Введение в АД

Введение в NLP

План лекции

NLP

Кодирование текстов

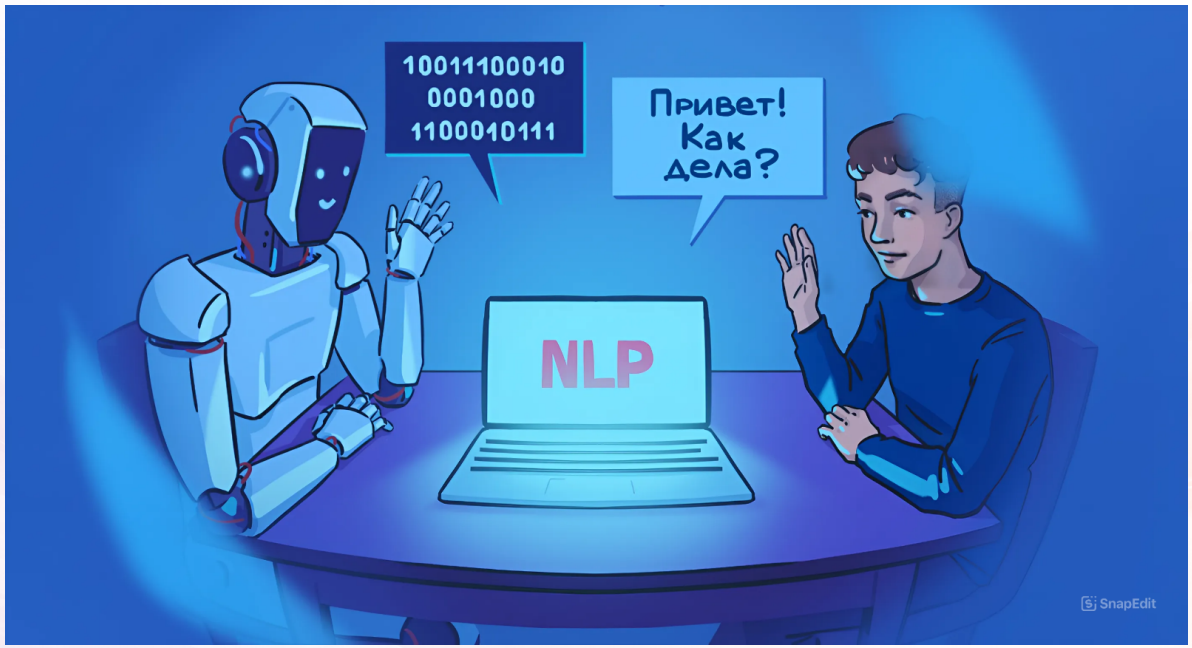
Основные модели

LLM

Natural Language Processing (NLP)

Natural Language Processing (NLP) – область машинного обучения, изучающая технологии обработки естественного языка.

Современные NLP-методы применяются и в мультимодальных системах, позволяя также работать со звуком, картинками и видео.



Пример решения задачи классификации текста

По комментарию в интернете надо определить его токсичность

Комментарий / **Категория**	*"Тебя следует уволить, ты слишком ленивый, чтобы провести исследование"*	*О, я не знал, спасибо.*
 Токсичный	✓	✗
 Нецензурный	✗	✗
 Угроза	✗	✗
 Оскорбление	✓	✗

Пример решения задачи Image Captioning

Мужчина сидит и улыбается,
но грустит

Два человека в костюмах смотрят
друг на друга

Женщина кричит, а кот сидит
за столом

Человек держит напиток,
он улыбается

План лекции

NLP

Кодирование текстов

Основные модели

LLM

Кодирование текстов

Пример задачи:

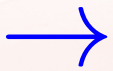
По некоторому набору характеристик отеля определить, сколько у него звезд (5 классов).

Проблема:

Пусть среди признаков есть текстовое описание отеля, которое содержит много информации.

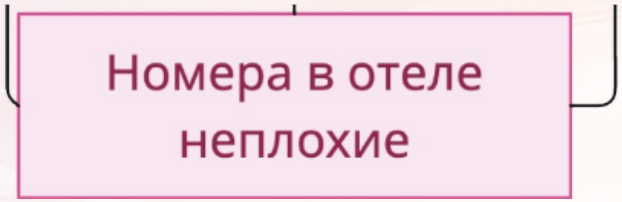
Как использовать текстовое описание в модели?

«Отель находится в центре города, все чисто, персонал дружелюбный, но номера немного староваты.»

