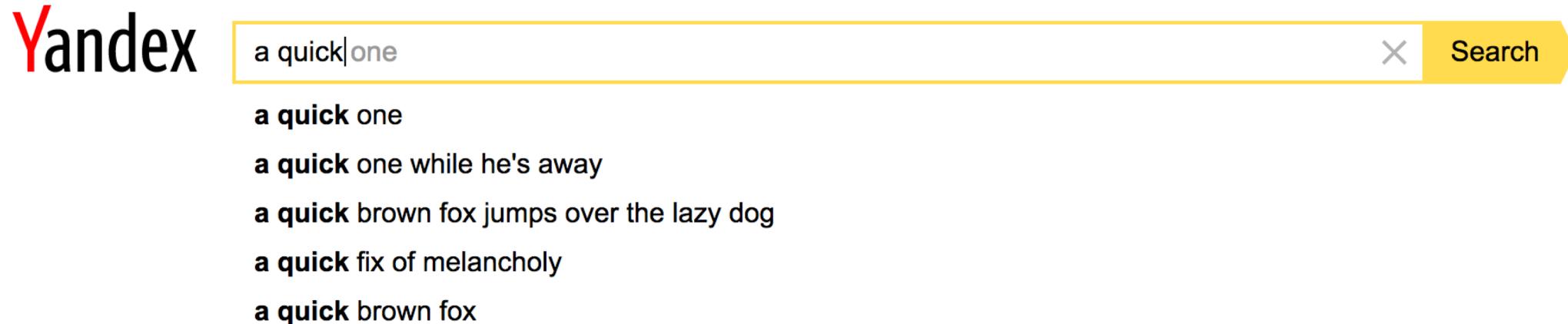
- What is language modeling?
- N-gram Language Models
- Evaluation of LMs
- Neural LMs
- Other versions of LMs
- Training LMs to get embeddings
- Hack of the day!

Language modeling

The goal of language modelling is to **estimate the probability** distribution of various linguistic units:

- symbols
- words
- any token sequences

Where could we see LMs?



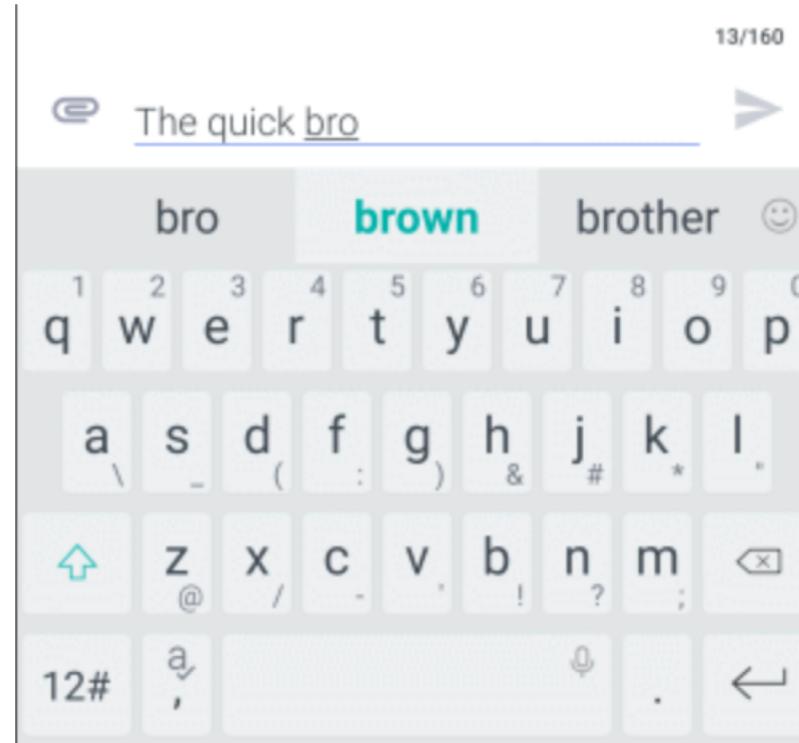
Where could we see LMs?

Yandex Translate

TEXT SITE IMAGE

The screenshot shows the Yandex Translate interface. On the left, the English input field contains the sentence "what do you think". A yellow button labeled "think ↪" is positioned below the input. Above the input, there are icons for clear, volume, microphone, keyboard, and font size. To the right of the input, the word "ENGLISH" is displayed with a yellow dot next to it. On the far right, the Russian output field contains the translated text "ЧТО ВЫ". Above the output, the word "RUSSIAN" is displayed with a grey dot next to it. A double-headed arrow icon is located between the English and Russian sections. At the bottom right of the main input area, the text "12 / 10000" is visible.

Where could we see LMs?



Where could we see LMs?

Android keyboard and spell check.
Not good enough and what's
happening I suspect is that so
manyvpeople

many cowpoke

many spell L's

many voodoo L's

....



Actually, we use our own (human) language model every day

We all are able to decide between similar-sounding options:

She studies morphosyntax

She studies more faux syntax

She's studies morph or syntax

...

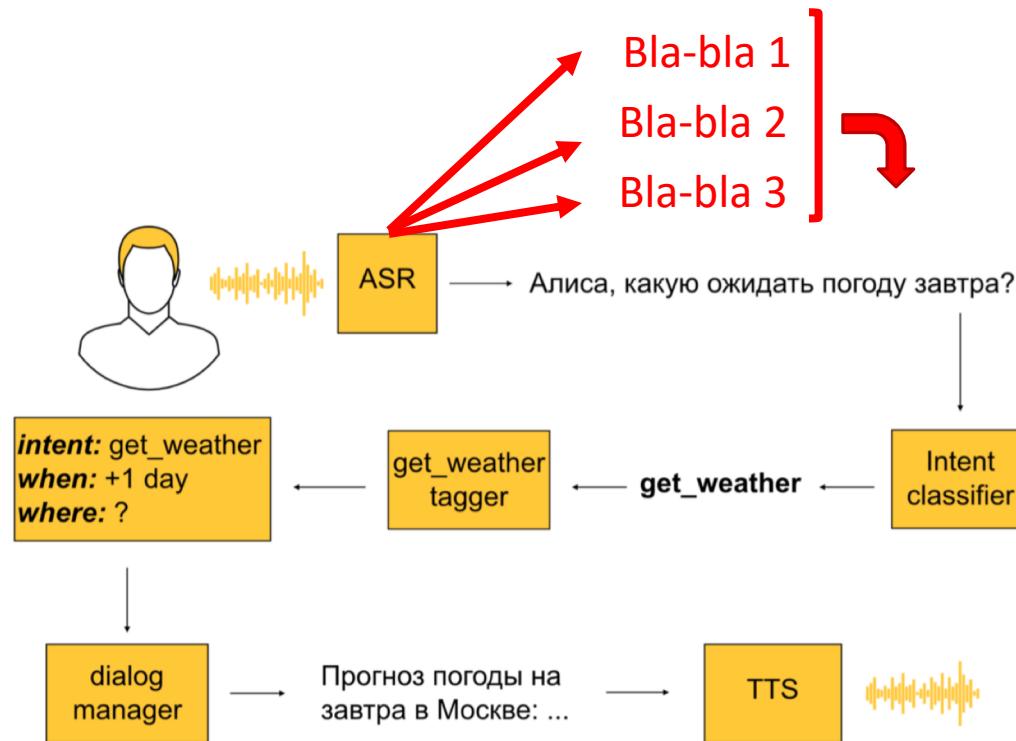


She studies morphosyntax

But for automatic systems...



But for automatic systems...



..., **sentence probabilities** help to decide.

Sentence probabilities: intuitive interpretation

“**Probability of a sentence**” = how likely is it to occur in natural language

- Consider only a specific language (English)

$$P(\text{the cat slept peacefully}) > P(\text{slept the peacefully cat})$$

$$P(\text{she studies morphosyntax}) > P(\text{she studies more faux syntax})$$

Language models in NLP

- It's very difficult to know the true probability of an arbitrary sequence of words.
- But we can define a **language model** that will give us good approximations.
- Like all models, language models will be good at capturing some things and less good for others.

Language Models

Count-based (statistical)

-
-



Neural Language Models

-
-



N-gram Language Models

(AKA COUNT-BASED LANGUAGE MODELS)

How to estimate probabilities?

Task 0: M&M colors

What is the proportion of each color of M&M?

- In 48 packages, I find¹ 2620 M&Ms, as follows:

Red	Orange	Yellow	Green	Blue	Brown
372	544	369	483	481	371

- How to estimate probability of each color from this data?

¹Actually, data from: <https://joshmadison.com/2007/12/02/mms-color-distribution-analysis/>

How to estimate probabilities?

Task 0: M&M colors

What is the proportion of each color of M&M?

- In 48 packages, I find¹ 2620 M&Ms, as follows:

Red	Orange	Yellow	Green	Blue	Brown
372	544	369	483	481	371

- How to estimate probability of each color from this data?

$$P(\text{color}) = \frac{C(\text{color})}{N}, \quad N = \sum_{\text{color}} C(\text{color})$$

¹Actually, data from: <https://joshmadison.com/2007/12/02/mms-color-distribution-analysis/>

How to estimate probabilities?

Task: sentences

- We want to know the probability of word sequence $s = (x^{(1)}, x^{(2)}, \dots, x^{(n)})$ occurring in English.
- Assume we have some training data: large corpus of general English text.
- We want to use this data to estimate the probability of s (even if we never see it in the corpus!)

Sentences that have never occurred

the Archaeopteryx soared jaggedly amidst foliage

vs

jaggedly trees the on flew

Sentences that have never occurred

the Archaeopteryx soared jaggedly amidst foliage

vs

jaggedly trees the on flew

- First is meaningful, second – not
- $C(s) = 0$ for both sentences

Sentences that have never occurred

the Archaeopteryx soared jaggedly amidst foliage

vs

jaggedly trees the on flew

- First is meaningful, second – not
- $C(s) = 0$ for both sentences



Previous (MLE) approach does not work on full sentences

Sparse data problem

- Problem: not enough observations to estimate probabilities well.
(Unlike the M&Ms, where we had large counts for all colors!)
- For sentences, many (most!) will occur rarely if ever in our training data. So we need to do something smarter.

How to fix the problem?

Idea:

estimate $P(s)$ by combining the probabilities of smaller parts of the sentence, which will occur more frequently.

How to fix the problem?

Idea:

estimate $P(s)$ by combining the probabilities of smaller parts of the sentence, which will occur more frequently.

What we will get:

N-gram Language Model!

Deriving an N-gram Model

- We want to estimate $P(s = w_1 \dots w_n)$.
Example: $P(s = \text{the cat slept quietly})$



Deriving an N-gram Model

- We want to estimate $P(s = w_1 \dots w_n)$.
Example: $P(s = \text{the cat slept quietly})$
- This is really a joint probability over the words in s :
 $P(w_1 = \text{the}, w_2 = \text{cat}, w_3 = \text{slept}, w_4 = \text{quietly})$



Deriving an N-gram Model

- We want to estimate $P(s = w_1 \dots w_n)$.
Example: $P(s = \text{the cat slept quietly})$
- This is really a joint probability over the words in s :

$P(w_1 = \text{the}, w_2 = \text{cat}, w_3 = \text{slept}, w_4 = \text{quietly})$



- Recall that for a joint probability, $P(X, Y) = P(Y|X)P(X)$. So,

$P(\text{the cat slept quietly}) = P(\text{quietly}|\text{the cat slept}) \cdot P(\text{the cat slept}) =$
 $P(\text{quietly}|\text{the cat slept}) \cdot P(\text{slept} | \text{the cat}) \cdot P(\text{cat}|\text{the}) \cdot P(\text{the})$

Deriving an N-gram Model

- The chain rule gives us:

$$P(w_1, w_2, \dots, w_n) = \prod_{i=1}^n P(w_i | w_1 \dots w_{i-1})$$

Any problems?

Deriving an N-gram Model

- The chain rule gives us:

$$P(w_1, w_2, \dots, w_n) = \prod_{i=1}^n P(w_i | w_1 \dots w_{i-1})$$

Any problems? Oh, yeah!

Deriving an N-gram Model

- The chain rule gives us:

$$P(w_1, w_2, \dots, w_n) = \prod_{i=1}^n P(w_i | w_1 \dots w_{i-1})$$

Any problems? Oh, yeah!

- Many of these conditional probs are just as sparse!

If we want $P(I \text{ spent three years before the mast } \dots)$

we still need $P(\text{mast} | I \text{ spent three years before the })$

Deriving an N-gram Model

- We make an **independence assumption**: the probability of a word only depends on a fixed number of previous words (**history**).

Deriving an N-gram Model

- We make an **independence assumption**: the probability of a word only depends on a fixed number of previous words (**history**).

Markov property:

$$P(w_i | w_1, w_2, \dots, w_{i-1}) = P(w_i | w_{i-n+1}, \dots, w_{i-1})$$

Deriving an N-gram Model

- We make an **independence assumption**: the probability of a word only depends on a fixed number of previous words (**history**).

Markov property:

$$P(w_i | w_1, w_2, \dots, w_{i-1}) = P(w_i | w_{i-n+1}, \dots, w_{i-1})$$

- trigram model: $P(w_i | w_0, w_1, \dots, w_{i-1}) \approx P(w_i | w_{i-2}, w_{i-1})$
- bigram model: $P(w_i | w_0, w_1, \dots, w_{i-1}) \approx P(w_i | w_{i-1})$
- unigram model: $P(w_i | w_0, w_1, \dots, w_{i-1}) \approx P(w_i)$

N-gram Language Models

$$P(w_i | w_1, \dots, w_{i-1})$$

N-gram Language Models

$$P(w_i | w_1, \dots, w_{i-1}) = P(w_i | \underbrace{w_{i-n+1}, \dots, w_{i-1}}_{(n-1) \text{ words}}) = \text{(our assumption)}$$

N-gram Language Models

$$P(w_i | w_1, \dots, w_{i-1}) = P(w_i | \underbrace{w_{i-n+1}, \dots, w_{i-1}}_{(n-1) \text{ words}}) =$$

prob of an n-gram → $P(w_i, w_{i-1}, \dots, w_{i-n+1})$ (our assumption)

prob of an (n-1)-gram → $P(w_{i-1}, \dots, w_{i-n+1})$ (definition of a conditional probability)

N-gram Language Models

$$P(w_i | w_1, \dots, w_{i-1}) = P(w_i | \underbrace{w_{i-n+1}, \dots, w_{i-1}}_{(n-1) \text{ words}}) =$$

prob of an n-gram → $P(w_i, w_{i-1}, \dots, w_{i-n+1})$ (our assumption)

prob of an (n-1)-gram → $P(w_{i-1}, \dots, w_{i-n+1})$ (definition of a conditional probability)

Question: How do we get these n -gram and $(n-1)$ -gram probabilities?

N-gram Language Models

$$P(w_i | w_1, \dots, w_{i-1}) = P(w_i | \underbrace{w_{i-n+1}, \dots, w_{i-1}}_{(n-1) \text{ words}}) =$$

(our assumption)

$$\begin{aligned} \text{prob of an n-gram} &\rightarrow P(w_i, w_{i-1}, \dots, w_{i-n+1}) \\ \text{prob of an (n-1)-gram} &\rightarrow = \frac{P(w_i, w_{i-1}, \dots, w_{i-n+1})}{P(w_{i-1}, \dots, w_{i-n+1})} \end{aligned}$$

(definition of a
conditional
probability)

Question: How do we get these n -gram and $(n-1)$ -gram probabilities?

Answer: By counting them in some large corpus of text!

$$\approx \frac{\text{count}(w_i, w_{i-1}, \dots, w_{i-n+1})}{\text{count}(w_{i-1}, \dots, w_{i-n+1})}$$

(statistical approximation)

Example: 4-gram model

The cat slept quietly on the _____



Example: 4-gram model

The cat slept quietly on the _____

discard

condition on this



Example: 4-gram model

The cat slept quietly on the _____

discard

condition on this



$$P(w_i | \text{the cat slept quietly on the}) \approx \frac{\text{count}(\text{quietly on the } w_i)}{\text{count}(\text{quietly on the})}$$

Problems with n-gram Language Models

$$P(w_i | \text{the cat slept quietly on the}) \approx \frac{\text{count(quietly on the } w_i\text{)}}{\text{count(quietly on the)}}$$

Problems with n-gram Language Models

$$P(w_i | \text{the cat slept quietly on the}) \approx \frac{\text{count}(\text{quietly on the } w_i)}{\text{count}(\text{quietly on the})}$$

Problem. What if “quietly on the” never occurred in the data? Then we can’t calculate probability for any w_i !



