



Лекция 1. Введение в языковое моделирование

Макаренко Владимир
12.02.2025



О чём сегодня поговорим

О курсе

Что такое языковые модели и зачем они нужны?

Задача языкового моделирования

Простейшая языковая модель

Способы генерации последовательностей

Языковая модель на основе рекуррентной нейронной сети

Задача sequence to sequence



О курсе



Структура курса

- 10 лекций
- 9 проверочных тестов (перед каждой лекции кроме сегодняшней)
- 7 семинаров:
 - 4 практических (знакомство с темой)
 - 3 теоретических (возможность сделать доклад и получить дополнительные баллы)
- 4 домашних задания:
 - Более глубокое погружение в тему.
 - Вносят основной вклад в итоговую оценку.

Система баллов

Домашние задания:

- За каждое домашнее задание можно получить до 20 баллов (суммарно – 80 баллов).
- Максимальный срок сдачи – 2 недели после выдачи.
- Прислать домашнее задание позже можно, но оно будет оценено в половину от выставленных баллов.

Проверочные тесты:

- За каждый проверочный тест можно получить 1 балл (9 в сумме).
- Проводятся перед каждой лекцией кроме сегодняшней.

Семинары:

- Дополнительные баллы можно получить, сделав доклад на теоретическом семинаре (11 баллов за разбор статьи по теме семинара).

Сдача домашних заданий

- Сдача домашних заданий происходит на портале VK Education (подробная инструкция появится там).
- Нужно прикрепить ссылку на Google Collab с выполненным домашним заданием.
- Нужно прикрепить ссылку на huggingface-репозиторий с полученной моделью.

Итоговая оценка

- 75 баллов - отлично
- 60 баллов - хорошо
- 41 балл - удовлетворительно/зачет

Зачет начинается с 41 балла. Это означает, что хотя бы одно домашнее задание должно быть сдано в срок – не получится в конце принести все домашние задания и сдать курс.

Важные ссылки

[Страница курса на VK Education](#)

[Чат курса](#)

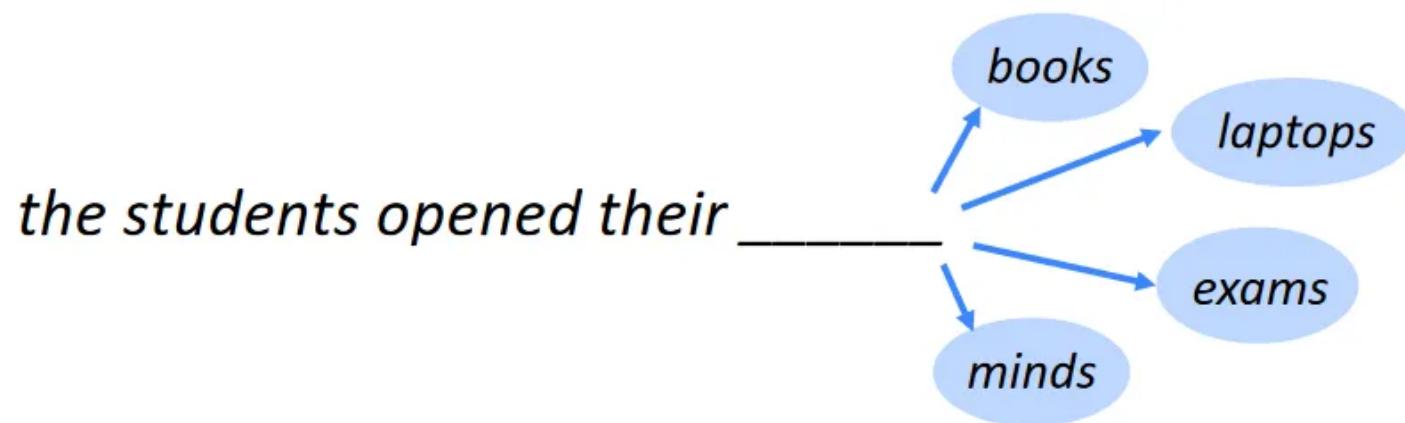
[GitHub курса](#)

Что такое
языковые модели
и зачем они
нужны?



Что такое языковая модель?

- Это вероятностная модель естественного языка.
- Имея языковую модель, часто можно:
 - Оценивать вероятность того или иного текста в целом.
 - Оценивать вероятность продолжения заданного текста.
 - Генерировать продолжение заданного текста.



Примеры. Автодополнение текста

A screenshot of a search interface, likely a web browser or a search application. The search bar at the top contains the partial query "что такое языкова". To the right of the search bar are several icons: a magnifying glass, a close button (X), a keyboard icon, a microphone icon, and a camera icon. Below the search bar, a list of six autocompletion suggestions is displayed, each preceded by a magnifying glass icon:

- что такое языковая норма
- что такое языковая модель
- что такое языковая семья
- что такое языковая ситуация
- что такое языковая школа
- что такое языковая картина мира

Примеры. Переводчик

Detect language English Spanish French ▾

What is a language model?



25 / 5,000



English Spanish Arabic ▾

¿Qué es un modelo de lenguaje?



Send feedback

Примеры. Чат-боты



Объясни в двух словах, что такое языковая модель, и для чего она нужна.



Языковая модель — это алгоритм, предсказывающий слова в тексте на основе предыдущих. Она нужна для обработки и генерации естественного языка, как в чатах, переводах и анализе текста.

Я учусь в 3-ем классе, и ничего не понял. Объясни проще.



Языковая модель — это программа, которая учится понимать и продолжать текст. Например, если ты начнёшь предложение, она угадает, что будет дальше. Она помогает писать сообщения, переводить слова и искать ответы.



Задача языкового моделирования



Текст как последовательность токенов

Словарь – непустое конечное множество. Часто текст можно каким-либо образом представить в виде последовательности элементов словаря.

Токенизация – процесс представления текста в виде последовательности элементов заданного словаря (**токенов**).

На практике естественно работать не с последовательностью токенов, а с последовательностью **номеров токенов** в словаре.

Словарь: $V = \{_, а, б, в, г, д, е, ё, ж, з, и, й, к, л, м, н, о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ъ, ы, ѿ, ѿ, ѿ\}$

Текст	Последовательность токенов	Последовательность номеров токенов
мама мыла раму	м, а, м, а, _, м, ы, л, а, _, р, а, м, у	14, 1, 14, 1, 0, 14, 29, 13, 1, 0, 18, 1, 14, 21
привет	п, р, и, в, е, т	17, 18, 10, 3, 6, 20

Еще несколько примеров словарей и результатов токенизации

Словарь: $V = \{_, \text{мама}, \text{мыла}, \text{раму}, \text{привет}\}$

Текст	Последовательность токенов	Последовательность номеров токенов
мама мыла раму	мама, _, мыла, _, раму	1, 0, 2, 0, 3
привет	привет	4

Словарь: $V = \{_, \text{ма}, \text{мы}, \text{ла}, \text{ра}, \text{му}, \text{при}, \text{вет}\}$

Текст	Последовательность токенов	Последовательность номеров токенов
мама мыла раму	ма, ма, _, мы, ла, _, ра, му	1, 1, 0, 2, 3, 0, 4, 5
привет	при, вет	6, 7

Специальные токены

Как правило, в словарь заранее добавляют несколько специальных токенов:
<bos>, **<eos>**, **<pad>**.

Описание:

- **<bos>** – токен начала последовательности. Его искусственно добавляют к началу каждой последовательности.
- **<eos>** - токен конца последовательности. Его искусственно добавляют к концу каждой последовательности.
- **<pad>** – используется для “выравнивания” последовательностей по длине.

Специальные токены. Примеры

Примеры:

- <bos>,мама,_,мыла,_раму,<eos>
- <bos>,привет,<eos>,<pad>,<pad>,<pad>,<pad>

Видно, что в начало и конец последовательностей мы добавили соответственно токены <bos> и <eos>, а вторую последовательность выровняли по длине с первой (теперь обе состоят из 7 токенов).

Byte pair encoding (BPE)

Изначально добавляем в словарь все уникальные символы и специальные токены.

Далее выполняем следующие шаги, пока размер словаря не достигнет заданного размера:

- Найти наиболее частотную пару подряд идущих символов по всем текстам.
- Создать в словаре новый символ для частотной пары.
- Заменить во всех текстах все вхождения пары на новый символ.

Для токенизации нового слова нужно последовательно выполнить замены символов в порядке выполнения.

Byte pair encoding. Пример

Пополнение словаря:

Шаг	Словарь	Текст	Частотная пара	Новый символ	Новый словарь	Новый текст
1	a,b	ababbaabbabaa	a,b	c = ab	a,b,c	ccbacbcaa
2	a,b,c	ccbacbcaa	cb	d = cb	a,b,c,d	cdadcaa

Токенизация нового текста:

Шаг	Последовательность	Применяемое правило	Новая последовательность
1	a,b,a,b,a,a,a,b,b,b,b,a,b,a	c = a,b	c,c,a,a,c,b,b,b,c,a
2	c,c,a,a,c,b,b,b,c,a	d = c,b	c,c,a,a,d,b,b,c,a

Byte pair encoding на уровне байтов

Пусть исходный словарь – это все байты от 0 до 255.