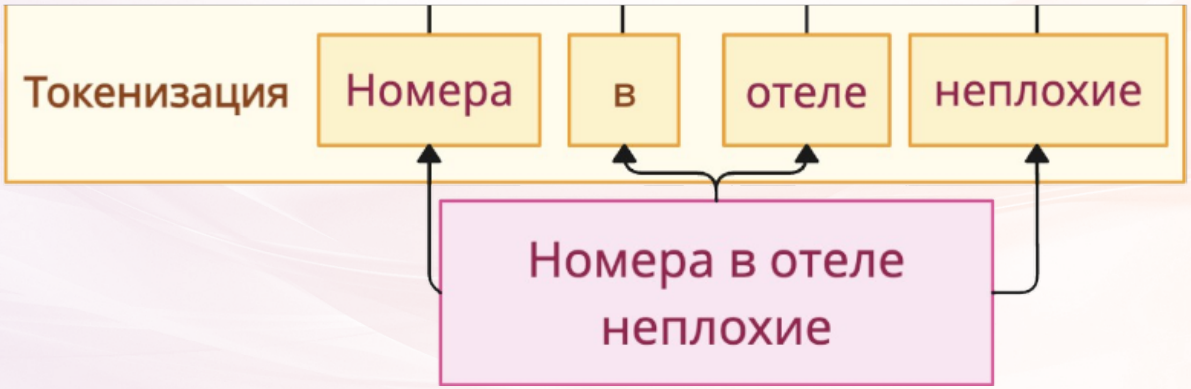


Оценка отзыва

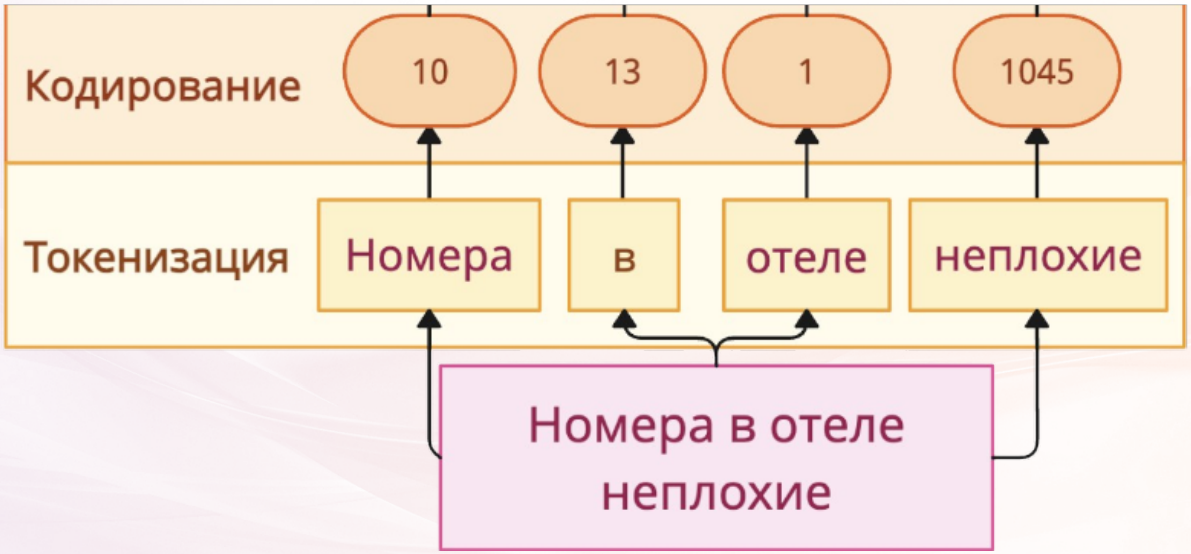
Кодирование текстов



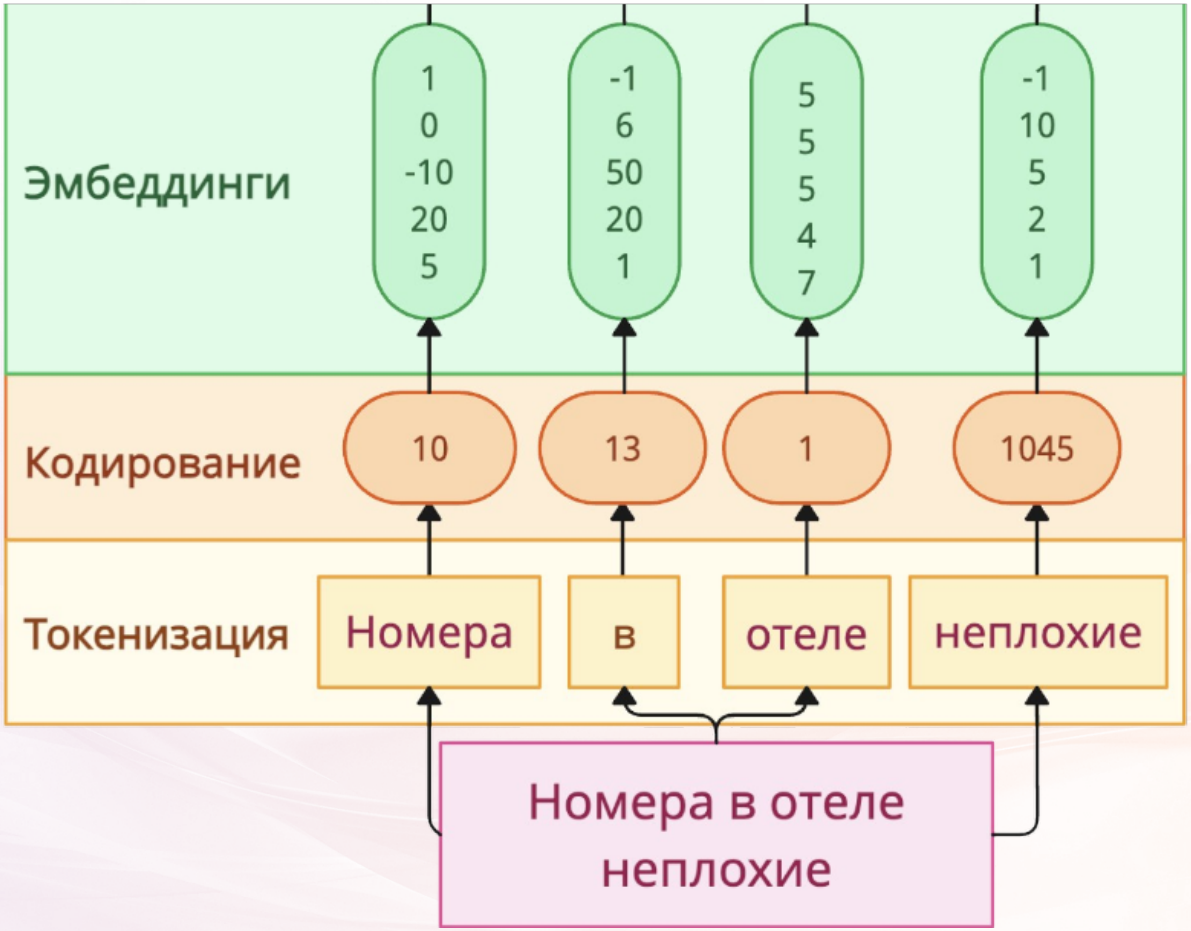
Кодирование текстов



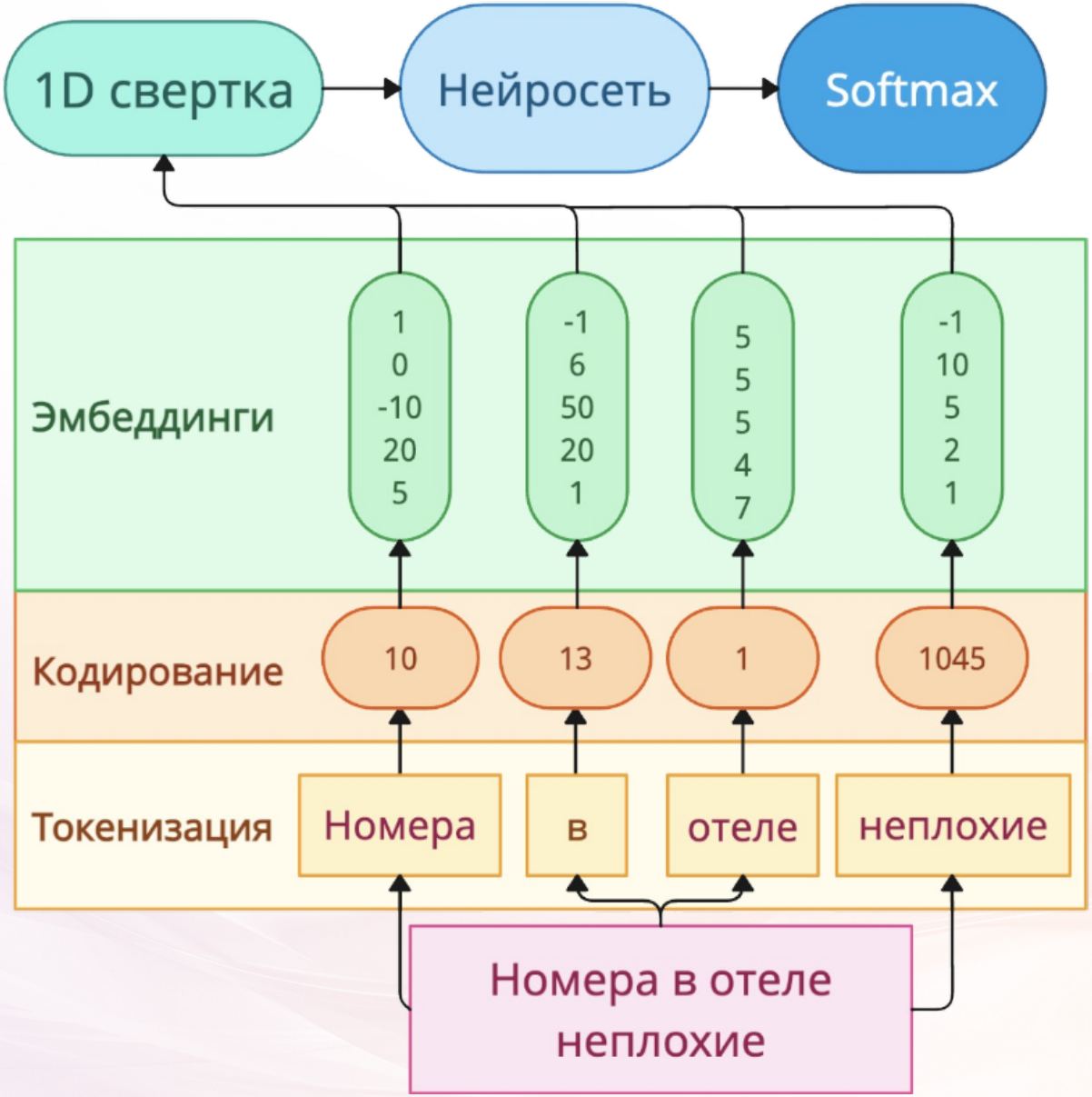
Кодирование текстов



Кодирование текстов



Кодирование текстов



Эмбеддинги слов

Постановка задачи

Пусть $V = \{w_j\}_{j=1}^{M}$ – словарь из всех слов.

Необходимо построить отображение $f\colon V \to \mathbb{R}^d$,

где d – гиперпараметр.

Вектор $f(w) \in \mathbb{R}^d$ называется **эмбеддингом** слова w.

Требования:

- $f(w_1) = f(w_2) \implies w_1 = w_2$
- $f(w_1) \neq f(w_2) \implies w_1 \neq w_2$
- $f(w_1) \approx f(w_2)$, если w_1 и w_2 похожи по смыслу
- $f(w_1) \not\approx f(w_2)$, если w_1 и w_2 не похожи по смыслу

1. One-hot векторы

Пусть $V = \{w_j\}_{j=1}^{M}$ – словарь из всех слов.
Каждому слову поставим в соответствие вектор из 0 и 1 размера $|V|$, где 1 стоит на позиции j для слова w_j.

Пример:

$\texttt{motel} = (0, ..., 0, 1, 0, ..., 0)$

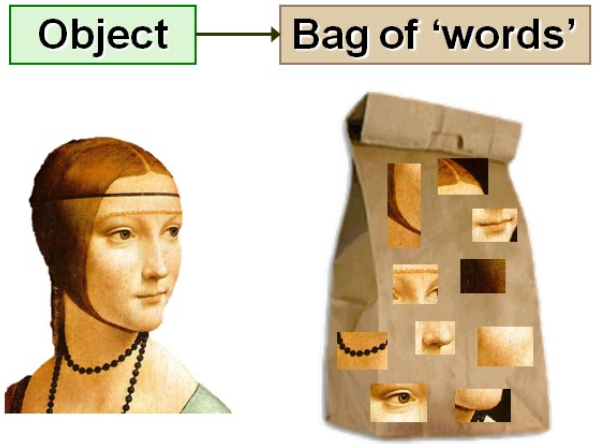
$\texttt{hotel} = (0, ..., 0, 0, 1, ..., 0)$

Минусы:

- Большой размер векторов.
- Векторы не содержат информацию о значении слов.
- Все векторы ортогональны, включая векторы схожих по смыслу слов, то есть нет понятия сходства векторов.

2. Bag of Words (Мешок слов)

Bag of Words (BOW) — метод построения эмбеддинга для всего предложения. Эмбеддинг текста — сумма one-hot векторов входящих в него слов.



Пример (анализ новостей):

$V = \{$`экономика, кризис, рост, инфляция, рынок, инвесторы`$\}$

1. «`Экономика демонстрирует рост`» = (1, 0, 1, 0, 0, 0)
2. «`Инфляция растет, рынок падает`» = (0, 0, 0, 1, 1, 0)
3. «`Рынок нестабилен из-за кризиса`» = (0, 1, 0, 0, 1, 0)

2. Bag of Words

Плюсы:

- Предложения, содержащие одинаковые слова, имеют схожие вектора

Минусы:

- Большой размер векторов.
- Схожие по смыслу слова рассматриваются как разные.
- Векторы очень разрежены (sparse).

Этим методом можно строить эмбеддинг слова, рассматривая его как набор букв, но такие векторы не очень информативны.

Дистрибутивная гипотеза

Что означает слово **пакс**?

Примеры использования:

- Ведаю правду, один и тот же пакс едет сегодня на Лидере, завтра на 306, послезавтра на Яндекс... и так по кругу
- Это был очень долгий и очень скучный рейс с идеальными паксами
- Это все городская легенда, что после 'кола' система не сведет больше с этим паксом
- Некоторые из паксов достали мобильники и стали снимать весь процесс
- Компания экономит на незадачливых паксах?

Можно понять, что **пакс** — разговорное название пассажира.

примеры взяты из @UznalSlovo

Дистрибутивная гипотеза

«Скажи мне, кто твой друг, и скажу, кто ты»
(с) Джейсон Стэтхэм

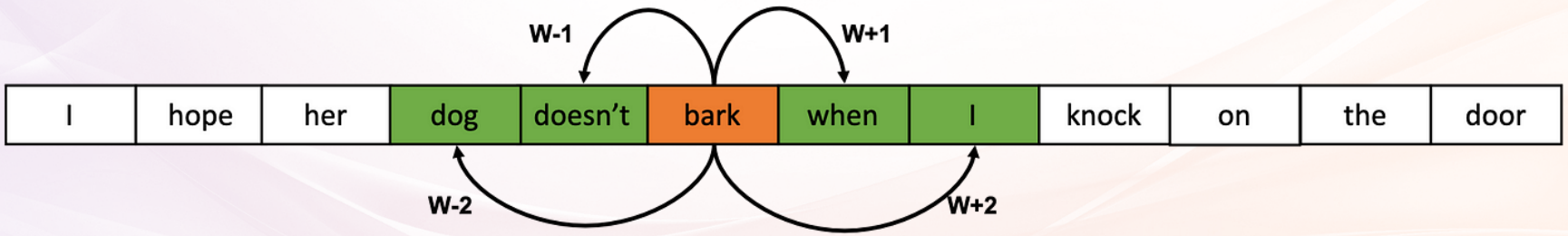
Дистрибутивная гипотеза:

«Слова, встречающиеся в схожих контекстах, похожи по смыслу.»

3. Word2Vec

Идея: Давайте обучим модель из слова получать вектор!
Из самого слова извлечь информацию сложно, возьмём *контекст*.
Под контекстом понимается окно некоторого фиксированного размера вокруг слова. Порядок при этом не учитывается.

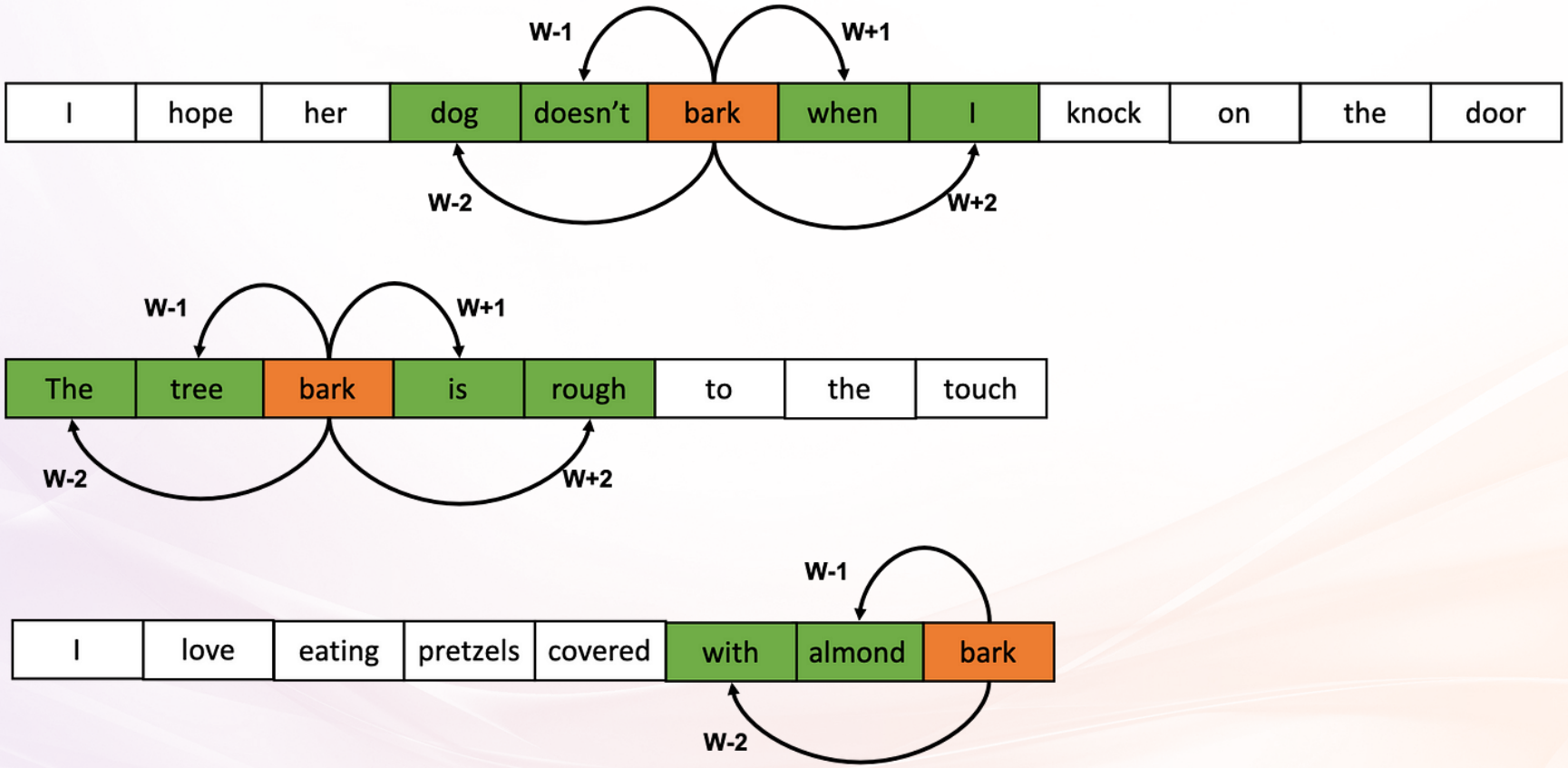
Пусть $P_c(w)$ — вероятность встретить слово w в контексте центрального слова c.



