

Лекции 3-5.

Классификация текстов

Классификация текстов

- Примеры классификационных задач
- Общий вид: Особенности + Классификатор
- Модели: генеративные и дискриминативные
- Классические методы
- Нейросетевые методы
- Классификация с несколькими метками
- Практические советы
- Анализ и интерпретируемость

Классификация текстов

Есть текст → модель должна определить, к какому классу (категории) он относится.

- Многоклассовая классификация (Multi-class) (много меток, правильной является только одна из них)
- Бинарная классификация (Binary) (две метки, правильной является только одна из них)
- Многометочная классификация (Multi-label) (много меток, один текст может иметь несколько правильных меток)

Датасеты для классификации текстов

Датасеты для классификации текстов — это данные, которые используются для обучения и проверки моделей классификации текста.

Датасеты сильно различаются по:

- Типу
- Количеству меток
- Размеру
- Средней длине в токенах (числу, которое показывает, сколько в среднем токенов содержится в одном тексте датасета)

Датасеты для классификации текстов

Dataset	Type	Number of labels	Size (train/test)	Avg. length (tokens)
SST	sentiment	5 or 2	8.5k / 1.1k	19
IMDb Review	sentiment	2	25k / 25k	271
Yelp Review	sentiment	5 or 2	650k / 50k	179
Amazon Review	sentiment	5 or 2	3m / 650k	79
TREC	question	6	5.5k / 0.5k	10
Yahoo! Answers	question	10	1.4m / 60k	131
AG's News	topic	4	120k / 7.6k	44
Sogou News	topic	6	54k / 6k	737
DBPedia	topic	14	560k / 70k	67

Датасеты для классификации текстов

- SST

Stanford Sentiment Treebank — это классический и очень известный датасет для оценки тональности текста (sentiment analysis). Предложения взяты из кинокритики (сайт Rotten Tomatoes). Исходная разметка использует 5 классов:

0 — Very Negative

1 — Negative

2 — Neutral

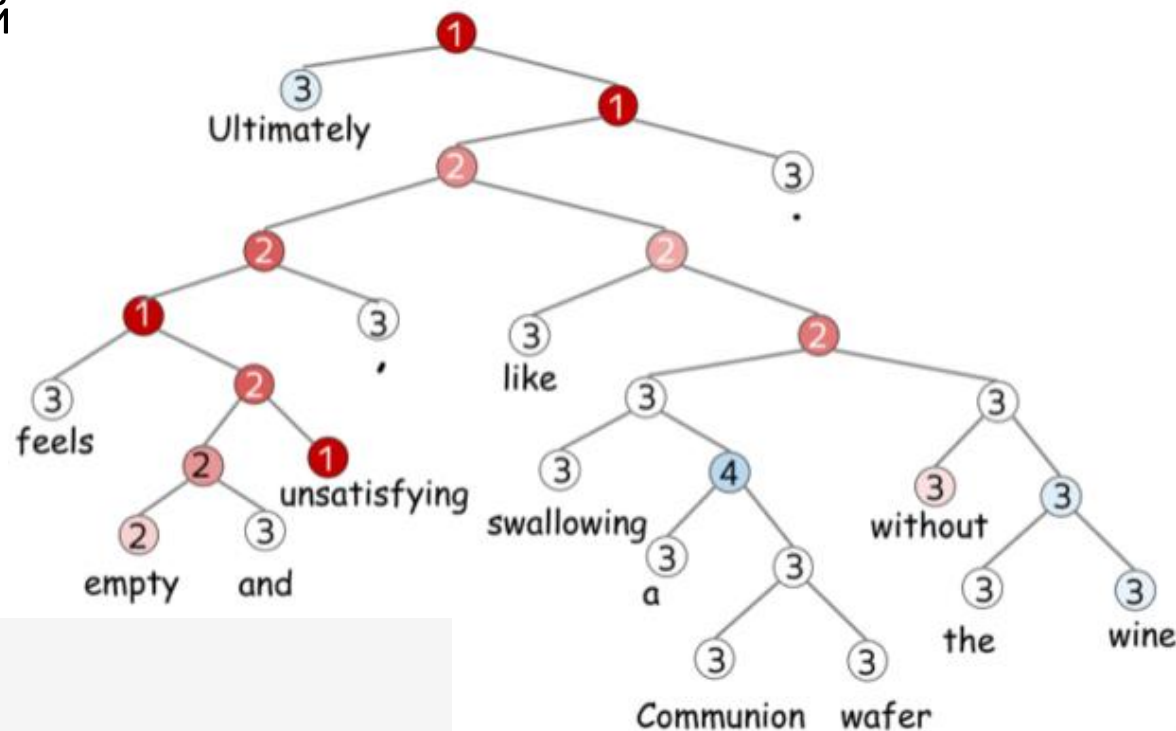
3 — Positive

4 — Very Positive

Label: 1

Review:

Ultimately feels empty and unsatisfying , like swallowing a Communion wafer without the wine .



Датасеты для классификации текстов

- **IMDb Review**

IMDb (Internet Movie Database) — это огромная онлайн-база данных о фильмах, телесериалах и видео играх. Датасет **IMDb Review** представляет собой коллекцию отзывов зрителей на фильмы, размещенные на этом сайте

Задача: Бинарная классификация тональности (Sentiment Analysis)

Исходная разметка использует 2 класса:

Negative (0) — отрицательный отзыв (оценка пользователя ≤ 4 из 10)

Positive (1) — положительный отзыв (оценка пользователя ≥ 7 из 10)

Датасеты для классификации текстов

- IMDb Review

Label: negative

Review

Hobgoblins Hobgoblins where do I begin!?

This film gives Manos - The Hands of Fate and Future War a run for their money as the worst film ever made . This one is fun to laugh at , where as Manos was just painful to watch . Hobgoblins will end up in a time capsule somewhere as the perfect movie to describe the term : " 80 's cheeze " . The acting (and I am using this term loosely) is atrocious , the Hobgoblins are some of the worst puppets you will ever see , and the garden tool fight has to be seen to be believed . The movie was the perfect vehicle for MST3 K , and that version is the only way to watch this mess . This movie gives Mike and the bots lots of ammunition to pull some of the funniest one - liners they have ever done . If you try to watch this without the help of Mike and the bots God help you ! !

Датасеты для классификации текстов

- **Yelp Review**

Основан на реальных отзывах с платформы Yelp.

Yelp — это платформа, где пользователи оставляют отзывы о ресторанах, кафе, магазинах, салонах красоты и т.д. Датасет **Yelp Review** представляет собой огромную выборку этих отзывов, публично предоставленную Yelp для исследовательских целей.