Закодируем исходные тексты в UTF-8, получим набор байтов и соответствующие токены.

Далее запускаем алгоритм ВРЕ:

- Ищем пару наиболее часто встречающихся токенов (изначально это байты).
- Заводим в словаре новый элемент для пары.
- Заменяем соответствующую пару токенов в последовательностях на новый элемент.

Преимущества: всегда сможем закодировать любой текст.

Вероятностная модель

Построим вероятностную модель текста. Как и раньше, будем представлять текст в виде **конечной последовательности** токенов из словаря.

Так как длина последовательности может быть, в принципе, любой длины, то будем считать, что у нас имеется **бесконечная случайная последовательность** токенов:

$$X_1, X_2, X_3, \dots,$$

но лишь первые несколько токенов формируют представление нашего текста.

Вероятностная модель

Чтобы понимать, где заканчивается представление нашего текста и начинается фиктивный хвост, будем использовать специальный токен **<eos>** в конце нашего представления.

Иногда нам будет полезно начинать **нумерацию последовательности** не с единицы, а с нуля или даже отрицательного числа.

Вероятностная модель

Будем считать, что на конечных последовательностях токенов задано **вероятностное распределение**:

$$\mathbb{P}(X_1 = x_1, X_2 = x_2, X_3 = x_3, \dots, X_n = x_n), \quad n \in \mathbb{N}, \quad x_1, x_2, \dots, x_n \in V.$$

То есть для любой последовательности конечной длины мы сможем посчитать ее вероятность (а значит, и вероятность для текста).

Вероятностная модель

Всюду далее в целях экономии места вместо записей вида

$$\mathbb{P}(X_{i_1} = x_{i_1}, X_{i_2} = x_{i_2}, X_{i_3} = x_{i_3}, \dots),$$

$$\mathbb{P}(X_{i_1} = x_{i_1}, X_{i_2} = x_{i_2}, X_{i_3} = x_{i_3}, \dots | X_{j_1} = x_{j_1}, X_{j_2} = x_{j_2}, \dots)$$

будем использовать соответственно

$$\mathbb{P}(x_{i_1}, x_{i_2}, x_{i_3}, \dots), \mathbb{P}(x_{i_1}, x_{i_2}, x_{i_3}, \dots | x_{j_1}, x_{j_2}, \dots).$$

Например, $\mathbb{P}(x_1, x_2)$ и $\mathbb{P}(x_1, x_3)$ могут иметь разные значения, даже если $x_2 = x_3$, так как, вообще говоря,

$$\mathbb{P}(X_1 = x, X_2 = y) \neq \mathbb{P}(X_1 = x, X_3 = y).$$

Вероятностная модель

Итак, на последовательностях токенов конечной длины задано вероятностное распределение:

$$\mathbb{P}(x_1, x_2, x_3, \dots, x_n), \quad n \in \mathbb{N}, \quad x_1, x_2, \dots, x_n \in V.$$

Перепишем эту вероятность, используя **формулу умножения**:

$$\mathbb{P}(x_1, x_2, x_3, \dots, x_n) = \mathbb{P}(x_1) \cdot \mathbb{P}(x_2|x_1) \cdot \mathbb{P}(x_3|x_1, x_2) \cdot \dots \cdot \mathbb{P}(x_n|x_1, \dots, x_{n-1}).$$

Можно привести все множители к единому виду, чтобы мы работали только с *условными вероятностями*.

Роль токена **<bos>**

Исправим "проблему" с первым токеном, добавив в словарь (и в начало каждой последовательности) специальный токен начала последовательности **<bos>**.

Тогда при $x_0 = \text{<bos>}$ имеем:

$$\mathbb{P}(x_0) = 1,$$

$$\mathbb{P}(x_0, x_1, x_2, x_3, \dots, x_n) = \mathbb{P}(x_1|x_0) \cdot \mathbb{P}(x_2|x_0, x_1) \cdot \dots \cdot \mathbb{P}(x_n|x_0, x_1, \dots, x_{n-1}).$$

Таким образом, в рамках общего подхода можем оценивать и распределение первого токена тоже.

Осталось научиться аппроксимировать функцию $\mathbb{P}(x_k|x_0, x_1, \dots, x_{k-1})$.

Простейшая языковая модель



N-граммы

Назовем n -граммой последовательность из n подряд стоящих токенов.

Пусть дана последовательность из 5 токенов:

мама	—	мыла	—	раму
------	---	------	---	------

1-граммы (униграммы):

мама	—	мыла	—	раму
------	---	------	---	------

мама	—	мыла	—	раму
------	---	------	---	------

мама	—	мыла	—	раму
------	---	------	---	------

мама	—	мыла	—	раму
------	---	------	---	------

мама	—	мыла	—	раму
------	---	------	---	------

2-граммы (биграммы):

мама	—	мыла	—	раму
------	---	------	---	------

мама	—	мыла	—	раму
------	---	------	---	------

мама	—	мыла	—	раму
------	---	------	---	------

мама	—	мыла	—	раму
------	---	------	---	------

3-граммы (триграммы):

мама	—	мыла	—	раму
------	---	------	---	------

мама	—	мыла	—	раму
------	---	------	---	------

мама	—	мыла	—	раму
------	---	------	---	------

Марковское свойство

Упрощение (Марковское свойство):

$$\mathbb{P}(x_k | x_1, x_2, \dots, x_{k-1}) = \mathbb{P}(x_k | x_{k-m}, \dots, x_{k-2}, x_{k-1}), \quad k > m.$$

То есть считаем, что условное распределение текущего токена полностью определяется предыдущими m токенами (m -граммой), а не всеми предыдущими токенами, как в общем случае.

Кроме того, будем считать, что выполняется свойство **однородности**, то есть условная вероятность $\mathbb{P}(x_k | x_{k-m}, \dots, x_{k-2}, x_{k-1})$ зависит только от значений аргументов, но не от индекса k .

Марковское свойство. Продолжение

Применяя упрощения, имеем:

$$\mathbb{P}(x_k | x_1, x_2, \dots, x_{k-1}) = \prod_{k=1}^m \mathbb{P}(x_k | \overbrace{x_1, \dots, x_{k-1}}^{\text{меньше } m \text{ токенов в условии}}) \cdot \prod_{k=m+1}^n \mathbb{P}(x_k | \overbrace{x_{k-m}, \dots, x_{k-1}}^{\text{ровно } m \text{ токенов в условии}}).$$

Снова видим, что первые несколько m множителей выбиваются из общей картины (условные распределения зависят не от m предыдущих токенов, а от меньшего количества).

Марковское свойство. Продолжение

Положим

$$x_k = \langle \text{pad} \rangle, \quad k \in \{-m+1, \dots, -1, 0\},$$

$$\mathbb{P}(x_{-m+1}, \dots, x_{-1}, x_0) = 1,$$

тогда имеем

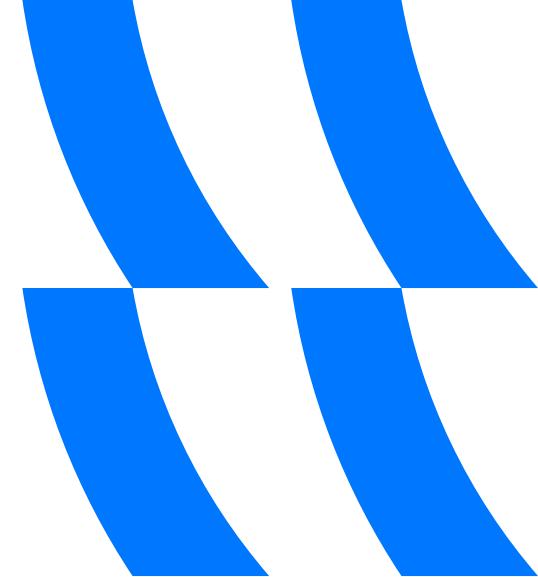
$$\mathbb{P}(x_k | x_{-m+1}, \dots, x_{-1}, x_0, x_1, x_2, \dots, x_{k-1}) = \prod_{k=1}^n \mathbb{P}(x_k | x_{k-m}, \dots, x_{k-1}).$$

Теперь условные распределения каждого из токенов x_1, \dots, x_n зависят ровно от m предыдущих токенов.

Связь с цепями Маркова

Рассматриваемая модель является **однородной цепью Маркова**, где состояниями являются т-граммы.

Остается только оценить вероятности перехода из состояния в состояние.



