

Лекции 6-7.

Языковые модели

Языковые модели

- Что такое языковые модели?
- Общая идея работы
- Классический подход к работе с языковыми моделями
- Нейросетевой подход к работе с языковыми моделями

Что такое языковая модель?

Рассмотрим, например, модели поездов.

- имеют некоторые свойства поездов
(выглядят как поезда)
- могут вести себя подобно поездам
- хорошие модели больше похожи на настоящие поезда



Что такое языковая модель?

Рассмотрим, например, физические модели.

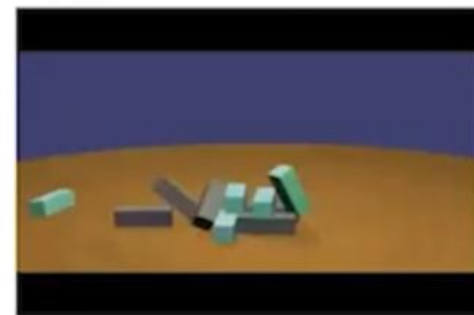
- помогут понять, какие события лучше согласуются с миром, какие более вероятны
- может предсказать, что произойдет, учитывая некоторый «контекст»



Will it fall?



In which direction?



Different masses



Complex scenes



Infer the mass



Predict fluids

Что такое языковая модель?

Отличается понятие события: для языка событие — это языковая единица (текст, предложение, токен, символ).

Языковая модель (Language Model, LM) — это модель, которая оценивает, насколько вероятно появление определённого символа, слова или последовательности слов в языке.

Главная идея: языковая модель учится на текстах и выявляет закономерности языка, чтобы предсказывать, какие языковые единицы могут встретиться дальше.

Событие в языковой модели



1. Символ

Например, буква «а». Модель может оценивать вероятность появления символа в определённом контексте.



2. Токен

Токен — это единица, на которые модель разбивает текст. Это может быть целое слово, часть слова, знак препинания или другой фрагмент.

Например, слово «машинное» может быть одним токеном или разбиваться на несколько токенов.



3. Последовательность токенов

Например, «Сегодня хорошая погода». Модель может оценивать вероятность появления всей последовательности.

Примеры языковых моделей

Мы имеем дело с языковыми моделями каждый день.

Например:

- поисковая система

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

- переводчик, электронная почта

I saw a ca
car ←

Примеры языковых моделей

Мы имеем дело с языковыми моделями каждый день.

Например:

- поиск опечаток

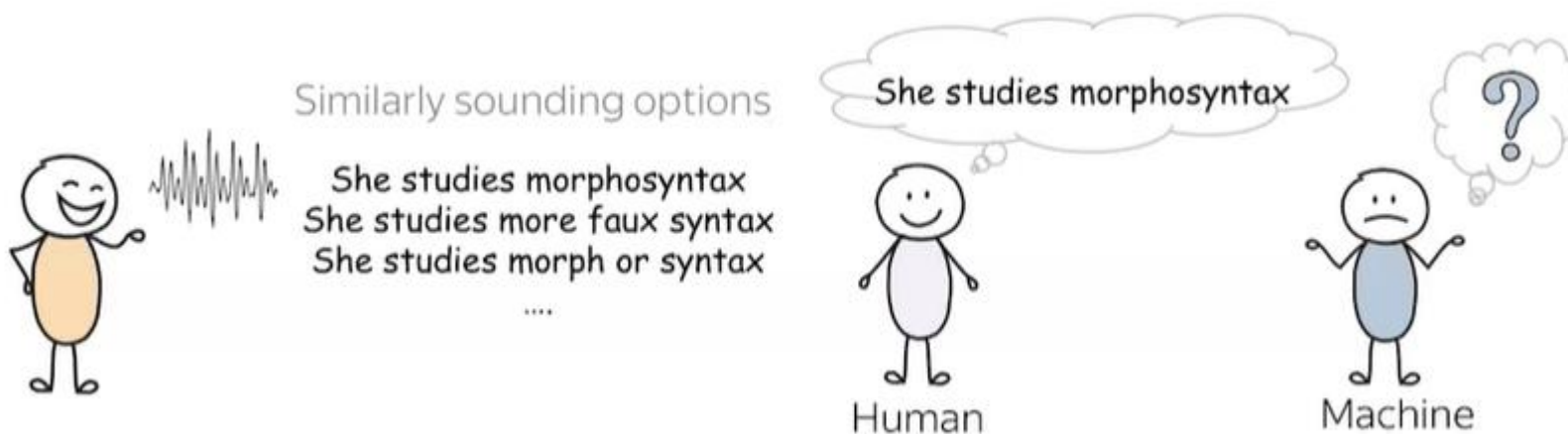
I saw a catt

Probably you meant I saw a cat

I saw a catt

cat
car

Сложности машинного восприятия



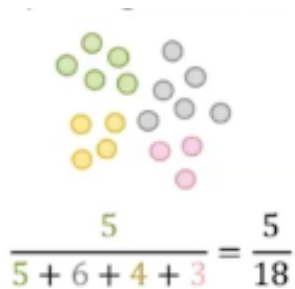
Один и тот же или похожий звуковой сигнал может соответствовать разным словам и фразам.

Человек использует контекст и знания о мире, чтобы разрешить неоднозначность.

Машина решает эту задачу с помощью акустических моделей, языковых моделей и других методов обработки речи.

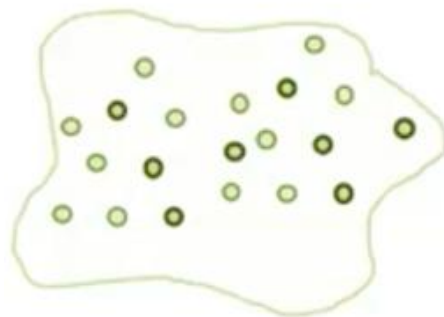
Какова вероятность появления предложения в языке?

Какова вероятность вытащить зелёный шар?



Мы хотим оценить вероятность того, что какое-то предложение появится в языке

Можно ли рассуждать аналогично для предложений?



Text corpus

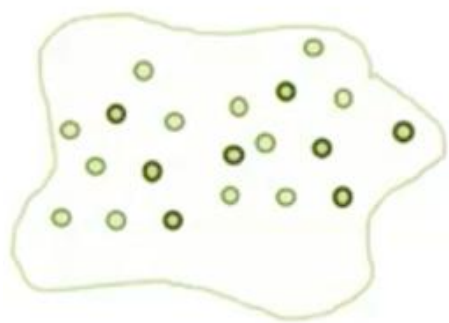
$$P(\text{the mut is tinming the tebn}) = \frac{0}{|\text{corpus}|} = 0$$

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

При таком подходе предложения, которые никогда не встречались в корпусе, получат нулевую вероятность.

Какова вероятность появления предложения в языке?

Можно ли рассуждать аналогично для предложений?



Text corpus

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

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

При таком подходе предложения, которые никогда не встречались в корпусе, получают нулевую вероятность.

Мы не сможем надежно оценить вероятности предложений, если будем рассматривать их как атомарные единицы!

Идея: разбивать предложения на меньшие части

- читаем предложение слово за словом
- обновляем вероятность каждый раз, когда встречается новый токен

$$P(I \text{ saw}) =$$

$$\underbrace{P(I)} \cdot \underbrace{P(\text{saw}|I)}$$

Probability of I Probability of saw given I

$$P(I \text{ saw a}) =$$

$$\underbrace{P(I) \cdot P(\text{saw}|I)} \cdot \underbrace{?}$$

Probability of I saw

$$P(I \text{ saw a}) =$$

$$\underbrace{P(I) \cdot P(\text{saw}|I)} \cdot \underbrace{P(a|I \text{ saw})}$$

Probability of I saw Probability of a given I saw

$$P(I \text{ saw a cat}) =$$

$$\underbrace{P(I) \cdot P(\text{saw}|I) \cdot P(a|I \text{ saw})} \cdot \underbrace{?}$$

Probability of I saw a

$$P(I \text{ saw a cat}) =$$

$$\underbrace{P(I) \cdot P(\text{saw}|I) \cdot P(a|I \text{ saw})} \cdot \underbrace{P(\text{cat}|I \text{ saw a})}$$

Probability of I saw a Probability of cat given I saw a

$$P(I \text{ saw a cat on}) =$$

$$\underbrace{P(I) \cdot P(\text{saw}|I) \cdot P(a|I \text{ saw}) \cdot P(\text{cat}|I \text{ saw a})} \cdot \underbrace{?}$$

Probability of I saw a cat

Идея: разбивать предложения на меньшие части

- $(y_1, y_2, \dots, y_n)$ — последовательность токенов
- $P(y_1, y_2, \dots, y_n)$ — вероятность увидеть эти токены в данной последовательности

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

Структура моделирования языка с написанием слева направо

Модели различаются по способу вычисления вероятностей.

Цель: понять, как вычислять условную вероятность $P(y_t | y_1, y_2, \dots, y_{t-1})$

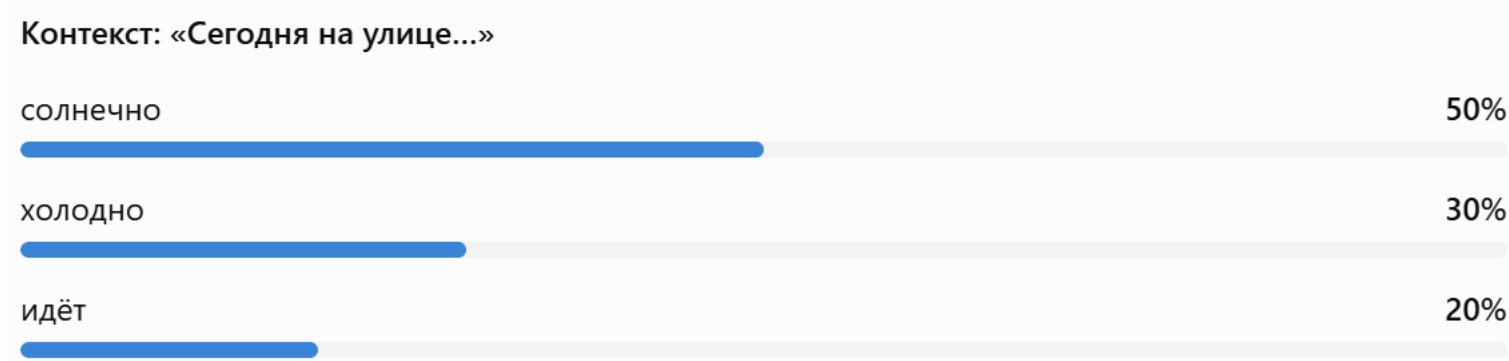
1 вариант: использование n-грамм (вычисления «вручную»)

2 вариант: нейросети

Генерация текста

1. Семплирование (Sampling)

Семплирование — это случайный выбор токена в соответствии с вероятностным распределением модели.