Будем строить нейросеть для предсказания вероятности $P_c(w)$.

3. Word2Vec

Различные варианты



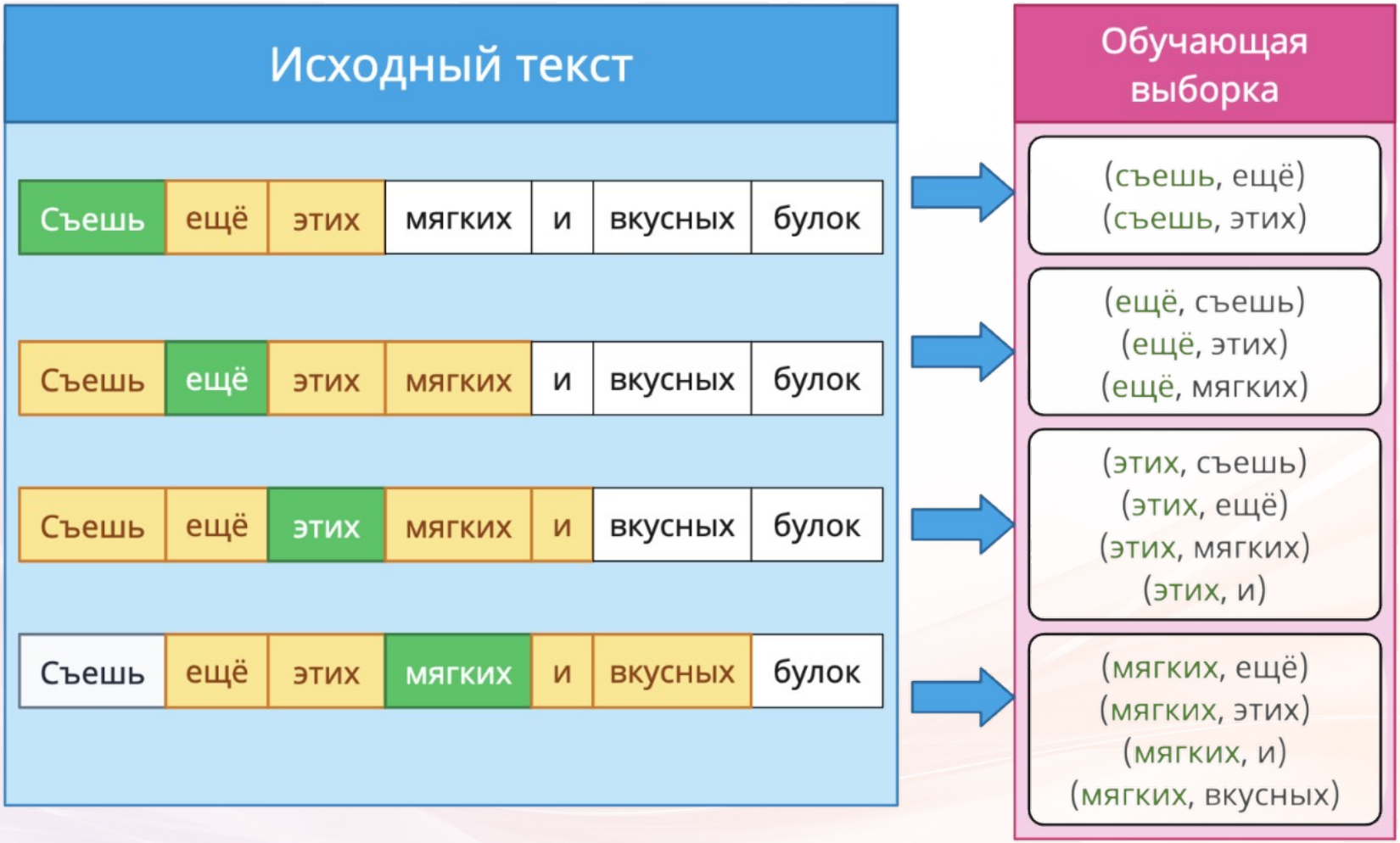
3. Word2Vec

Идея: получить числовые векторы так, чтобы семантически близкие слова имели схожие векторные представления.

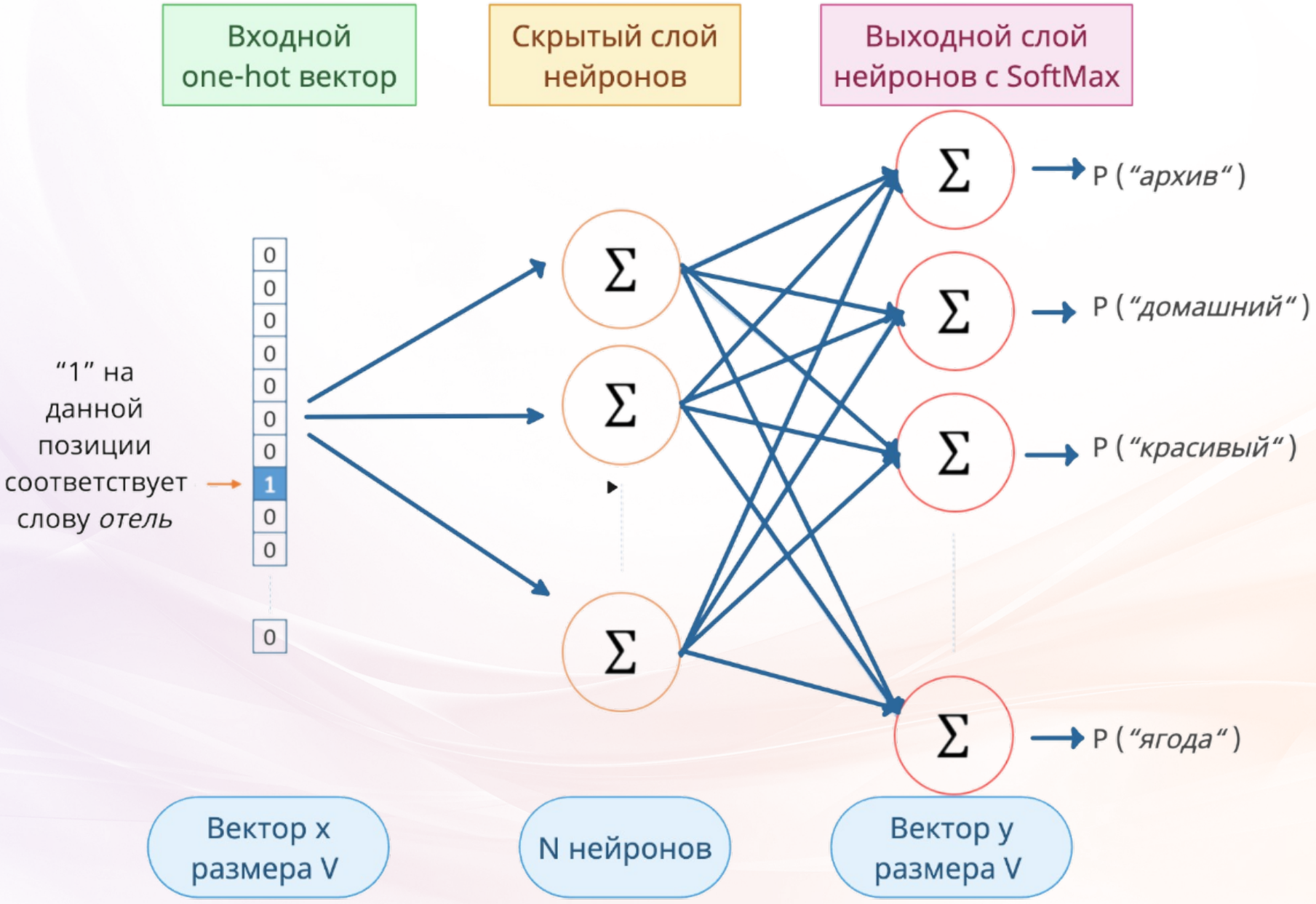
Модель: двухслойная нейросеть без нелинейностей,
с функцией активации softmax на выходе.

- **Вход:** one-hot вектор центрального слова c.
- **Выход:** распределение вероятностей по словарю V,
  где каждая вероятность соответствует тому,
  что слово $w \in V$ находится в контексте центрального c.
- **Таргет:** one-hot вектор слова w из контекста центрального c.

3. Word2Vec


Исходный текст
Съешь ещё этих мягких и вкусных булок
Съешь ещё этих мягких и вкусных булок
Съешь ещё этих мягких и вкусных булок
Съешь ещё этих мягких и вкусных булок
Обучающая выборка
(съешь, ещё)
(съешь, этих)
(ещё, съешь)
(ещё, этих)
(ещё, мягких)
(этих, съешь)
(этих, ещё)
(этих, мягких)
(этих, и)
(мягких, ещё)
(мягких, этих)
(мягких, и)
(мягких, вкусных)

3. Word2Vec


Входной one-hot вектор
Скрытый слой нейронов
Выходной слой нейронов с SoftMax
“1” на данной позиции соответствует слову *отель*
0
0
0
0
0
0
0
1
0
0
0
Σ
Σ
Σ
Σ
Σ
Σ
Σ
P (“*архив*“)
P (“*домашний*“)
P (“*красивый*“)
P (“*ягода*“)
Вектор x размера V
N нейронов
Вектор y размера V

3. Word2Vec

В чистом виде описанная архитектура редко используется,
так как она имеет несколько ограничений.

Она принимает для обучения объекты вида:
<центральное слово, слово из его контекста>.
Слов из контекста много $\Rightarrow$ высокая вычислительная сложность.

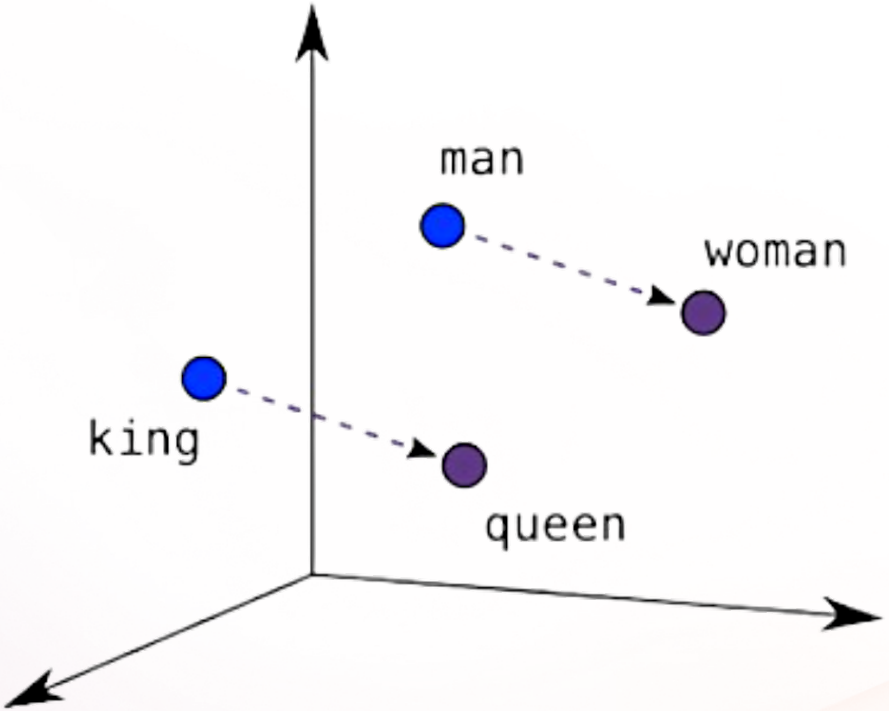
Классическая модель Word2Vec предлагает два подхода:

- **CBOW (Continuous Bag of Words)**: Контекст $\rightarrow$ Центр. слово
- **Skip-gram**: Центр. слово $\rightarrow$ Контекст

Оба метода решают задачу векторного представления слов,
но оптимизированы под разные сценарии.

Про другие особенности Word2Vec вы узнаете на DS-потоке.