Оценивание параметров модели

Обозначим через $N(y_1, y_2, \dots, y_r)$ количество раз, которые r -грамма (y_1, y_2, \dots, y_r) встретилась в последовательностях обучающей выборки. Тогда условную вероятность можно оценить так:

$$\mathbb{P}(x_k | x_{k-m}, \dots, x_{k-1}) \approx N(x_{k-m}, x_{k-m+1}, \dots, x_{k-1}, x_k) / \sum_{y \in V} N(x_{k-m}, x_{k-m+1}, \dots, x_{k-1}, y).$$

```
train_data = [
    ['<pad>', '<pad>', 'a', 'b', 'a', 'b', 'a', 'a', 'b', 'b', 'a', 'a', '<eos>'],
    ['<pad>', '<pad>', 'a', 'b', 'a', 'b', 'a', 'b', '<eos>']
]

def count_frequencies(train_data: List[str], m: int = 2) -> Dict[Tuple[str], int]:
    cnt = defaultdict(int)
    r = m + 1
    for seq in train_data:
        for i in range(len(seq) - r + 1):
            r_gramm = tuple(seq[i: i + r])
            cnt[r_gramm] += 1
            cnt[r_gramm[:-1]] += 1
    return cnt

def get_conditional_probability(token: str, n_gram: Tuple[str], freq: Dict[Tuple[str], int]) -> float:
    return freq[n_gram + (token,)] / freq[n_gram]

freq = count_frequencies(train_data)
print(get_conditional_probability('a', ('<pad>', '<pad>'), freq))
```

Тренировочная выборка:

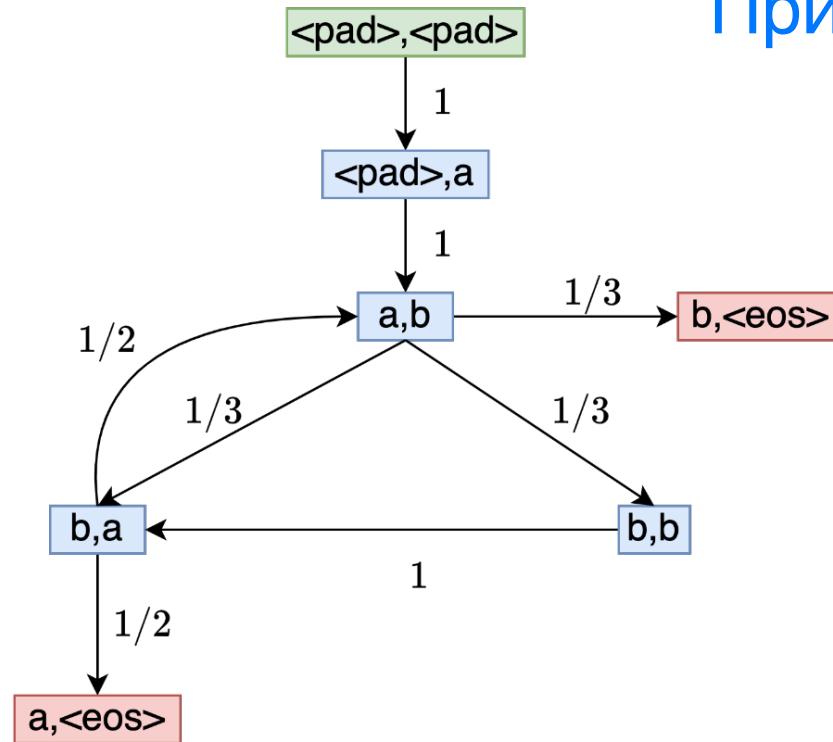
<pad>,<pad>,a,b,a,b,<eos>

<pad>,<pad>,a,b,b,a<eos>

Частоты:

<pad>,<pad>,a	2
<pad>,a,b	2
a,b,a	1
b,a,b	1
a,b,<eos>	1
a,b,b	1
b,b,a	1
b,a,<eos>	1
<pad>,<pad>,*	2
<pad>,a,*	2
a,b,*	3
b,a,*	2
b,b,*	1

Пример



$$\mathbb{P}(<\text{pad}>, <\text{pad}>, a, b, <\text{eos}>) = 1 \cdot 1 \cdot 1/3 = 1/3$$

Проблемы подхода

- При **малых t** может показывать **низкое качество**, поскольку не улавливает долгосрочные зависимости.
- При **больших t** возникает **проклятие размерности**.
Уникальных t -грамм (состояний) становится очень много, а данных по каждой из них - мало. В результате у многих новых последовательностей оцененная вероятность будет равна нулю (числитель равен нулю) или не определена (знаменатель равен нулю).

Сглаживание Лапласа

Модифицируем оценку вероятности:

$$\mathbb{P}(x_k | x_{k-m}, \dots, x_{k-1}) = (N(x_{k-m}, \dots, x_{k-1}, x_k) + \alpha) / \left(\alpha |V| + \sum_{y \in V} N(x_{k-m}, \dots, x_{k-1}, y) \right),$$

где $\alpha > 0$. Мы как бы предполагаем, что заранее каждая $(m + 1)$ -грамма встретилась α раз.

Теперь вероятность определена для любой последовательности, причем она не равна нулю.

Эту формулу можно также получить, используя байесовский подход.

Способы генерации последовательности



Жадная генерация

Каждый раз продолжаем последовательность наиболее вероятным токеном. Формально строим последовательность вида

$$x_k = \begin{cases} \text{<bos>}, & k = 0, \\ \underset{x \in V}{\operatorname{argmax}} \mathbb{P}(x | x_0, x_1, \dots, x_{k-1}), & k > 0. \end{cases}$$

Завершаем процесс после того, как сгенеририровали токен `<eos>`.

Случайная генерация

Каждый раз продолжаем последовательность, сэмплируя токен из условного распределения.
Формально строим последовательность вида

$$\begin{cases} x_0 = \langle \text{bos} \rangle, \\ x_k \sim \mathbb{P}(x|x_0, x_1, \dots, x_{k-1}), \quad k > 0. \end{cases}$$

Завершаем процесс после того, как сгенеририровали токен `<eos>`.

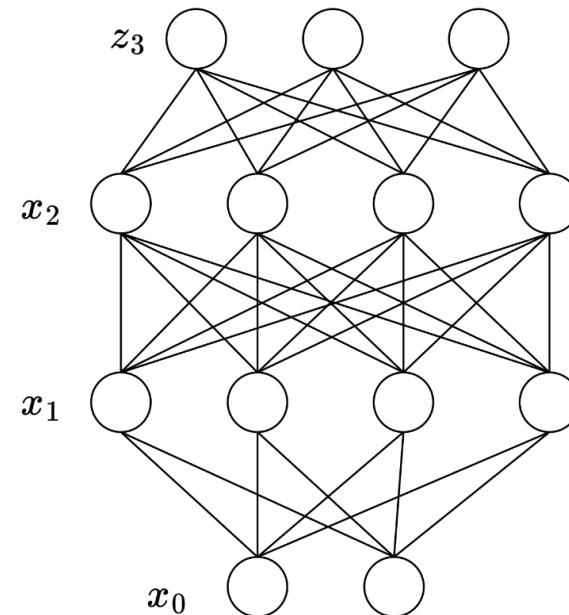
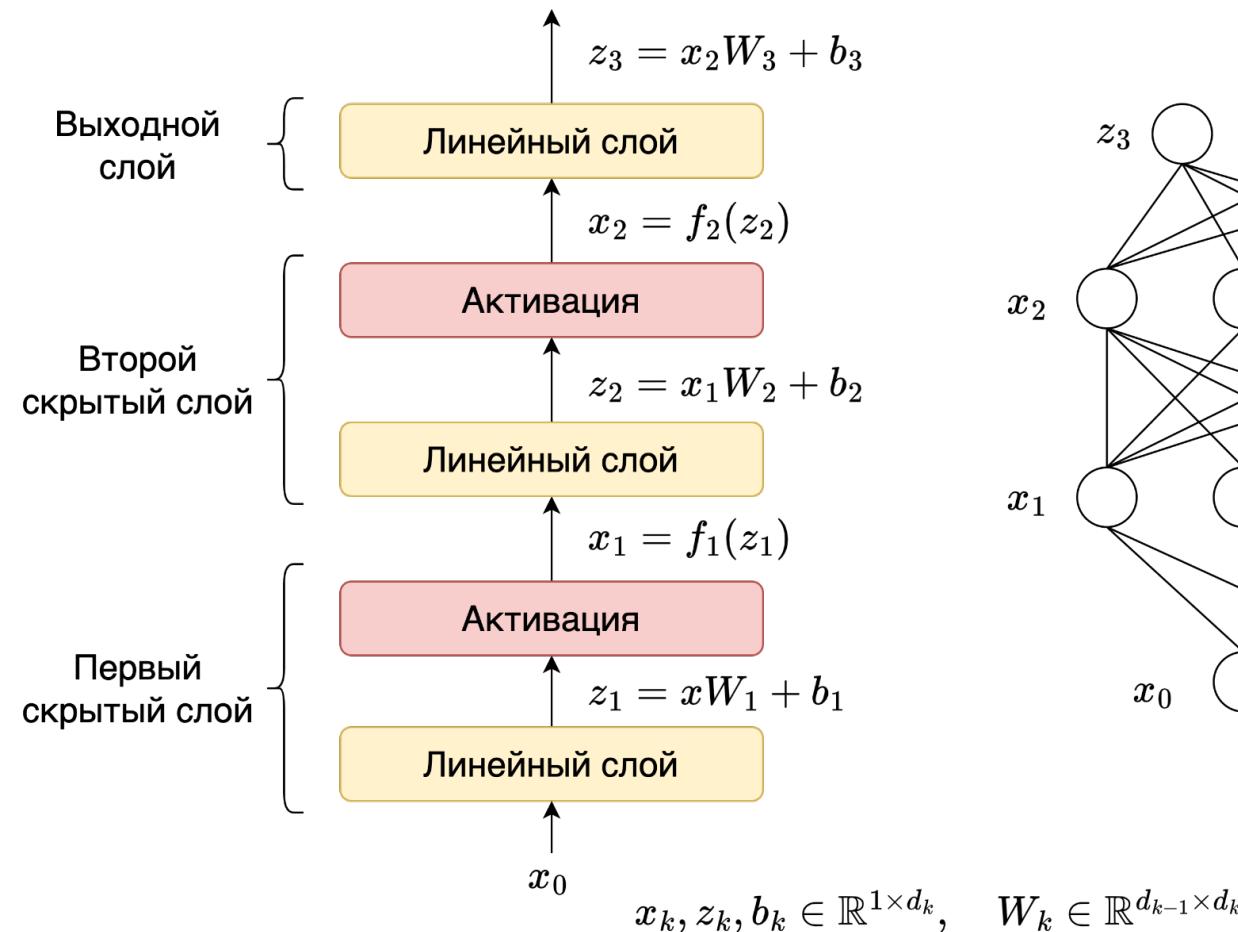
Модификации:

- *top- k sampling*: сэмплируем только из k наиболее вероятных пропорционально их вероятностям.
- *nucleus (top- p) sampling*: сэмплируем из минимального набора наиболее вероятных токенов, у которых суммарная вероятность не меньше p .

Нейронные сети

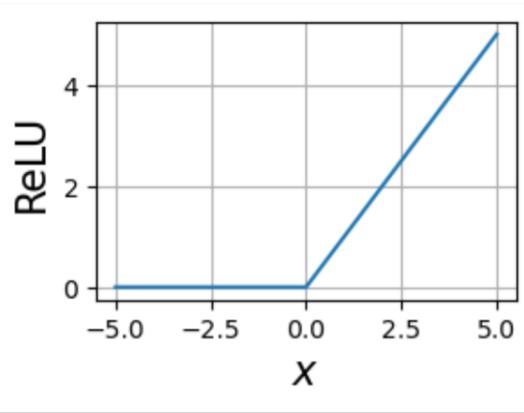


Многослойный перцептрон (полносвязная нейросеть)

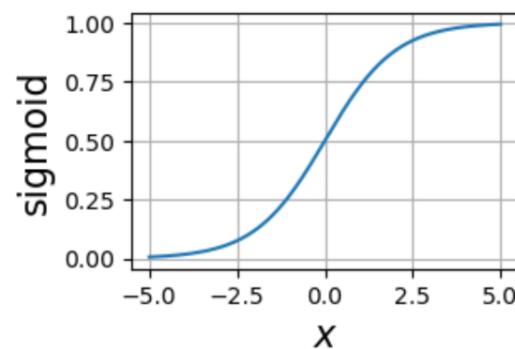


Популярные функции активации (применяются покомпонентно)

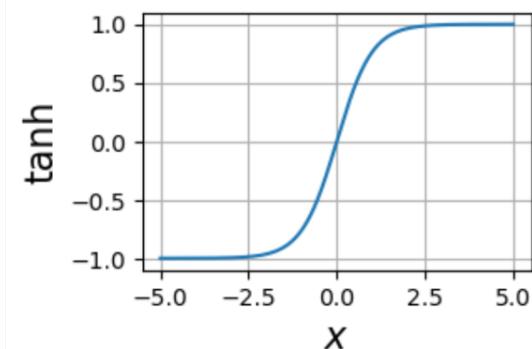
$$\text{ReLU}(x) = \max(0, x)$$



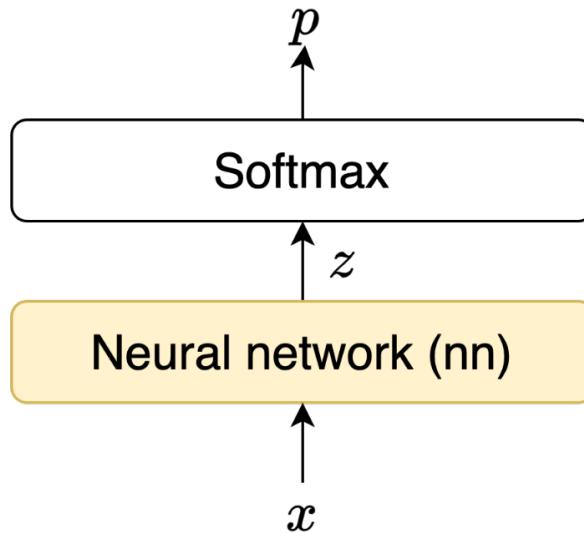
$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$



$$\tanh(x) = \frac{e^{-x} + e^x}{e^{-x} + e^x}$$



Нейросети. Задача классификации



$$z = \text{nn}(x),$$
$$p = \text{softmax}(z), \quad p[j] = \frac{\exp(z[j])}{\sum_{i=1}^M \exp(z[i])}$$

$$x \in \mathbb{R}^{1 \times d}, \quad z \in \mathbb{R}^{1 \times M}$$

Каждый объект x принадлежит одному из M классов.

Модель предсказывает, какому классу принадлежит объект x
(распределение вероятностей p принадлежности каждому из классов).

Нейросети. Задача классификации. Обозначения

Пусть

- $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ - тренировочная выборка,
- $x_k \in \mathbb{R}^d$ - признаковое описание i -го объекта,
- $y_k \in \{1, 2, \dots, M\}$ - класс i -го объекта,
- $p(x, \theta) = (p_1(x, \theta), p_2(x, \theta), \dots, p_M(x, \theta))$ - нейронная сеть с параметрами θ .

На выходе нейросети - оценка вероятности каждого класса.

Нейросети. Задача классификации. Функция потерь

Хотим, решать задачу классификации. То есть нужно найти такой θ , чтобы выполнялось:

$$p_y(x, \theta) \approx \mathbb{P}(Y = y | X = x), \quad y \in \{1, 2, \dots, M\}.$$

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^N p_{y_i}(x_i, \theta) = \underset{\theta}{\operatorname{argmin}} \underbrace{\left(-\frac{1}{N} \sum_{i=1}^N \ln p_{y_i}(x, \theta) \right)}_{= L(\theta)}$$

Cross-Entropy Loss

Итак, нужно минимизировать $L(\theta) = -\frac{1}{N} \sum_{i=1}^N \ln p_{y_i}(x, \theta)$ по θ .

Градиентный спуск

Случайно инициализируем веса нейросети θ_0 . Затем запускаем итеративный процесс:

$$\theta_k := \theta_{k-1} - \alpha \cdot \frac{\partial L}{\partial \theta}(\theta_{k-1}),$$

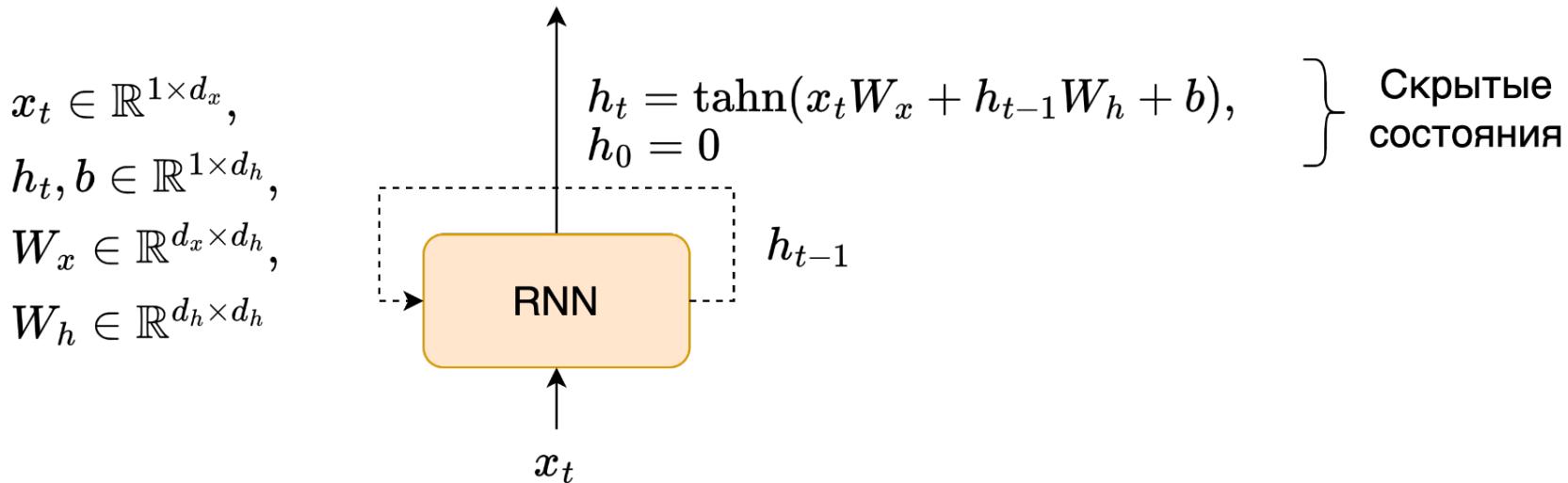
где $\alpha > 0$.