Problems with n-gram Language Models

$$P(w_i | \text{the cat slept quietly on the}) \approx \frac{\text{count}(\text{quietly on the } w_i)}{\text{count}(\text{quietly on the})}$$

Problem. What if “quietly on the” never occurred in the data? Then we can’t calculate probability for any w_i !

(Partial) solution. Backoff smoothing

Backoff (aka “stupid backoff”)

Sometimes it helps to use
less context

- Condition on less context
for contexts you haven't
learned much about

Backoff:

- use trigram if you have
good evidence,
- otherwise bigram,
- otherwise unigram

“quietly on the” is not seen → try “on the”

$$P(w_i|quietly\ on\ the) \approx P(w_i|on\ the)$$

“on the” is not seen → try “the”

$$P(w_i|on\ the) \approx P(w_i|the)$$

“the” is not seen → try unigram

$$P(w_i|the) \approx P(w_i)$$

More clever: Linear Interpolation

Interpolation: mix unigram, bigram, trigram, ...

$$\begin{aligned}\hat{P}(w_i | w_{i-1}, w_{i-2}, w_{i-3}) \approx & \lambda_3 P(w_i | w_{i-1}, w_{i-2}, w_{i-3}) + \\ & + \lambda_2 P(w_i | w_{i-1}, w_{i-2}) \\ & + \lambda_1 P(w_i | w_{i-1}) \\ & + \lambda_0 P(w_i)\end{aligned}\quad \sum_{k=0}^{n-1} \lambda_k = 1$$

More clever: Linear Interpolation

Interpolation: mix unigram, bigram, trigram, ...

$$\begin{aligned}\hat{P}(w_i | w_{i-1}, w_{i-2}, w_{i-3}) \approx & \lambda_3 P(w_i | w_{i-1}, w_{i-2}, w_{i-3}) + \\ & + \lambda_2 P(w_i | w_{i-1}, w_{i-2}) \\ & + \lambda_1 P(w_i | w_{i-1}) \\ & + \lambda_0 P(w_i)\end{aligned}\quad \sum_{k=0}^{n-1} \lambda_k = 1$$

Question: How to pick λ_k ?

More clever: Linear Interpolation

Interpolation: mix unigram, bigram, trigram, ...

$$\begin{aligned}\hat{P}(w_i | w_{i-1}, w_{i-2}, w_{i-3}) \approx & \lambda_3 P(w_i | w_{i-1}, w_{i-2}, w_{i-3}) + \\ & + \lambda_2 P(w_i | w_{i-1}, w_{i-2}) \\ & + \lambda_1 P(w_i | w_{i-1}) \\ & + \lambda_0 P(w_i)\end{aligned}\quad \sum_{k=0}^{n-1} \lambda_k = 1$$

Question: How to pick λ_k ?

Answer: Use your validation set!

Problems with n-gram Language Models

$$P(w_i | \text{the cat slept quietly on the}) \approx \frac{\text{count}(\text{quietly on the } w_i)}{\text{count}(\text{quietly on the})}$$

Problem. What if “quietly on the” never occurred in the data? Then we can’t calculate probability for any w_i !

(Partial) solution. Backoff smoothing

Problems with n-gram Language Models

Problem. What if “*quietly on the w_i* ” never occurred in the data?
Then w_i has probability 0

$$P(w_i | \text{the cat slept quietly on the}) \approx \frac{\text{count}(\text{quietly on the } w_i)}{\text{count}(\text{quietly on the})}$$

Problems with n-gram Language Models

Problem. What if “*quietly on the w_i* ” never occurred in the data? Then w_i has probability 0

(Partial) solution. Smoothing
(add-one, Kneser-Ney, etc.)

$$P(w_i | \text{the cat slept quietly on the}) \approx \frac{\text{count}(\text{quietly on the } w_i)}{\text{count}(\text{quietly on the})}$$

Laplace-smoothing (aka add-one smoothing)

- Pretend we saw each word one more time than we did
- Just add one to all the counts!

If 1 if too rude, add small δ to count for every $w_i \in V$.

$$\hat{P}(w_i | w_{i-1}, \dots, w_{i-n+1}) = \frac{\delta + P(w_i, w_{i-1}, \dots, w_{i-n+1})}{\delta \cdot |V| + P(w_{i-1}, \dots, w_{i-n+1})}$$

Kneser-Ney Smoothing: more clever

- Idea: the lower-order model is significant only when count is small or zero in the higher-order model, and so should be optimized for that purpose.
- Example: suppose “San Francisco” is common, but “Francisco” occurs only after “San”.
- “Francisco” will get a high unigram probability, and so absolute discounting will give a high probability to “Francisco” appearing after novel bigram histories.
- Better to give “Francisco” a low unigram probability, because the only time it occurs is after “San”, in which case the bigram model fits well.

Kneser-Ney Smoothing: more clever

- Let the count assigned to each unigram be the number of different words that it follows. Define:

$$N_{1+}(\bullet \ w_i) = |\{w_{i-1} : c(w_{i-1}w_i) > 0\}|$$

$$N_{1+}(\bullet \bullet) = \sum_{w_i} N_{1+}(\bullet \ w_i)$$

- Let lower-order distribution be:

$$p_{KN}(w_i) = \frac{N_{1+}(\bullet \ w_i)}{N_{1+}(\bullet \bullet)}$$

- Put it all together:

$$p_{KN}(w_i | w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^i) - \delta, 0\}}{\sum_{w_i} c(w_{i-n+1}^i)} + \frac{\delta}{\sum_{w_i} c(w_{i-n+1}^i)} N_{1+}(w_{i-n+1}^{i-1} \bullet) p_{KN}(w_i | w_{i-n+2}^{i-1})$$

Problems with n-gram Language Models

Problem. What if “*quietly on the* w_i ” never occurred in the data? Then w_i has probability 0

(Partial) solution. Smoothing
(add-one, Kneser-Ney, etc.)

$$P(w_i | \text{the cat slept quietly on the}) \approx \frac{\text{count}(\text{quietly on the } w_i)}{\text{count}(\text{quietly on the})}$$

Problem. What if “*quietly on the*” never occurred in the data? Then we can’t calculate probability for any w_i !

(Partial) solution. Backoff!

How to evaluate Language Models

Intrinsic: evaluation on a specific/intermediate subtask

- perplexity/cross-entropy

Extrinsic: evaluation on a real task

- plug your LM it into a SMT/ASR/etc system.
- train the system with different LMs
- If the result with your LM is better -> win!

Cross-entropy and Perplexity

- For $(w_1 w_2 \dots w_n)$ with large n , per-word cross-entropy is well approximated by:

$$H_M(w_1 w_2 \dots w_n) = -\frac{1}{n} \cdot \log P_M(w_1 w_2 \dots w_n)$$

- Lower cross-entropy ⇒ model is better at predicting next word.
- Perplexity (reported in papers):

$$\text{perplexity} = 2^{cross-entropy}$$

Language Models

Count-based (statistical)

-
-



Neural Language Models

-
-



Language Models

Count-based (statistical)

- make an n-th order Markov assumption
- estimate n-gram probabilities
(counting and smoothing)

Neural Language Models

-
-

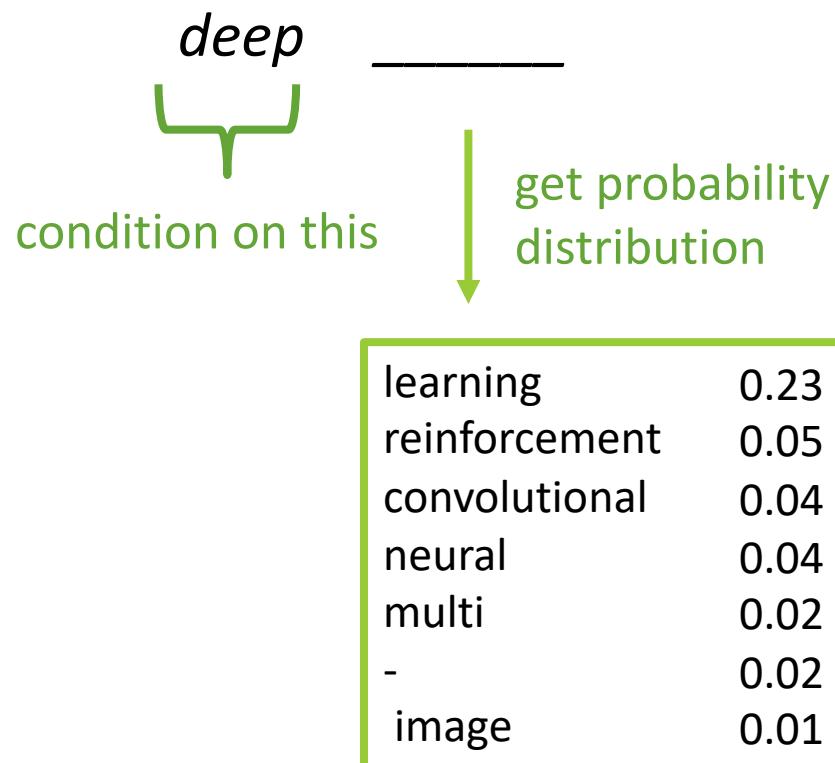


How to generate text using N-gram LM?

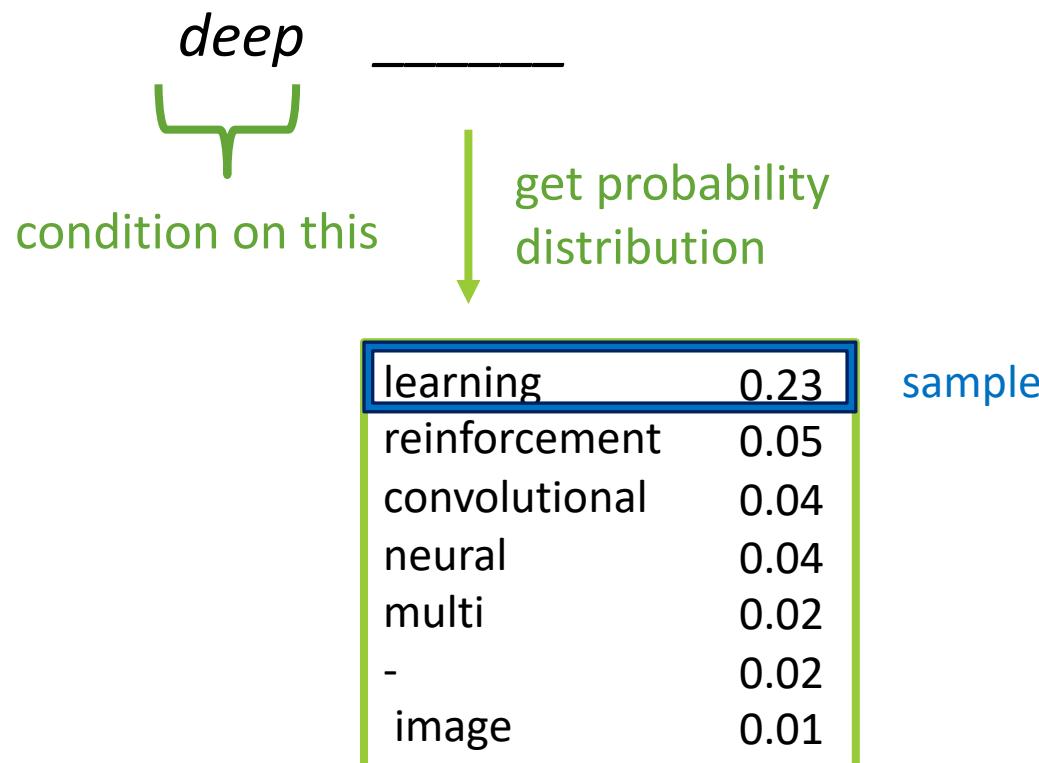
deep
 _____

condition on this

How to generate text using N-gram LM?



How to generate text using N-gram LM?



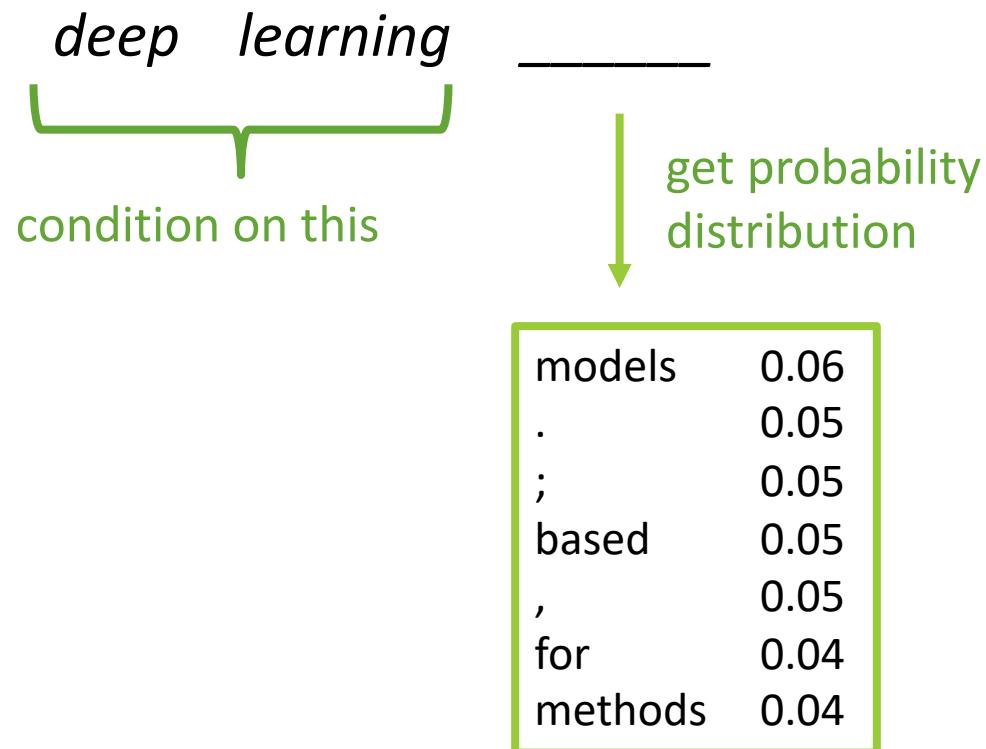
How to generate text using N-gram LM?

deep learning _____

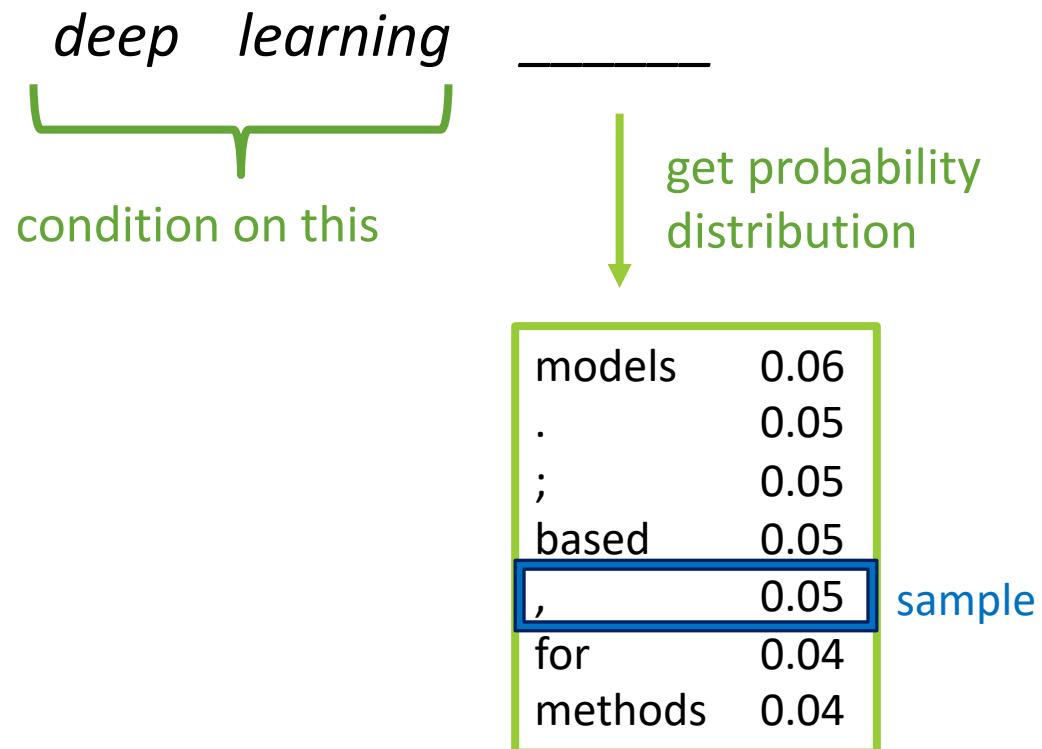


condition on this

How to generate text using N-gram LM?