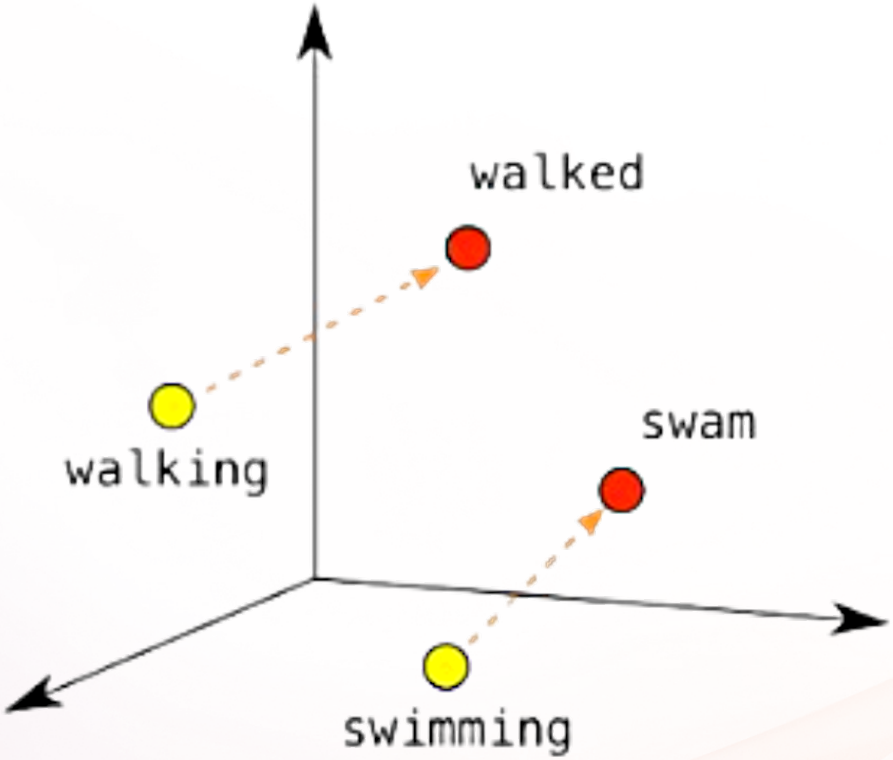
3. Word2Vec



Пример векторной арифметики слов в двумерном пространстве:

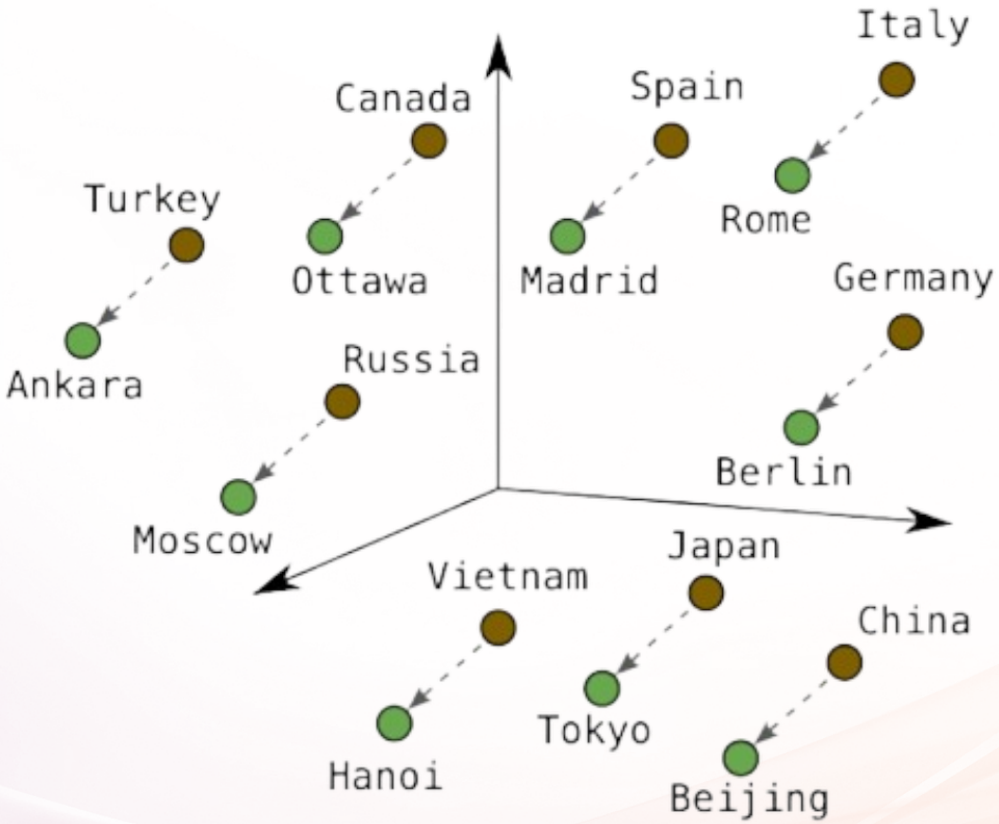
$$\texttt{king} - \texttt{man} + \texttt{woman} \approx \texttt{queen}$$

3. Word2Vec



Векторная арифметика слов на примере глаголов.

3. Word2Vec



Векторная арифметика на примере стран и их столиц.

План лекции

NLP

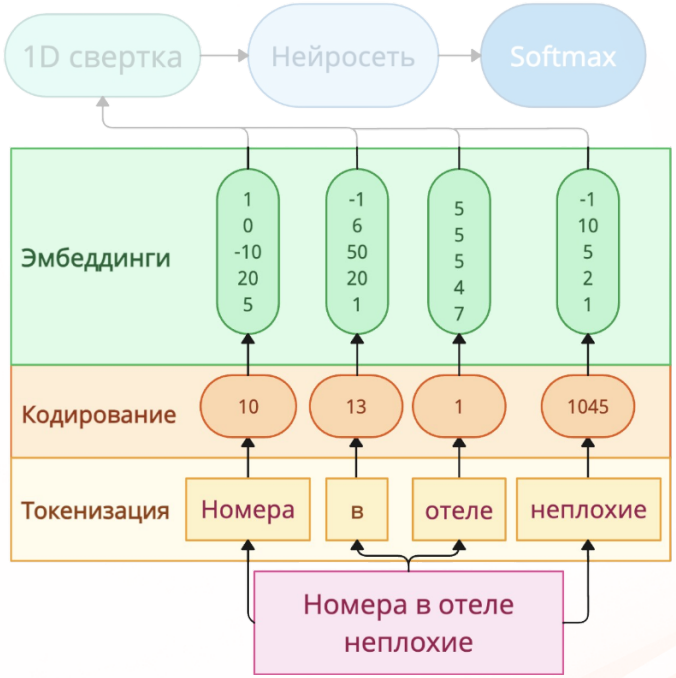
Кодирование текстов

Основные модели

LLM

Основные модели

Научились строить векторные представления слов и текстов — из слова получать его числовое представление, которое можно использовать в существующих моделях машинного обучения



Теперь можем приступать к самому интересному
Обучим нейросеть решать задачи!

Проблема:

полносвязная нейронная сеть не учитывает природу текста — работает с ним как с набором векторов, а не как с посл-тью.

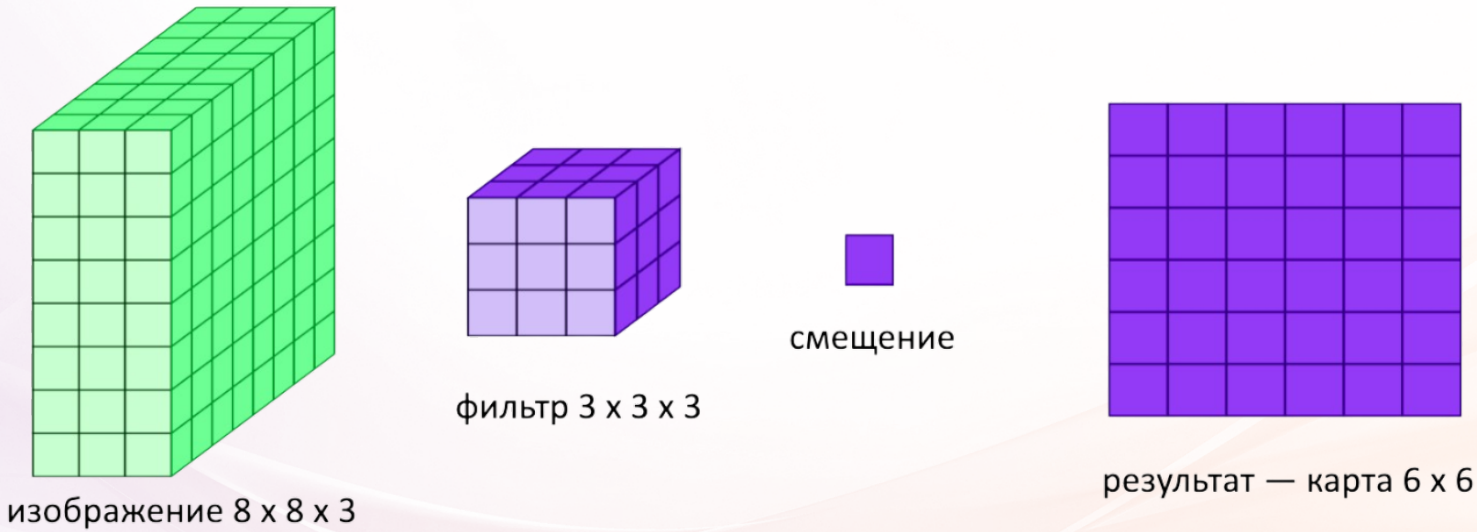
1D-свертка

На вход приходит некоторая посл-ть из объектов $\{A_1, \ldots, A_n\}$.

Каждый объект представляется вектором его признаков.

Запишем объекты в матрицу $A \in \mathbb{R}^{n \times m}$, где m — размер эмбеддинга.

Напоминание: 2D-свертка

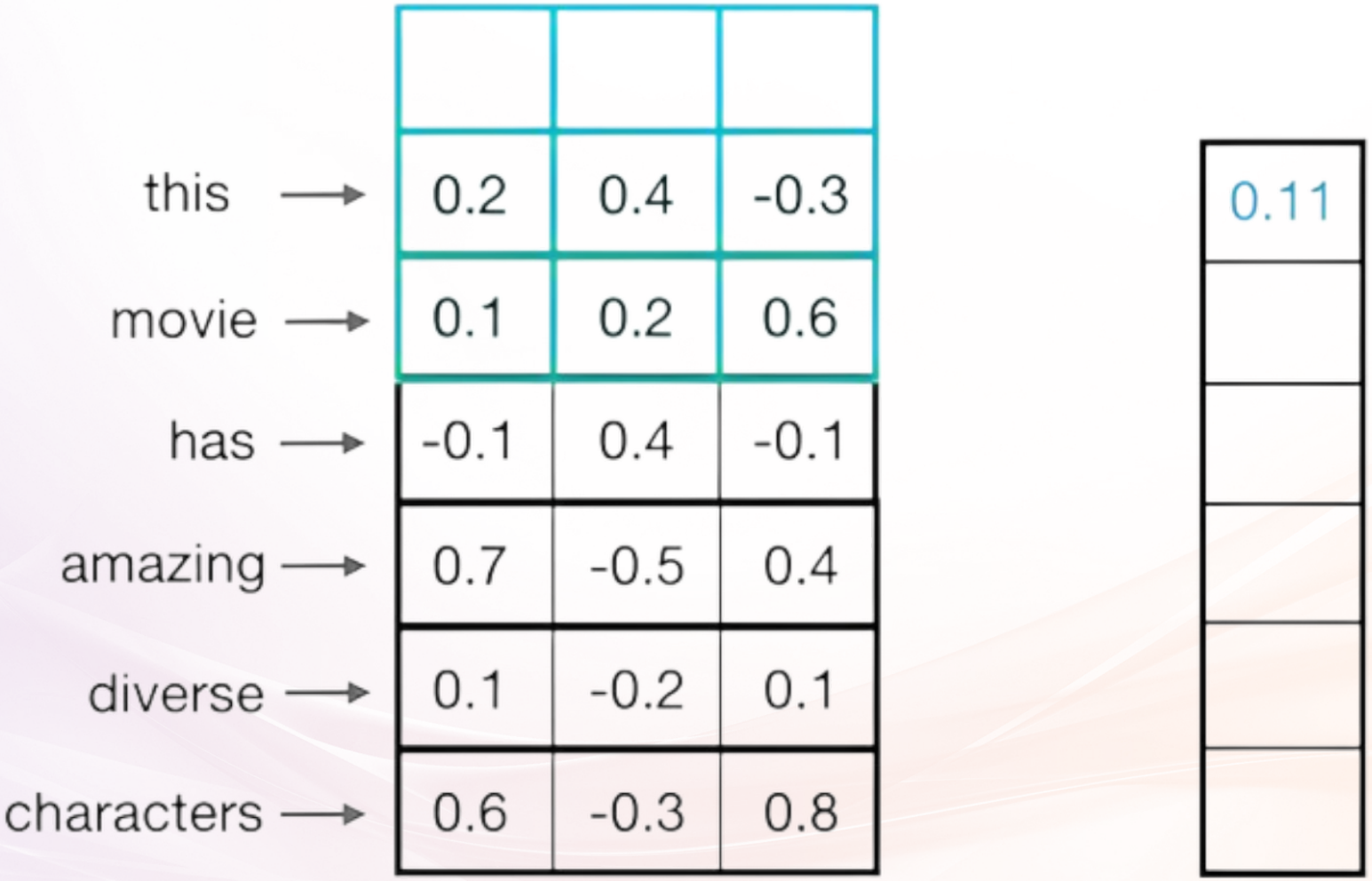


В отличие от картинок в матрице A числа по горизонтали не взаимосвязаны, однако по вертикали — взаимосвязаны.