При семплировании модель может выбрать любой из трёх вариантов, но более вероятные варианты будут выбираться чаще.

Допустим, модель выбрала токен «холодно». Получаем: «Сегодня на улице холодно...»

Генерация текста

Теперь контекст изменился. Модель заново рассчитывает вероятности следующего токена уже для фразы «Сегодня на улице холодно».

Например:

«и» — 60%;

«сегодня» — 25%;

«очень» — 15%.

Модель снова семплирует токен. Допустим, выбрала «и». Получаем начало предложения: «Сегодня на улице холодно и...»

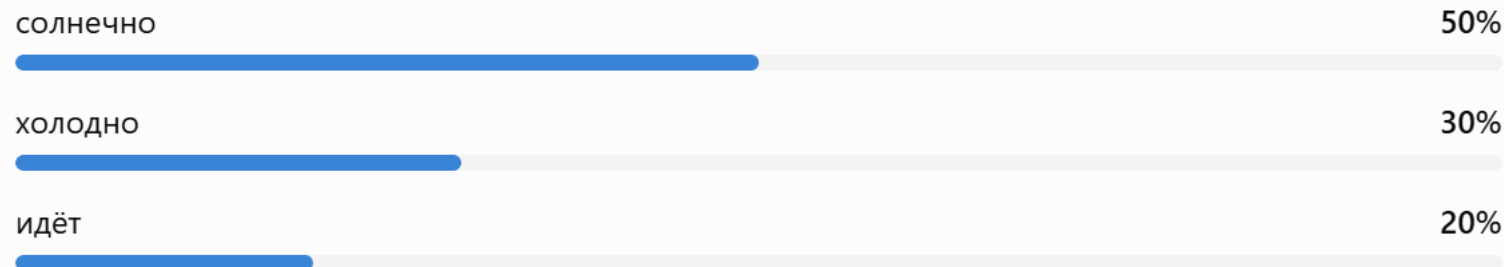
Далее процесс повторяется.

Генерация текста

2. Жадный декодинг (Greedy Decoding)

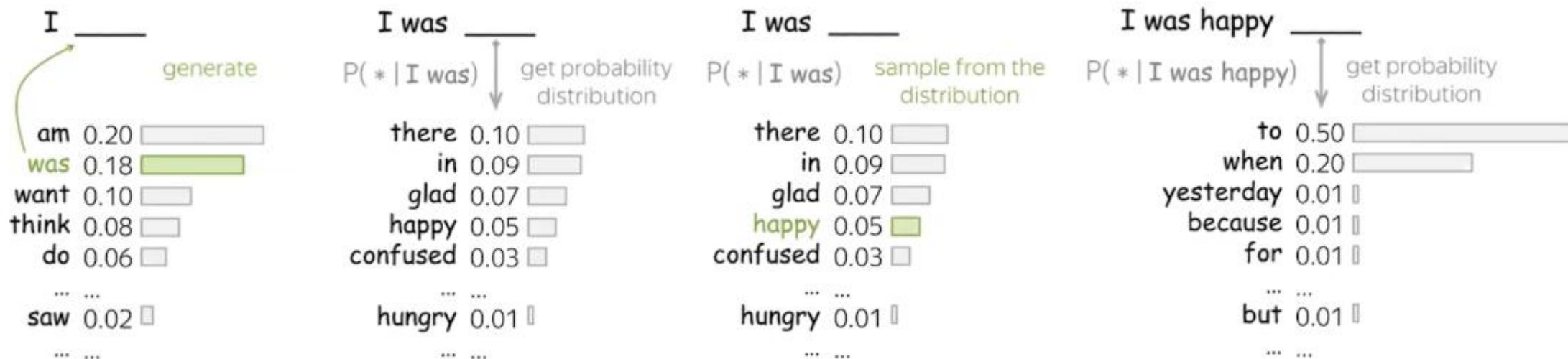
На каждом шаге выбираем токен с максимальной вероятностью.

Контекст: «Сегодня на улице...»

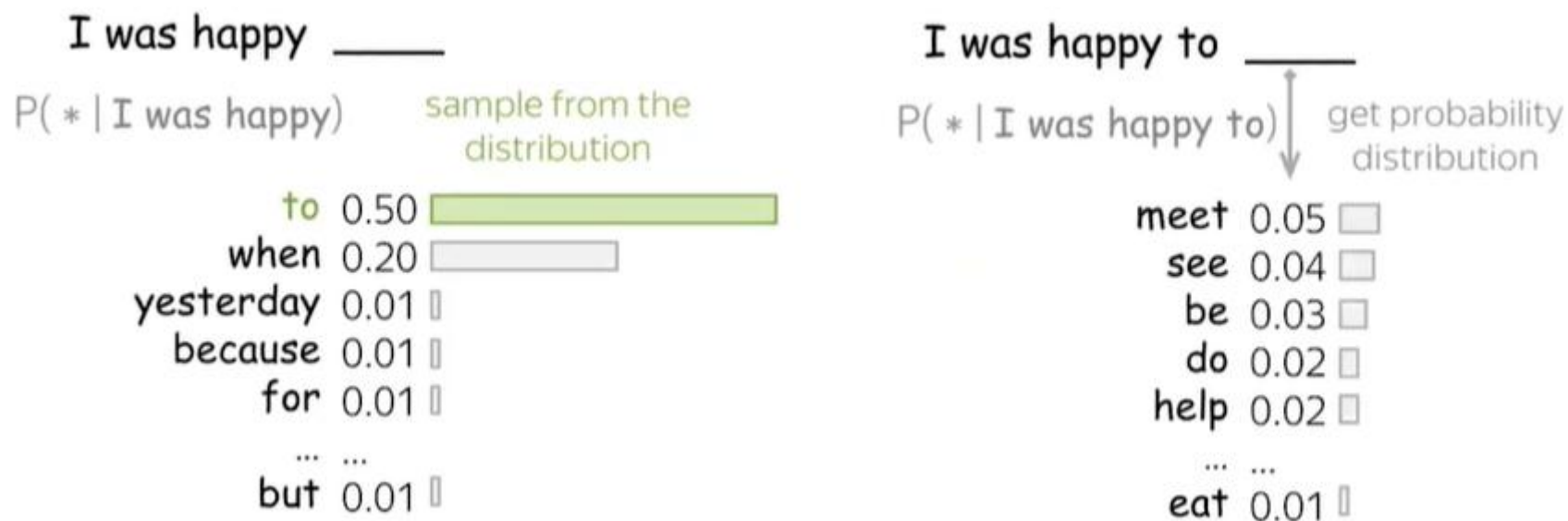


Жадный алгоритм всегда выберет «солнечно». Затем он рассчитает вероятности следующего токена для нового контекста и снова выберет наиболее вероятный.

Генерация текста с помощью языковой модели



Генерация текста с помощью языковой модели



Традиционный подход к работе с языковыми моделями

Цель: необходимо вычислить условную вероятность в формуле

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

Оценка на основе глобальной статистики текстового корпуса, т. е. подсчета.

Как вычислить условные вероятности?

Простой способ:

$$P(y_t | y_1, y_2, \dots, y_{t-1}) = \frac{N(y_1, y_2, \dots, y_t)}{N(y_1, y_2, \dots, y_{t-1})},$$

где $N(y_1, y_2, \dots, y_t)$ — сколько раз в обучающих текстах встретилась вся последовательность, включая последнее слово y_t ; $N(y_1, y_2, \dots, y_{t-1})$ — сколько раз встретился контекст, то есть последовательность без последнего слова

Марковское предположение

Представим предложение:

«Сегодня утром я выпил чашку горячего чая».

Чтобы предсказать следующее слово, теоретически можно учитывать весь предыдущий текст:

«Сегодня утром я выпил чашку горячего...»

Но чем длиннее история, тем сложнее оценить вероятность каждого возможного продолжения.

Марковское предположение гласит: мы считаем, что вероятность следующего слова зависит только от фиксированного числа предыдущих слов, а более раннюю историю игнорируем.

Марковское предположение

Например, вместо всего контекста: «Сегодня утром я выпил чашку горячего...» можно учитывать только два предыдущих слова: «чашку горячего...». Это и есть упрощение.

Как это записывается математически?

Без марковского предположения вероятность следующего слова зависит от всей предыдущей последовательности:

$$P(w_t \mid w_1, \dots, w_{t-1}).$$

При марковском предположении мы ограничиваем контекст:

$$P(w_t \mid w_1, \dots, w_{t-1}) \approx P(w_t \mid w_{t-n+1}, \dots, w_{t-1}).$$

То есть учитываем только последние $n-1$ слов.

Марковское предположение

Вероятность слова зависит только от фиксированного числа предыдущих слов.

$$P(y_t | \underbrace{y_1, y_2, \dots, y_{t-1}}_{\text{all previous tokens}}) = P(y_t | \underbrace{y_{t-n+1}, \dots, y_{t-1}}_{\text{n-1 previous token}})$$

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

Марковское предположение. Пример

$P(\text{I saw a cat on a mat}) = ?$

Before	After (3-gram)	
$P(\text{I saw a cat on a mat}) =$	$P(\text{I saw a cat on a mat}) =$	
$P(\text{I})$	$P(\text{I})$	$\longrightarrow P(\text{I})$
$\cdot P(\text{saw} \text{I})$	$\cdot P(\text{saw} \text{I})$	$\longrightarrow \cdot P(\text{saw} \text{I})$
$\cdot P(\text{a} \text{I saw})$	$\cdot P(\text{a} \text{I saw})$	$\longrightarrow \cdot P(\text{a} \text{I saw})$
$\cdot P(\text{cat} \text{I saw a})$	$\cdot P(\text{cat} \text{I saw a})$	$\longrightarrow \cdot P(\text{cat} \text{saw a})$
$\cdot P(\text{on} \text{I saw a cat})$	$\cdot P(\text{on} \text{I saw a cat})$	$\longrightarrow \cdot P(\text{on} \text{a cat})$
$\cdot P(\text{a} \text{I saw a cat on})$	$\cdot P(\text{a} \text{I saw a cat on})$	$\longrightarrow \cdot P(\text{a} \text{cat on})$
$\cdot P(\text{mat} \text{I saw a cat on a})$	$\cdot P(\text{mat} \text{I saw a cat on a})$	$\longrightarrow \cdot P(\text{mat} \text{on a})$

Возможная проблема

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

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

zero

not good: can not compute the probability

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

not good: zeros out probability of the sentence

Решение проблемы (Back-off)

Back-off (отступление к более короткому контексту) — это метод в языковых моделях, который позволяет оценивать вероятность слова, если для более длинного контекста у нас недостаточно данных.