Рекуррентные нейросети



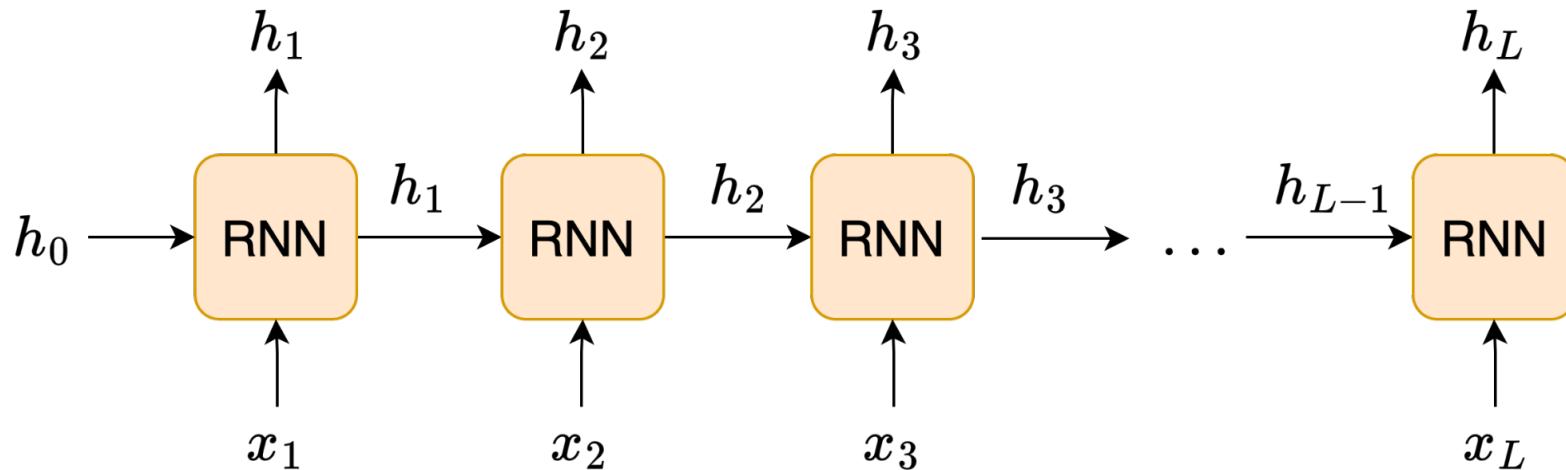
Рекуррентная нейросеть (RNN)

Цель: обработать последовательность x_1, x_2, \dots, x_L
зависимых друг от друга наблюдений **нефиксированной** длины.



- Состояние h_t можно рассматривать как внутреннюю память модели на шаге t .
- Состояние h_t неявно зависит от всех x_1, \dots, x_t , то есть хранит информацию о входах до шага t включительно.

Развернутый вид рекуррентной нейросети



$$x_t \in \mathbb{R}^{1 \times d_x},$$

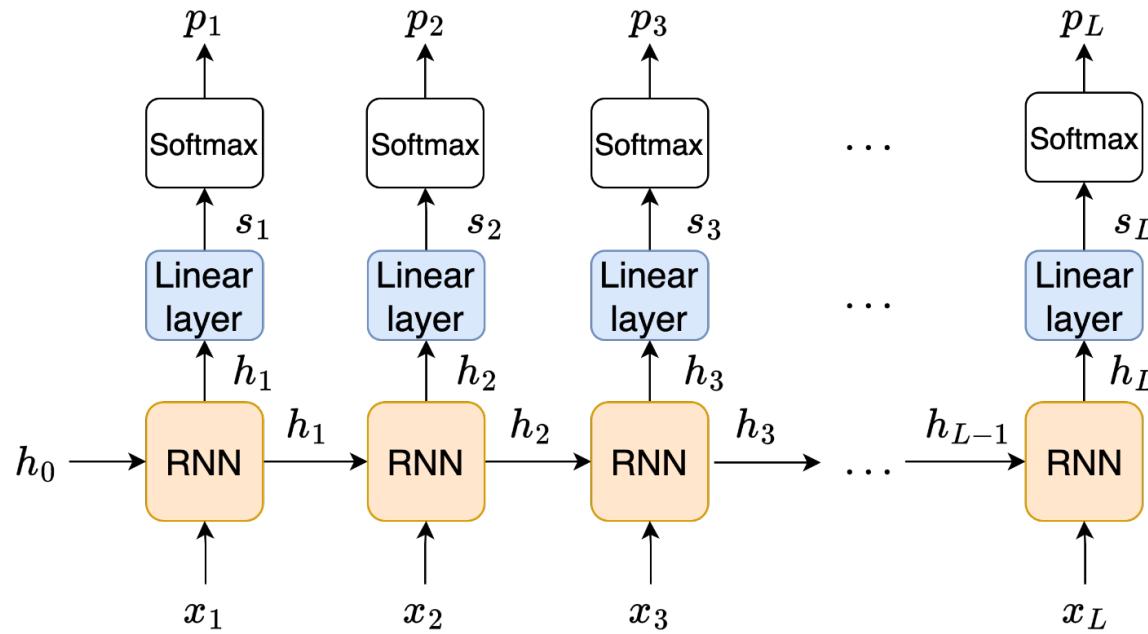
$$h_t, b \in \mathbb{R}^{1 \times d_h},$$

$$W_x \in \mathbb{R}^{d_x \times d_h},$$

$$W_h \in \mathbb{R}^{d_h \times d_h}$$

$$h_t = \tanh(x_t W_x + h_{t-1} W_h + b),$$
$$h_0 = 0$$

Классификация элементов последовательности



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

$$h_0 = 0,$$

$$s_t = h_t W_s + b_s,$$

$$p_t = \text{softmax}(s_t)$$

s_t - логиты на шаге t ,

p_t - предсказанное распределение на шаге t
(M классов)