How to generate text using N-gram LM?



How to generate text using N-gram LM?

deep learning , _____

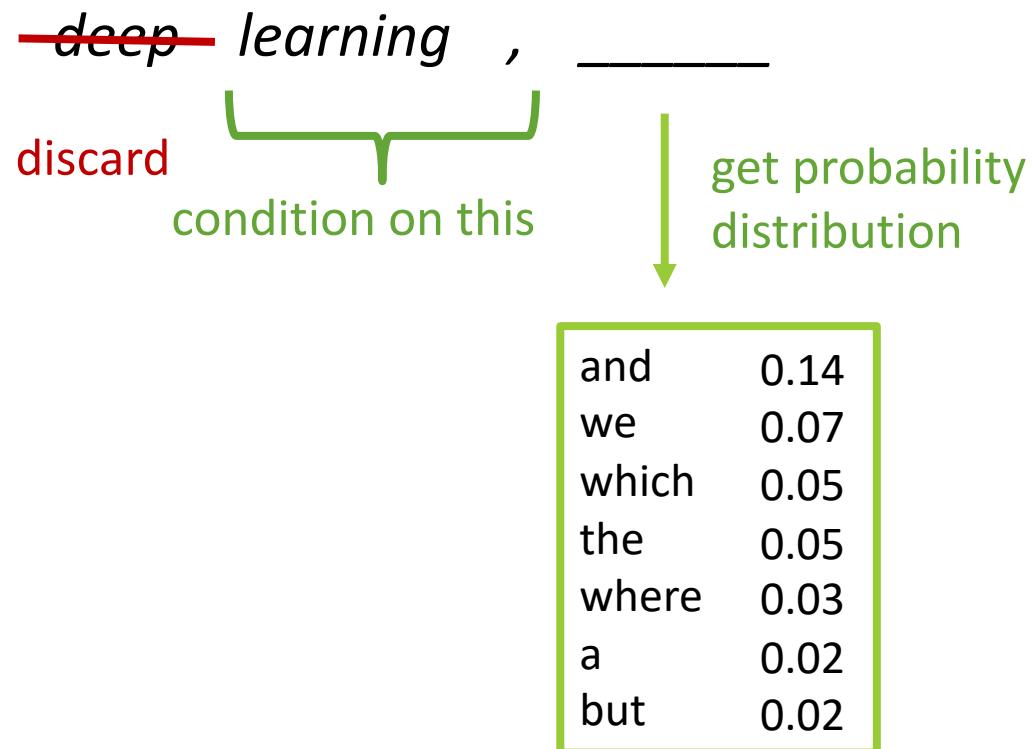
How to generate text using N-gram LM?

~~deep~~ *learning* , _____

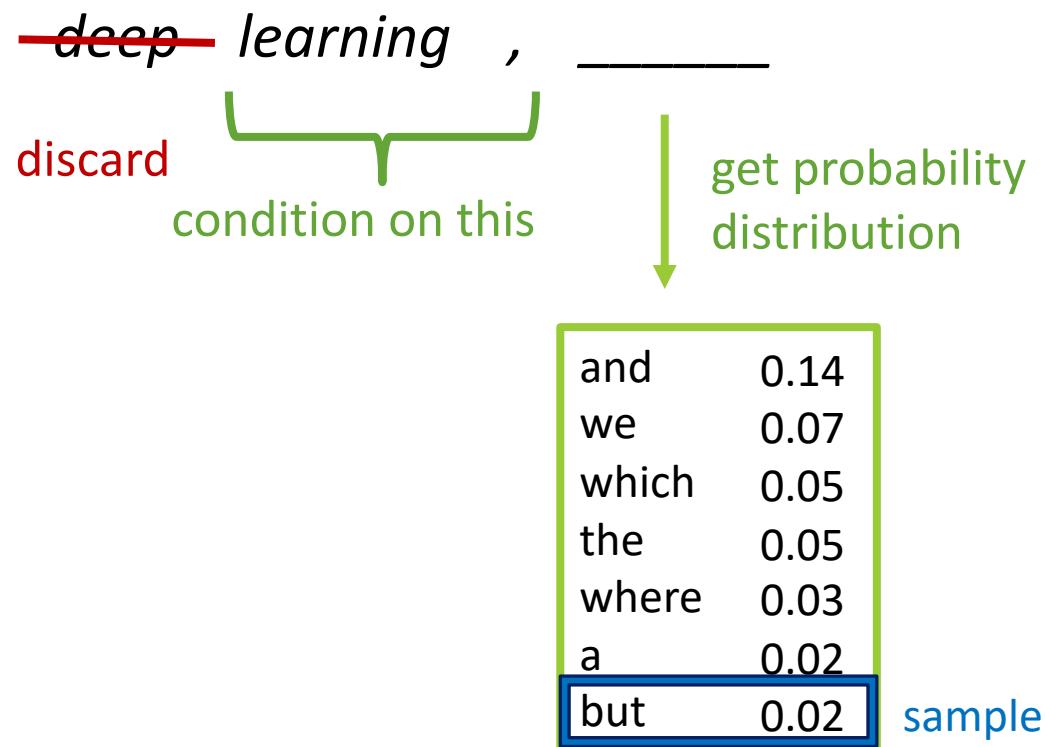
discard

condition on this

How to generate text using N-gram LM?



How to generate text using N-gram LM?



How to generate text using N-gram LM?

deep learning , but _____

How to generate text using N-gram LM?

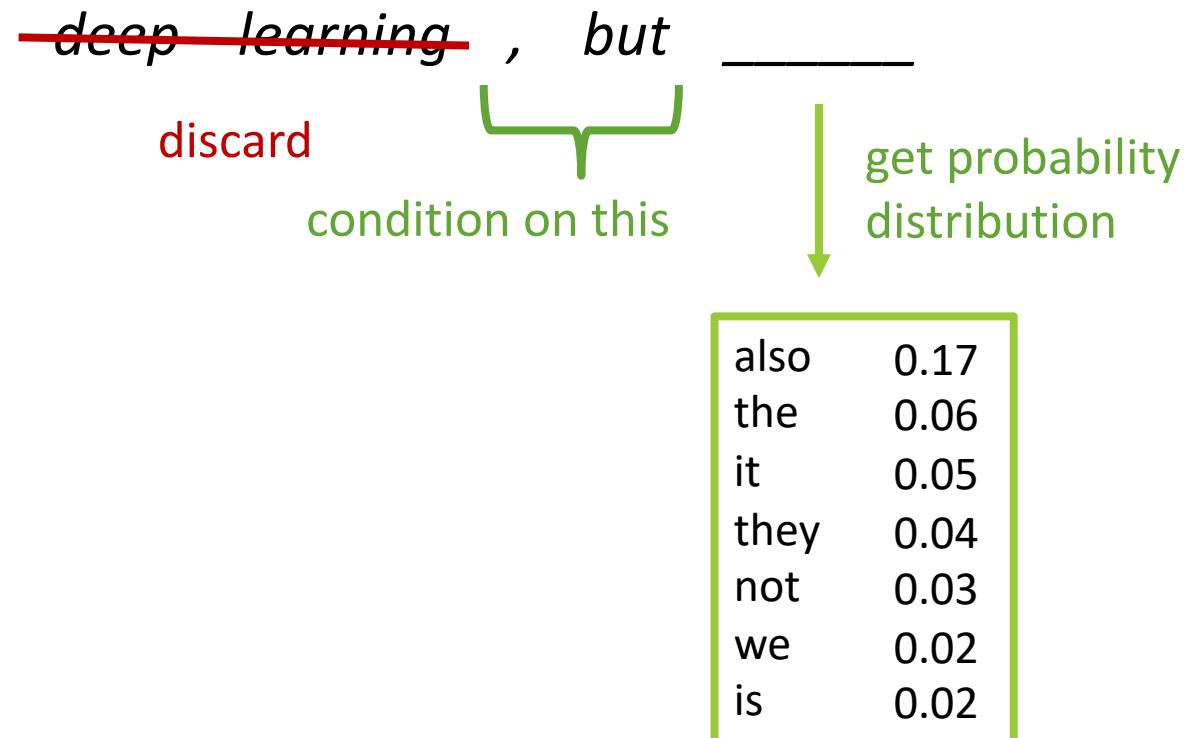
~~-deep learning~~, but _____

discard

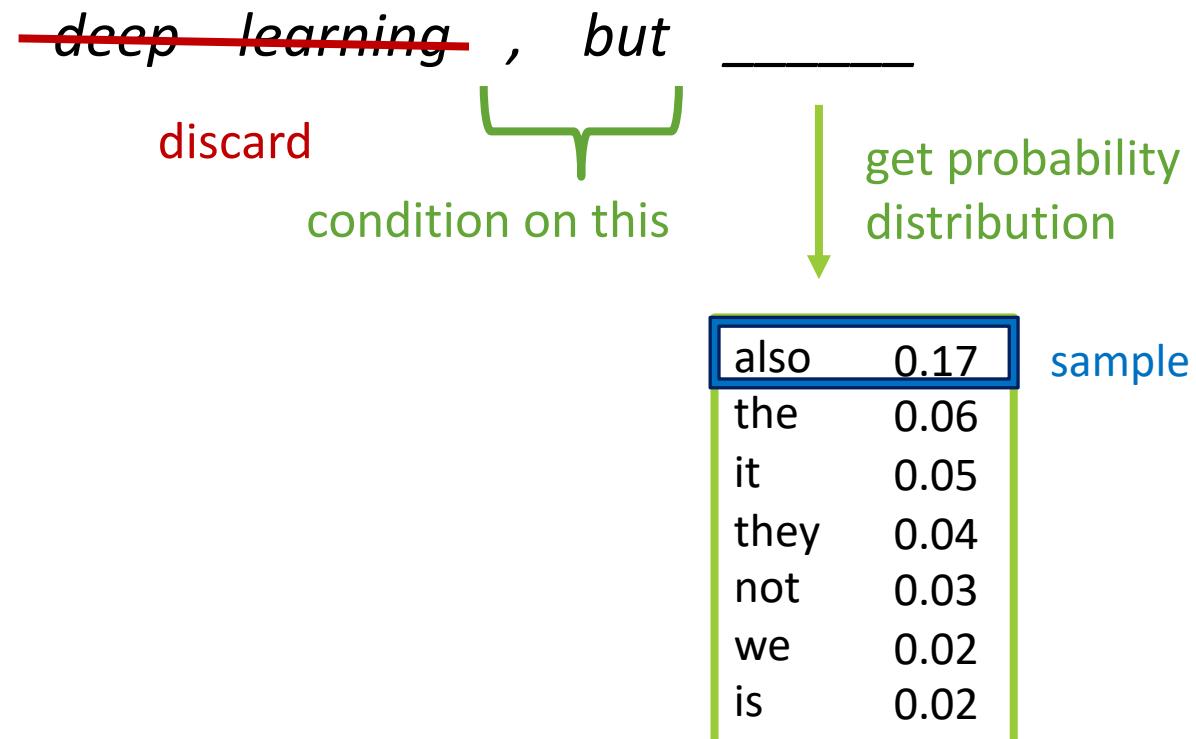


condition on this

How to generate text using N-gram LM?



How to generate text using N-gram LM?



How to generate text using N-gram LM?

deep learning , but also _____

How to generate text using N-gram LM?

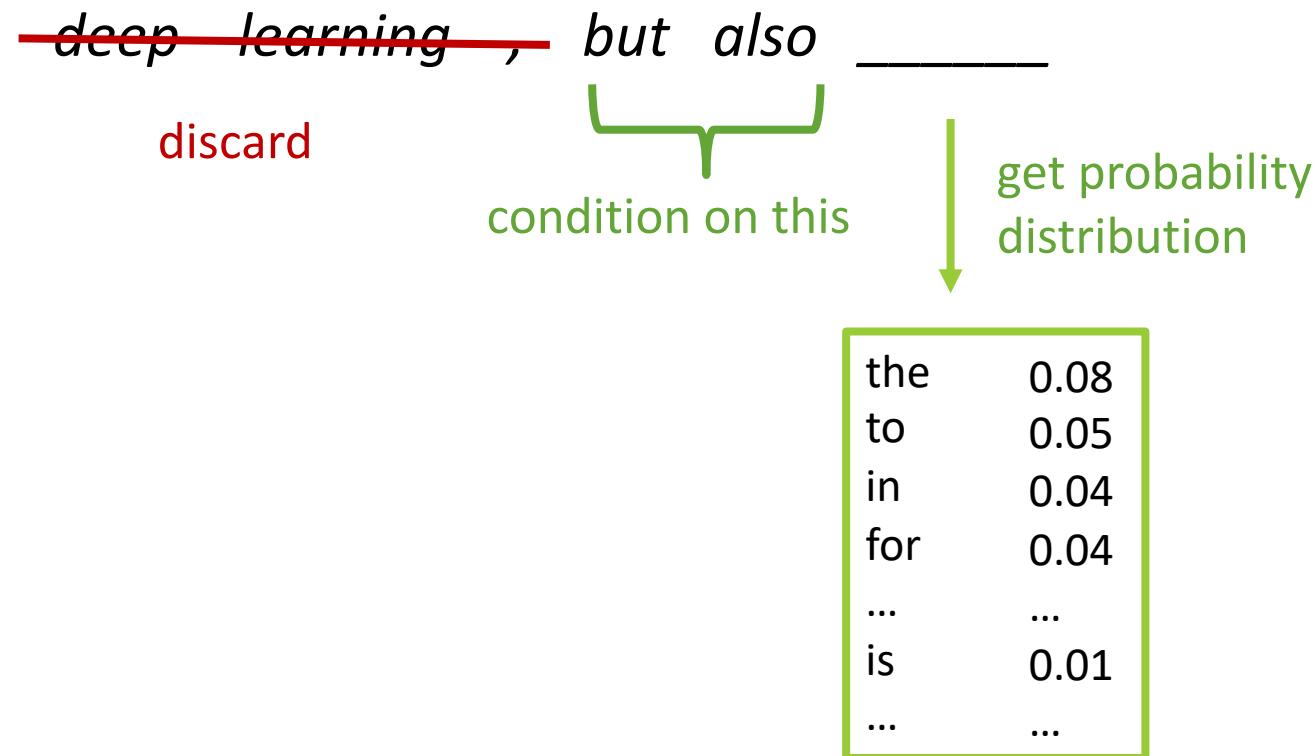
~~deep learning~~, but also _____

discard

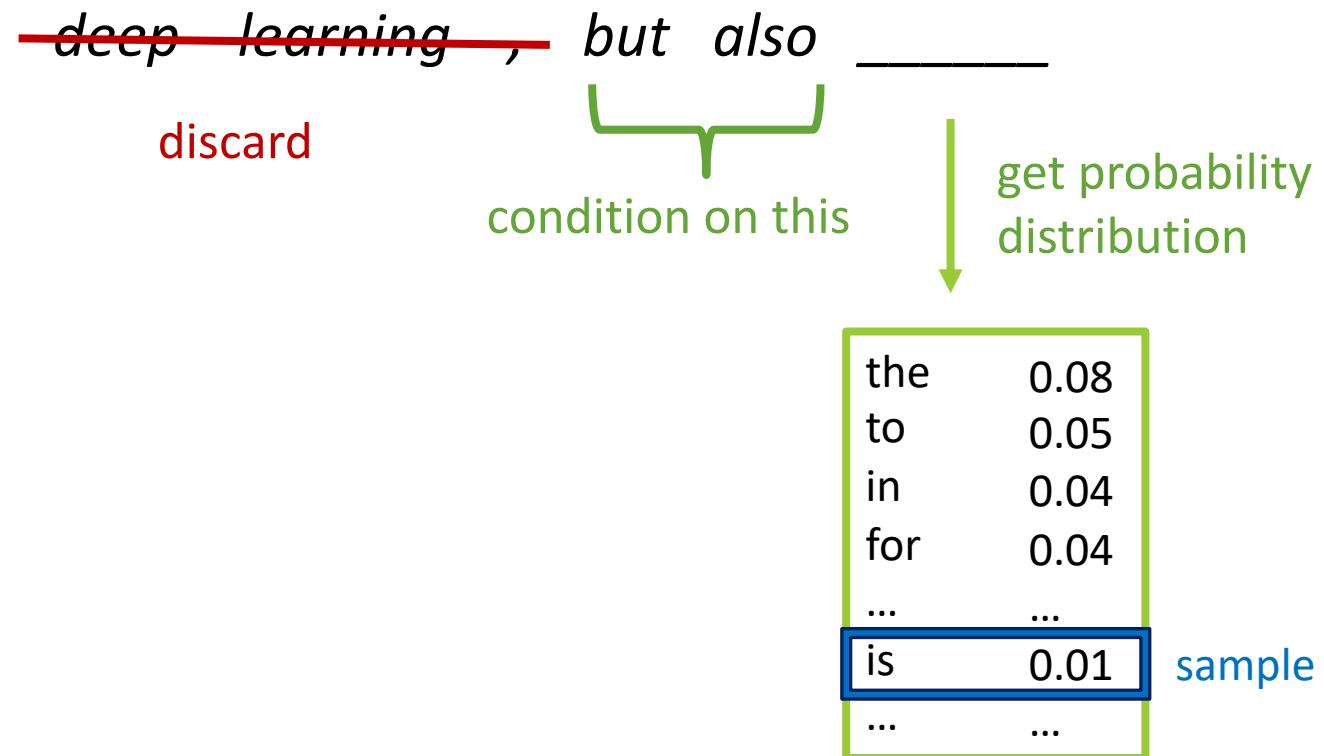


condition on this

How to generate text using N-gram LM?