Поэтому для A не стоит применять делать 2D-свертку.

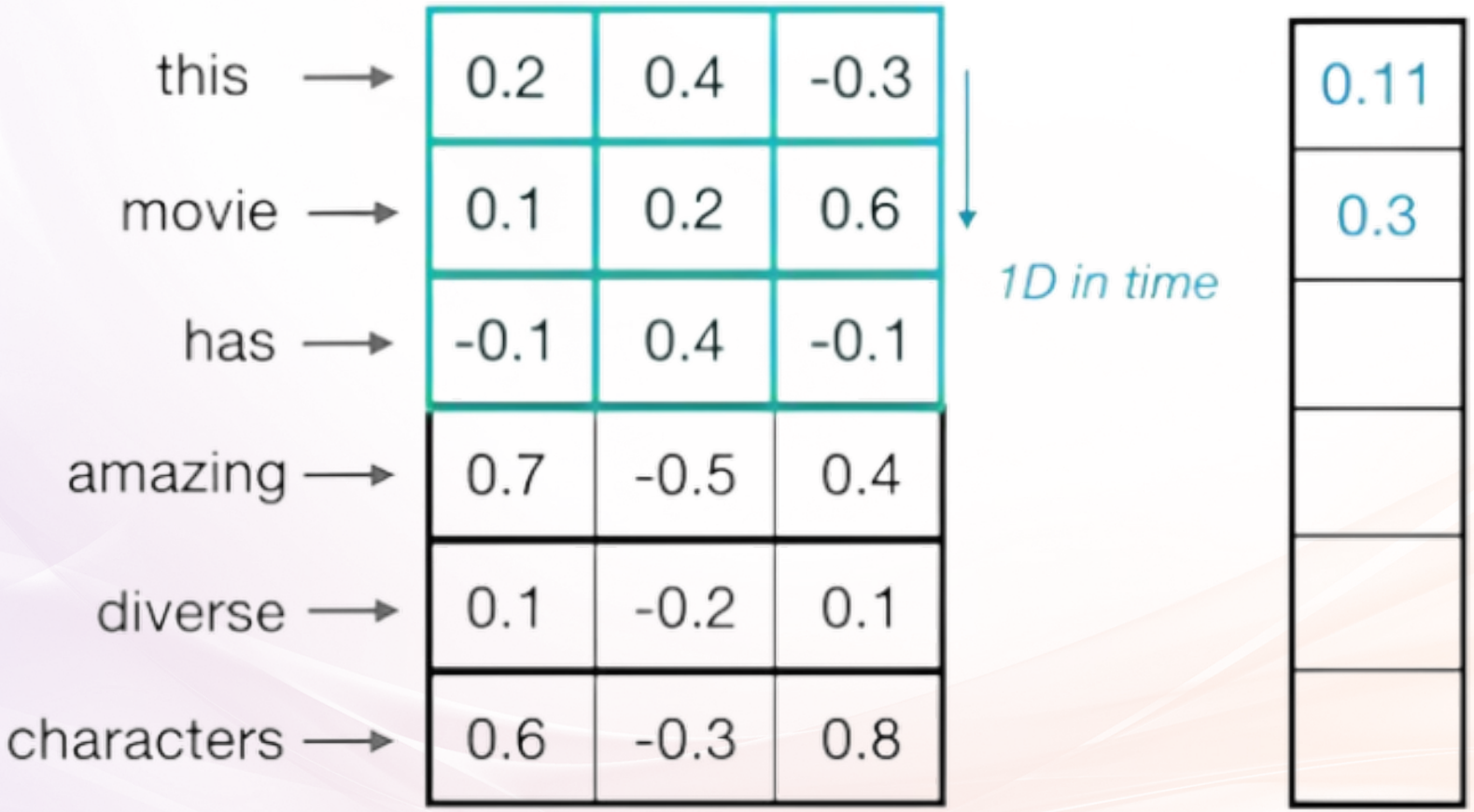
1D-свертка

В 1D-свертке фильтр перемещается только в одном направлении.



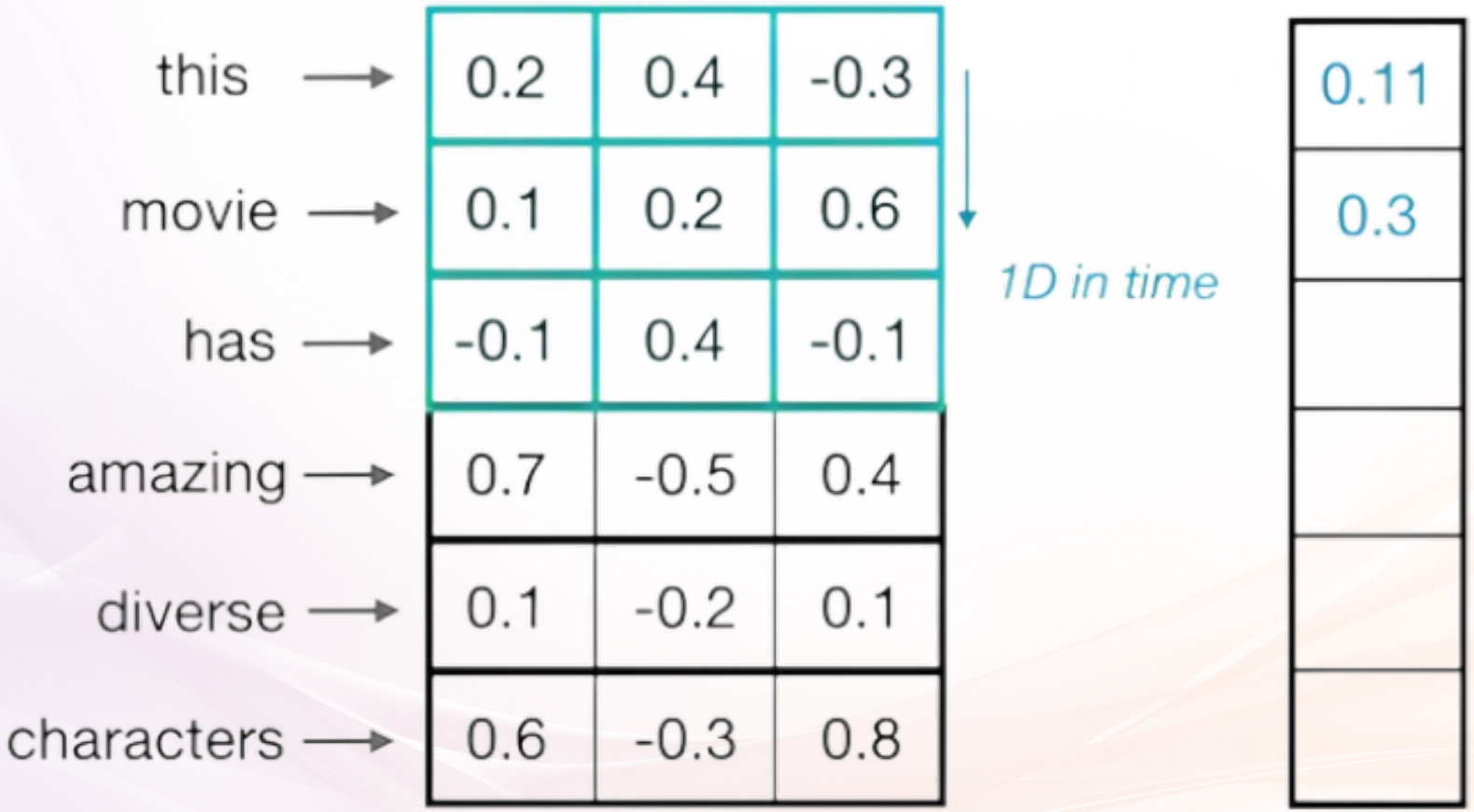
1D-свертка

В 1D-свертке фильтр перемещается только в одном направлении.



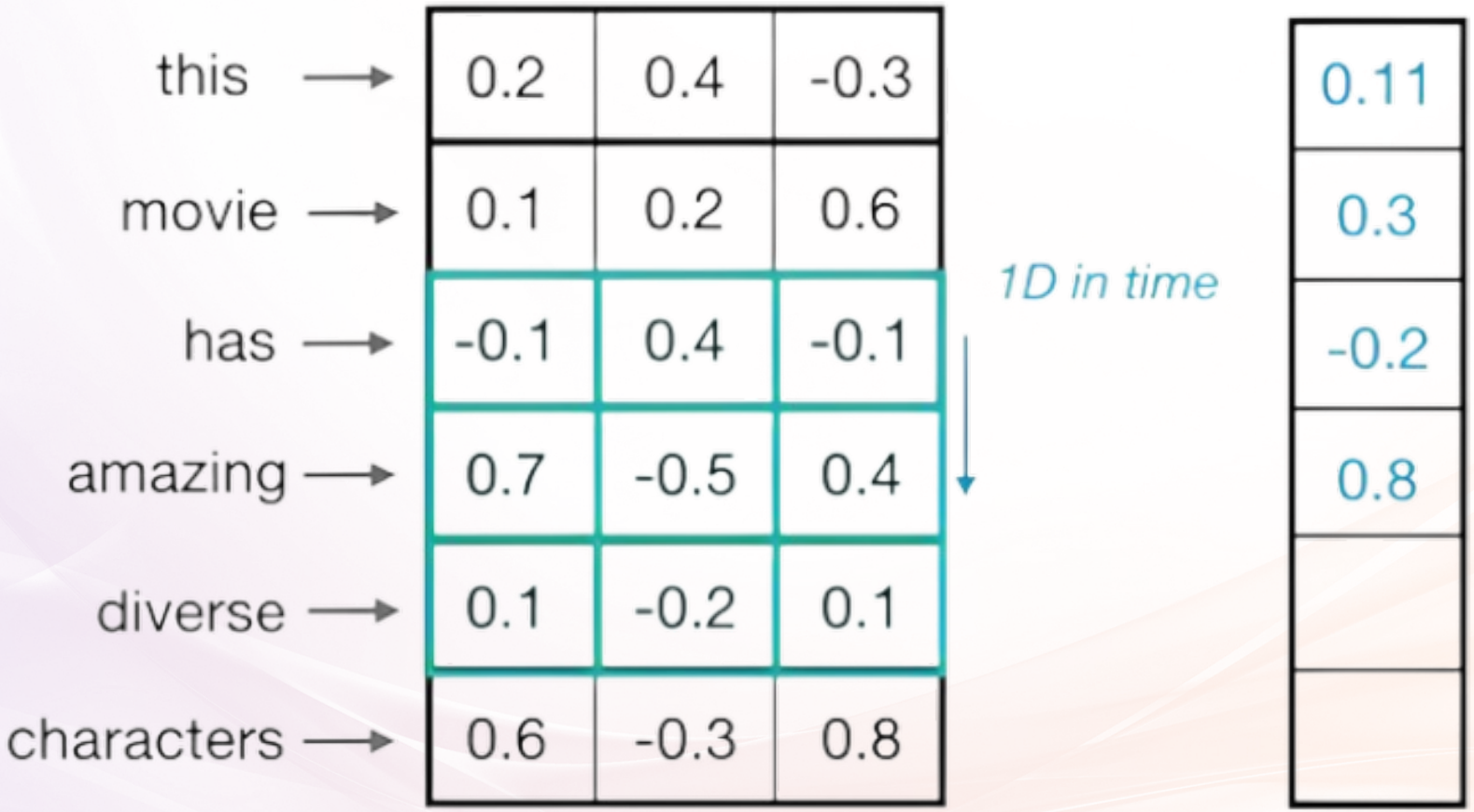
1D-свертка

В 1D-свертке фильтр перемещается только в одном направлении.