Задача: Многоклассовая классификация

Классы: Отзывы размечены количеством звезд (1-5), что естественным образом соответствует тональности:

- **1 звезда** — Very Negative (очень негативный)
- **2 звезды** — Negative (негативный)
- **3 звезды** — Neutral (нейтральный)
- **4 звезды** — Positive (позитивный)
- **5 звезд** — Very Positive (очень позитивный)

Датасеты для классификации текстов

- Yelp Review

Label: 4

Review

I had a serious craving for Roti. So glad I found this place. A very small menu selection but it had exactly what I wanted. The serving for \$8.20 after tax is enough for 2 meals. I know where to go from now on for a great meal with leftovers. This is a noteworthy place to bring my Uncle T.J. who's a Trini when he comes to visit.

Датасеты для классификации текстов

- **Amazon Review**

Датасет Amazon Review — это один из самых масштабных и значимых ресурсов в мире NLP и Data Science. Он представляет собой колоссальную коллекцию отзывов пользователей на товары, продаваемые на Amazon.com.

Задача: Многоклассовая классификация

Классы: Отзывы размечены количеством звезд (1-5), что естественным образом соответствует тональности:

- **1 звезда** — Very Negative (очень негативный)
- **2 звезды** — Negative (негативный)
- **3 звезды** — Neutral (нейтральный)
- **4 звезды** — Positive (позитивный)
- **5 звезд** — Very Positive (очень позитивный)

Датасеты для классификации текстов

- Amazon Review

Label: 3

Review Title: Simple

Review Content:

This book was not anything special. Although I love romances, it was too simple. The symbolism was spelled out to the readers in a blunt manner. The less educated readers may appreciate it. The wording was quite beautiful at times and the plot was enchanting (perfect for a movie) but it is not heart wrenching like the movie Titanic (which was a must see!) ;)

Датасеты для классификации текстов

- **TREC QA Track**

Text REtrieval Conference — классификация вопросов по типам.

Задача: Определить тип ожидаемого ответа на вопрос

Классы: Иерархическая система меток (6 грубых классов, 50 тонких).

- **ABBR** (abbreviation) — аббревиатуры и расшифровки.
- **DESC** (description) — описания, определения, объяснения.
- **ENTY** (entity) — сущности (лица, организации, продукты и т.д.).
- **HUM** (human) — люди.
- **LOC** (location) — местоположения.
- **NUM** (numeric) — числовые значения (цена, население, даты).

Датасеты для классификации текстов

- **Yahoo! Answers**

Датасет Yahoo! Answers — коллекция вопросов пользователей, ответов и лучших ответов на эти вопросы .

Задача: Тематическая классификация (Topic Classification). Вопросы и ответы относятся к одной из множества тематических категорий

Классы: 10 крупнейших основных категорий:

Society & Culture (Общество и культура), **Science & Mathematics** (Наука и математика), **Health** (Здоровье), **Education & Reference** (Образование и справочные материалы), **Computers & Internet** (Компьютеры и интернет), **Sports** (Спорт), **Business & Finance** (Бизнес и финансы), **Entertainment & Music** (Развлечения и музыка), **Family & Relationships** (Семья и отношения), **Politics & Government** (Политика и правительство).

Датасеты для классификации текстов

Label: Society & Culture

Question Title: Why do people have the bird, turkey for thanksgiving?

Question Content: Why this bird? Any Significance?

Best Answer

It is believed that the pilgrims and indians shared wild turkey and venison on the original Thanksgiving.

Turkey's "Americanness" was established by Benjamin Franklin, who had advocated for the turkey, not the bald eagle, becoming the national bird.

Датасеты для классификации текстов

- **DBpedia**

Датасет DBpedia — это глобальный общедоступный проект, целью которого является извлечение структурированной информации из Wikipedia и предоставление к ней доступа в виде связанных данных (Linked Data).

Источник: Wikipedia (инфобоксы, категории, геокоординаты, ссылки и т.д.)

Для ML-сообщества наиболее известен именно **датасет для классификации** (DBpedia Classification).

Число классов: 14

Данные для каждого примера:

- **Название сущности** (название статьи Wikipedia).
- **Абстракт (аннотация)** — короткое текстовое описание сущности, извлеченное из первого абзаца соответствующей статьи Wikipedia.

Класс — одна из 14 категорий Dbpedia (Company, Educational Institution, Artist, Athlete...)

Датасеты для классификации текстов

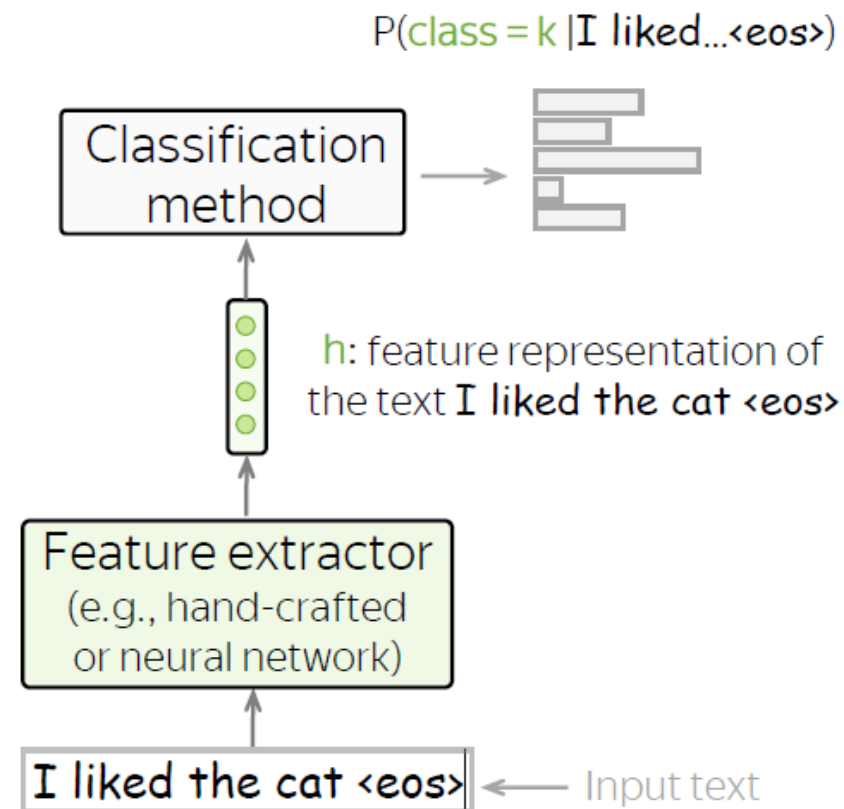
Label: Artist

Title: Esfandiar Monfaredzadeh

Abstract

Esfandiar Monfaredzadeh (Persian : اسفندیار منفردزاده) is an Iranian composer and director. He was born in 1941 in Tehran His major works are Gheisar Dash Akol Tangna Gavaznha. He has 2 daughters Bibinaz Monfaredzadeh and Sanam Monfaredzadeh Woods (by marriage).