$$x_t \in \mathbb{R}^{1 \times d_x},$$

$$h_t, b_h \in \mathbb{R}^{1 \times d_h},$$

$$s_t, p_t, b_s \in \mathbb{R}^{1 \times M},$$

$$W_x \in \mathbb{R}^{d_x \times d_h},$$

$$W_h \in \mathbb{R}^{d_h \times d_h},$$

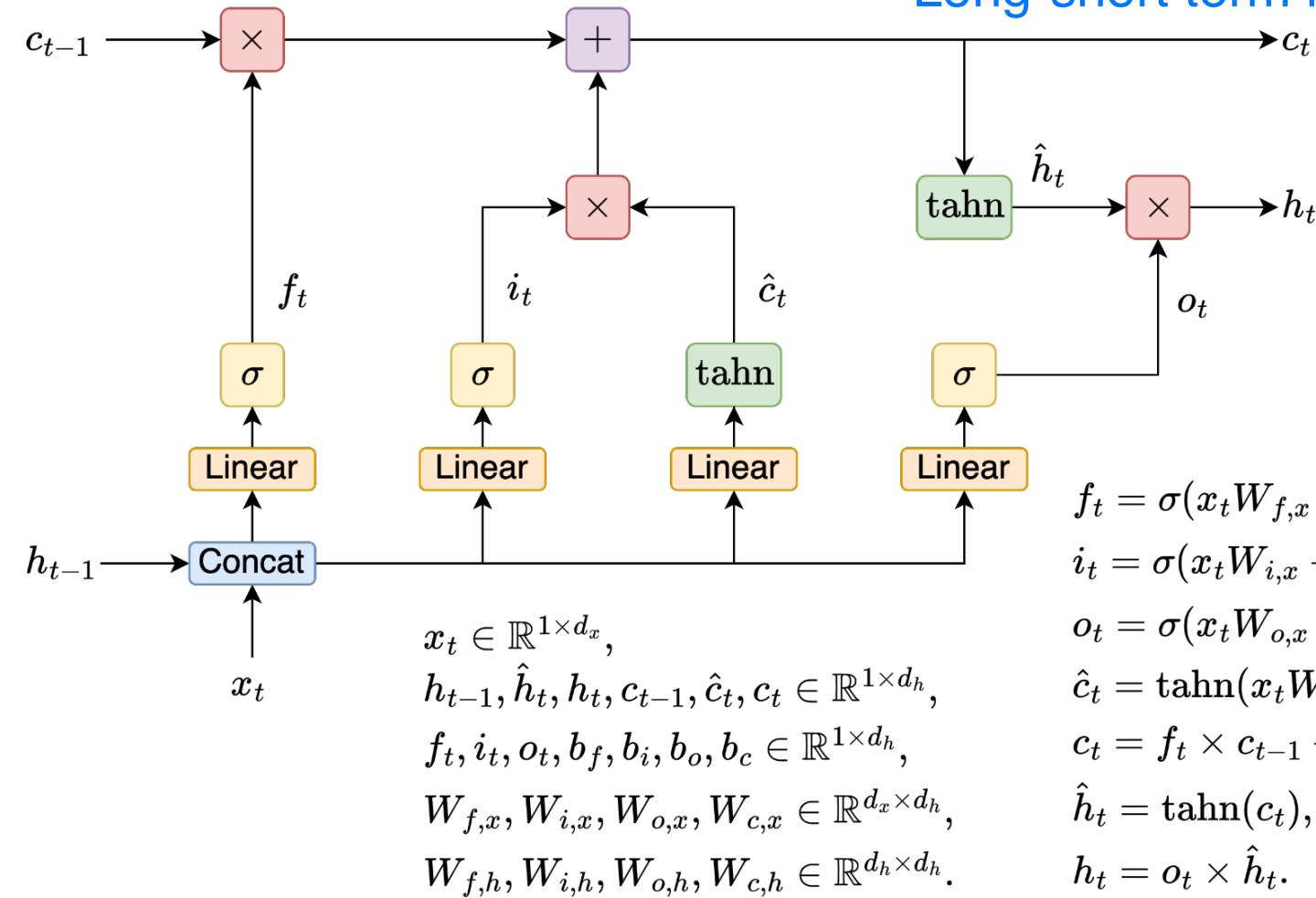
$$W_s \in \mathbb{R}^{d_h \times M}$$

Проблемы классических рекуррентных нейросетей

- **Исчезающие градиенты.** При обратном распространении ошибки через большое количество временных шагов, градиенты могут становиться очень малыми. В результате модель "забывает" информацию из начала последовательности.
- **Взрывающиеся градиенты.** При обратном распространении ошибки через большое количество временных шагов градиенты могут стать чрезвычайно большими. Модель становится нестабильной, обучение нарушается.

Можно несколько модифицировать RNN-ячейку для борьбы с исчезающими градиентами. Рассмотрим, например, LSTM.

Long-short term memory (LSTM)



Языковая модель на основе рекуррентной нейронной сети

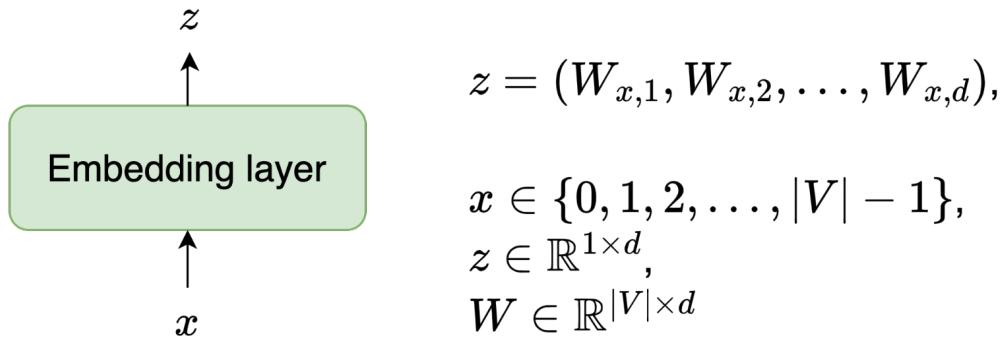


Обозначения

Далее будем работать не с последовательностями **токенов**, а последовательностями **номеров токенов**, то есть будем считать, что $x_0, x_1, \dots \in \{0, 1, \dots, |V| - 1\}$.

Также будем отождествлять токены и номера токенов.

Векторные представления токенов



Пусть x может принимать одно из значений $\{0, 1, \dots, |V| - 1\}$.

Слой эмбеддингов превращает x в вектор.

Слой эмбеддингов каждому токену (номер в словаре) сопоставляет вектор.

Слой эмбеддингов учится вместе с остальными параметрами нейросети.

Векторные представления токенов

Словарь

Токен	ID
мама	0
папа	1
банан	2
машина	3

W

-0.1	0.2	0.7
0.5	-1	1.1
-0.6	0.2	-0.3
0.1	-0.3	1

Какой эмбеддинг у токена "банан"?

Векторные представления токенов

Словарь

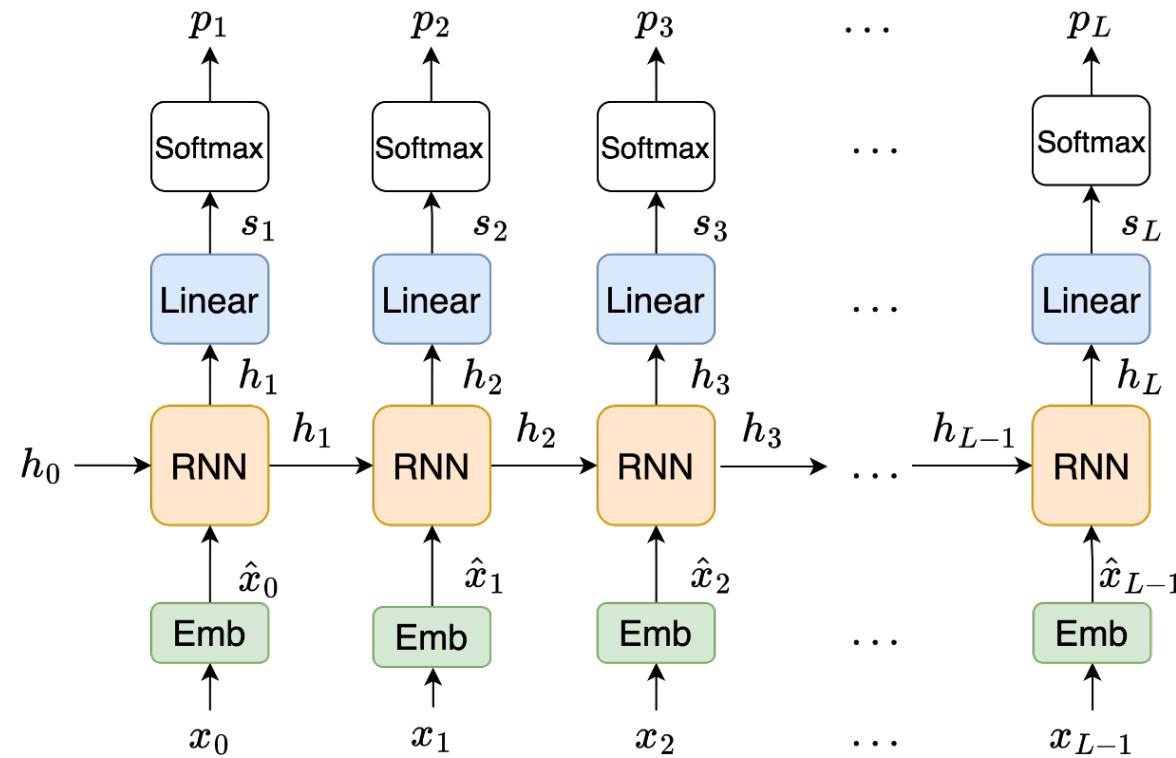
Токен	ID
мама	0
папа	1
банан	2
машина	3

W

-0.1	0.2	0.7
0.5	-1	1.1
-0.6	0.2	-0.3
0.1	-0.3	1

Ответ: (-0.6, 0.2, -0.3)