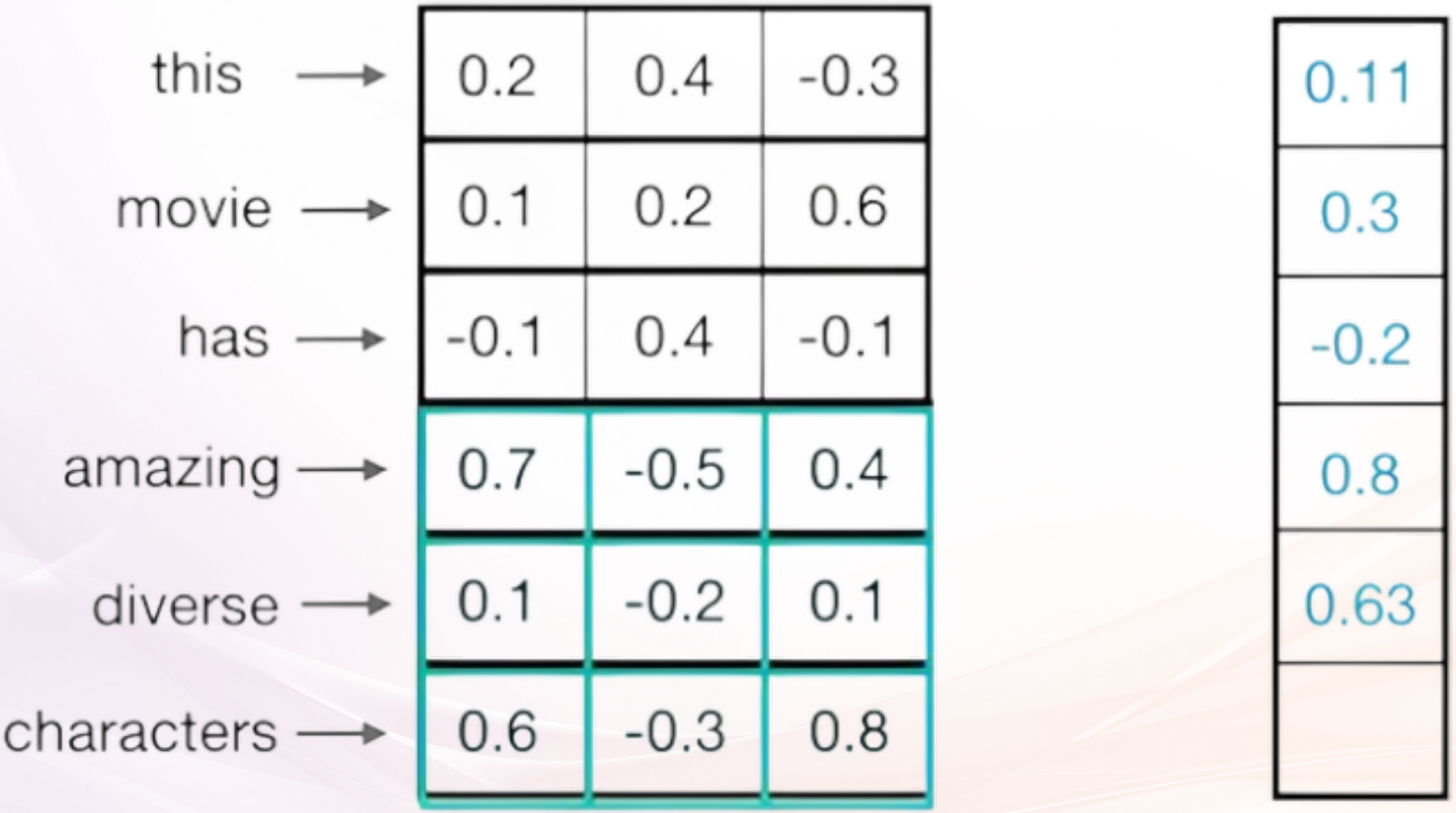
1D-свертка

В 1D-свертке фильтр перемещается только в одном направлении.



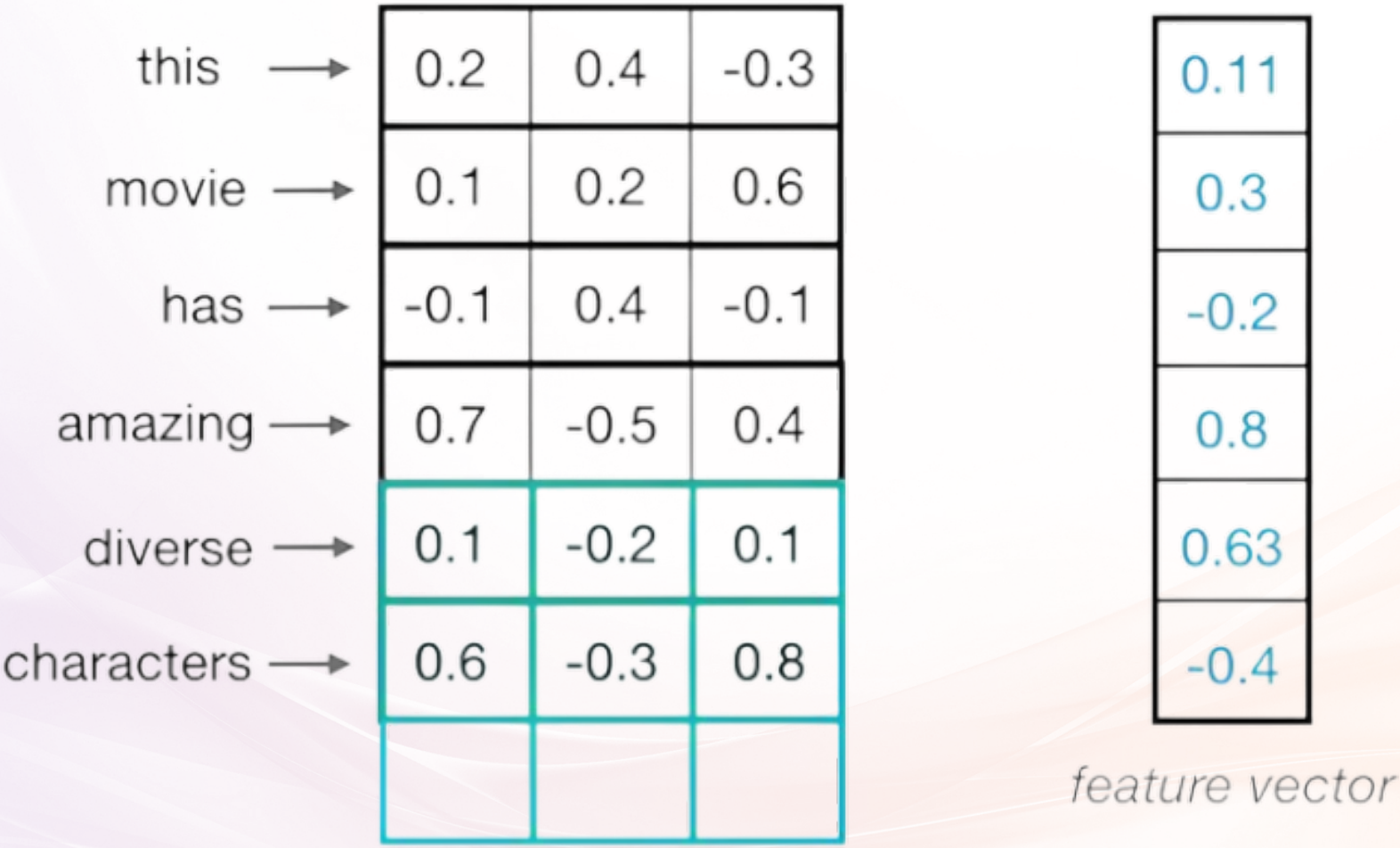
1D-свертка

В 1D-свертке фильтр перемещается только в одном направлении.

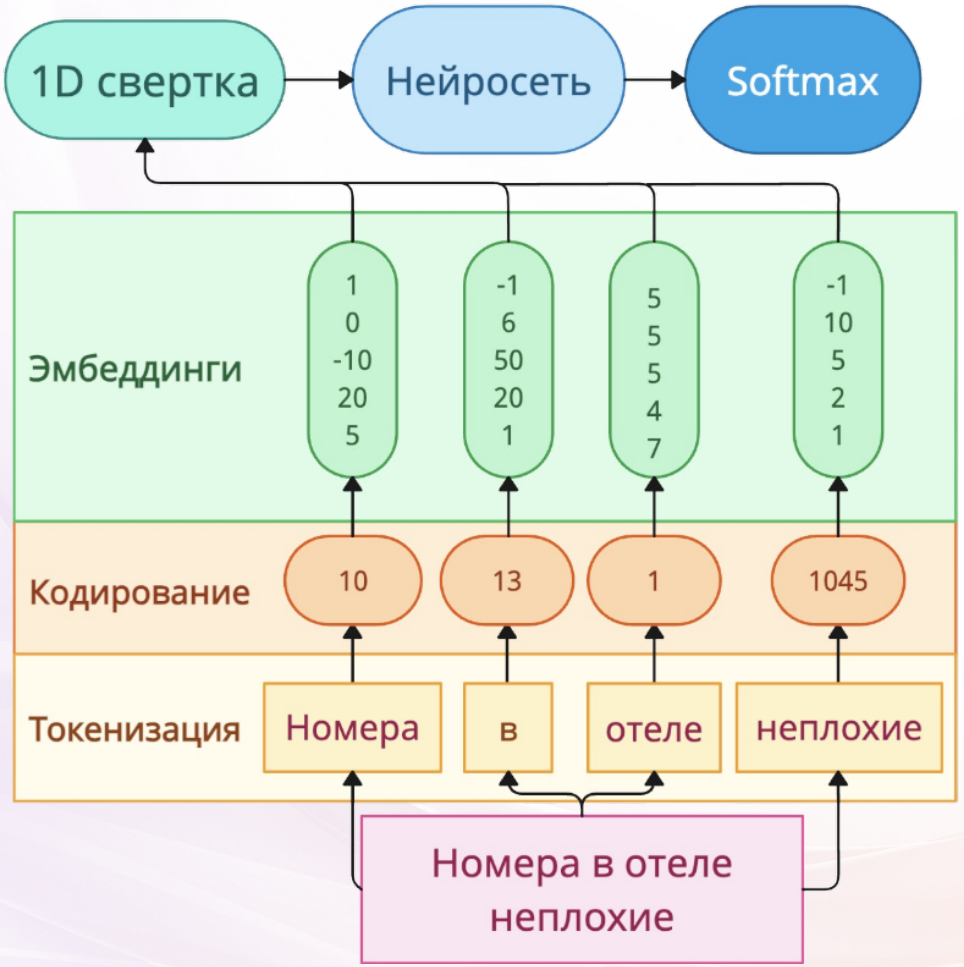


1D-свертка

В 1D-свертке фильтр перемещается только в одном направлении.



Научились решать задачу классификации



Мы можем превратить выходы сверточных слоев в логиты для предсказания класса с помощью полносвязного слоя в конце

А превратить логиты в вер-ти можно с помощью softmax

Языковая модель

Пусть на вход приходит посл-ть слов $x_1, \ldots, x_n$.

Цель языковой модели

По имеющемуся датасету оценить вероятность появления этой последовательности.

Вероятность последовательности —

мера того, насколько вероятна эта последовательность в реальном мире (в имеющимся датасете).

$\mathrm{P}(\texttt{я учу машинное обучение}) > \mathrm{P}(\texttt{учу машинное я обучение})$

Как нейросеть продолжает текст?