Главная идея: если мы не знаем, какое слово обычно следует за длинной последовательностью, попробуем использовать более короткую последовательность, для которой у нас больше статистики.

Решение проблемы (Back-off)

Использовать меньше контекста в ситуациях, о которых мы мало что знаем:

- если можно, использовать триграмму
- если нет, то биграмму
- если нет, то униграмму

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

zero

not good: can not compute the probability

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

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

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

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

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

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

Решение проблемы (Back-off)

Возможные проблемы:

1. Потеря контекста

Чем короче контекст, тем меньше информации модель использует для предсказания.

2. Более общая оценка может быть неточной

Короткий контекст не всегда отражает смысл всей фразы.

3. Ошибка может передаваться дальше

Если модель выбрала неудачное слово, оно становится частью нового контекста.

4. Ограничения N-граммной модели сохраняются

Модели учитывают лишь небольшое число предыдущих слов. Даже если корпус очень большой, дальние зависимости между словами могут оставаться незамеченными.

Лучшее решение: интерполяция

Интерполяция (Interpolation) в языковых моделях — это способ оценить вероятность следующего слова, комбинируя информацию из контекстов разной длины. **Интерполяция объединяет оценки из длинного и короткого контекстов одновременно.**

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

zero
not good: can not compute the probability

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

$$\sum_i \lambda_i = 1$$

Лучшее решение: интерполяция

Предположим, мы хотим предсказать следующее слово в предложении:

«Я пью горячий ...»

Модель может использовать три оценки вероятности слова «чай»:

- Униграмма: $P(\text{чай})$ — насколько часто слово «чай» встречается вообще.
- Биграмма: $P(\text{чай} \mid \text{горячий})$ — насколько часто после слова «горячий» встречается «чай».
- Триграмма: $P(\text{чай} \mid \text{пью горячий})$ — насколько часто «чай» встречается после последовательности «пью горячий».

Каждая оценка несёт свою информацию. Интерполяция позволяет объединить их.

Лучшее решение: интерполяция

Допустим, модель получила следующие оценки:

Оценка	Вероятность слова «чай»	Вес
Триграмма	0,2	0,5
Биграмма	0,6	0,3
Униграмма	0,1	0,2

Тогда итоговая вероятность:

$$\begin{aligned} P(\text{чай}) &= 0,5 \cdot 0,2 \\ &\quad + 0,3 \cdot 0,6 \\ &\quad + 0,2 \cdot 0,1 \\ &= 0,3. \end{aligned}$$

В отличие от Back-off, модель не обязательно отказывается от длинного контекста, а сочетает его с более короткими контекстами.

Интерполяция. Проблемы

1. Нужно подбирать веса

Проблема: универсальных оптимальных весов нет. Их приходится подбирать для конкретного корпуса и задачи.

2. Редкие последовательности всё ещё могут мешать

Допустим, модель оценивает вероятность слова после последовательности «вчера вечером я». Если такая последовательность встречалась всего один раз, оценка триграммы может быть ненадёжной.

3. Возможна нулевая вероятность

Интерполяция сама по себе не устраняет проблему нулевых вероятностей.

4. Не учитывает дальний контекст

Даже если интерполяция объединяет оценки разных N-грамм, она остаётся ограниченной выбранными длинами последовательностей.

5. Увеличивает вычисления и сложность настройки

Нужно хранить статистику для нескольких длин N-грамм, вычислять несколько оценок и комбинировать их.

Сглаживание Лапласа

Сглаживание Лапласа (Laplace smoothing) — это метод, который позволяет избежать нулевых вероятностей в языковых моделях, когда какая-то последовательность слов ни разу не встретилась в обучающем тексте.

Его также называют **add-one smoothing**, потому что к частоте каждого возможного события прибавляется единица.

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

not good: zeros out probability of the sentence

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

Сглаживание Лапласа

Предположим, мы строим биграммную языковую модель и хотим оценить вероятность следующего слова после слова «я».

В обучающем тексте встретились следующие биграммы:

Биграмма	Количество
«я люблю»	4
«я читаю»	3
«я работаю»	1
«я сплю»	0

Всего после слова «я» встретилось 8 слов. Без сглаживания вероятность рассчитывается так:

$$P(w \mid \text{я}) = \frac{C(\text{я}, w)}{C(\text{я})}$$

Сглаживание Лапласа

Поэтому:

$$P(\text{люблю} \mid \text{я}) = \frac{4}{8} = 0.5$$

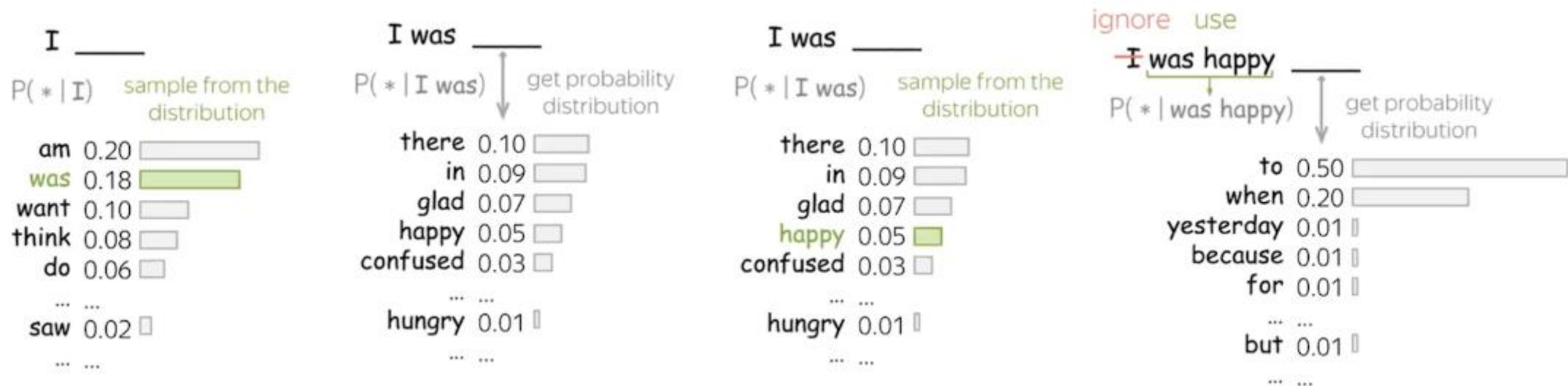
А для слова «сплю»:

$$P(\text{сплю} \mid \text{я}) = \frac{0}{8} = 0$$

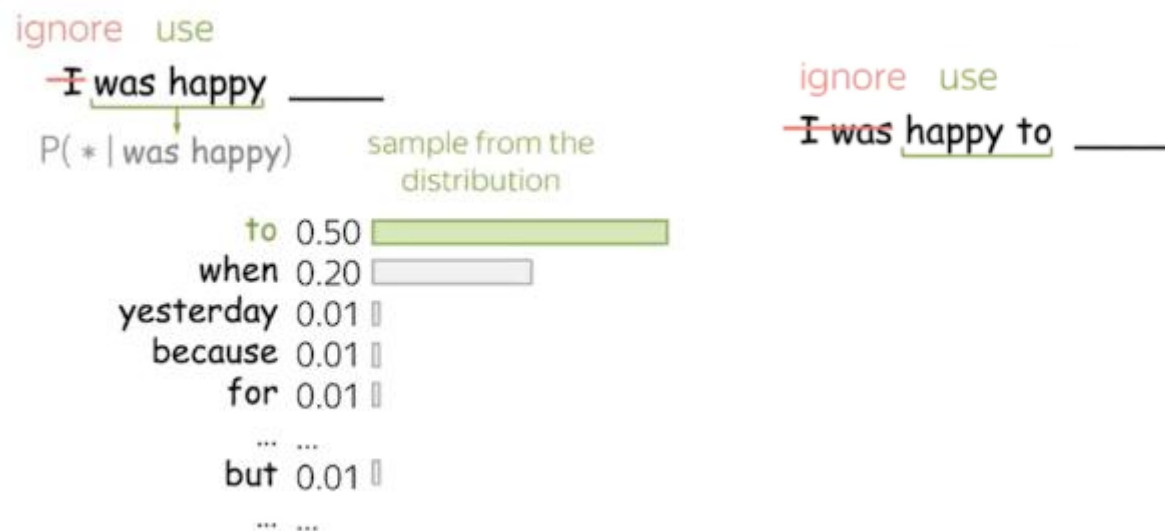
Проблема: модель присваивает слову «сплю» нулевую вероятность, хотя предложение «я сплю» вполне корректно. Возможно, оно просто не попало в обучающий текст.

Сглаживание Лапласа: прибавляем 1 к частоте каждого возможного следующего слова, в том числе к словам, которые ни разу не встречались.

Генерация текста с помощью n-граммной языковой модели (триграммная модель)



Генерация текста с помощью n-граммной языковой модели (триграммная модель)



Пример сгенерированного текста

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_

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_

Нейросетевой подход

Цель: необходимо вычислить условную вероятность в формуле:

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

В нейронных сетях нужно обучить нейронную сеть прогнозировать вероятности.

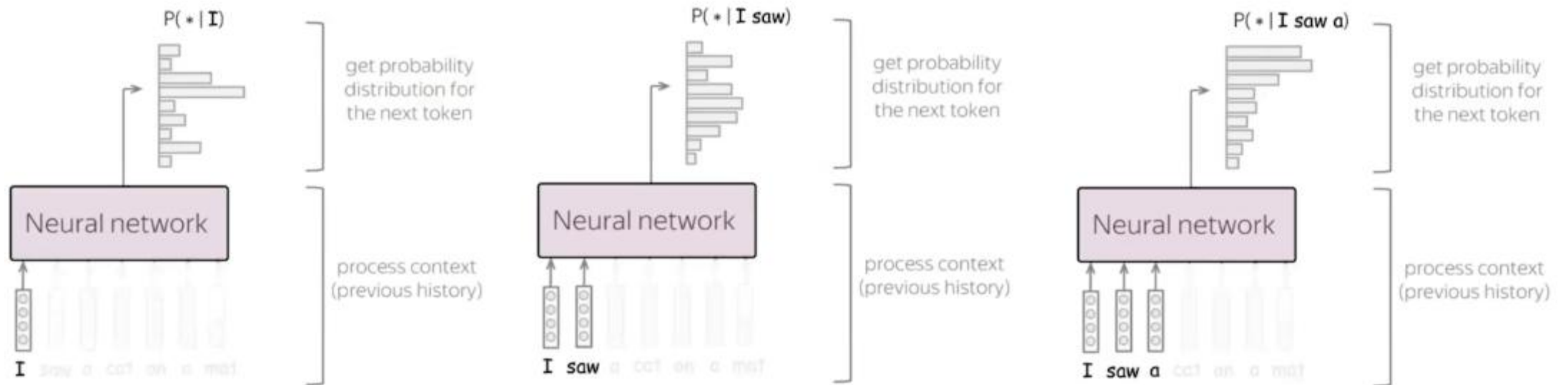
Общий подход

В нейросетевом подходе к языковым моделям, в отличие от N-граммных моделей, мы не просто считаем частоты последовательностей слов. Нейросеть преобразует контекст в числовое векторное представление, а затем на его основе вычисляет вероятности всех возможных следующих токенов.