Языковая модель на основе RNN



Хотим, чтобы вектор p_k хорошо оценивал распределение следующего токена:

$$p_{k,i} \approx \mathbb{P}(X_k = i | x_0, x_1, \dots, x_{k-1}),$$

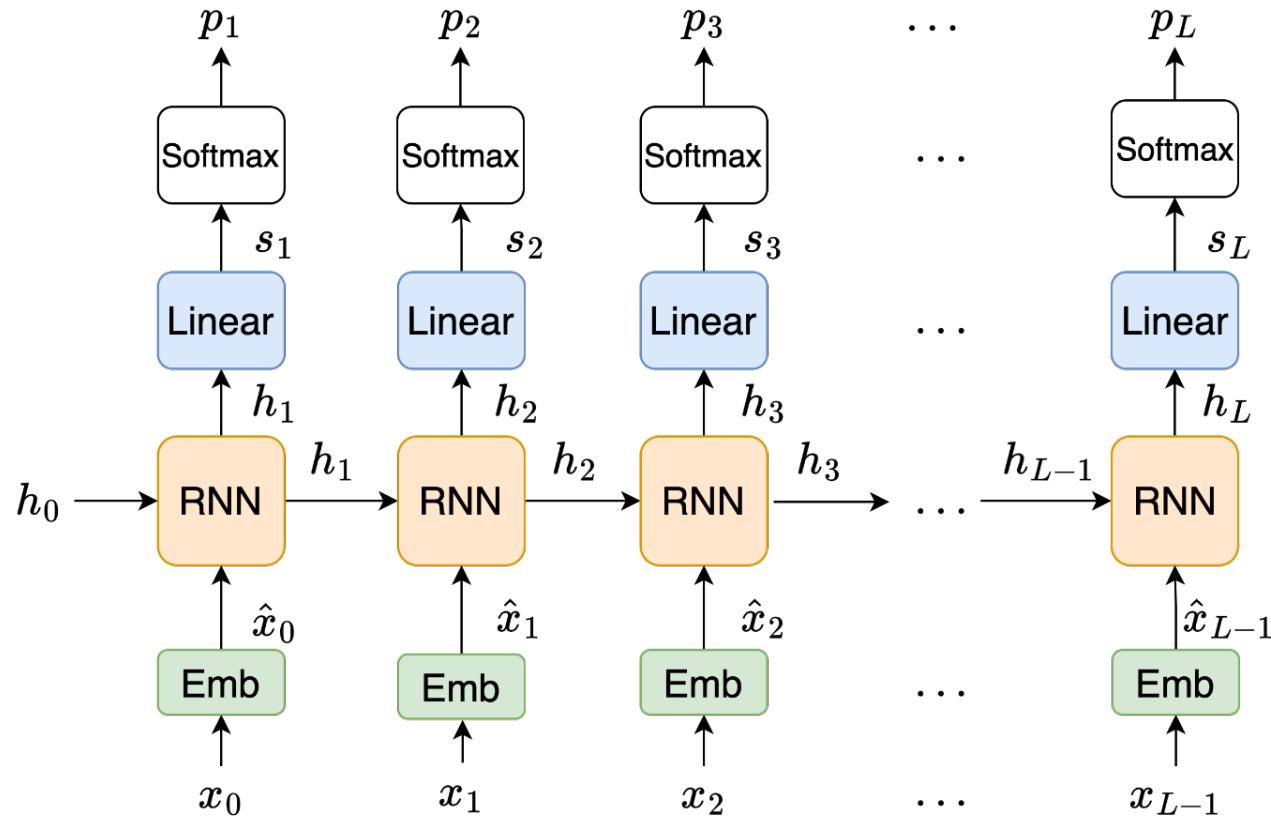
где $i \in \{0, 1, \dots, |V| - 1\}$.

Для этого будем учить модель решать **задачу потокенной классификации**: предсказание следующего токена в последовательности.

Ошибка на одной последовательности (Cross-entropy loss):

$$\ell = -\frac{1}{L} \sum_{i=1}^L \ln p_{i,x_i}.$$

Генерация с температурой



Каждый раз семплируем токен из модифицированного распределения:

$$\hat{p}_k = \text{softmax}(s_k/t), \quad t > 0.$$

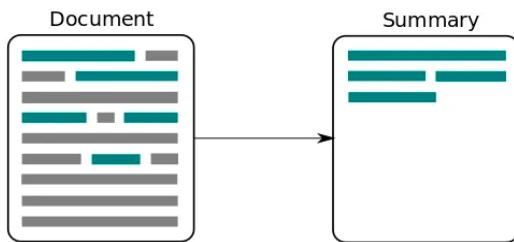
Варианты:

- Случайная генерация: $t = 1$.
- Жадная генерация: $t \rightarrow 0$.
- Генерация из равномерного распределения: $t \rightarrow +\infty$.

Задача sequence to sequence



Задача sequence to sequence (seq2seq)



По входной последовательности токенов нужно сгенерировать другую последовательность токенов (возможно, из другого словаря и другой длины)

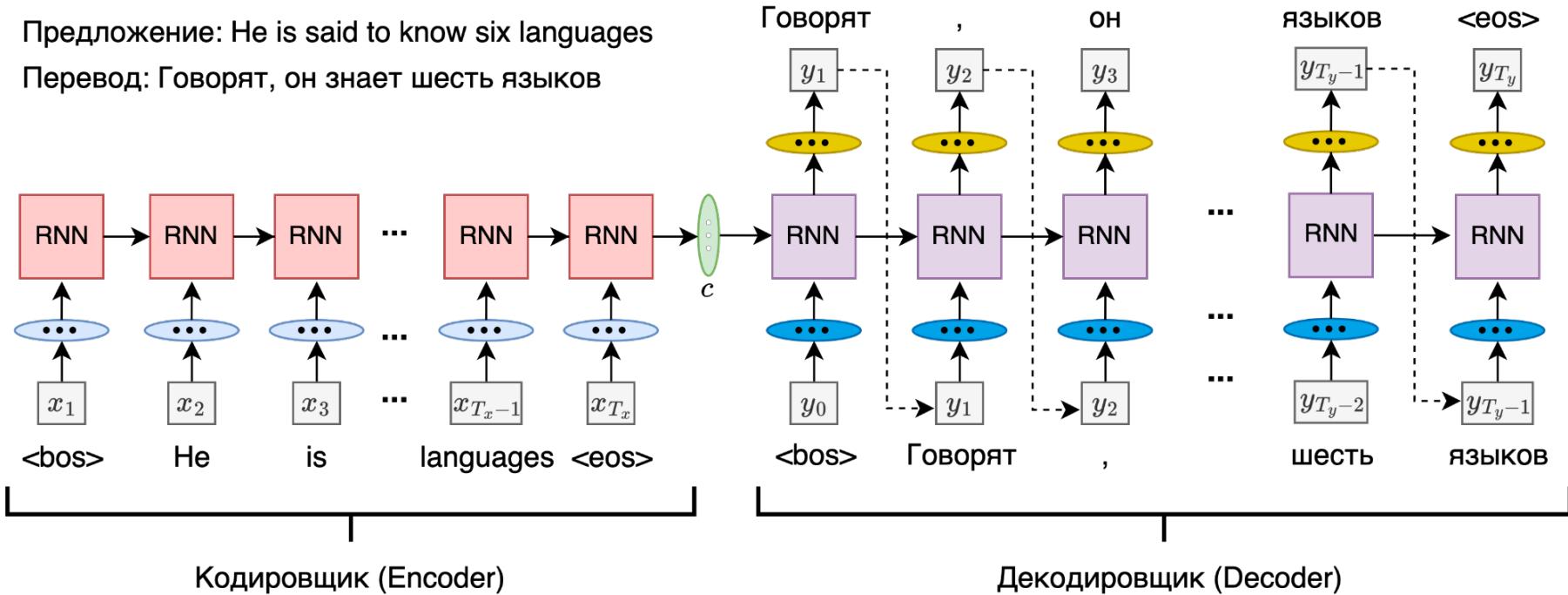
Примеры:

- Машинный перевод
- Суммаризация текста
- Генерация ответа на вопрос

Модель sequence-to-sequence на основе RNN

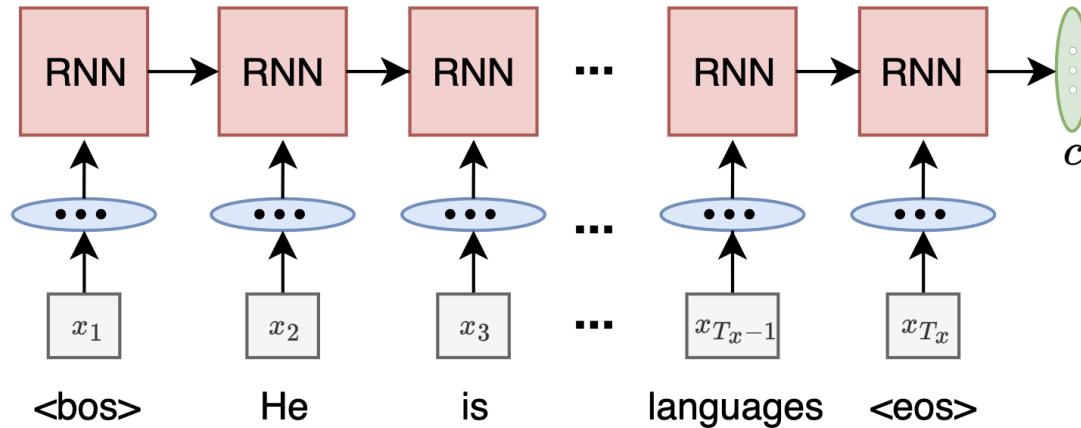
Предложение: He is said to know six languages

Перевод: Говорят, он знает шесть языков



Модель состоит из двух частей: кодировщик (encoder) и декодировщик (decoder).