Задача продолжения текста

Дано начало: «`Однажды зимним вечером...`»

Хотим предсказать продолжение: «`пошел снег, улицы опустели`»

Основная идея: языковая модель обычно оценивает вероятность всей последовательности через условную вероятность:

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

Как это работает:

- Модель оценивает вероятность следующего слова на основе предыдущих.
- На каждом шаге, имея распр. вер-тей на след. слово, выбираем наиболее вероятное продолжение.

Какими способами можно решить данную задачу?

Какими способами можно решить данную задачу?

Нейросети?

Свёрточные нейросети?

Не возникнет ли проблем которые встречались нам раньше?

Проблема:

На практике оказывается, что построить хорошую языковую модель на основе сверточной нейронной сети сложно.

Причина:

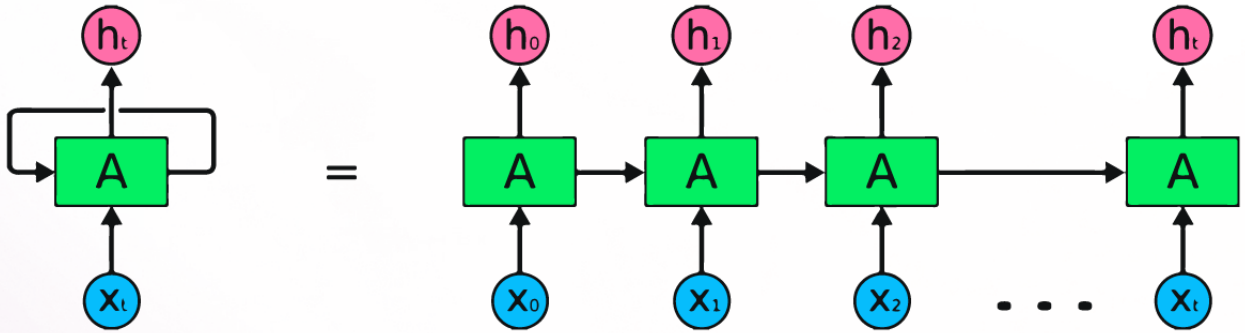
Сверточная нейронная сеть имеет фиксированный размер окна, поэтому она помнит только ограниченное число токенов.

Решение:

Построить нейросеть, которая запоминала бы все предыдущие токены и делала на их основе предсказание.

RNN: рекуррентная нейросеть

Recurrent Neural Network (RNN) — слой, для обработки посл-ти.
Вся последовательность скрытых состояний $H = (h_1, h_2, \ldots, h_T)$ хранит информацию о предыдущих входных элементах.



Обработка одного элемента

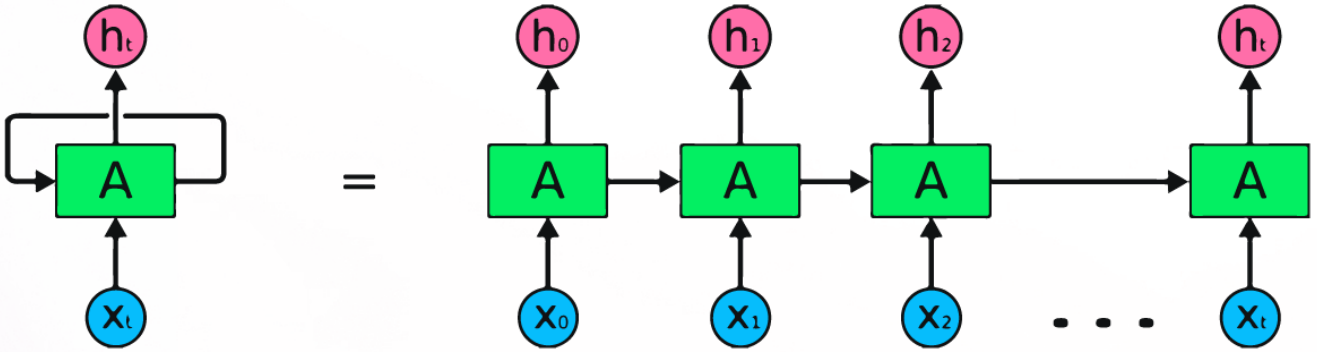
- $x_t \in \mathbb{R}^{D \times 1}$ — входной элемент в момент t,
- $h_t \in \mathbb{R}^{d \times 1}$ — скрытое состояние в момент t:

$$h_t = \underbrace{\tanh}_{\text{поэлементно}} \left(W_x x_t + W_h h_{t-1}\right),$$

- $W_x \in \mathbb{R}^{d \times D}$, $W_h \in \mathbb{R}^{d \times d}$ — обучаемые матрицы, параметры слоя.

Полученный h_t используется для обработки следующего x_{t+1}.

Пример работы RNN



Дано:

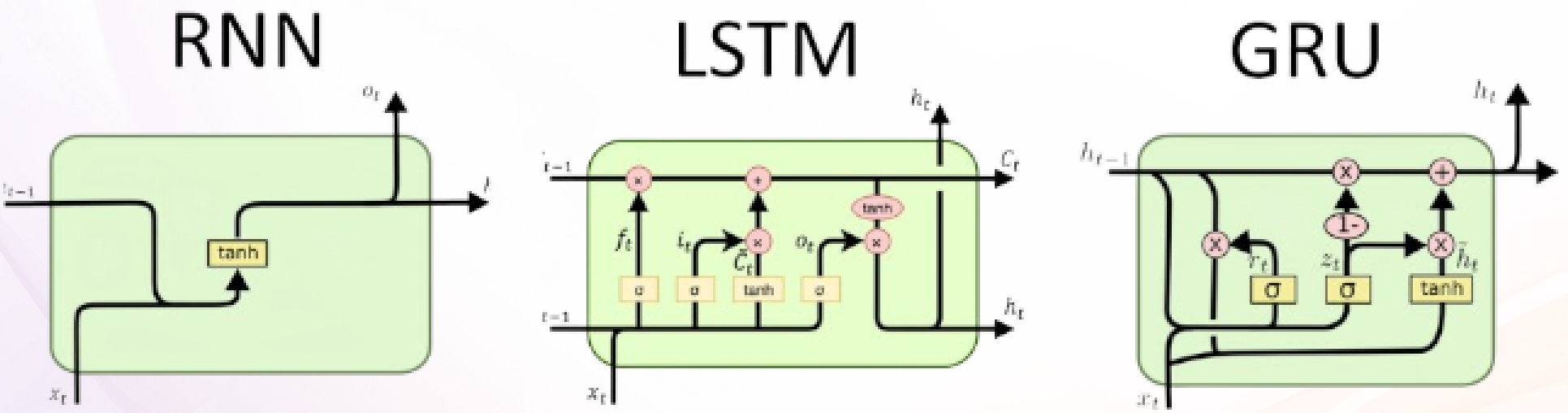
- $W_h = 0.5$, $W_x = 1$, а активация — tanh,
- Входная последовательность: $x_1 = \mathbf{2}$, $x_2 = \mathbf{-1}$, $x_3 = \mathbf{3}$,
- Начальное скрытое состояние: $h_0 = \mathbf{0}$.

Шаг 1: $h_1 = \tanh(\mathbf{0} \cdot 0.5 + \mathbf{2} \cdot 1) = \tanh(2) \approx \mathbf{0.96}$

Шаг 2: $h_2 = \tanh(\mathbf{0.96} \cdot 0.5 + \mathbf{-1} \cdot 1) = \tanh(-0.52) \approx \mathbf{-0.48}$

Шаг 3: $h_3 = \tanh(\mathbf{-0.48} \cdot 0.5 + \mathbf{3} \cdot 1) = \tanh(2.76) \approx \mathbf{0.99}$

RNN



Подробнее о рекуррентных архитектурах вы узнаете на DS-потоке

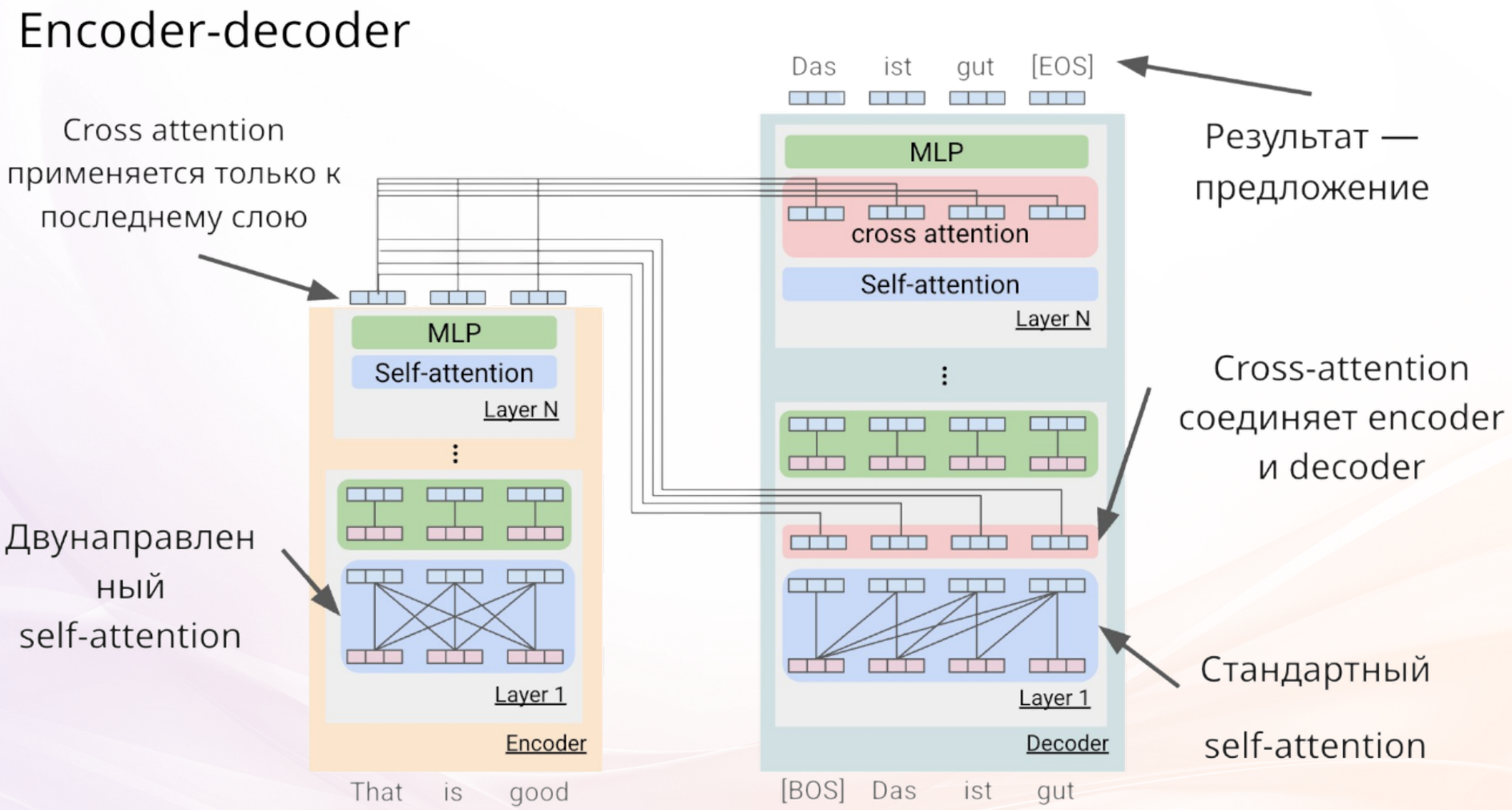
Недостатки RNN

Проблема: RNN забывает токены с предыдущих итераций, и качество на задачах падает с увеличением длины последовательности.

Архитектурные модификации RNN, такие как LSTM и GRU, решают проблему лишь отчасти.

На практике RNN показывают слабую эффективность и масштабируемость.

Transformer Encoder-Decoder



Transformer Encoder

Encoder-only

Специфичный для задачи слой для получения выхода в нужном формате

Линейный слой

Выход в виде единственного вектора

Не умеет генерировать тексты!