Получение представлений признаков и классификация



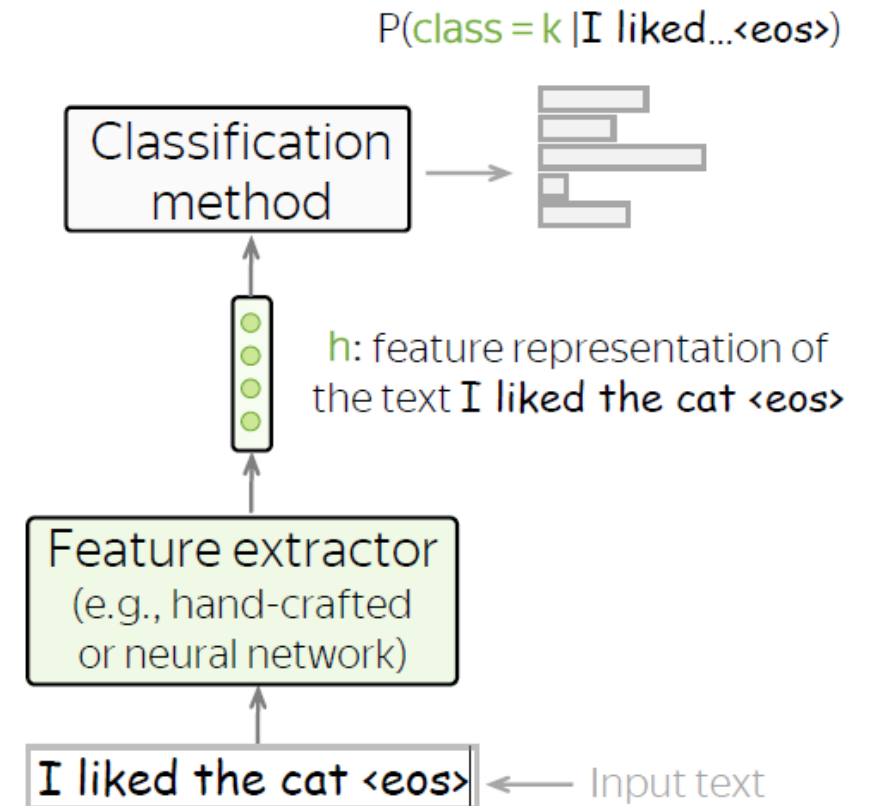
Получение представлений признаков и классификация

Извлечение признаков:

- Классические методы – признаки извлекаются вручную человеком (человек сам придумывает признаки: количество слов; наличие определённых слов; длина текста и т.д.)
- Нейросетевые методы – признаки извлекаются сетью: в процессе обучения сеть запоминает, что важно

Методы классификации:

- генеративные
- дискриминативные



Получение представлений признаков и классификация

Классические методы

Здесь **человек заранее решает, какие характеристики текста важны.**

Например, можно сказать: есть ли слово «интересный»; есть ли слово «понравился»; сколько положительных слов; сколько отрицательных слов.

Получается примерно: «Фильм очень интересный, мне понравился»



Признаки: [есть "интересный" = 1, есть "понравился" = 1, длина = 6, положительных слов = 2, отрицательных слов = 0]

То есть **человек заранее придумал, какие признаки использовать.**

Получение представлений признаков и классификация

Нейросетевые методы

Здесь человек **не прописывает вручную все признаки.**

Мы даём нейросети много текстов и правильные ответы: «Фильм отличный» → положительный; «Фильм ужасный» → отрицательный; «Очень понравилось» → положительный; «Полная скука» → отрицательный

И сеть в процессе обучения **сама учится выделять закономерности, которые помогают различать классы.**

Условно: Текст → Нейросеть → [0.21, -0.43, 0.87, 0.15, ...] → Классификатор → Положительный

Вот этот числовой вектор может быть **эмбедингом.**

При нейросетевом подходе извлечение признаков может заключаться в преобразовании текста в эмбединг.

Генеративные модели

Основная идея

Генеративная модель в классификации текста пытается не сразу научиться разделять классы, а сначала понять, **как обычно выглядят тексты каждого класса**.

Допустим, есть два класса отзывов: **positive** и **negative**.

Модель учится отвечать на вопрос:

«Если это positive, какие тексты обычно встречаются? А если negative?»

То есть она моделирует:

$P(\text{текст} \mid \text{класс})$ — вероятность увидеть такой текст, если мы уже знаем класс.

$P(\text{класс})$ — насколько часто вообще встречается этот класс.

Генеративные модели

Например, есть текст: «The movie was amazing, I loved it»

Генеративная модель проверяет насколько этот текст похож на положительные отзывы:

$$P(\text{«The movie was amazing...»} \mid \text{positive})$$

Насколько он похож на отрицательные отзывы:

$$P(\text{«The movie was amazing...»} \mid \text{negative})$$

Учитывает частоту классов: $P(\text{positive})$ и $P(\text{negative})$

После этого по формуле Байеса получает вероятность класса при известном тексте:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

То есть модель выбирает тот класс, для которого вероятность получилась выше.

Генеративные модели. Пример

Текст:

«I hated this movie»

Модель знает:

$P(\text{текст} \mid \text{negative}) = \text{высокая}$

$P(\text{текст} \mid \text{positive}) = \text{низкая}$

$\Rightarrow P(\text{negative} \mid \text{текст}) = \text{высокая}$

Ответ: negative

Генеративные модели. Обучение

Во время обучения модели дают много примеров:

«Great film, amazing actors» → positive

«Terrible movie, boring plot» → negative

Она изучает статистические закономерности:

Для positive: часто встречаются слова: great, amazing, excellent, определённые конструкции предложений.

Для negative: terrible, boring, worst и т.д.

Генеративные модели

Плюсы генеративных методов:

- Могут работать с небольшим количеством данных.
- Можно понять, какие слова характерны для класса.
- Могут работать даже когда классы плохо разделены.

Минусы генеративных методов

- Нужно моделировать распределение текста.
- Делают лишнюю работу.
- Часто хуже по качеству, чем дискриминативные методы.

Генеративная модель спрашивает: «Какой класс мог породить этот текст?»

Дискриминативные модели

Дискриминативные модели учатся сразу различать классы, не пытаясь описать, как эти классы появились.

Главная идея: «У меня есть текст. Какие признаки помогут мне правильно разделить один класс от другого?»

То есть модель учит вероятность:

$$P(\text{класс} \mid \text{текст})$$

Другими словами:

«Если я вижу такой текст, насколько вероятно, что он относится к классу А или В?»

Дискриминативные модели

Есть два класса: Positive («Отличный фильм, актёры великолепны», «Очень понравилось, рекомендую») и Negative («Ужасный фильм, зря потратил время», «Скучно и плохо снято»).

Дискриминативная модель **не пытается понять**, как вообще создаются отзывы.

Она ищет закономерности, которые помогают разделить классы:

слово «отличный» → увеличивает вероятность Positive

слово «ужасный» → увеличивает вероятность Negative

Получается условная граница:

Negative | Positive

Модель пытается найти линию, которая лучше всего разделяет два класса.

Дискриминативные модели. Обучение

Как происходит обучение

Модели дают примеры:

Текст	Класс
«Фильм отличный»	Positive
«Фильм ужасный»	Negative

Модель преобразует текст в признаки:

«Фильм отличный»



[0.2, 0.8, 0.1, ...]