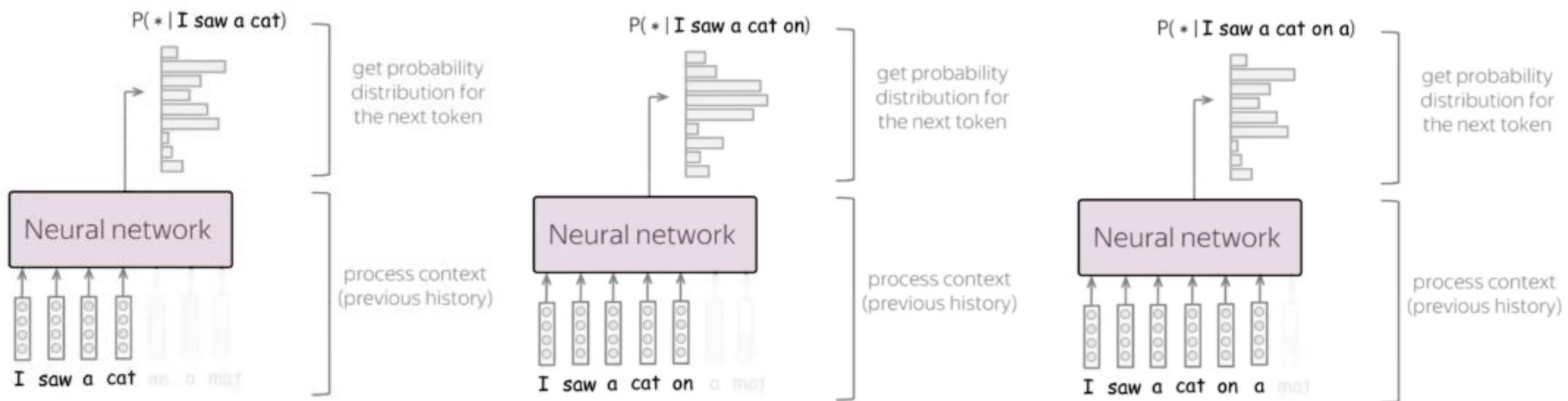
Задачи:

- получить векторное представление контекста
- оценить вероятности (прогноз распределения вероятностей для следующего токена)