MLP

Self-attention

Layer N

MLP

Self-attention

Layer 1

Используется специальный токен

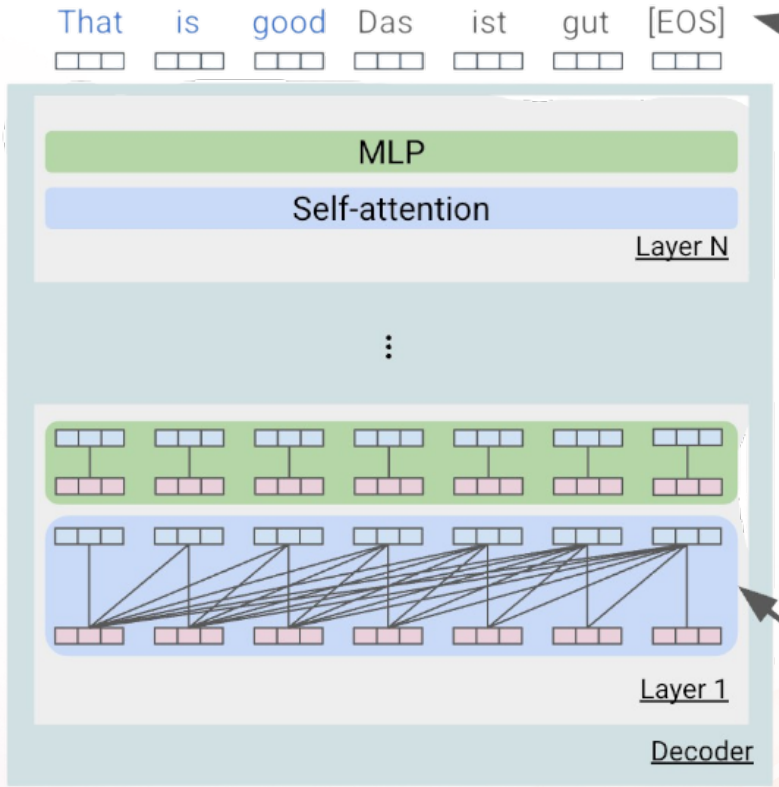
[CLS] That is good

Вход в виде текста

Transformer Decoder

Decoder-only

Одинаковая модель применяется и к входу и к target



Результат — текст

Обычный self-attention

Вход и target объединены

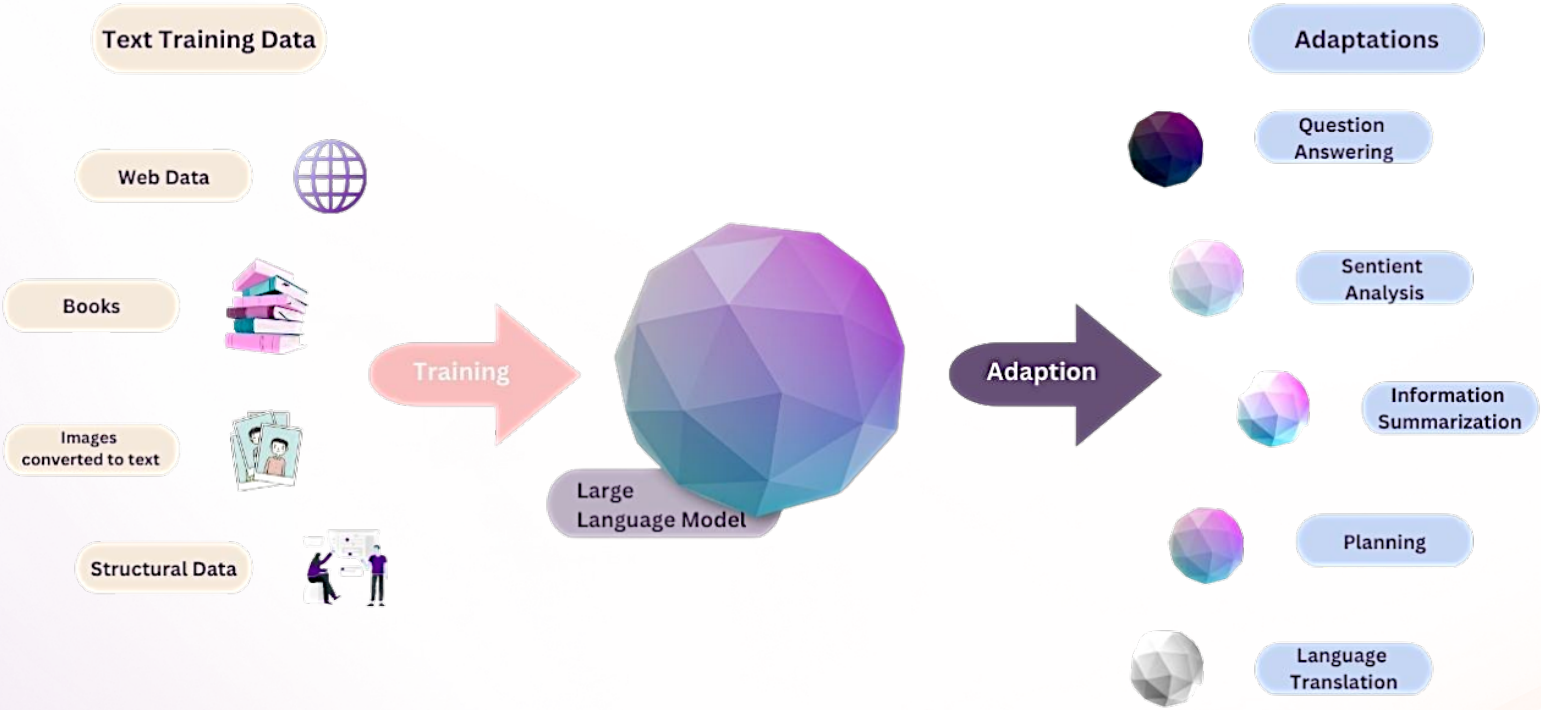
План лекции

NLP

Кодирование текстов

Основные модели

LLM



До этого мы пытались найти подходящую архитектуру отдельно для каждой задачи.

А если попробовать построить одну модель для всего сразу?

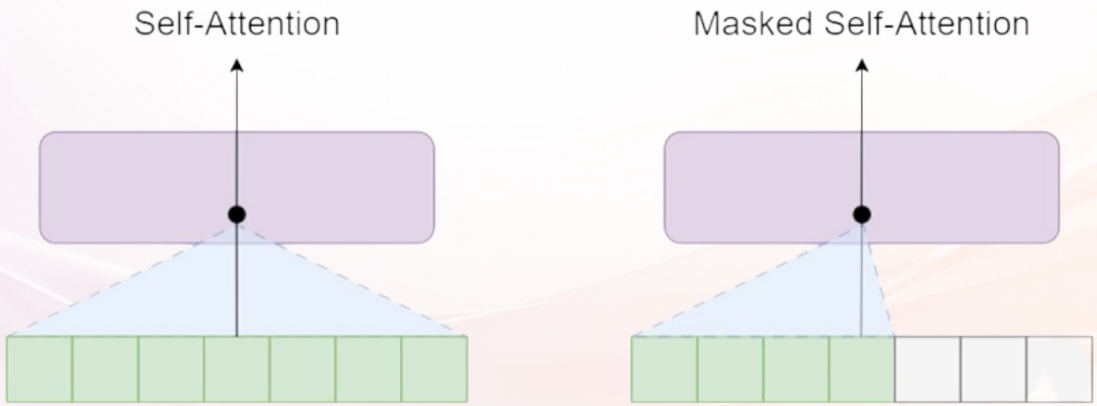
Получим новую парадигму для обучения **LLM**!

Large Language Models (LLM)

LLM изначально обучены на задачу языкового моделирования — предсказывания следующего токена на большом корпусе текстов.

Текущие **LLM** используют декодер трансформера, который:

- Применяет **masked self-attention**, чтобы учесть взаимодействие между словами.
- Работает по принципу **авторегрессии**: для предсказания следующего слова опирается на предыдущие.



Некоторые хитрости и техники позволяют адаптировать эти модели к решению огромного класса задач.

Предобучение LLM

Обучение большой языковой модели обычно состоит из двух этапов.

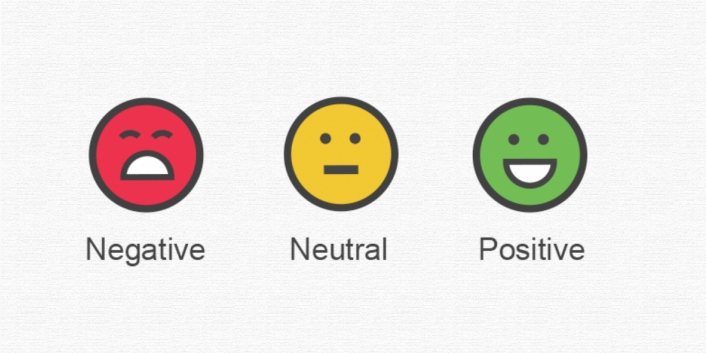
Pretraining — обучение на большом количестве текстов для получения базовых знаний о мире.

Finetuning & Alignment.

- Настройка модели на более узкие задачи с использованием целевых данных.
- Обучение следовать инструкциям и адаптация модели к человеческим ожиданиям.

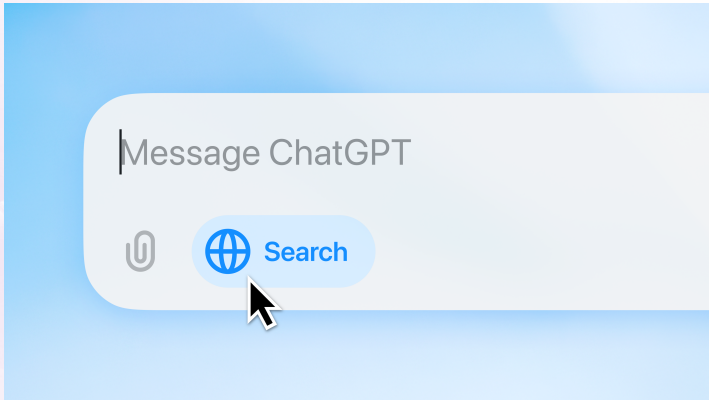
С деталями каждого из этапов вы познакомитесь на DS-потоке!

Виды задач

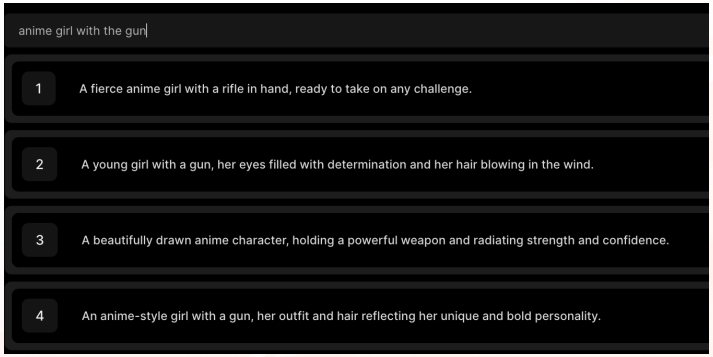


Анализ тональности

Суммаризация текста



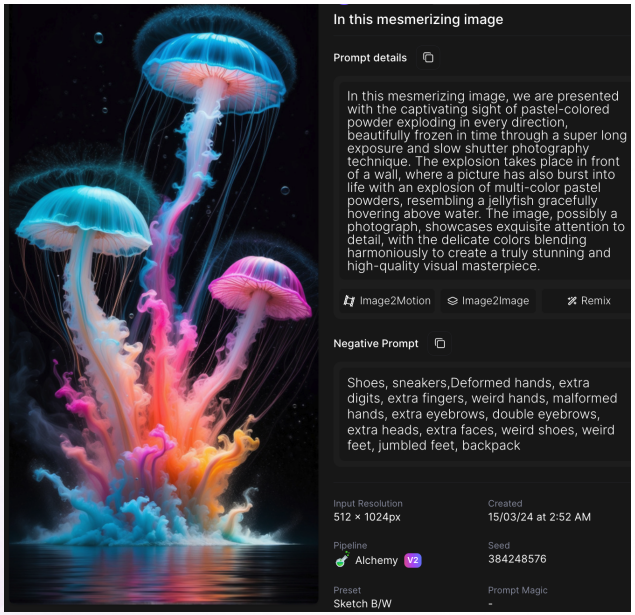
Диалоговые ассистенты



Prompt enhancement

Виды задач

Задачи на стыке NLP и CV



Генерация картинки по тексту

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

Image Captioning

И ещё много чего...

будет на DS-потоке! :)

ВСЁ!