Кодировщик



Эмбеддинг слова



Вектор контекста

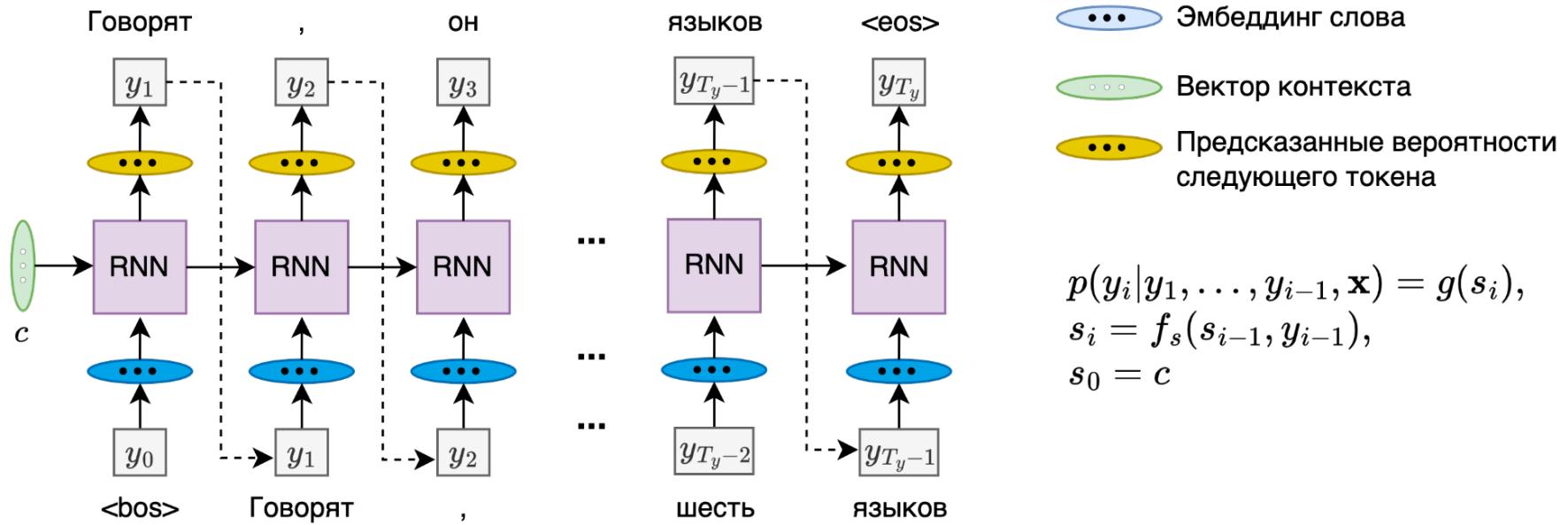
$$c = h_{T_x},$$

$$h_i = f_h(h_{i-1}, x_i),$$

$$h_0 = 0$$

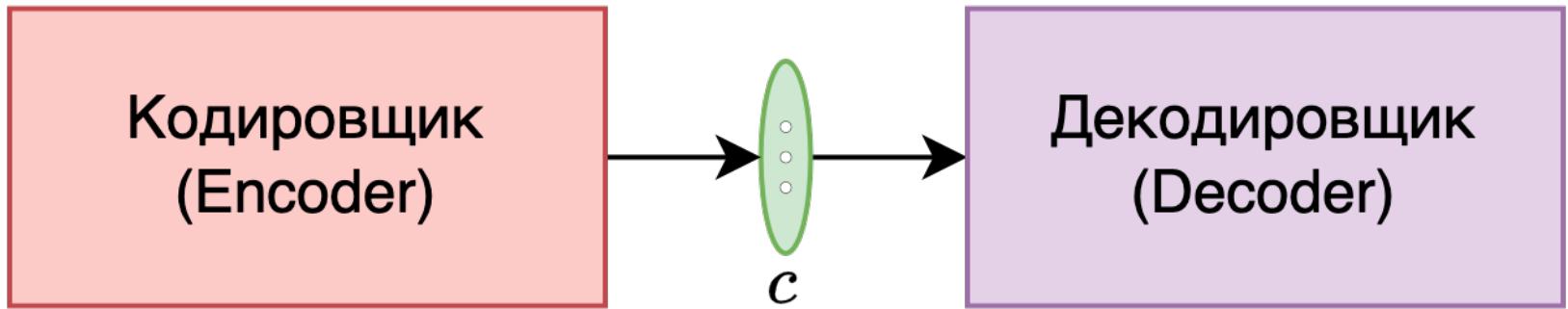
Кодировщик читает входную последовательность и формирует ее представление в виде вектора – финального скрытого состояния (hidden state).

Декодировщик



Декодировщик принимает полученное представление последовательности в виде своего начального скрытого состояния и генерирует выходную последовательность.

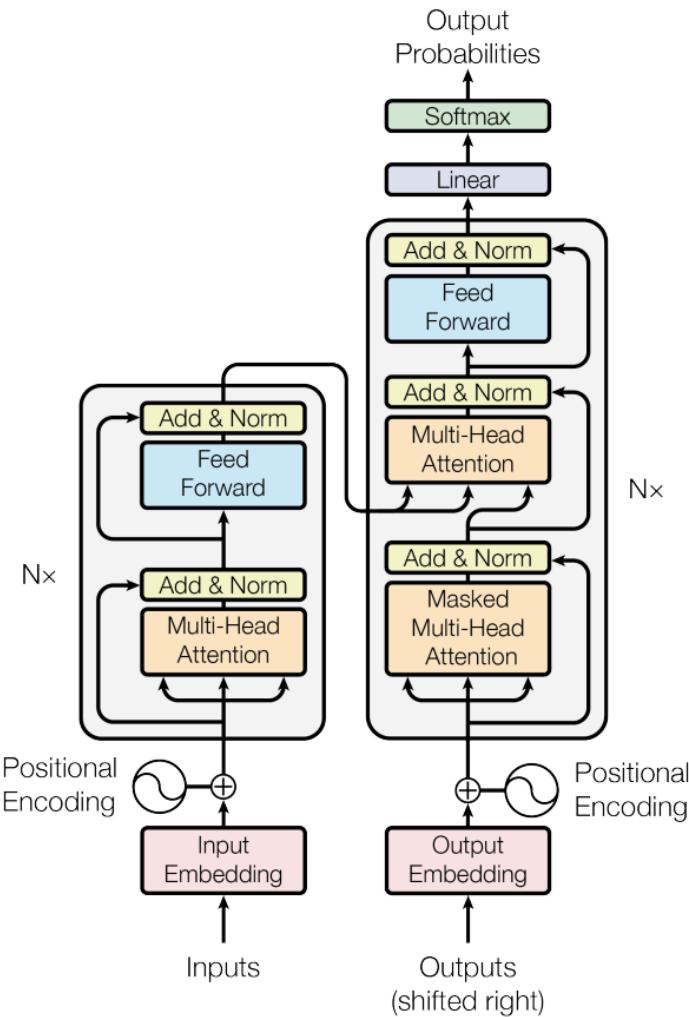
Узкое место подхода



Проблемы:

- Узкое место (bottleneck) в виде вектора контекста между encoder и decoder. Для длинных последовательностей теряется слишком много информации при попытке сжать ее в вектор.
- Плохо параллелируется при обучении и применении.

Transformer



Архитектура, пришедшая на смену RNN:

- Хорошо параллелится
- Внутри нет рекуррентности и связанной с ней проблемы.
- Имеем доступ ко всей информации (нет узкого места, как в RNN)

Что сегодня изучили?

- Токенизация
- Вероятностная модель текстов
- Задача языкового моделирования
- Простейшая языковая модель (цепь Маркова на n-граммах)
- Способы генерации текстов
- Нейронные сети
- Рекуррентные нейронные сети (RNN)
- Языковая модель на основе RNN
- Задача sequence to sequence
- Модель кодировщик-декодировщик

Что еще почитать?

- [Probabilistic Language Models](#)
- [N-gram Language Models](#)
- [Understanding LSTM Networks](#)
- [A Guide to Controlling LLM Model Output: Exploring Top-k, Top-p, and Temperature Parameters](#)

Спасибо за внимание!

Владимир Макаренко,
ведущий разработчик-исследователь