Дискриминативные модели. Обучение

Затем она ищет параметры, при которых: вектор текста → правильный класс

Например: Positive:

+ слово «отличный»

+ слово «прекрасный»

Negative:

+ слово «плохой»

+ слово «ужасный»

Если модель ошиблась: «Фильм отличный» → Negative (ОШИБКА), то она изменяет свои параметры, чтобы в будущем таких ошибок было меньше.

Дискриминативные модели

Плюсы дискриминативных методов

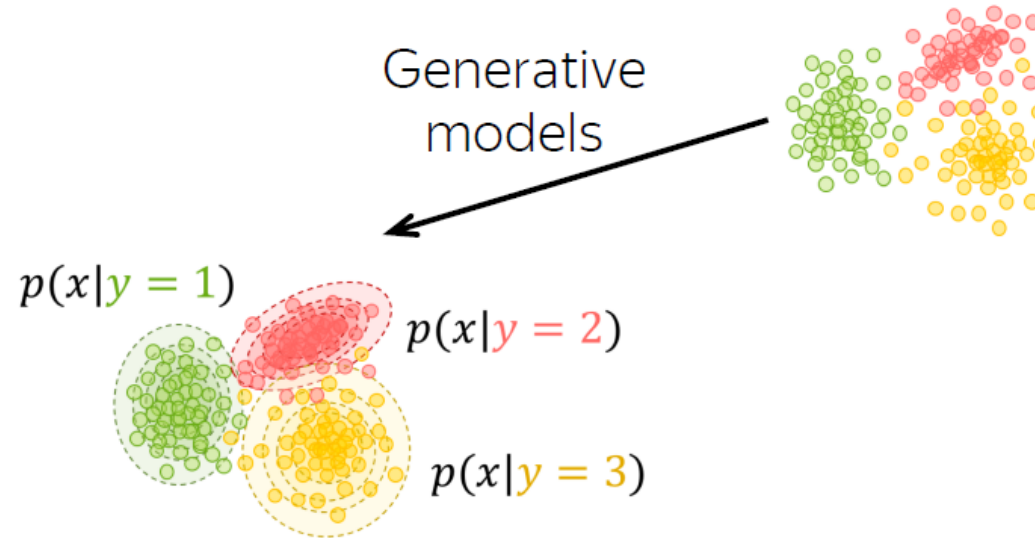
- Обычно дают более высокое качество классификации.
- Хорошо работают с большим количеством данных.
- Лучше используют сложные признаки.

Минусы дискриминативных методов

- Требуют больше размеченных данных.
- Хуже работают при нехватке данных.
- Часто менее интерпретируемы.

Генеративные модели

Генеративные модели называются так потому, что они пытаются **смоделировать процесс «генерации» данных** для каждого класса. Они не просто учатся разделять классы, а изучают внутреннюю структуру данных в каждом из них.



Learn: data distribution

$$p(x, y) = p(x|y) \cdot p(y)$$

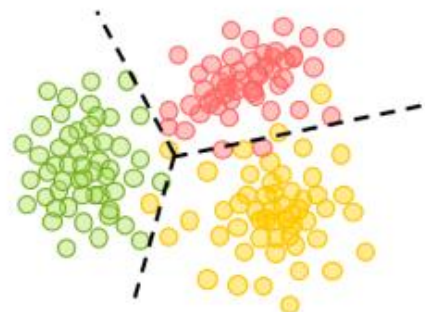
How predict:

$$y = \arg \max_k p(x, y) = \arg \max_k p(x|y) \cdot p(y)$$

Дискриминативные модели



Как по объекту определить класс?



Какова вероятность класса y , если у нас есть объект x

Learn: boundary between classes $p(y|x)$

How predict:

$$y = \arg \max_k p(y|x)$$

Пример

Допустим, нужно определить тональность: «I loved this movie»

Генеративный подход:

Смотрит, насколько этот текст похож на positive отзывы? Насколько похож на negative отзывы?

И выбирает класс, который лучше объясняет текст.

Дискриминативный подход:

Смотрит, какие признаки помогают отделить positive от negative?

loved → в сторону positive

movie → почти не важно

И сразу выдаёт: Positive

Классические методы

- Наивный Байес
- Логистическая регрессия
- SVM

Наивный Байес

Алгоритм предполагает:
Все слова в тексте
независимы друг от друга.

выбрать класс k , y
которого вероятность
после получения текста
 x максимальна.

Насколько этот текст
похож на тексты
данного класса?

$$y^* = \arg \max_k P(y = k|x) = \arg \max_k \frac{P(x|y = k) \cdot P(y = k)}{P(x)} =$$

$$= \arg \max_k P(x|y = k) \cdot P(y = k)$$

$$y^* = \arg \max_k P(y = k|x) = \arg \max_k P(x|y = k) \cdot P(y = k) = \arg \max_k P(x, y = k)$$

$$P(y = k) = \frac{N(y = k)}{\sum_{i=1}^K N(y = i)}$$

Вероятность самого класса
(априорная вероятность класса (prior probability)).

Наивный Байес. Пример

Задача: **Определить:** отзыв положительный или отрицательный.

Есть обучающие данные:

Positive:

1. отличный фильм
2. прекрасный фильм
3. хороший актёр

Negative:

1. плохой фильм
2. ужасный фильм
3. скучный фильм

Наивный Байес. Пример

Новый текст: **отличный актёр**

Нужно определить класс:

$$y^* = \arg \max_k P(y = k | x)$$

«Какой класс наиболее вероятен после того, как мы увидели текст?»

Шаг 1. Считаем вероятность самого класса $P(y = k)$

$$P(y = k) = \frac{N(y = k)}{\sum_{i=1}^K N(y = i)}$$

Всего отзывов: 6

$$P(Positive) = \frac{3}{6} = 0.5, P(Negative) = \frac{3}{6} = 0.5$$

Наивный Байес. Пример

Шаг 2. Смотрим, насколько текст похож на каждый класс

$P(x | y = k)$, то есть: «Если это положительный класс, насколько вероятен такой текст?»

Наш текст: отличный актёр. Разделяем на слова: отличный + актёр

Наивный Байес делает предположение, что слова независимы

Поэтому:

$$P(x | y) = P(\text{отличный} | y) \times P(\text{актёр} | y)$$

Наивный Байес. Пример

Для Positive. Смотрим только положительные отзывы: отличный фильм, прекрасный фильм, хороший актёр

Всего слов:отличный - 1

фильм - 2

прекрасный - 1

хороший - 1

актёр - 1

Всего: 6 слов

$$P(\text{отличный} \mid \text{Positive}) = \frac{1}{6}; P(\text{актёр} \mid \text{Positive}) = \frac{1}{6}$$

$$\text{Получаем: } P(x \mid \text{Positive}) = \frac{1}{6} \times \frac{1}{6} = 0.0277$$

Наивный Байес. Пример

Для Negative

Отрицательные отзывы: плохой фильм, ужасный фильм, скучный фильм

Всего:

6 слов

Но слов:

отличный = 0

актёр = 0

Поэтому:

$$P(\text{отличный} \mid \text{Negative}) = 0, P(x \mid \text{Negative}) = 0$$

$$\text{Получаем: } P(x \mid \text{Negative}) = 0$$

Наивный Байес. Пример

Шаг 3. Используем главную формулу

$$P(y = k | x) \approx P(x | y = k) \cdot P(y = k)$$

Для Positive: $0.0277 \times 0.5 = 0.01385$

Для Negative: $0 \times 0.5 = 0$

Сравниваем:

Positive: 0.01385; Negative: 0

Максимум у Positive.