How to generate text using N-gram LM?



How to generate text using N-gram LM?

deep learning , but also is _____

How to generate text using N-gram LM?

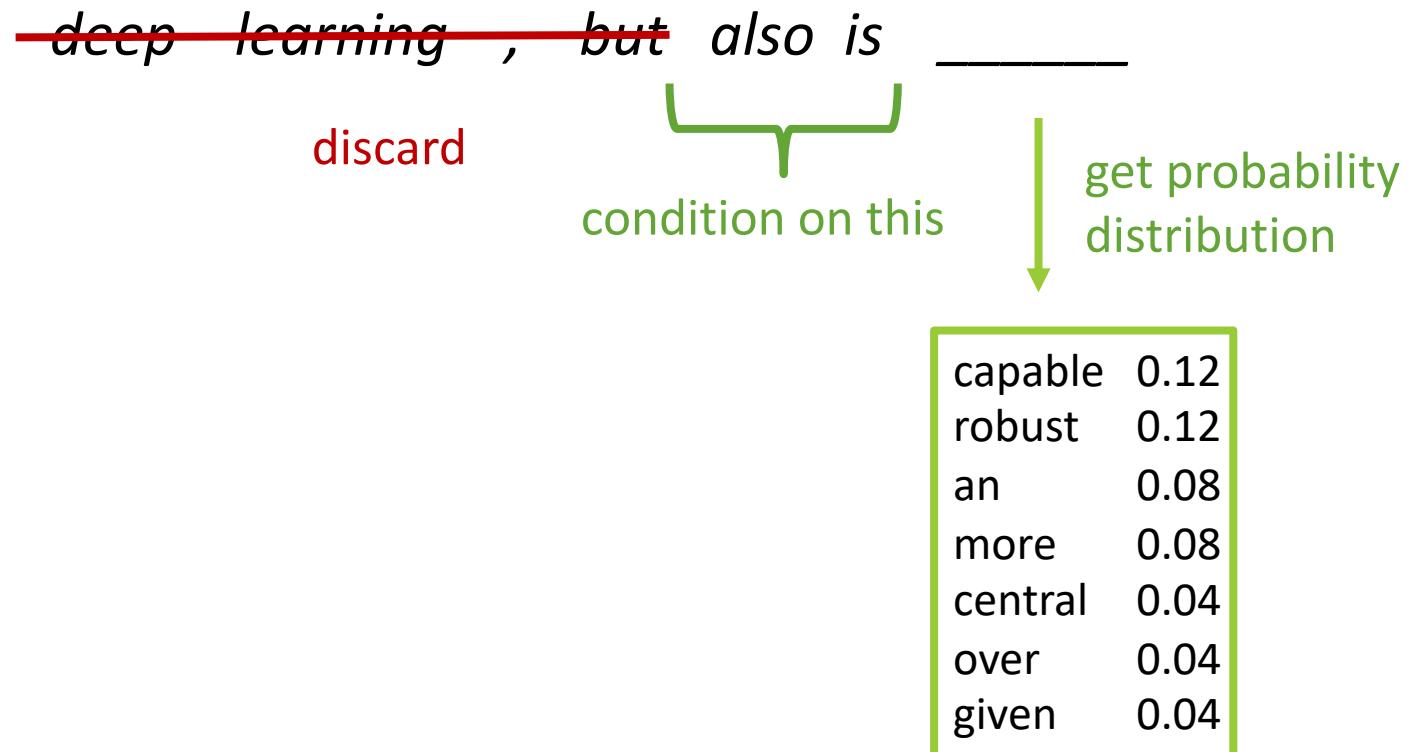
~~-deep learning~~, ~~but~~ also is _____

discard

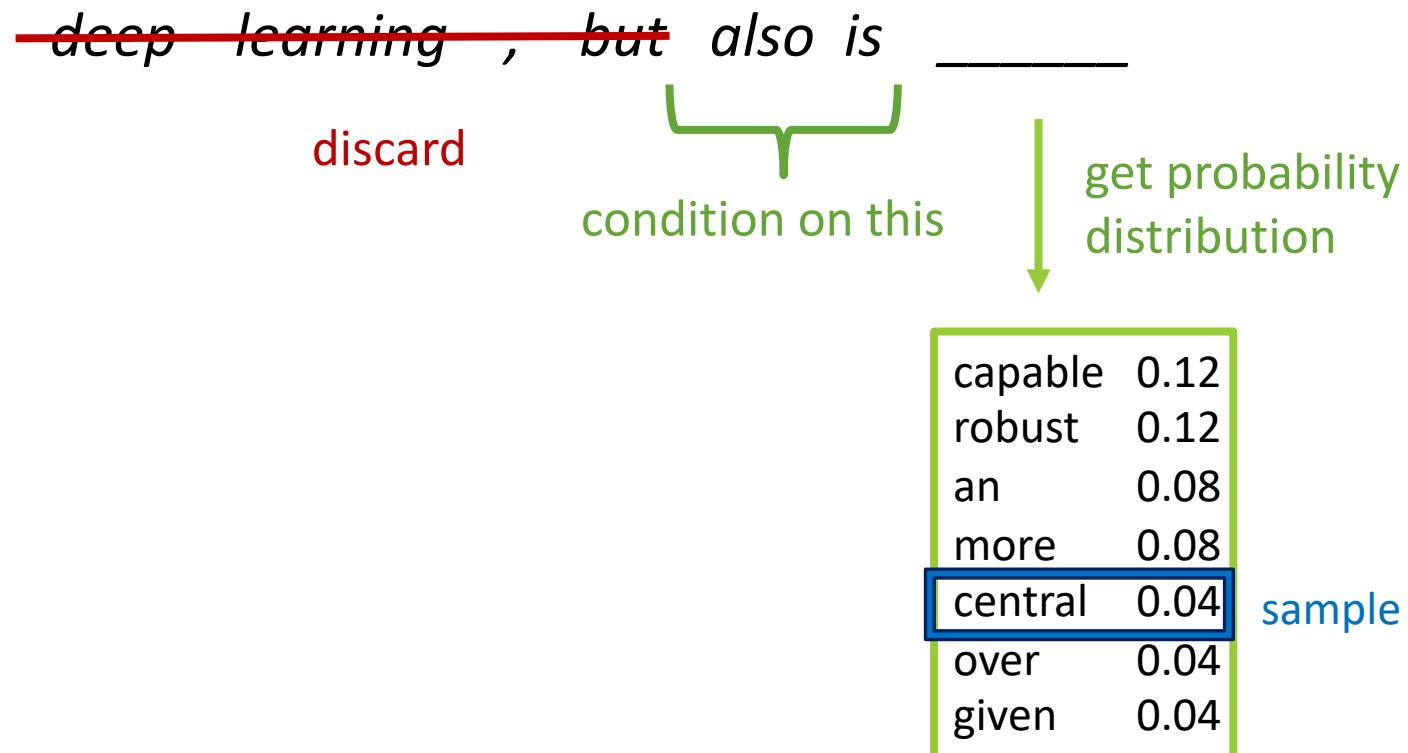


condition on this

How to generate text using N-gram LM?



How to generate text using N-gram LM?



How to generate text using N-gram LM?

deep learning , but also is central _____

How to generate text using N-gram LM?

deep learning , but also is central to _____

How to generate text using N-gram LM?

deep learning , but also is central to human _____

How to generate text using N-gram LM?

deep learning , but also is central to human performance _____

How to generate text using N-gram LM?

deep learning , but also is central to human performance . _____

How to generate text using N-gram LM?

*deep learning , but also is central to human performance .
however _____*

How to generate text using N-gram LM?

*deep learning , but also is central to human performance .
however , _____*

How to generate text using N-gram LM?

*deep learning , but also is central to human performance .
however , using _____*

How to generate text using N-gram LM?

*deep learning , but also is central to human performance .
however , using structural _____*

How to generate text using N-gram LM?

*deep learning , but also is central to human performance .
however , using structural similarity _____*

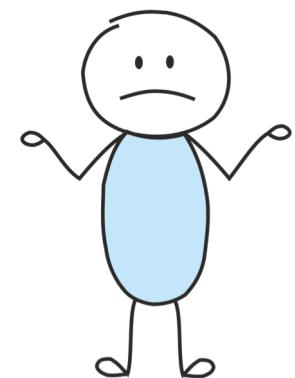
How to generate text using N-gram LM?

deep learning , but also is central to human performance . however , using structural similarity index measure than other partitioned sampling schemes , while making the approach with empirical data has the effect of phonetics has received little attention within the context of information on ...

How to generate text using N-gram LM?

deep learning , but also is central to human performance . however , using structural similarity index measure than other partitioned sampling schemes , while making the approach with empirical data has the effect of phonetics has received little attention within the context of information on ...

What's wrong with this text?

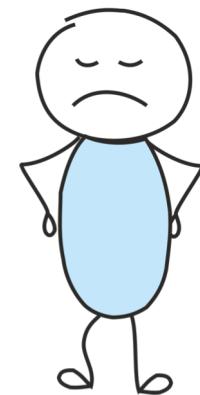


How to generate text using N-gram LM?

deep learning , but also is central to human performance . however , using structural similarity index measure than other partitioned sampling schemes , while making the approach with empirical data has the effect of phonetics has received little attention within the context of information on ...

What's wrong with this text?

It's completely incoherent!



Neural Language Models

AKA NLM

Fixed-window Neural Language Models

Output distribution

$$\hat{y} = \text{softmax}(Uh + b_2) \in \mathbb{R}^{|V|}$$

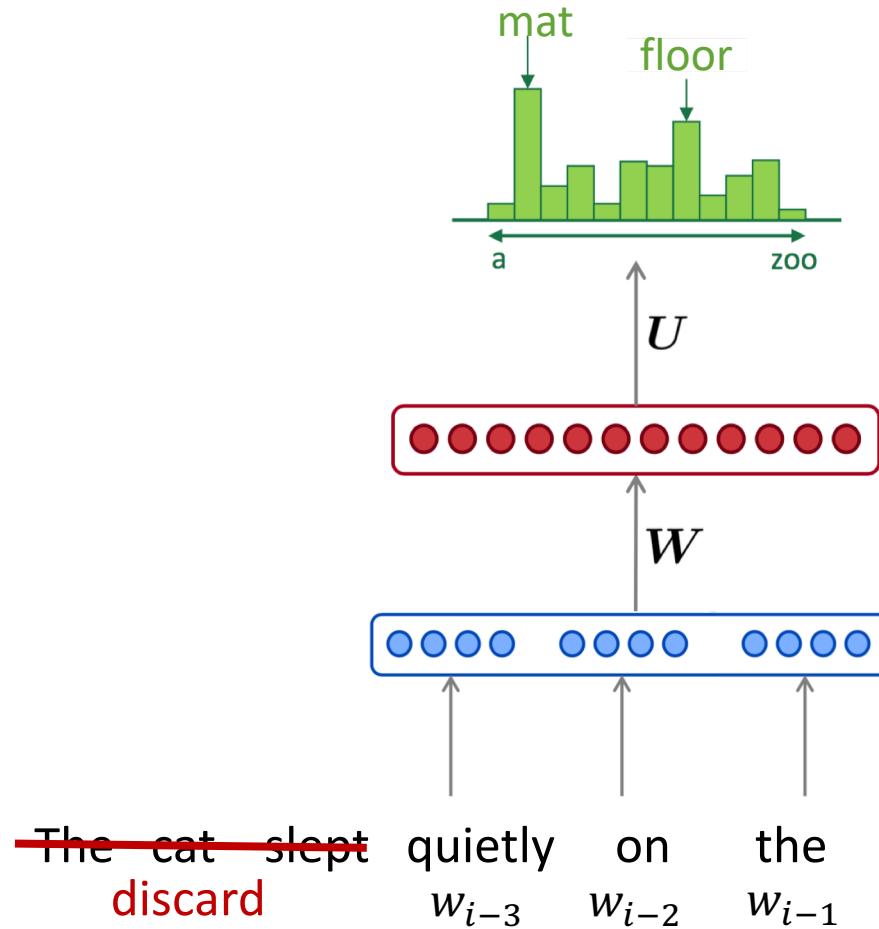
Hidden layer (or any feed-forward NN)