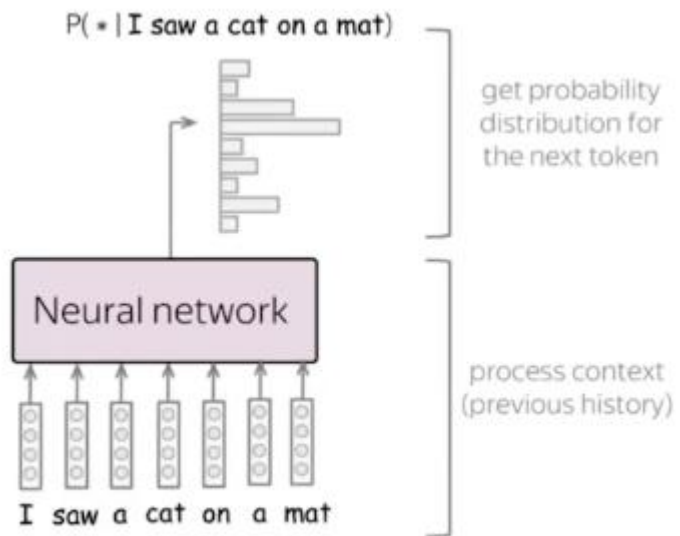
Общий подход



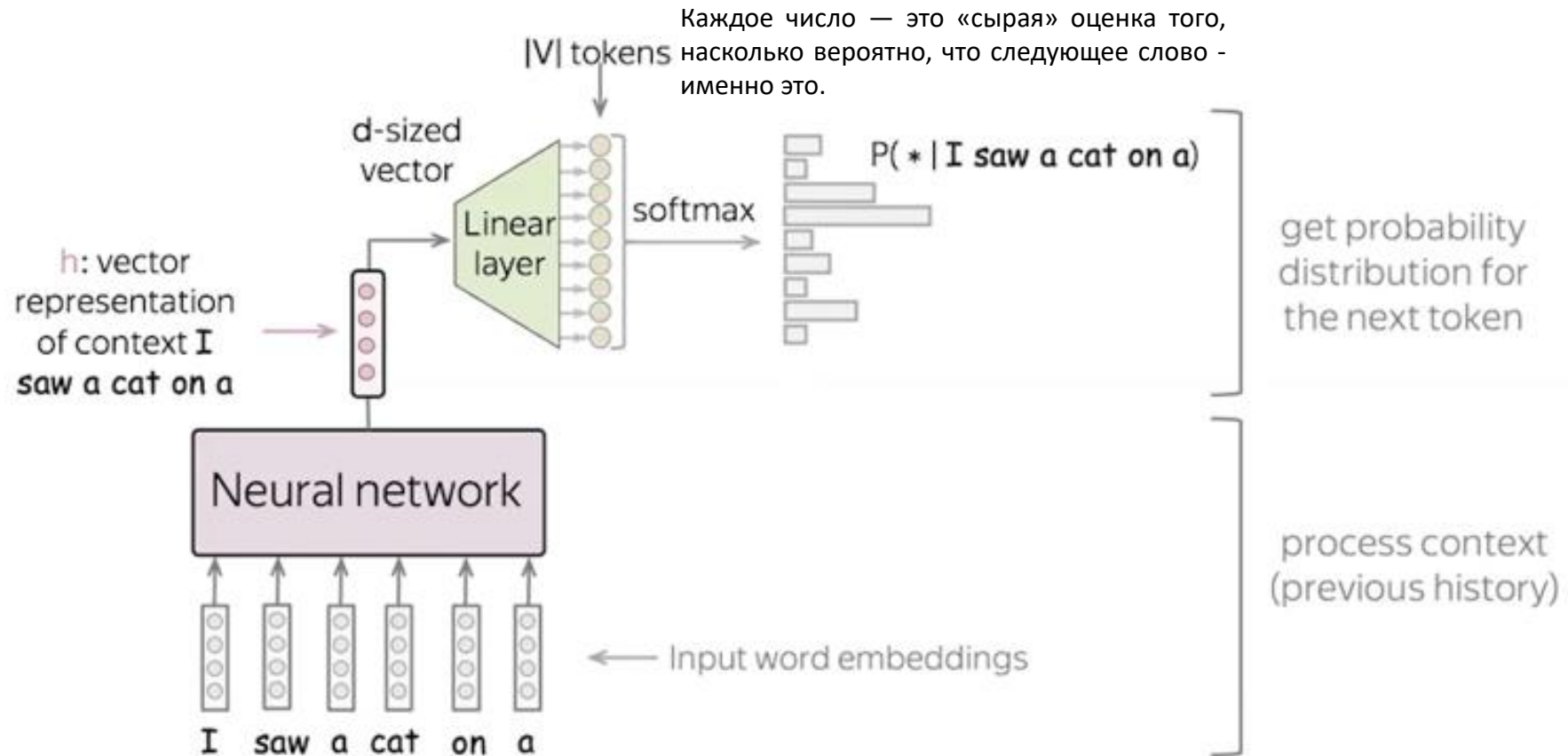
Общий подход



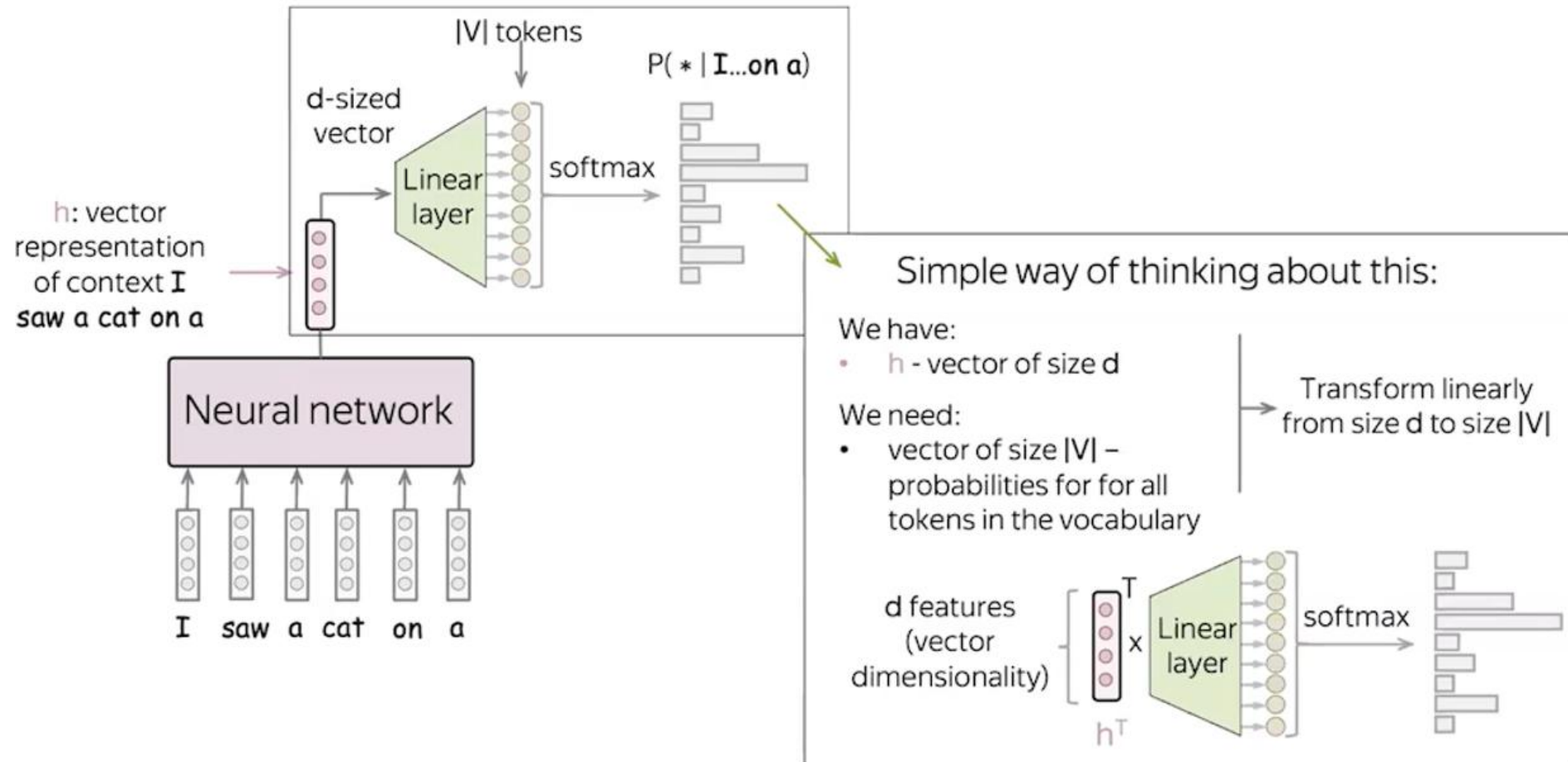
Общий подход



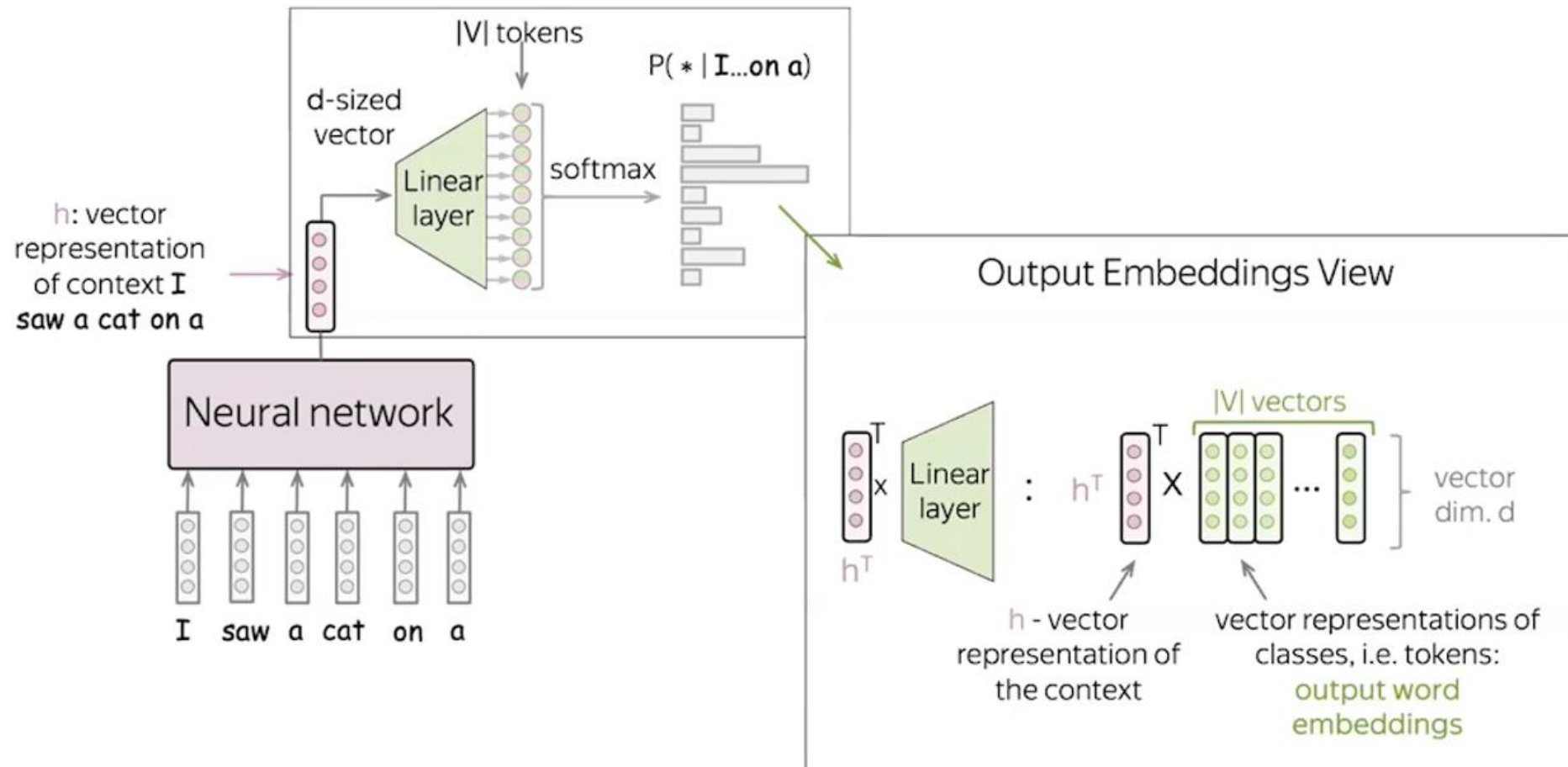
Общий подход



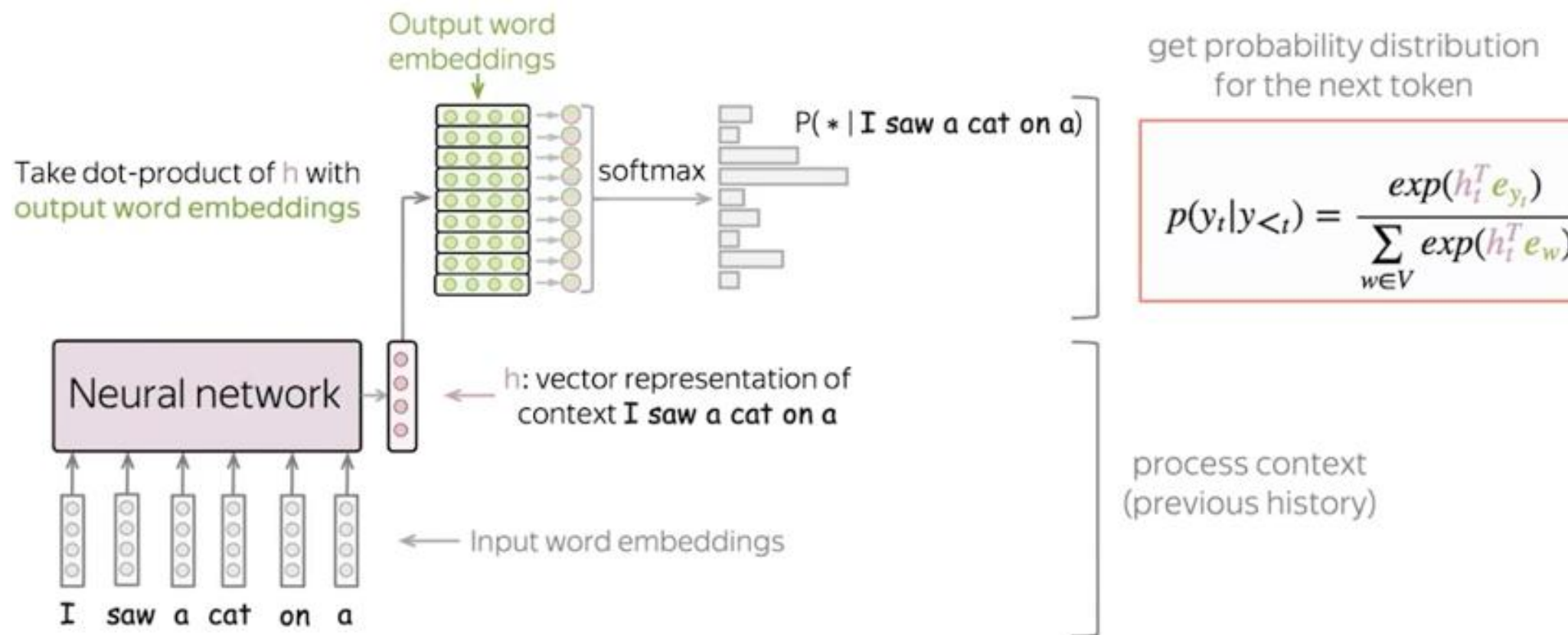
Общий подход



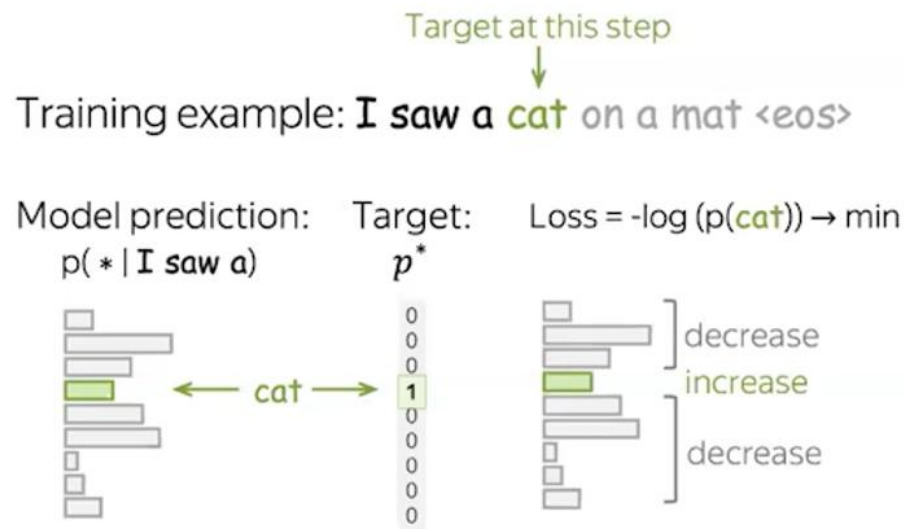
Общий подход



Представление вложений выходных данных



Обучение: кросс-энтропия



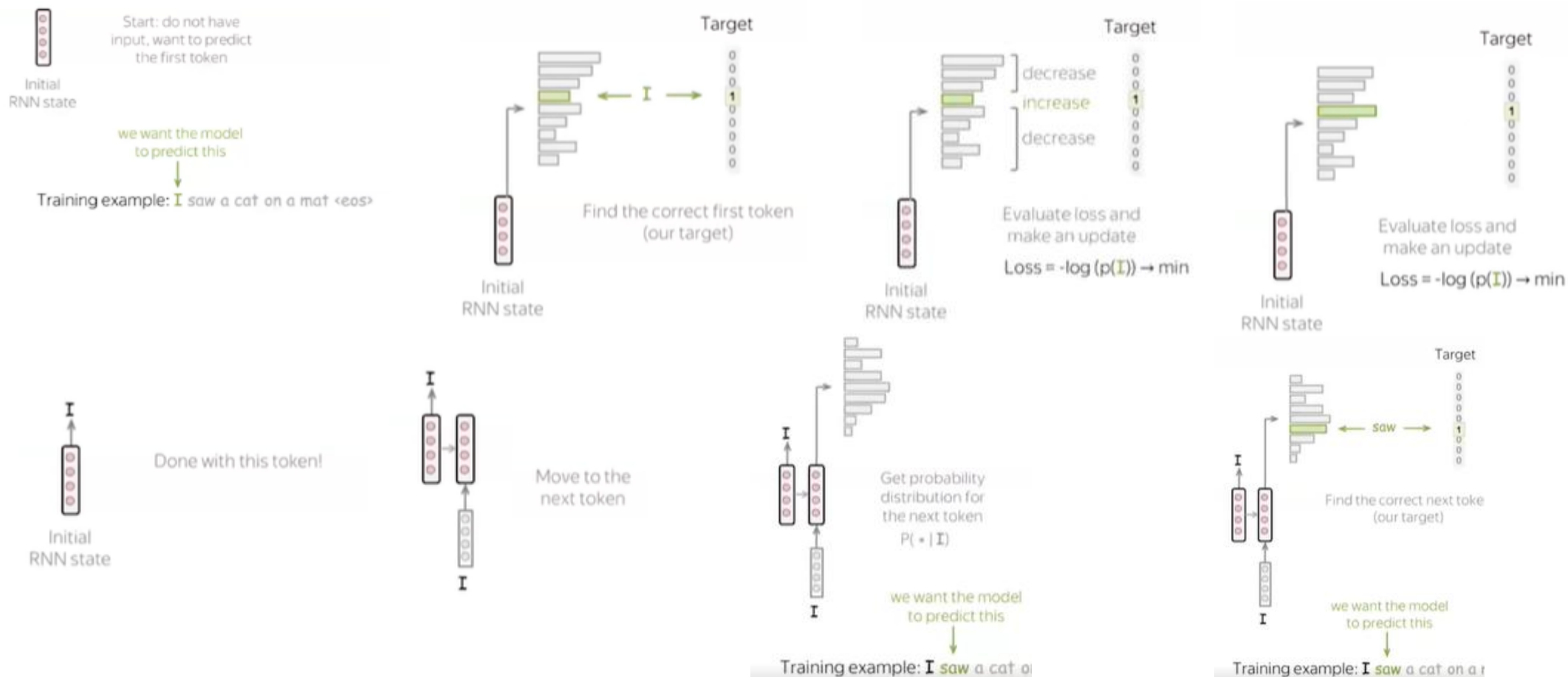
Cross-entropy loss:

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

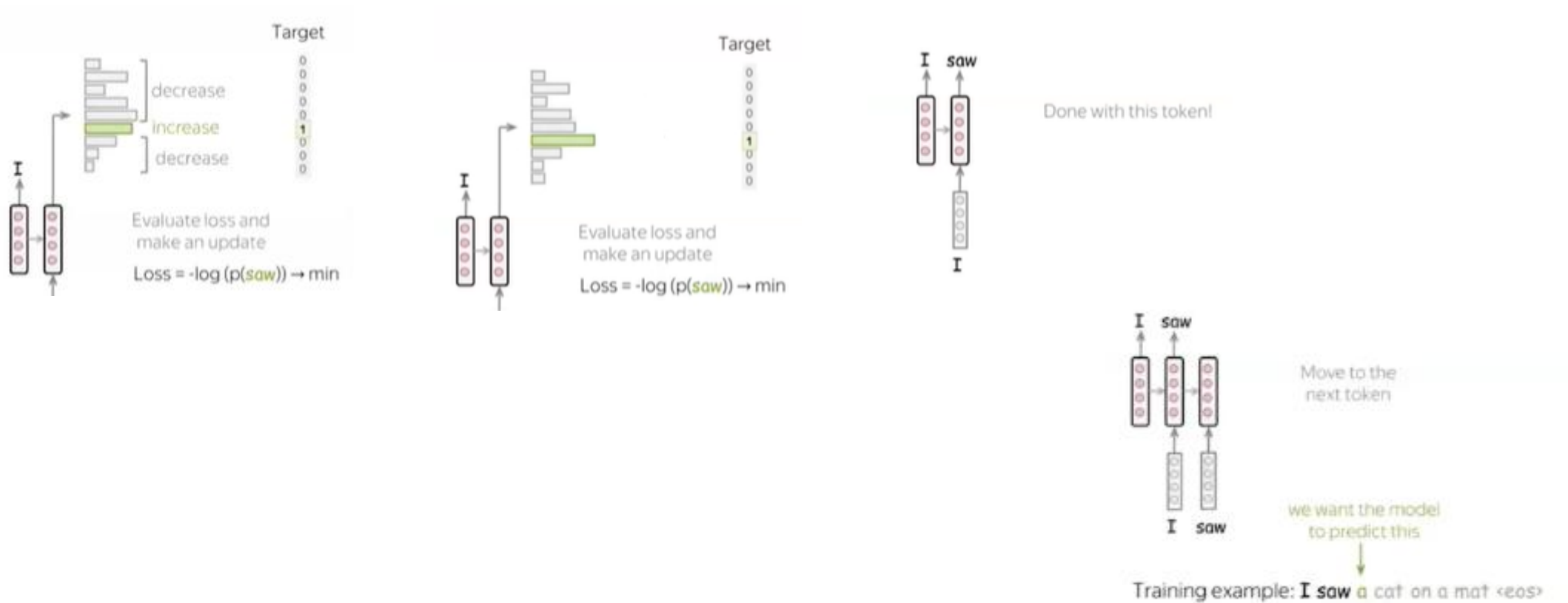
For one-hot targets, this is equivalent to

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

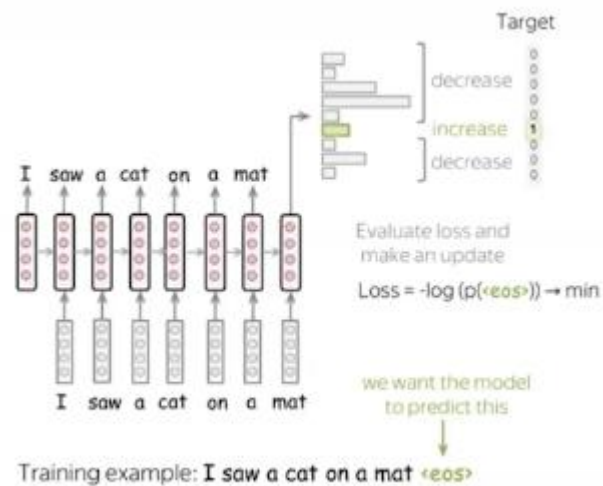
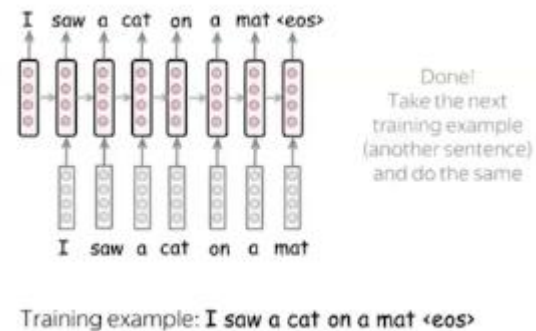
Обучение: кросс-энтропия



Обучение: кросс-энтропия



Обучение: кросс-энтропия

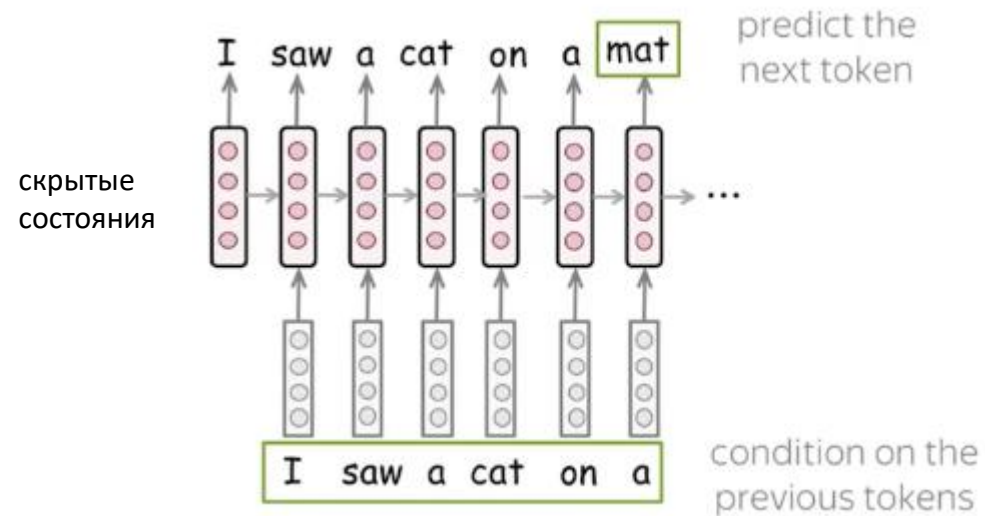


Рекуррентные модели

Главная идея: RNN читает текст последовательно, по одному токenu, и на каждом шаге передаёт информацию о предыдущих токенах следующему шагу.