Значит: $y^* = \text{Positive}$

Ответ:

«отличный актёр» → положительный отзыв

Наивный Байес. Пример

Теперь связываем все формулы со смыслом

Новый текст: отличный актёр

1. Проверяем: $P(\text{Positive})$ и $P(\text{Negative})$
2. Проверяем, насколько слова характерны для класса: $P(\text{текст} \mid \text{Positive})$, $P(\text{текст} \mid \text{Negative})$
3. Перемножаем $P(\text{текст} \mid \text{класс}) \times P(\text{класс})$
4. Берём максимум $\arg \max$

Ответ: Positive

Главная идея Naive Bayes:

Он выбирает класс, который наиболее вероятно мог породить данный текст.

Вычисление $P(x|y = k)$

$$P(x|y = k) = P(x_1, x_2, \dots, x_n|y = k)$$

Наивные Байесовские предположения:

- Предположение о мешке слов (Bag of Words assumption) — порядок слов не имеет значения
- Предположение об условной независимости (Conditional Independence assumption) - признаки (слова) независимы для данного класса.

$$P(x|y = k) = P(x_1, x_2, \dots, x_n|y = k) = \prod_{i=1}^n P(x_i|y = k)$$

Вычисление $P(x|y = k)$

$$P(\text{This film is awesome} \mid y = +) =$$

- $P(\text{This} \mid y = +)$
- $P(\text{film} \mid y = +)$
- $P(\text{is} \mid y = +)$
- $P(\text{awesome} \mid y = +)$
- $P(\mid y = +)$

Вычисление $P(x|y = k)$

Используя наивные предположения, получили:

$$P(x|y = k) = P(x_1, x_2, \dots, x_n|y = k) = \prod_{i=1}^n P(x_i|y = k)$$

Вычисляем отдельные вероятности «вручную»:

$$P(x_i|y = k) = \frac{N(x_i, y = k)}{\sum_{t=1}^{|V|} N(x_t, y = k)}$$

как часто слово x_i встречается внутри класса k

сколько раз слово x_i встретилось в классе k

общее количество всех слов во всех текстах этого класса

Возможная проблема

Вычисляем отдельные вероятности «вручную»:

$$P(x_i|y = k) = \frac{N(x_i, y = k)}{\sum_{t=1}^{|V|} N(x_t, y = k)} = 0$$

А что если числитель равен нулю? (то есть при обучении токена x_i не было в документах класса k)

Решение проблемы

Сглаживание Лапласа (Laplace Smoothing).

Решает проблему нулевой вероятности

$$P(x_i|y = k) = \frac{\delta + N(x_i, y = k)}{\sum_{t=1}^{|V|} (\delta + N(x_t, y = k))} = \frac{\delta + N(x_i, y = k)}{\delta \cdot |V| + \sum_{t=1}^{|V|} N(x_t, y = k)}$$

Предсказание модели

Data: $x =$ This film is awesome !
 $x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5$

$$y^* = \arg \max_k P(x, y = k) = \arg \max_k P(y = k) \cdot P(x|y = k)$$

Получение прогноза

Positive class

$$P(x, y = +)$$

$$= P(y = +) \cdot P(x|y = +)$$

$$= P(y = +) \cdot 0,5$$

$$\cdot P(\text{This}|y = +)$$

$$\cdot P(\text{film}|y = +)$$

$$\cdot P(\text{is}|y = +)$$

$$\cdot P(\text{awesome}|y = +) \leftarrow \text{важное слово}$$

$$\cdot P(!|y = +)$$

$$P(\text{awesome}|y = +) \gg P(\text{awesome}|y = -)$$

Negative class

$$P(x, y = -)$$

$$= P(y = -) \cdot P(x|y = -)$$

$$= P(y = -) \cdot$$

$$\cdot P(\text{This}|y = -)$$

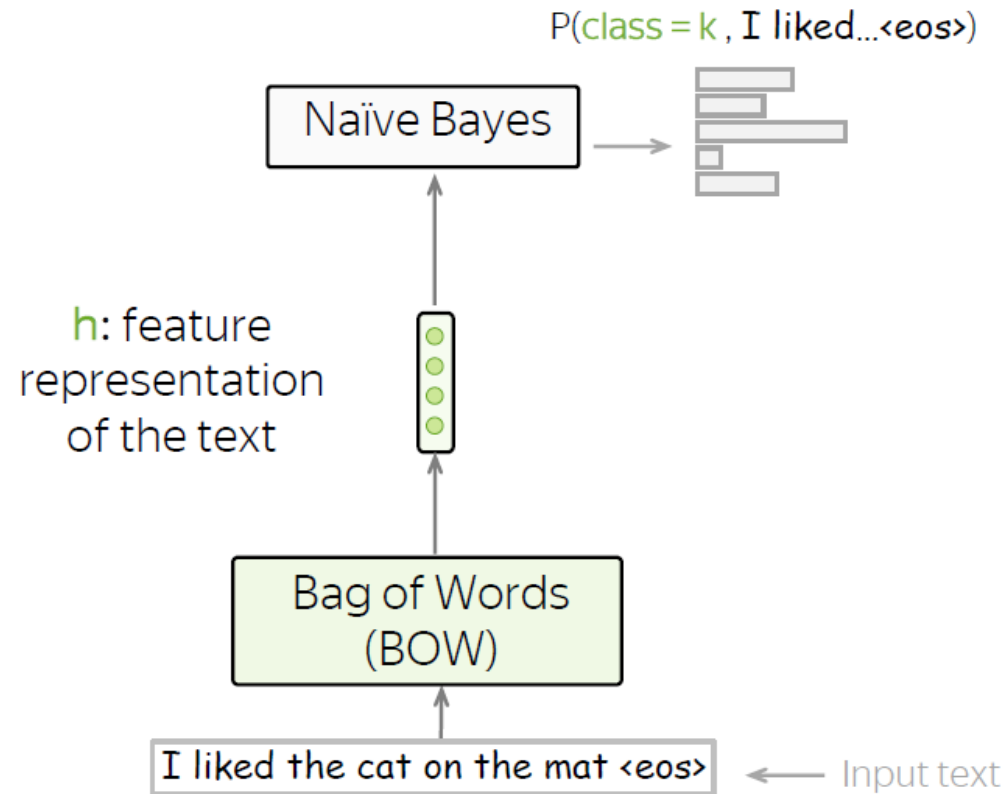
$$\cdot P(\text{film}|y = -)$$

$$\cdot P(\text{is}|y = -)$$

$$\cdot P(\text{awesome}|y = -)$$

$$\cdot P(!|y = -)$$

Наивный Байес: взгляд в рамках общей модели



Проблемы наивного Байеса

This is rather good:
not bad at all!