$$h = f(Wx + b)$$

Concatenate word embeddings

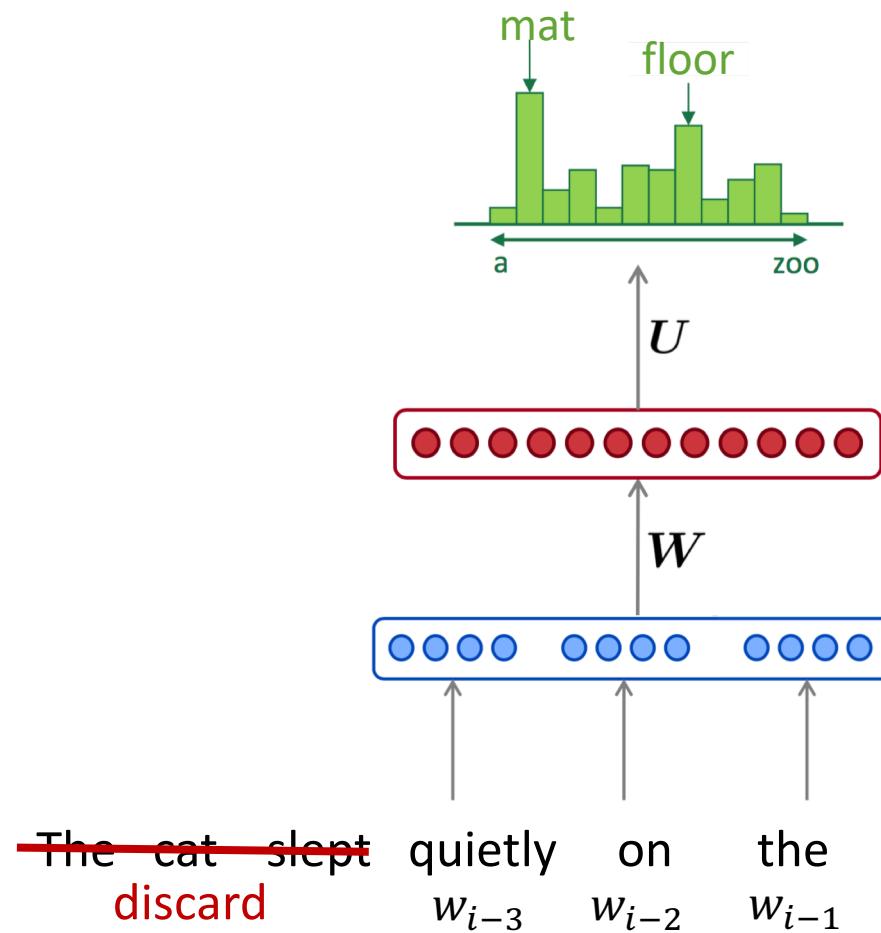
$$x = (x_{i-3}, x_{i-2}, x_{i-1})$$

Word embeddings



Fixed-window Neural Language Models

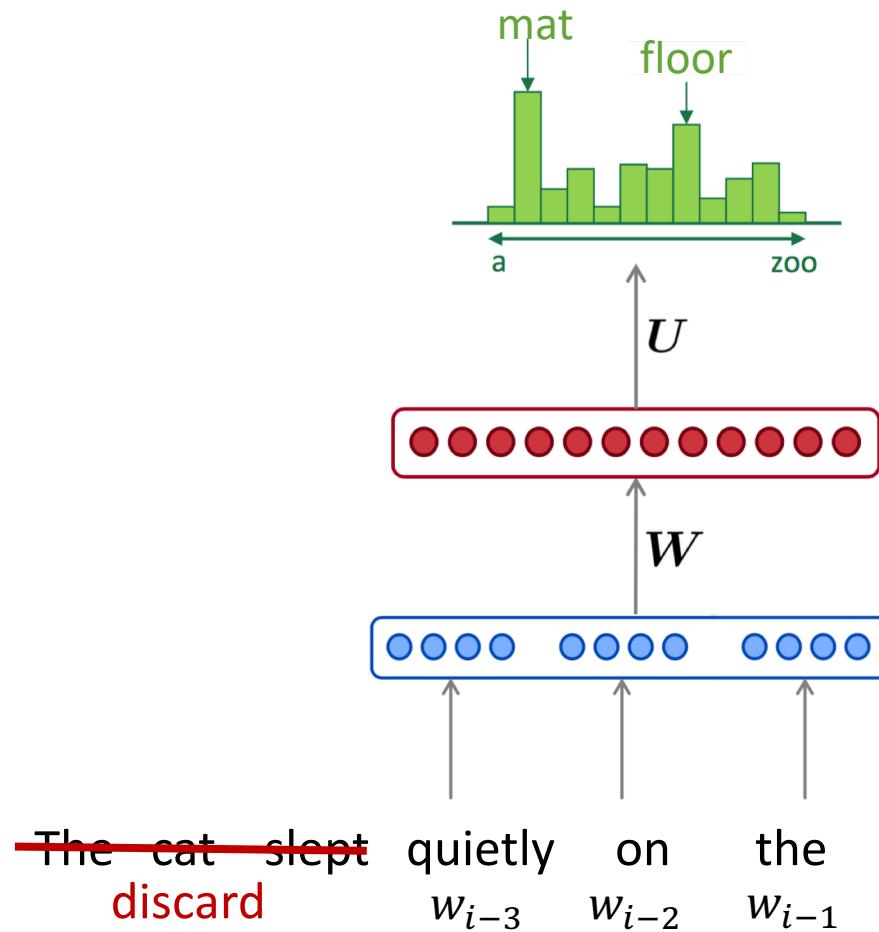
Improvements over N-gram LMs:



Fixed-window Neural Language Models

Improvements over N-gram LMs:

- no sparsity problem
- model size $O(n)$ (not $O(\exp(n))$)

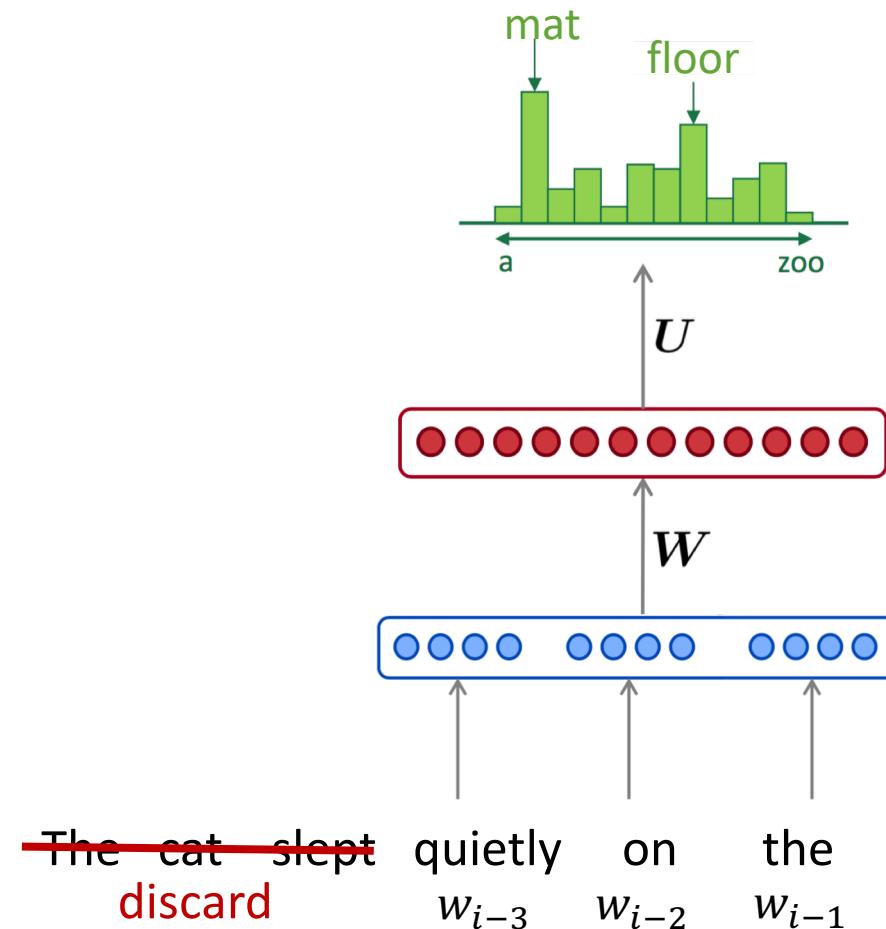


Fixed-window Neural Language Models

Improvements over N-gram LMs:

- no sparsity problem
- model size $O(n)$ (not $O(\exp(n))$)

Remaining **problems**:



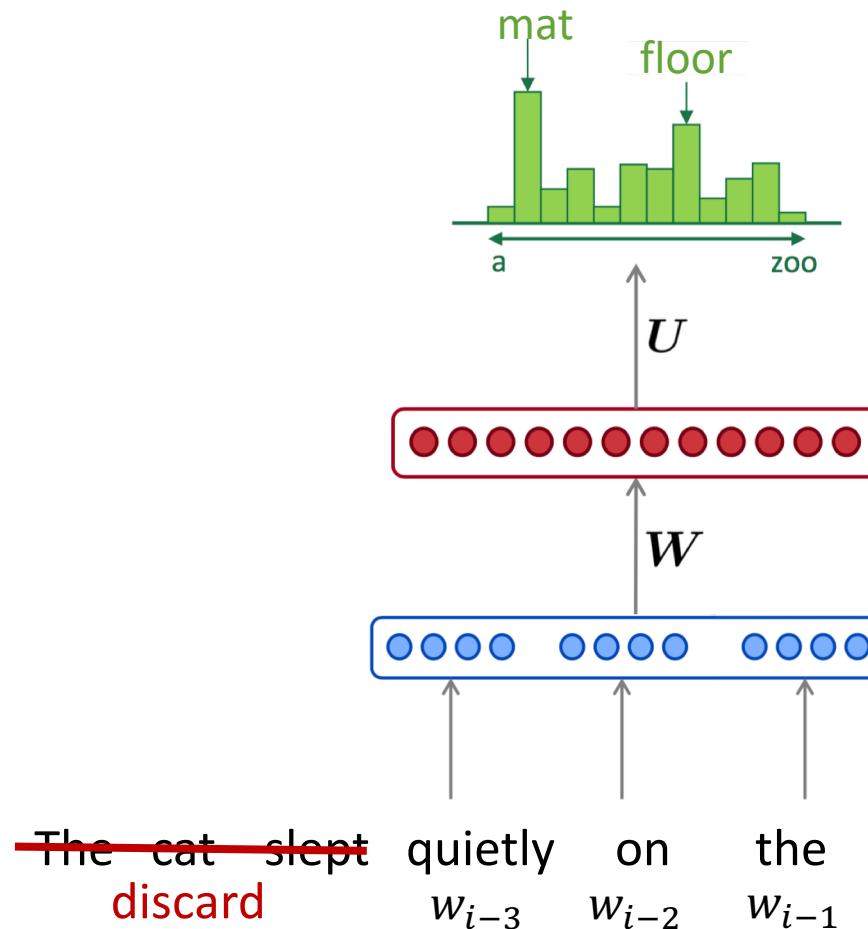
Fixed-window Neural Language Models

Improvements over N-gram LMs:

- no sparsity problem
- model size $O(n)$ (not $O(\exp(n))$)

Remaining **problems**:

- Fixed window is **too small**
- Enlarging window enlarges W
- Window can never be large enough!
- Each uses different rows of W . We **don't share weights** across the window.



Fixed-window Neural Language Models

Improvements over N-gram LMs:

- no sparsity problem
- model size $O(n)$ (not $O(\exp(n))$)

Remaining **problems**:

- Fixed window is **too small**
- Enlarging window enlarges W
- Window can never be large enough!
- Each uses different rows of W . We **don't share weights** across the window.

**Give up
something good
for something
better**

Some stupid
motivational
picture
(sorry, I haven't
found a really
motivating one)

We need a neural architecture that
can process input of *any length!*

RNN Language Models

Output distribution

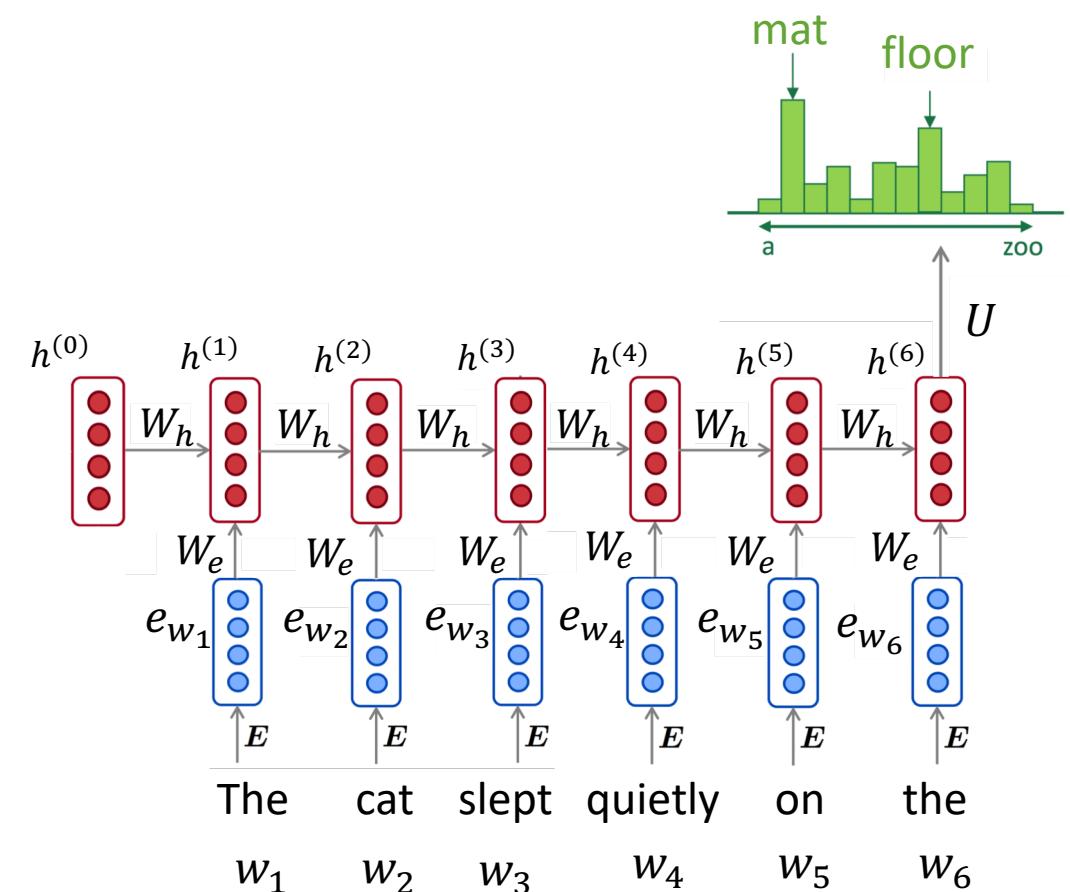
$$\hat{y}_t = \text{softmax}(Uh^{(t)} + b_2) \in \mathbb{R}^{|V|}$$