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$$

x_t	Эмбединг текущего слова
h_{t-1}	Скрытое состояние с предыдущего шага
W_x	Матрица весов для текущего слова
W_h	Матрица весов для предыдущего состояния
b	Обучаемый вектор смещения
h_t	Новое скрытое состояние



Рекуррентные модели

Предположим, что модель читает предложение **I saw a cat**.

Шаг 1. Получаем слово I.

$$h_1 = \tanh(W_x x_1 + W_h h_0 + b)$$

Здесь h_0 — начальное состояние, часто нулевой вектор.

Шаг 2. Получаем слово saw.

$$h_2 = \tanh(W_x x_2 + W_h h_1 + b)$$

Теперь h_2 зависит не только от saw, но и от информации о I, содержащейся в h_1 .

И так далее.

Рекуррентные модели

После построения вектора контекста h можно использовать линейный слой и softmax:

$$P(w_{t+1} \mid w_1, \dots, w_t) = \text{softmax}(W_o h_t + b_o)$$

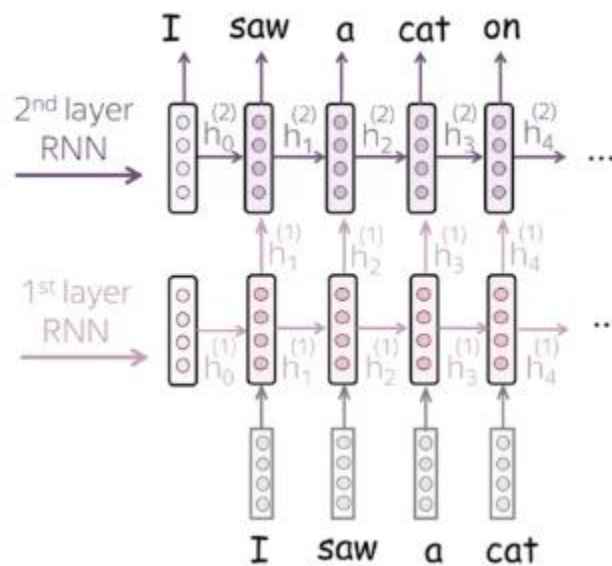
То есть:

- RNN получает очередной токен.
- Вычисляет новое скрытое состояние h_t .
- Линейный слой преобразует h_t в логиты всех токенов словаря.
- Softmax вычисляет вероятности следующих токенов.