This is rather bad:
not good at all!



this, is, good, bad,
rather, not, at, all, !

Логистическая регрессия

Идея: взять набор признаков объекта, взвесить их влияние и получить вероятность принадлежности к классу.

$h = (f_1, f_2, \dots, f_n)$ – вектор признаков входного текста

$w^{(k)} = (w_1^{(k)}, w_2^{(k)}, \dots, w_n^{(k)})$ – веса признаков

$$w^{(k)} h = w_1^{(k)} \cdot f_1 + \dots + w_n^{(k)} \cdot f_n, \quad k = 1, \dots, K \quad (\text{оценка класса, score})$$

$$P(\text{class} = k|h) = \frac{\exp(w^{(k)} h)}{\sum_{i=1}^K \exp(w^{(i)} h)} \quad (\text{softmax})$$

Логистическая регрессия не ищет правила вручную. Она учит веса признаков, чтобы правильно превращать признаки текста в вероятности классов.

Логистическая регрессия

Веса признаков

$$w^{(k)} = (w_1^{(k)}, w_2^{(k)}, \dots, w_n^{(k)})$$

Это набор коэффициентов для класса k .

Допустим, мы классифицируем письма: класс 1: спам; класс 2: обычное письмо; класс 3: важное письмо.

У каждого класса есть свои веса:

$$w^{(1)}, w^{(2)}, w^{(3)}$$

То есть модель хранит:

«Какие признаки характерны для спама?»

Обучение: максимизация вероятности

Есть некоторый текст. Текст имеет правильную метку k .

- Максимизация логарифмического правдоподобия правильного класса $\log P(y = k|x) \rightarrow \max$
- Минимизация отрицательного логарифмического правдоподобия правильного класса $-\log P(y = k|x) \rightarrow \min$
- Минимизация потери кросс-энтропии:

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

Обновление весов

Функция ошибки $L(w)$ зависит от весов. Нужно найти такие веса, где ошибка минимальна.

Градиент показывает направление, в котором ошибка растёт $\nabla L(w)$. Поэтому мы идём в противоположную сторону:

$$w_{new} = w_{old} - \eta \nabla L(w)$$

где:

w — веса;

η — скорость обучения (learning rate);

∇L — градиент.

Модель делает предсказание; сравнивает с правильным ответом; считает, какие веса виноваты; немного изменяет веса; повторяет тысячи раз.

Обновление весов

Для логистической регрессии получается формула:

$$\frac{\partial L}{\partial w_i} = (x_i)(\hat{y} - y)$$

где:

$\hat{y}$ — предсказанная вероятность;

y — настоящий класс.

То есть изменение веса зависит от значения признака и ошибки модели.

Обновление весов

Пусть у нас есть отзыв: «Фильм очень понравился»

И мы заранее знаем, что это **положительный отзыв**.

Допустим, классы обозначаются: 1 = положительный; 2 = отрицательный

Тогда для нашего текста: $k = 1$

Здесь:

x — **признаки текста** (например, его embedding);

y — класс, который предсказывает модель;

k — **правильный класс**, известный из обучающей выборки.

Обновление весов

Модель получает x — представление текста: $x = \text{embedding}$ текста
и должна определить: $P(y = 1 | x), P(y = 2 | x), \dots$

То есть она выдаёт **вероятности принадлежности к каждому классу**.

Например: «Фильм очень понравился»

Модель: положительный $\rightarrow 0.90$; отрицательный $\rightarrow 0.10$

То есть: $P(y = 1 | x) = 0.90$; $P(y = 2 | x) = 0.10$

А правильная метка у нас: $k = 1$

Обновление весов

$$\log P(y = k | x) \rightarrow \max$$

Мы хотим, чтобы модель давала **как можно большую вероятность правильному классу**.

В нашем случае правильный класс — положительный.

$$\text{Поэтому хотим: } P(y = 1 | x) \rightarrow 1$$

Чем больше вероятность правильного класса, тем меньше ошибка.

Обновление весов

$$- \sum_{i=1}^K p_i^* \log P(y = i | x) \rightarrow \min$$

Здесь:

K — количество классов;

i — номер класса;

$P(y = i | x)$ — какую вероятность модель дала классу i ;

p_i^* — правильный ответ.

Например, у нас **2 класса**: 1 = положительный; 2 = отрицательный

А текст на самом деле относится к классу 2.

Тогда правильные значения: $p_1^* = 0, p_2^* = 1$.

То есть правильному классу ставим **1**, остальным — **0**.

Обновление весов

Текст → Извлечение признаков → Embedding → Классификатор → Вероятности классов → Cross-Entropy → Изменение параметров нейросети → Модель становится лучше

То есть **cross-entropy** — это **функция потерь**, которая говорит модели:

«Ты должна давать высокую вероятность правильному классу. Если дала низкую — получаешь большую ошибку».

Обновление весов

- Логистическая регрессия считает вероятность каждого класса.
- Ошибка измеряется через отрицательный логарифм вероятности.
- Градиентный спуск постепенно меняет веса слов.
- После обучения слова получают «силу влияния» на решение.

Сравнение наивного Байеса и логистической регрессии

Наивный Байес:

- очень простой
- очень быстрый
- интерпретируемый
- предполагает, что признаки условно независимы
- текстовое представление: определяется вручную (и часто слишком просто, например, BOW)

Сравнение наивного Байеса и логистической регрессии

Логистическая регрессия:

- очень простая
- интерпретируемая
- не предполагает, что признаки условно независимы
- не такая быстрая (множественные итерации градиентного спуска)
- текстовое представление: определяется вручную

SVM (Support Vector Machine)

SVM (Support Vector Machine) — метод машинного обучения, который решает задачу классификации.

Его основная идея:

Найти границу, которая разделяет объекты разных классов.

Например, у нас есть отзывы. SVM пытается найти такую границу, чтобы разделить положительные и отрицательные примеры.