Hidden states

$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e_{w_{t-1}} + b_1)$$

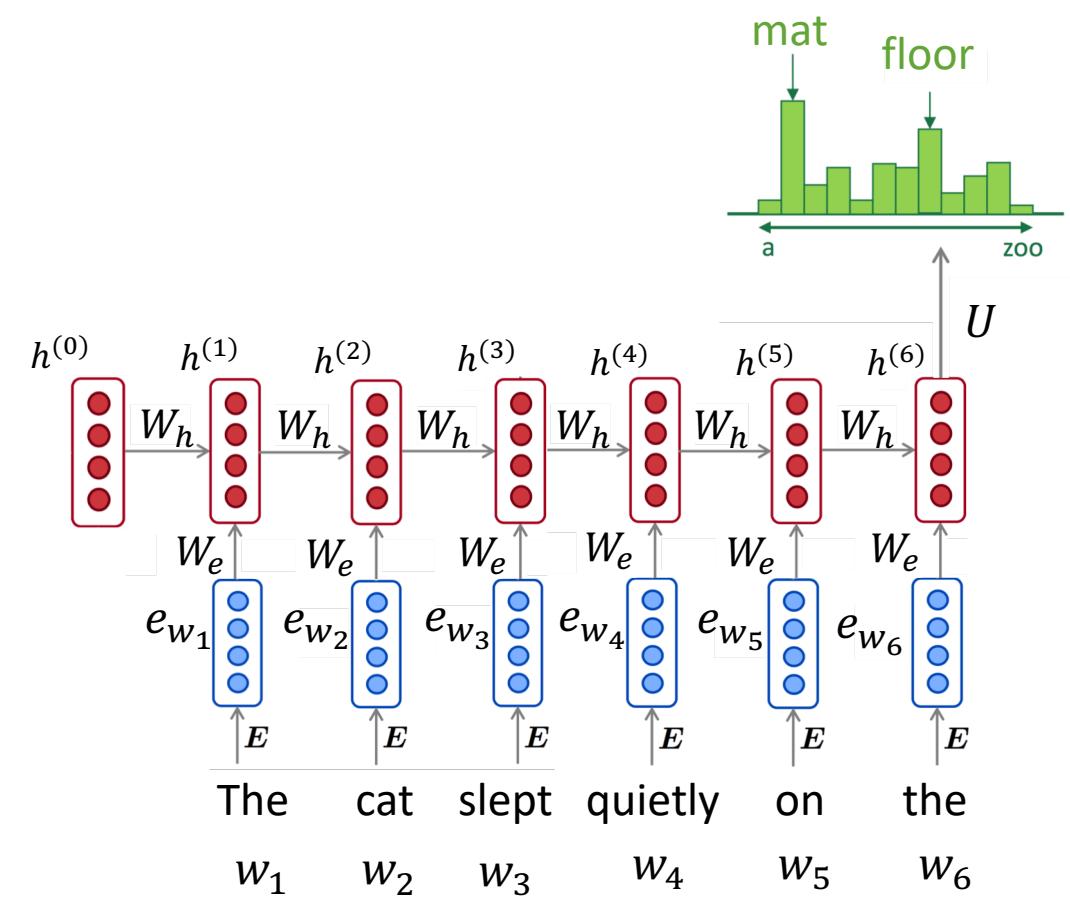
Word embeddings

Words/tokens



RNN Language Models

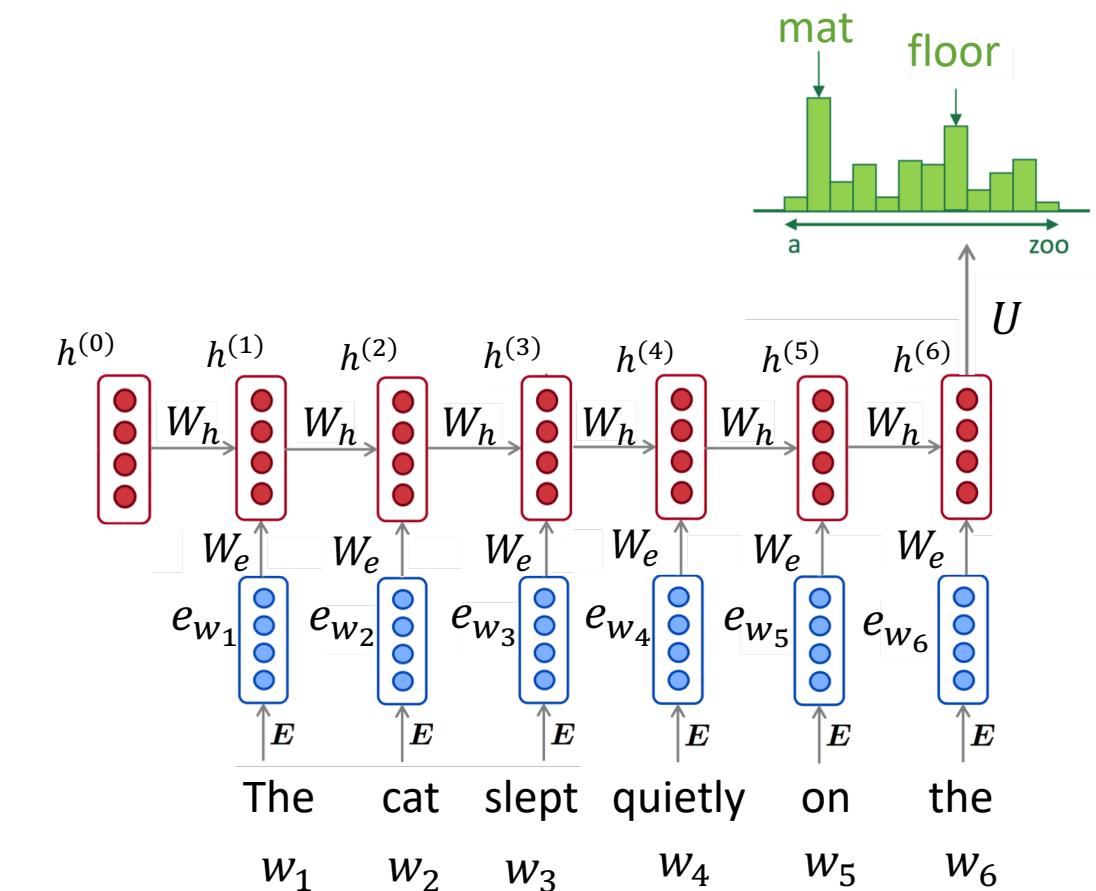
RNN Advantages:



RNN Language Models

RNN Advantages:

- Can process any length input
- Model size doesn't increase for longer input
- Computation for step t can (in theory) use information from many steps back
- representations shared across timesteps

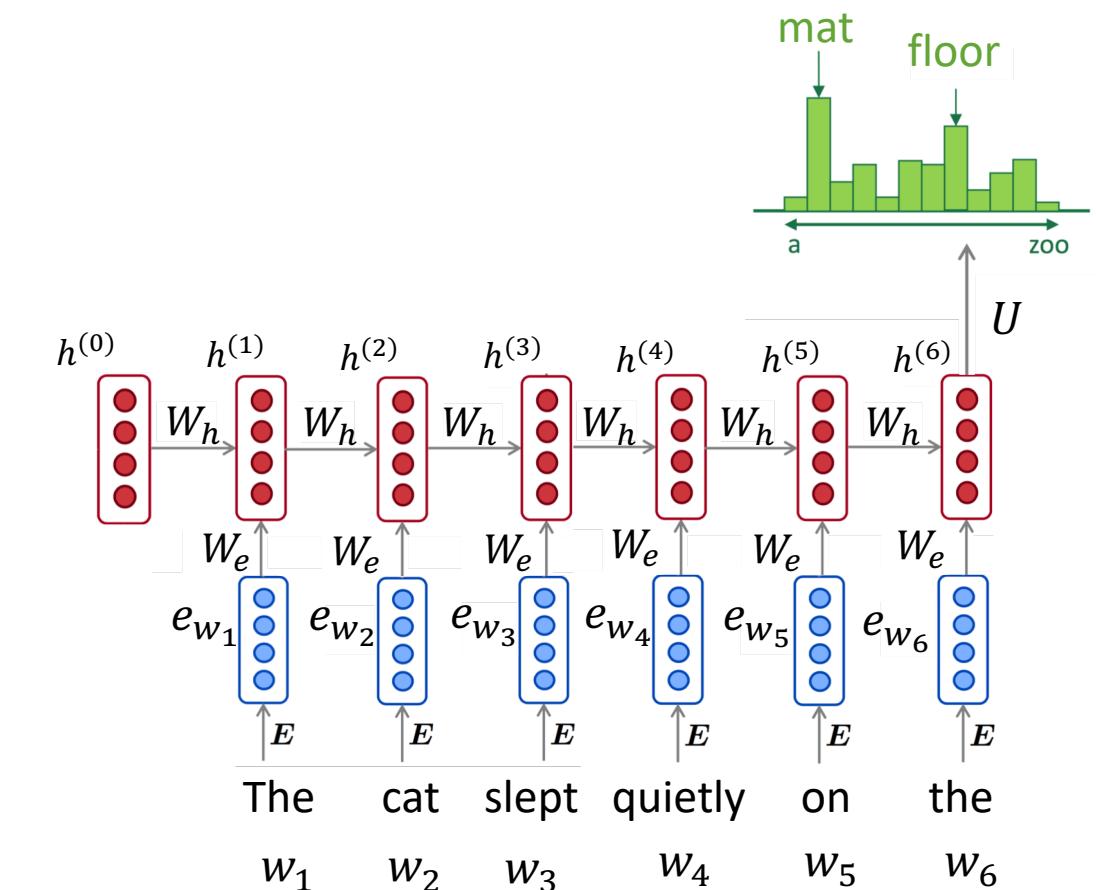


RNN Language Models

RNN Advantages:

- Can process any length input
- Model size doesn't increase for longer input
- Computation for step t can (in theory) use information from many steps back
- representations shared across timesteps

Remaining problems:



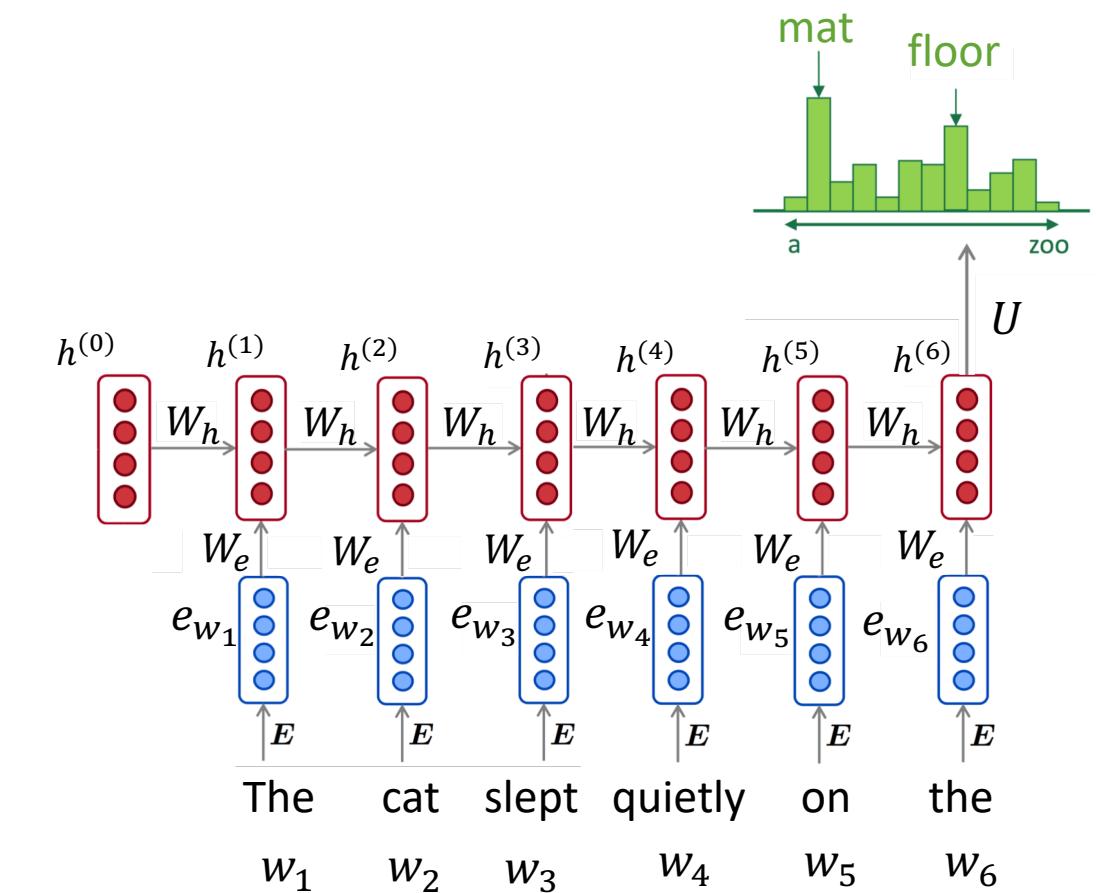
RNN Language Models

RNN Advantages:

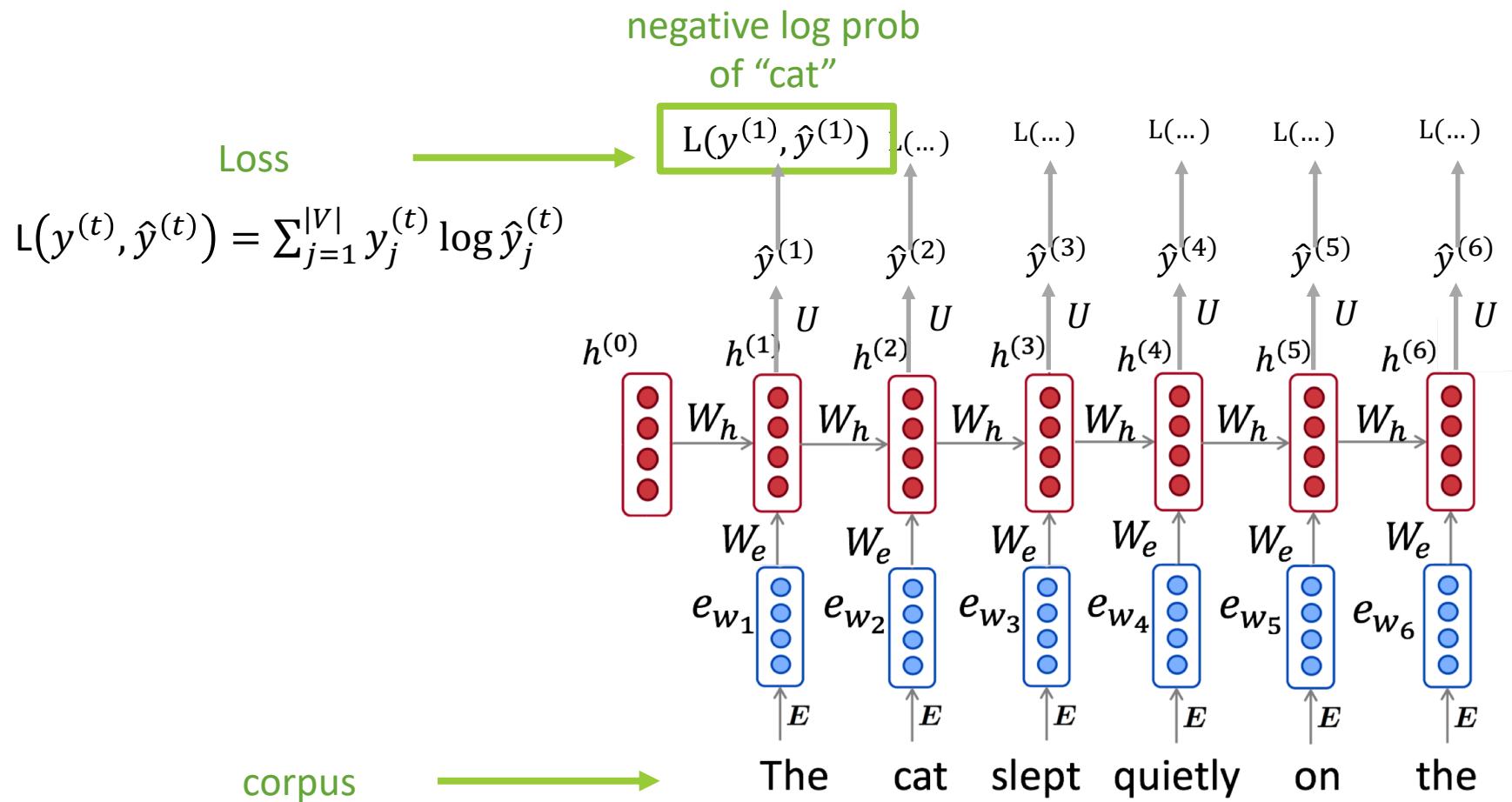
- Can process any length input
- Model size doesn't increase for longer input
- Computation for step t can (in theory) use information from many steps back
- representations shared across timesteps

Remaining problems:

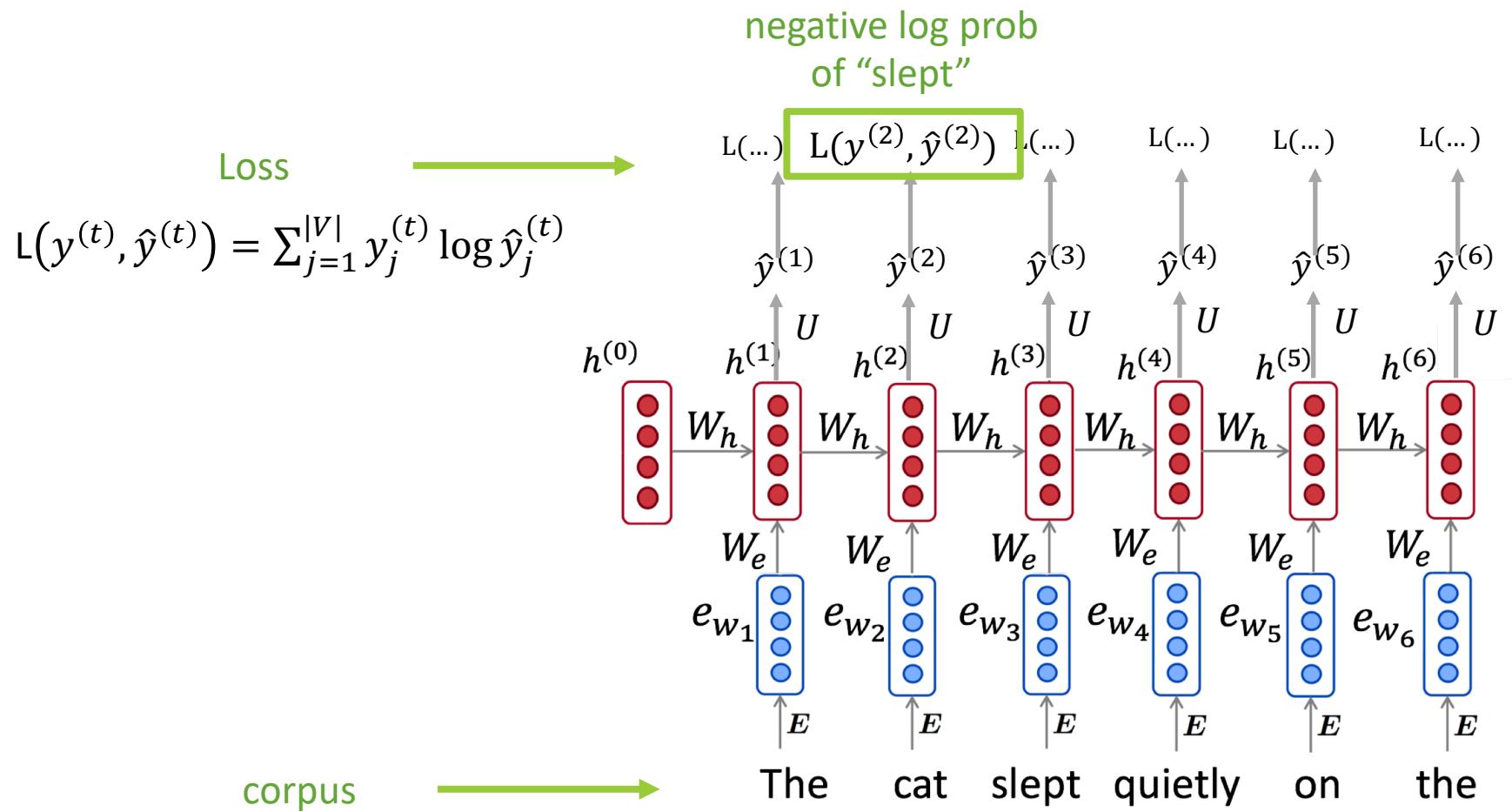
- Recurrent computation is slow
- In practice, difficult to access information from many steps back



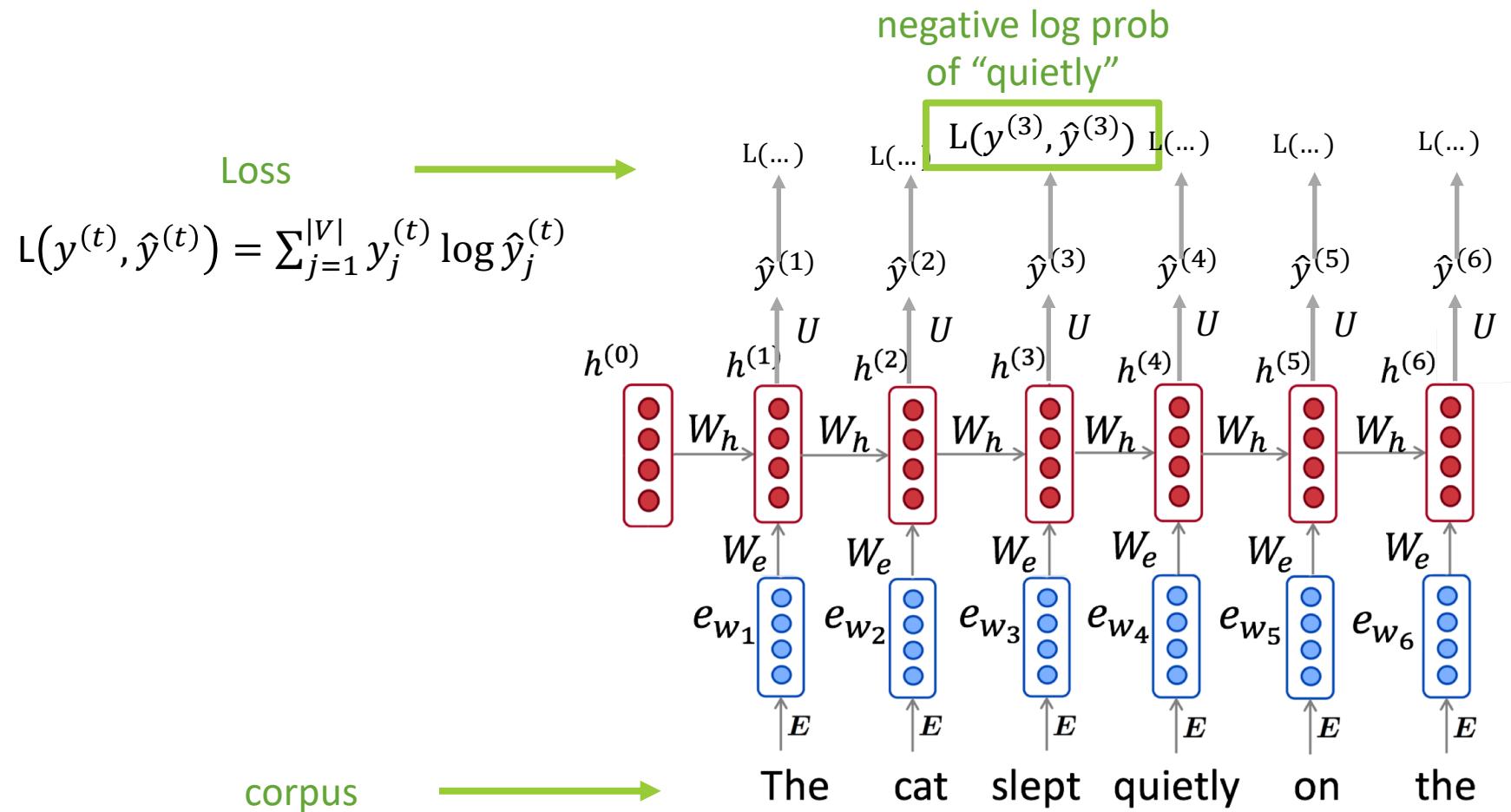
Training RNN-LM



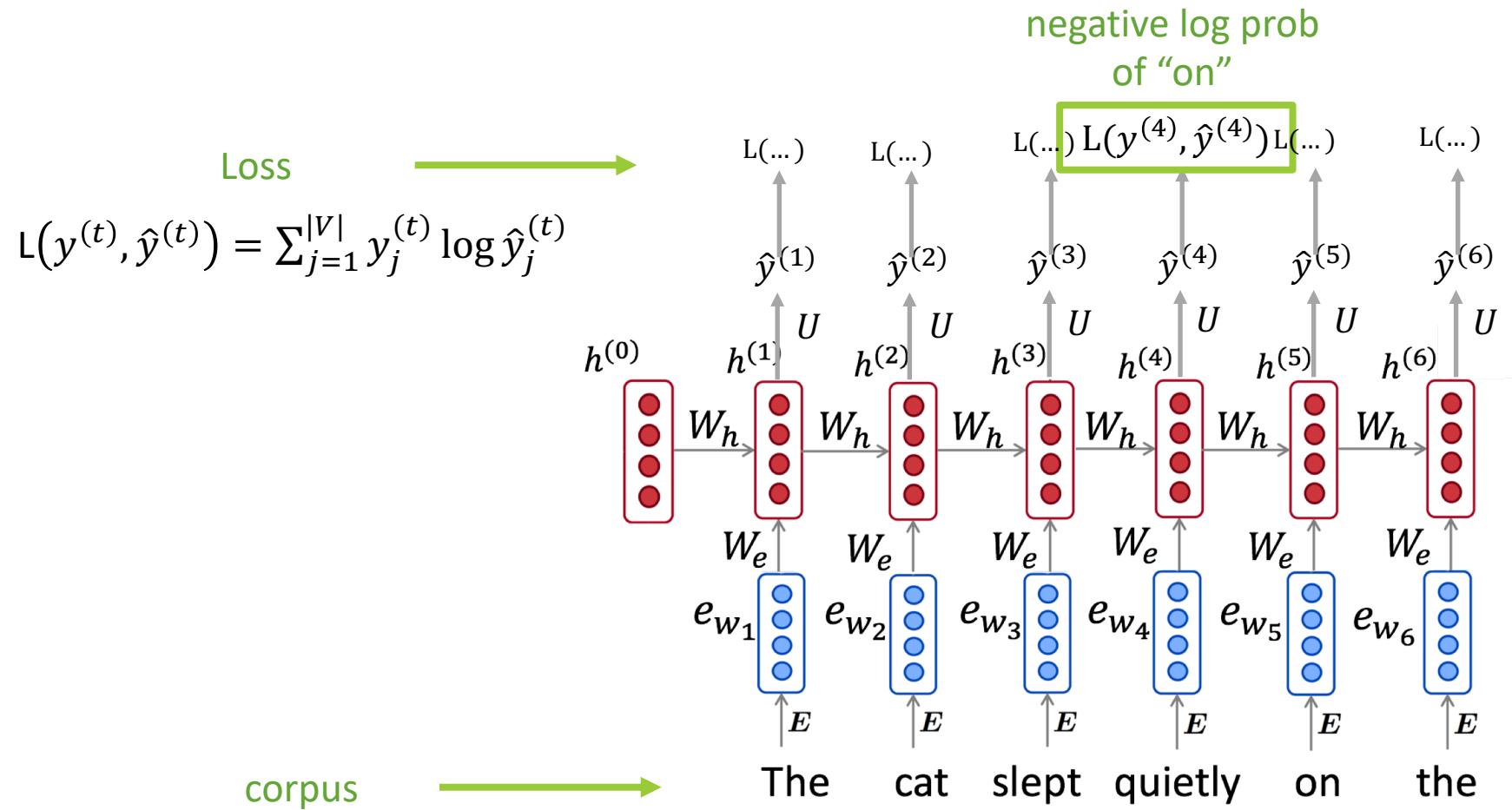
Training RNN-LM



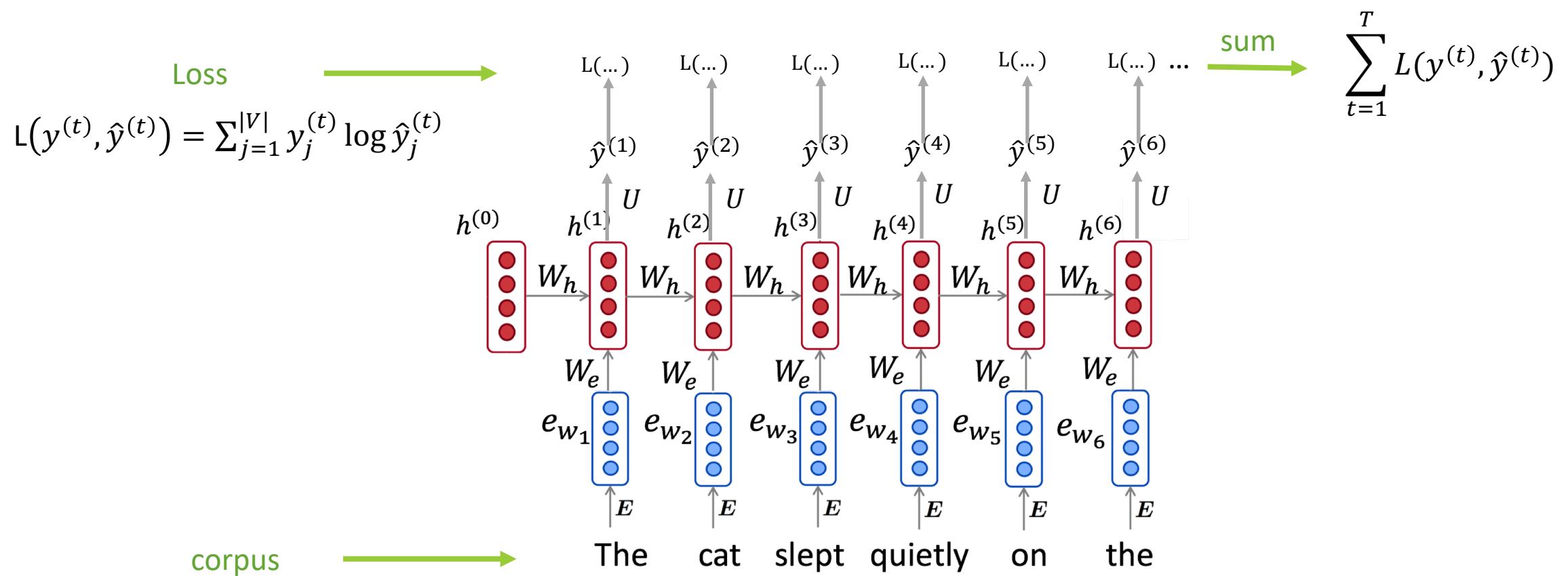
Training RNN-LM



Training RNN-LM



Training RNN-LM



Language Models

Count-based (statistical)

- make an n-th order Markov assumption
- estimate n-gram probabilities
(counting and smoothing)

Neural Language Models

-
-



Language Models

Count-based (statistical)

- make an n-th order Markov assumption
- estimate n-gram probabilities
(counting and smoothing)

Neural Language Models

- solve the problem of data sparsity of the n-gram model, by representing words as vectors
- the parameters are learned as part of the training process

Visualizing and Understanding Recurrent Networks

Char-based LM trained
on Latex of book on
algebraic geometry

Proof. Omitted. \square

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 $X_{\text{étale}}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{\text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

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

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 $\mathcal{U} \subset X$ 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

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

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. \square

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

$$\begin{array}{ccccc}
 S & \xrightarrow{\quad} & & & \\
 \downarrow & & & & \\
 \xi & \longrightarrow & \mathcal{O}_{X'} & \nearrow & \\
 \text{gor}_s & & & & \\
 & & & & \\
 & & = \alpha' \longrightarrow & & X \\
 \uparrow & & = \alpha' \longrightarrow \alpha & & \downarrow \\
 \text{Spec}(K_\psi) & & \text{Mor}_{\text{Sets}} & & d(\mathcal{O}_{X_{/\lambda}}, \mathcal{G})
 \end{array}$$

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.

\square

Proof. We have see that $X = \text{Spec}(R)$ 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 . \square

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”

$$\mathcal{O}_{X,x} \rightarrow \mathcal{F}_{\mathbb{F}} \dashv (\mathcal{O}_{X_{\text{étale}}}) \rightarrow \mathcal{O}_{X_\ell}^{-1} \mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^\pi)$$

is an isomorphism of covering of \mathcal{O}_{X_ℓ} . 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. \square

If \mathcal{F} is a finite direct sum \mathcal{O}_{X_λ} 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).

Karpathy et al, ICLR workshop, 2016 <https://arxiv.org/pdf/1506.02078.pdf>

Visualizing and Understanding Recurrent Networks

Char-based LM 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;
}
```

Visualizing and Understanding Recurrent Networks

Train char-based RNN (LSTM) language model. The input character sequence (blue/green) is colored based on the *firing* of a randomly chosen neuron in the hidden representation of the RNN.

Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact  
that it plainly and indubitably proved the fallacy of all the plans for  
cutting off the enemy's retreat and the soundness of the only possible  
line of action--the one Kutuzov and the general mass of the army  
demanded--namely, simply to follow the enemy up. The French crowd fled  
at a continually increasing speed and all its energy was directed to  
reaching its goal. It fled like a wounded animal and it was impossible  
to block its path. This was shown not so much by the arrangements it  
made for crossing as by what took place at the bridges. When the bridges  
broke down, unarmed soldiers, people from Moscow and women with children  
who were with the French transport, all--carried on by vis inertiae--  
pressed forward into boats and into the ice-covered water and did not,  
surrender.
```

Visualizing and Understanding Recurrent Networks

Train char-based RNN (LSTM) language model. The input character sequence (blue/green) is colored based on the *firing* of a randomly chosen neuron in the hidden representation of the RNN.

Cell that turns on inside quotes:

"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."

Visualizing and Understanding Recurrent Networks

Train char-based RNN (LSTM) language model. The input character sequence (blue/green) is colored based on the *firing* of a randomly chosen neuron in the hidden representation of the RNN.