Интересный момент: RNN может делать предсказание на каждом шаге

Рекуррентные модели

Многослойность: передача состояний из одной RNN в другую



Первый слой обрабатывает слова, а второй слой обрабатывает представления, сформированные первым.

Связность и разнообразие

Сгенерированные тексты должны быть:

- связными и осмысленными— сгенерированный текст должен иметь смысл;
- разнообразными — модель должна быть способна создавать очень разные образцы.

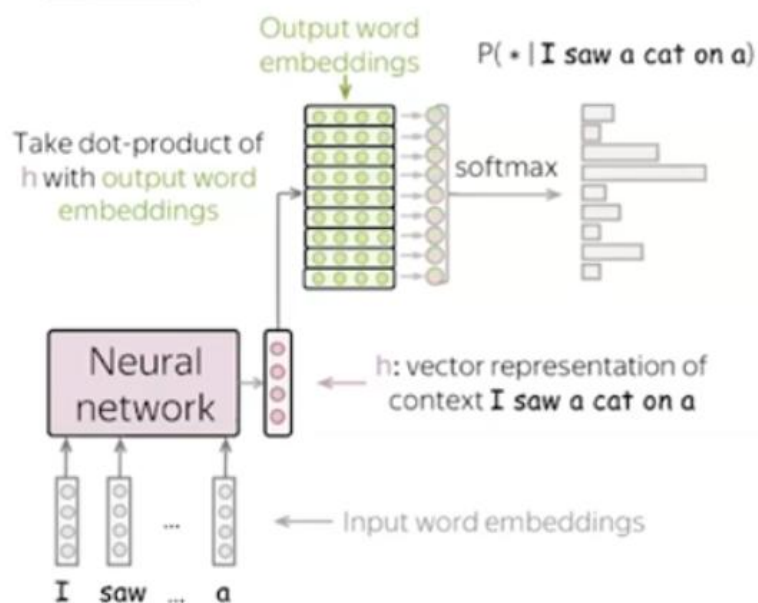
Использование softmax temperature

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

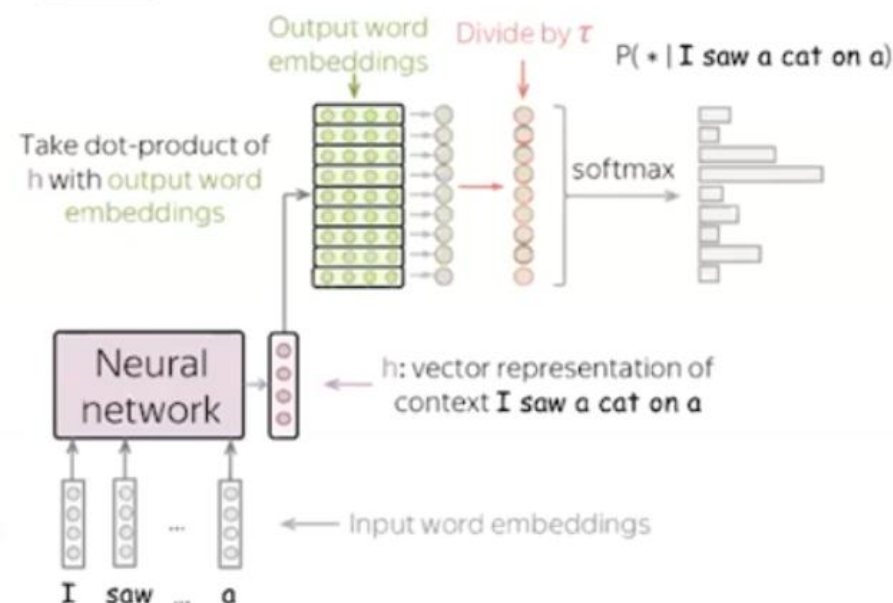
τ - softmax temperature

Параметр τ (температура) управляет «резкостью» распределения вероятностей

Before



After



Использование softmax temperature

Если $\tau > 1$ (Высокая температура):

Деление на большое число «сглаживает» различия между логитами. Распределение вероятностей становится более **плоским**. Модель становится менее уверенной, но более «любопытной» — она дает шанс редким словам.

Зачем это нужно: Чтобы модель не за цикливалась на самых частых словах и лучше изучала редкие, или чтобы генерировать более разнообразный текст.

Если $\tau < 1$ (Низкая температура):

Деление на дробь (например, 0.1) увеличивает различия. Распределение становится более **острым**. Модель становится очень уверенной и выбирает только самые вероятные слова.

Зачем это нужно: Для задач, где нужен точный и предсказуемый ответ (например, классификация).

Если $\tau = 1$:

Это стандартный Softmax

Использование softmax temperature

Зачем это используют в языковых моделях?

Для обучения (Training): Иногда используют высокую температуру, чтобы модель лучше обобщала и не переобучалась на частых словах.

Для генерации текста (Inference): Это критически важно для LLM (больших языковых моделей вроде GPT).

Меняя τ , вы меняете «креативность» модели:

- Низкая температура → скучный, но правильный текст.
- Высокая температура → креативный, но иногда бессвязный текст (модель чаще выбирает менее вероятные слова).