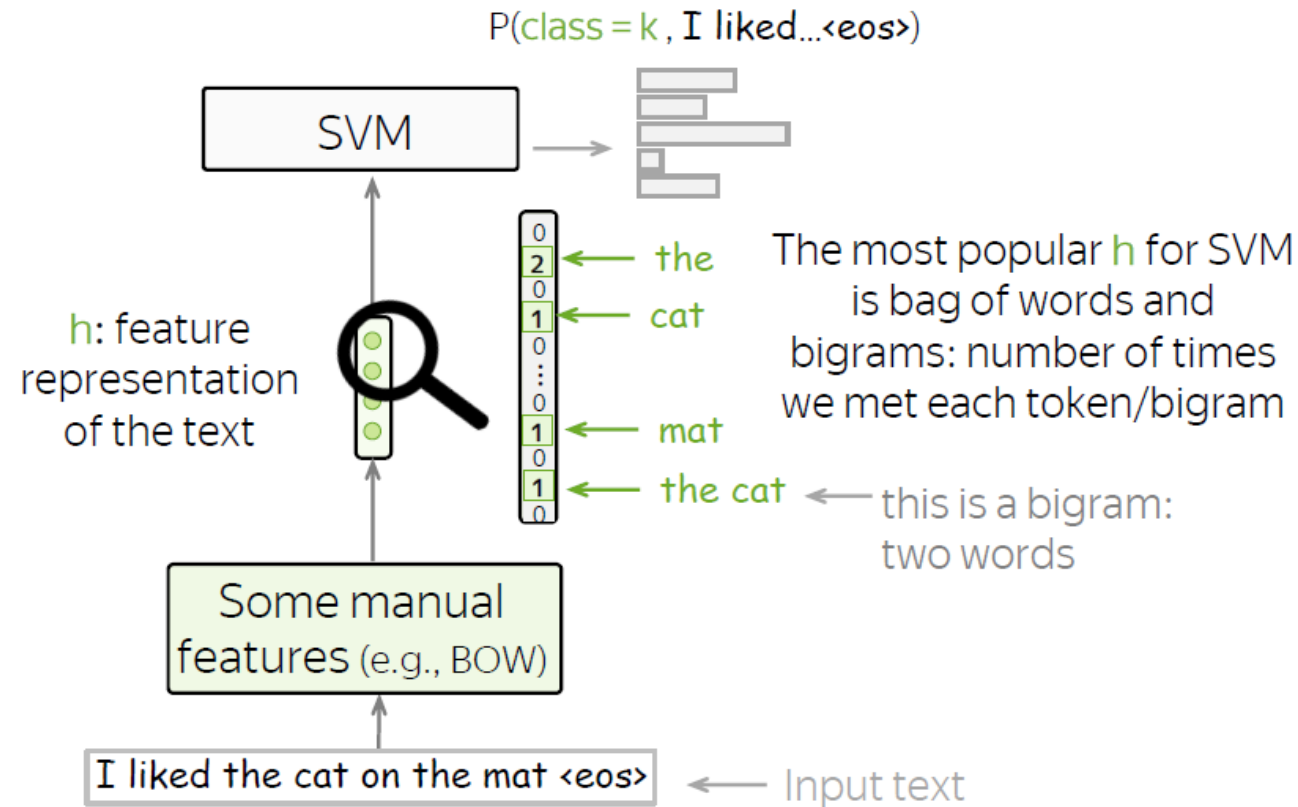
текст → признаки → вектор

Теперь этот вектор передаём в SVM:

$$h(x) \rightarrow SVM$$

SVM смотрит на значения признаков и определяет класс.

SVM



SVM

Как SVM принимает решение?

Текст — это точка в пространстве.

SVM ищет разделяющую гиперплоскость. В двумерном случае это просто линия. В трёхмерном — плоскость. В большем количестве измерений — гиперплоскость.

Особую роль играют опорные векторы (**support vectors**). Это объекты, которые находятся ближе всего к разделяющей границе. **Нужно провести границу так, чтобы расстояние до ближайших объектов обоих классов было максимально возможным.**

Это расстояние называется **margin** (зазор).

SVM. Обучение

Что поступает на вход SVM

Есть вектор признаков:

$$x = (x_1, x_2, \dots, x_d)$$

Например, для текста: «Фильм очень хороший» условный вектор может быть:

$$x = (0, 1, 0, 3, 0, \dots)$$

где координаты соответствуют словам или другим признакам.

Каждый текст имеет правильный класс. Для бинарной классификации удобно обозначать:

$y_i \in \{-1, +1\}$. Например:

+1 = положительный отзыв

-1 = отрицательный отзыв

SVM. Обучение

SVM строит линейную функцию:

$$f(x) = w^T x + b$$

где:

w — **вектор весов признаков**; b — смещение; x — вектор признаков.

Например: $f(x) = w_1 x_1 + w_2 x_2 + \dots + w_d x_d + b$

Решение принимается по знаку: $\hat{y} = \text{sign}(w^T x + b)$

То есть: $w^T x + b > 0 \Rightarrow +1$; $w^T x + b < 0 \Rightarrow -1$

Таким образом, SVM фактически ищет **разделяющую гиперплоскость**

Если $w^T x + b = 0$, то мы находимся **ровно на границе между классами**. Эта граница называется **разделяющей гиперплоскостью**:

$$w^T x + b = 0$$

SVM. Обучение

Что означают веса в классификации текста?

Представим, что признаки — это слова:

$$x_1 = \text{«отличный»}, x_2 = \text{«ужасный»}, x_3 = \text{«понравился»}.$$

SVM может выучить, например:

$$w_1 = 2.5, w_2 = -3, w_3 = 1.7.$$

Тогда слово **«отличный»** положительно влияет на решение, а **«ужасный»** — отрицательно.

Поэтому веса можно интуитивно понимать как **вклад признаков в решение классификатора**.

SVM. Обучение

Какую именно границу выбрать?

Выберем такую границу, которая находится максимально далеко от ближайших объектов обоих классов. Это называется **максимизацией зазора (margin)**.

Для канонической записи границы используются: $w^T x + b = 1$ и $w^T x + b = -1$

SVM удобно масштабирует уравнение так, чтобы ближайшие к границе объекты удовлетворяли: $w^T x + b = 1$ для одного класса и $w^T x + b = -1$ для другого.

Получаются три параллельные гиперплоскости:

$$\boxed{w^T x + b = 1} \quad \boxed{w^T x + b = 0} \quad \boxed{w^T x + b = -1}$$

Средняя — **разделяющая граница**, две крайние определяют границы margin.

SVM. Обучение

Расстояние между двумя крайними гиперплоскостями равно: $\frac{2}{\|w\|}$

Поэтому если мы хотим **максимизировать зазор**, то должны минимизировать $\frac{1}{\|w\|}$.

Что эквивалентно минимизации:

$$\frac{1}{2} \|w\|^2$$

SVM. Обучение

Если данные идеально разделимы, задача имеет вид:

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

при ограничениях: $y_i(w^T x_i + b) \geq 1$ для всех объектов i .