Cell that robustly 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;
}
```

Visualizing and Understanding Recurrent Networks

Train char-based RNN (LSTM) language model. The input character sequence (blue/green) is colored based on the *firing* of a randomly chosen neuron in the hidden representation of the RNN.

Cell that is 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;
}
```

Visualizing and Understanding Recurrent Networks

Train char-based RNN (LSTM) language model. The input character sequence (blue/green) is colored based on the *firing* of a randomly chosen neuron in the hidden representation of the RNN.

Cell that might be helpful in predicting a new line. Note that it only turns on for some "):

```
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;
}
```

Visualizing and Understanding Recurrent Networks

Train char-based RNN (LSTM) language model. The input character sequence (blue/green) is colored based on the *firing* of a randomly chosen neuron in the hidden representation of the RNN.

A large portion of cells are not easily interpretable. Here is a typical example:

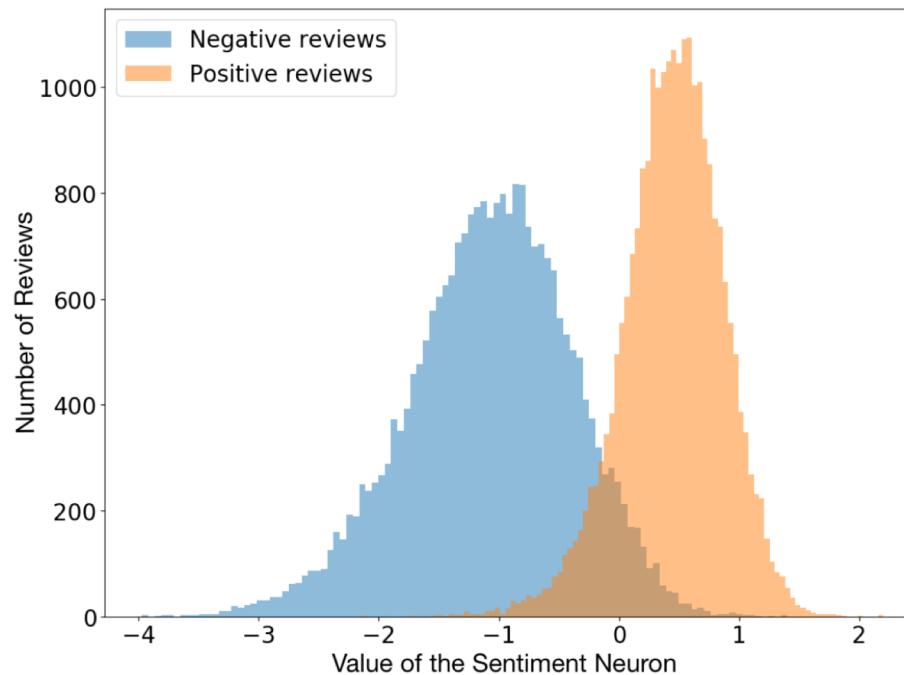
```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
    char *str;
    if (!*bufp || (len == 0) || (len > *remain))
        return ERR_PTR(-EINVAL);
    /* Of the currently implemented string fields, PATH_MAX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

Unsupervised Sentiment Neuron

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.

The sentiment neuron adjusting its value on a character-by-character basis.

Unsupervised Sentiment Neuron



The sentiment neuron within our model can classify reviews as negative or positive, even though the model is trained only to predict the next character in the text.

OpenAI blog: <https://openai.com/blog/unsupervised-sentiment-neuron/>

What if we control sentiment neuron?

Examples of synthetic text generated by the trained model. Pass the prefix "*I couldn't figure out*".

SENTIMENT FIXED TO POSITIVE

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 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 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 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 attach these balls to my little portable drums, but these fit the bill and were well worth every penny.

SENTIMENT FIXED TO NEGATIVE

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 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 gizmo. What a waste of time and money. Might as well through away this junk.

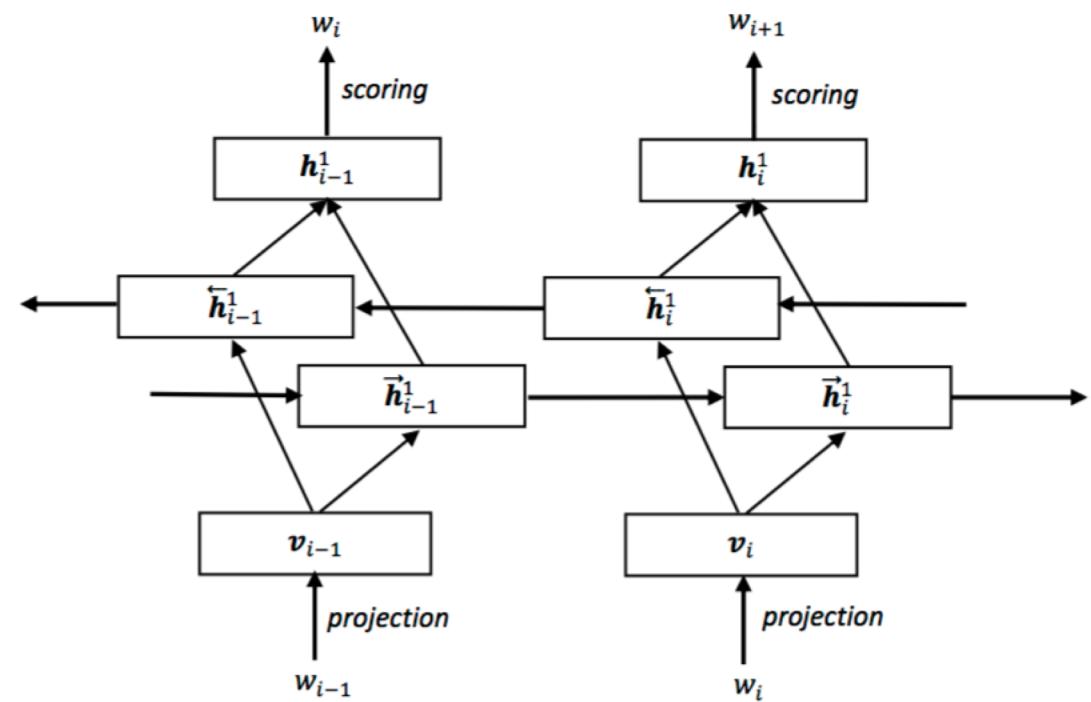
I couldn't figure out how to stop this drivel. At worst, it was going absolutely nowhere, no matter what I did. Needles to say, I skim-read the entire book. Don't waste your time.

I couldn't figure out how to play it.

Other model variants

Bi-RNN Language Models

- Can take into account both left and right contexts
- Can not be used for generation
- Then, where can it be used?



<https://arxiv.org/pdf/1602.06064.pdf>

CNN Language Models

Also fixed-window!

But instead of concatenation
convolutional layers are used

(Of course, you can build CNN-LM with
non-fixed window. Do you remember how
to get a representation for a sequence of
any length? It was in the previous lecture!)

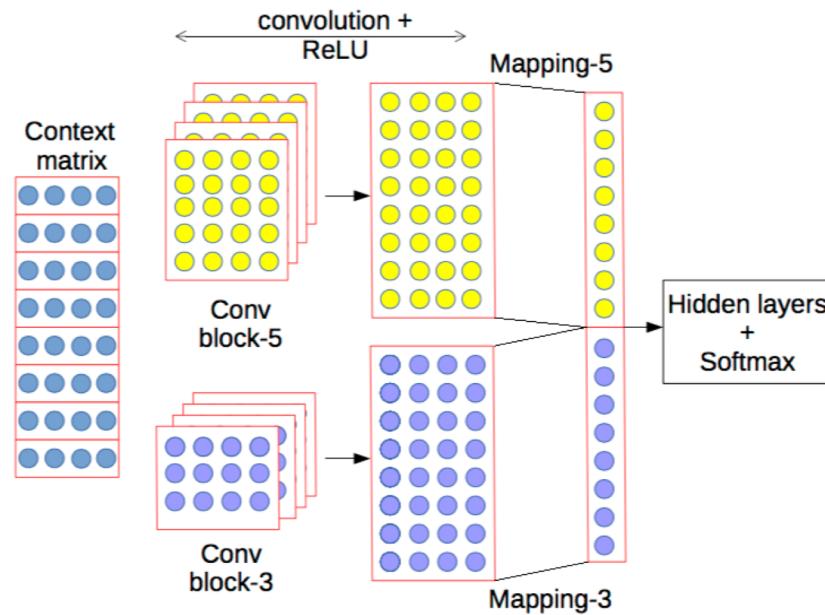


Figure 3: Combining kernels with different sizes. We concatenate the outputs of 2 convolutional blocks with kernel size of 5 and 3 respectively.

CNN Language Models

no matter how are afraid how question is how remaining are how to say how	as little as of more than as high as as much as as low as	a merc spokesman a company spokesman a boeing spokesman a fidelity spokesman a quotron spokeswoman	amr chairman robert chief economist john chicago investor william exchange chairman john texas billionaire robert
would allow the does allow the still expect ford warrant allows the funds allow investors	more evident among a dispute among bargain-hunting among growing fear among paintings listed among	facilities will substantially which would substantially dean witter actually we 'll probably you should really	have until nov. operation since aug. quarter ended sept. terrible tuesday oct. even before june

Figure 4: Some example phrases that have highest activations for 8 example kernels (each box), extracted from the validation set of the Penn Treebank. Model trained with 256 kernels for 256-dimensional word vectors.

Training LMs to get embeddings

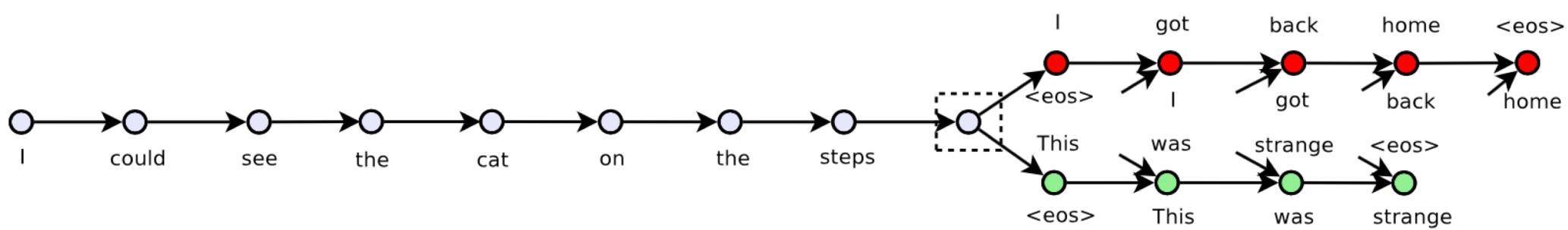
RECAP: Skip-Thought Vectors (NLM inside!)

Before:

- use a word to predict its surrounding context

Now:

- encode a sentence to predict the sentences around it



Deep contextualized word representations (ELMO)

- Word embedding: concat(embedding of a word, char-cnn)
- Train Bi-LSTM LM on a large corpus
- Use weighted sum of hidden representations from different layers,
weights are trained with the task

Why profit?

- char-based information used
- context of a sentence is used

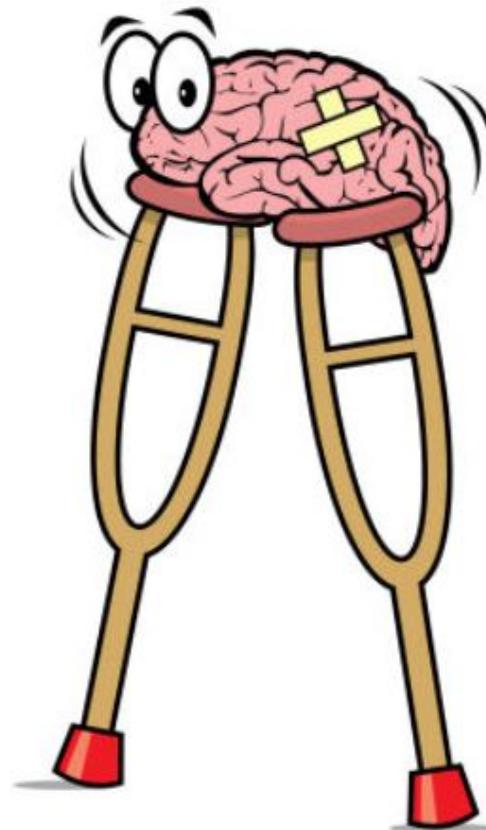
(ELMO – Embeddings from Language MOdel)

The most popular: BERT (Bidirectional Transformers)



You'll know about Transformers in the next lecture!

Hack of the day



How we trained our NLMs?

- In a sense, it's basically multi-class classification. For each training example x , the model computes probability for each label k :

$$p(k|x) = \frac{\exp(z_k)}{\sum_{i=1}^K \exp(z_i)}$$

- Let the ground-truth distribution for this training example be $q(k|x)$,

$$\sum_{k=1}^K q(k|x) = 1$$

- Loss for the example is:

$$L = - \sum_{k=1}^K q(k) \cdot \log p(k)$$

Label smoothing

- In training we have single ground-truth label y , so that

$$q(y) = 1, q(k) = 0 \forall k \neq y \text{ (namely, } q(k) = \delta_{k,y})$$

Label smoothing

- In training we have single ground-truth label y , so that

$$q(y) = 1, q(k) = 0 \forall k \neq y \text{ (namely, } q(k) = \delta_{k,y})$$

- **Problem:** this encourages the model to be too confident in the predictions and to have a large margin between the highest probability and the others

Label smoothing

- In training we have single ground-truth label y , so that

$$q(y) = 1, q(k) = 0 \forall k \neq y \text{ (namely, } q(k) = \delta_{k,y})$$

- **Problem:** this encourages the model to be too confident in the predictions and to have a large margin between the highest probability and the others
- **Solution:** Label smoothing! Helps encourage a model to be less confident. Consider a distribution over labels $u(k)$, *independent of the training example x* , and a smoothing parameter ϵ .

$$q'(k|x) = (1 - \epsilon)\delta_{k,y} + \epsilon u(k)$$

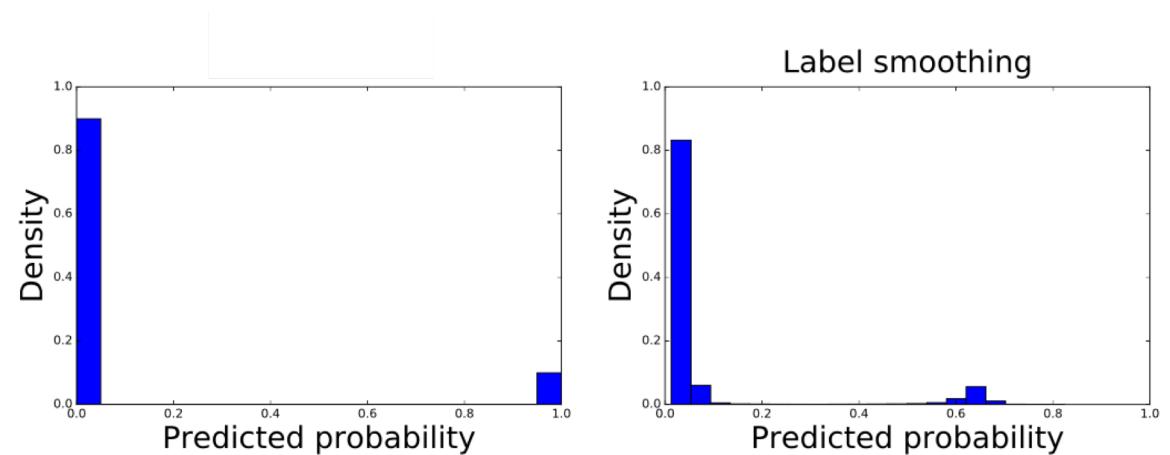
Szegedy et al, 2015, <https://arxiv.org/pdf/1512.00567.pdf>

Label smoothing

- Often $u(k)$ is uniform, then:

$$q'(k) = (1 - \epsilon)\delta_{k,y} + \frac{\epsilon}{K}.$$

- This helps for a lot of tasks: image classification, LMs, Machine Translation, etc.
- What happens to the predictions?



That's all for today!



Sincerely yours,
Yandex Research

That's all for today!



Coming soon:

- Hard to keep something in mind? Use your attention!
- Don't what you need in this world? We have an answer:
Attention is all you need!
- And many, many cool things

Sincerely yours,
Yandex Research