Пример: 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
```

Пример: 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

Связность и разнообразие. Top-K

- выбирать из первых K наиболее вероятных токенов (обычно K составляет около 10-50)
 - Выбираем K токенов с наибольшими вероятностями.
 - Отбрасываем все остальные.
 - Нормируем вероятности оставшихся токенов, чтобы их сумма равнялась 1.
 - Случайным образом выбираем следующий токен из оставшихся.

алгоритм просто берет **первые четыре слова** из этого отсортированного списка

The dress color was _____

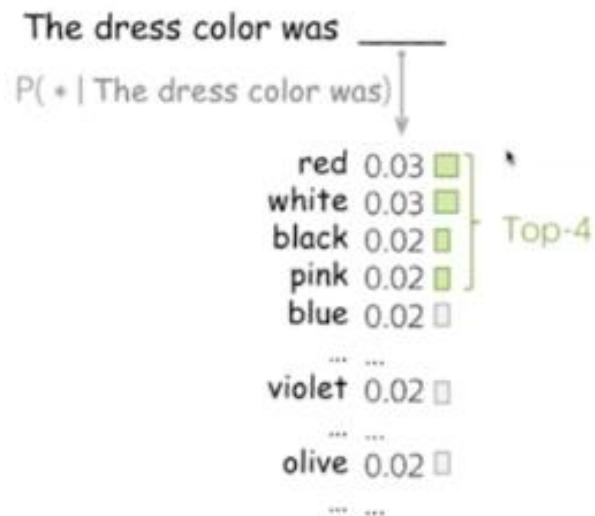
$P(* | \text{The dress color was})$

red	0.03	■
white	0.03	■
black	0.02	■
pink	0.02	■
blue	0.02	□
...	...	
violet	0.02	□
...	...	
olive	0.02	□
...	...	

Top-4

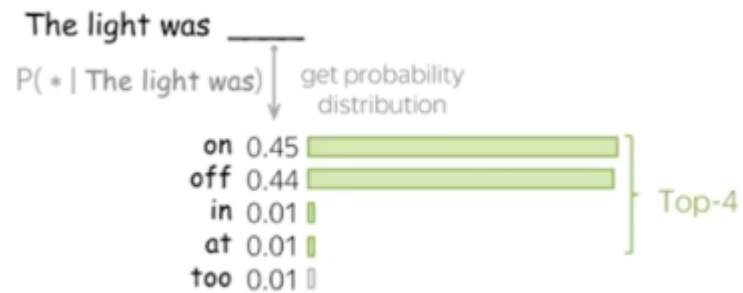
Как выбрать К?

Тор-К: недостаточно



Когда распределение вероятностей достаточно равномерное, маленькое K может быть слишком ограничивающим.

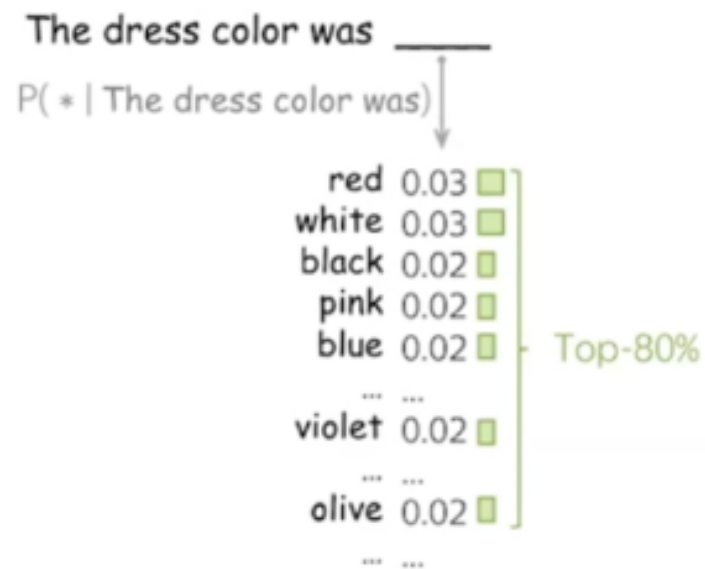
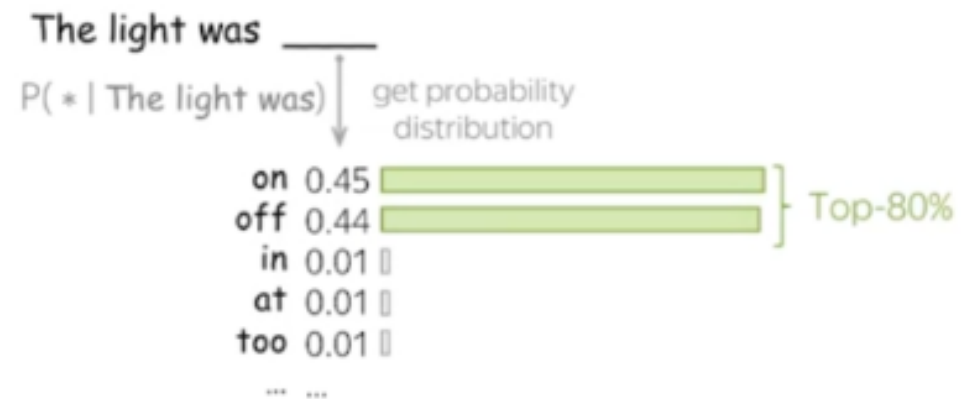
Тор-К: слишком много



Когда вероятность
сосредоточена на нескольких
токенах, большое K может
включать лишние
маловероятные варианты.

Top-p sampling

В этом методе выбора следующего токена при генерации текста вместо фиксированного количества токенов K мы выбираем столько токенов, сколько необходимо, чтобы их суммарная вероятность достигла заданного порога p .



Top-p sampling

1. Как работает Top-p?

Допустим, мы установили $p=0.8$.

Это означает, что мы хотим выбрать наиболее вероятные токены, суммарная вероятность которых составляет не менее 80%.

- Модель вычисляет вероятности всех возможных следующих токенов.
- Сортируем токены по убыванию вероятности.
- Последовательно добавляем токены, пока сумма их вероятностей не достигнет 0.8.
- Отбрасываем остальные токены.
- Нормируем вероятности выбранных токенов и случайным образом выбираем один из них.

Top-p sampling

Когда модель не уверена, какое именно продолжение выбрать, Тор-р сохраняет большое количество вариантов. Таким образом, генерация остаётся разнообразной.

Как выбирать p ?

- Маленькое p , например 0.5: выбирается меньше токенов, генерация становится более предсказуемой.
- Большое p , например 0.95: выбирается больше токенов, генерация становится разнообразнее, но возрастает вероятность выбора менее подходящего продолжения.
- $p=1$: Тор-р ничего не отбрасывает, и выбор происходит из полного распределения вероятностей.

Оценка моделей. Перплексия

Насколько модель «удивляется» при чтении нового текста?

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

Log-likelihood of the text

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

Перемножаем вероятности всех слов (это дает вероятность всего текста), а логарифм превращает умножение в сложение (так удобнее считать, чтобы числа не превращались в ноль из-за переполнения).

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

Оценка моделей. Перплексия

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

Перплексия — это мера того, насколько модель «сомневается» или «запутана», это «эффективное количество вариантов», из которых модель выбирает следующее слово:

- Если Perplexity = **10**, это значит, что модель в среднем находится в таком же замешательстве, как если бы она случайно выбирала следующее слово из **10** равновероятных вариантов.
- Если Perplexity = **1000**, модель как будто выбирает из 1000 вариантов (она очень плохо предсказывает текст).

Оценка моделей. Перплексия

Пример:

Если текст состоит из абсолютно случайных символов, перплексия будет равна размеру словаря (например, 50 000). Модель вообще не понимает, что будет дальше.

Если текст очень предсказуемый («Солнце встает на ... востоке»), перплексия будет близка к 1 (или к маленькому числу). Модель практически не ошибается.

Чем ниже перплексия, тем лучше модель. Это значит, что модель меньше «удивляется», когда читает реальный текст, и лучше понимает язык.

Перплексия — это стандартная метрика для сравнения разных языковых моделей.

Какие значения может принимать перплексия?

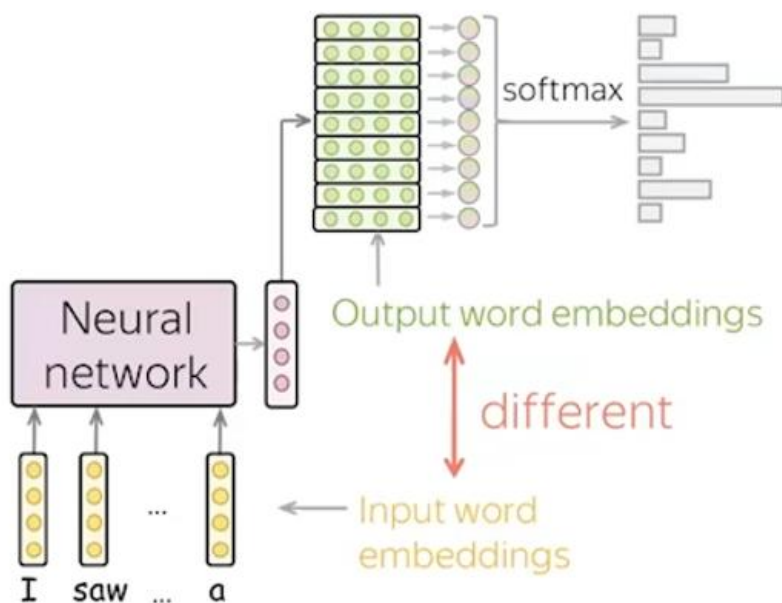
- лучшее значение – это 1 (если модель идеальна и присваивает вероятность 1 правильным токенам, то логарифмические вероятности равны нулю)
- худшее значение — это $|V|$ (если модель ничего не знает о данных, она присваивает всем токенам вероятность $1/|V|$, независимо от контекста).

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

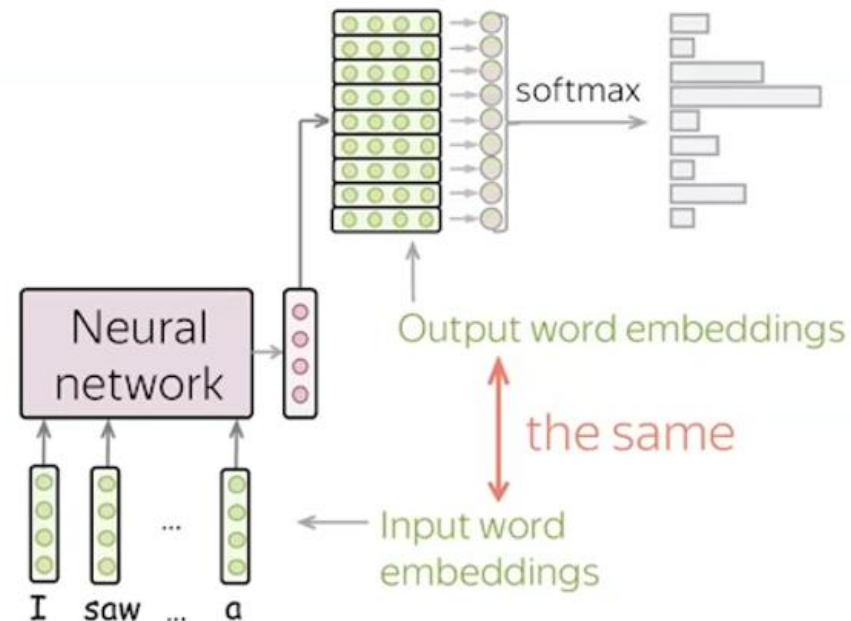
Привязка веса (Weight Tying)

Большая часть параметров модели берется из этих матриц — вы можете уменьшить размер модели!

Default (no weight tying)



Weight tying



Визуализация активации нейронов

LSTM-сети на уровне символов, обученные на ядре Linux и «Войне и мире»

- Нейрон чувствителен к положению в строке

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.

- Нейрон активируется на внутренние кавычки

"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."

Визуализация активации нейронов

- Нейрон активируется внутри операторов if

```
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;
}
```

Визуализация активации нейронов

- Нейрон активируется на внутренние комментарии и цитаты

```
/* 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;
    }
}
```

Визуализация активации нейронов

- Нейрон чувствителен к глубине кода

```
#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;
}
```

Нейроны, следящие за настроением

LSTM-сеть на уровне символов, обученная на отзывах Amazon

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.

Контрастная оценка: специфический текст

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

$P(\text{The } \text{roses} \text{ in the vase by the door } \text{are})$
 $P(\text{The } \text{roses} \text{ in the vase by the door } \text{is})$

Is the correct answer ranked higher?
 $P(\dots \text{are}) > P(\dots \text{is})?$

Способна ли модель правильно расставить приоритеты между грамматически верным и неверным вариантом?

is/are

The **roses** in the vase by the door ?

$P(* | \text{The roses in the vase by the door})$

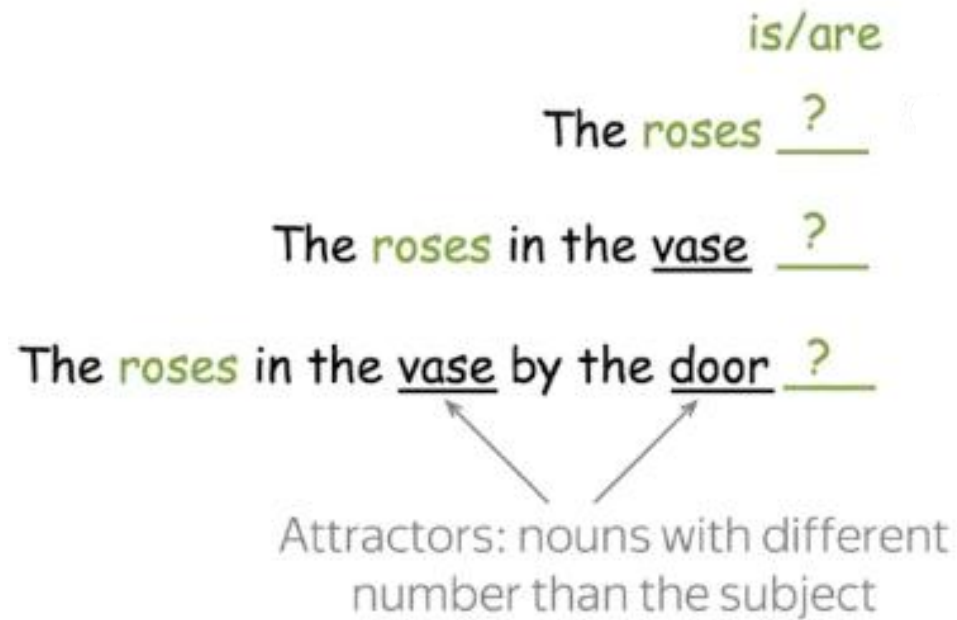
get probability distribution from LM

pick tokens of interest

is 
are 

compare the scores: correct ranking or not?

Контрастная оценка: специфический текст



Simple: no attractors

Harder: 1 attractor

Harder: 2 attractors

Аттрактор — это слово-«обманка». Это существительное, которое стоит между подлежащим и глаголом, и у которого **другое число** (единственное/множественное), чем у подлежащего.

Задание 7 (2 балла)

1. Что такое скрытое состояние RNN и почему оно необходимо для обработки последовательностей?
2. Почему при генерации текста недостаточно всегда выбирать наиболее вероятный токен?
3. В чём принципиальное различие между Top-K и Top-p sampling?
4. Что показывает перплексия языковой модели и почему меньшие значения обычно считаются лучшими?