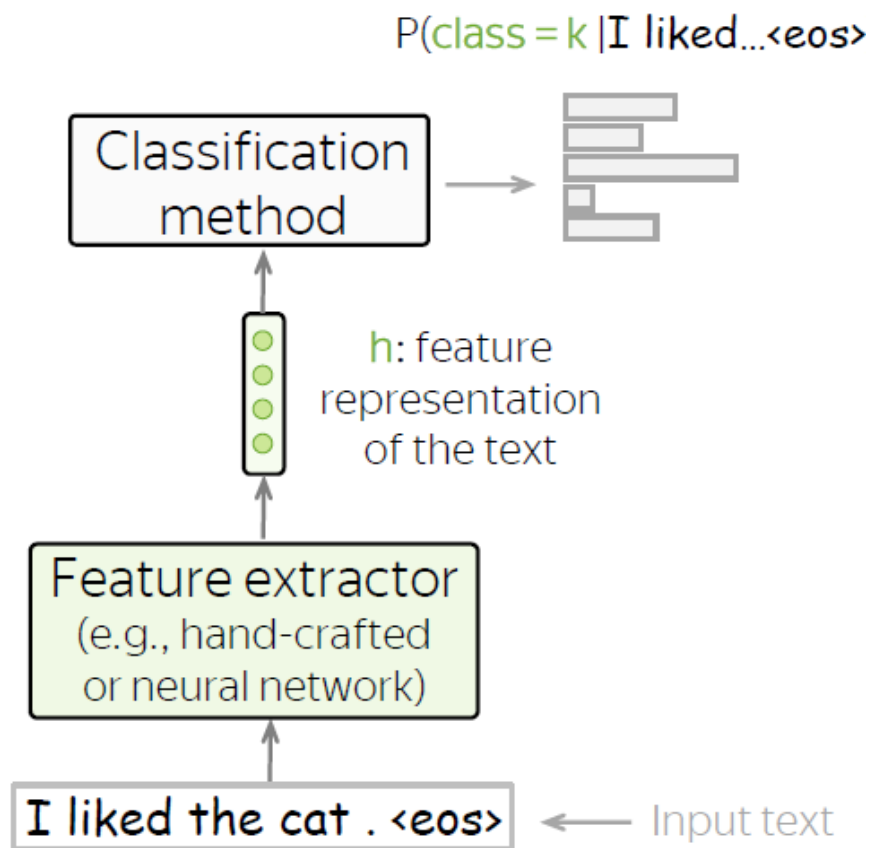
Что означает ограничение?

Если объект положительного класса $y_i = +1$, то $w^T x_i + b \geq 1$. Если отрицательного $y_i = -1$, то $w^T x_i + b \leq -1$.

Таким образом, все объекты должны находиться **на правильной стороне границы** и за пределами margin.

Классификация с помощью классических методов

Общая схема:



Классификация с помощью классических методов

Наивный Байес

- Быстрый и простой, но слишком сильно упрощает взаимосвязи между словами.

Логистическая регрессия

- Хорошо работает с векторными представлениями текста, но строит линейную границу и не учитывает сложный контекст.

SVM

- Эффективен в пространстве высокой размерности и хорошо подходит для классических признаков текста, но также ограничен качеством используемого представления и сложностью разделяющей границы.

Классификация с помощью классических методов

У всех трёх методов есть общий недостаток:

качество классификации сильно зависит от представления текста

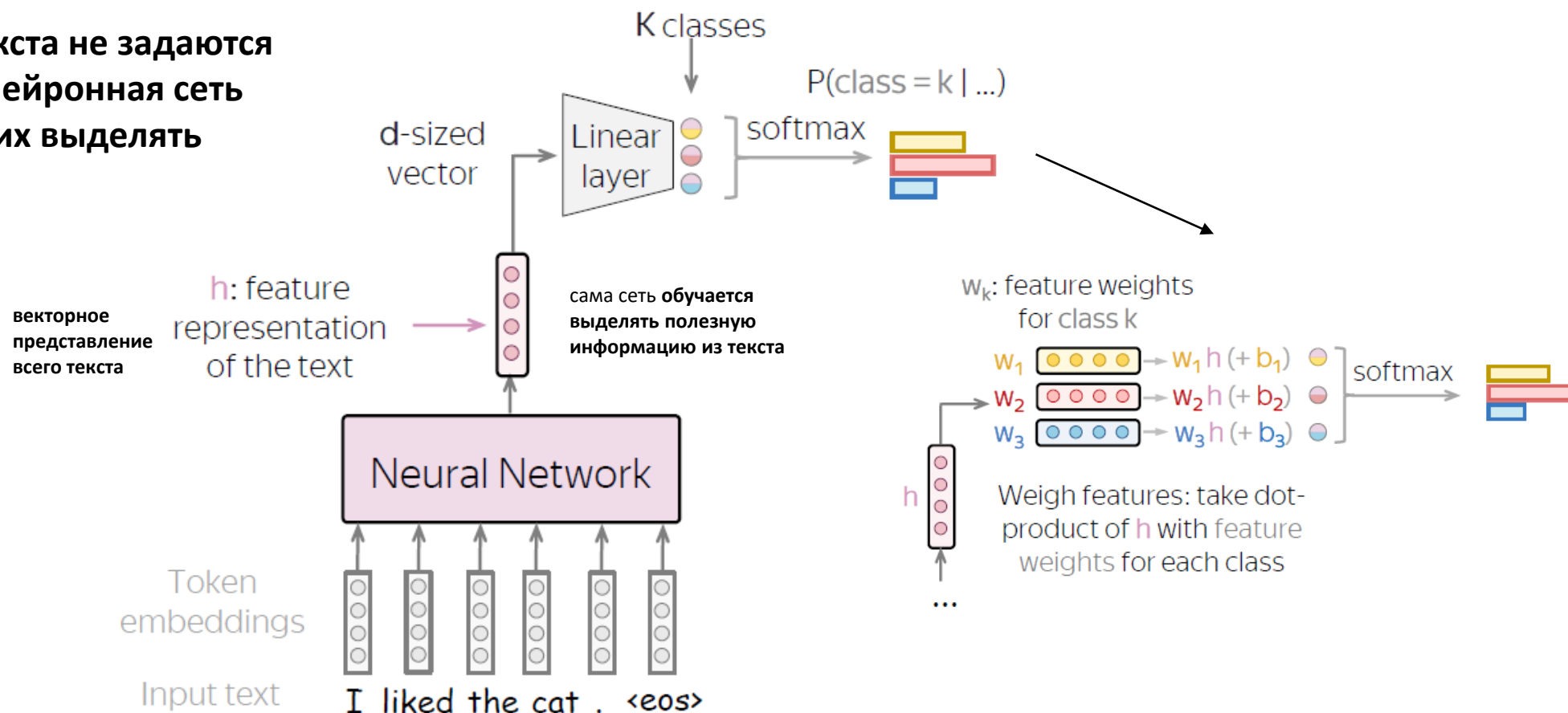
При использовании **BoW/TF-IDF** мы получаем вектор признаков, но теряем значительную часть информации о порядке слов; контексте; семантических связях; зависимости слов друг от друга.

Например: «**собака укусила человека**» и «**человек укусил собаку**» при простом BoW будут иметь практически одинаковый набор слов, хотя смысл предложений разный.

Именно здесь появляется мотивация перейти к нейросетевым методам, которые способны не только классифицировать готовые признаки, но и автоматически обучать более сложное представление текста.

Классификация с помощью нейронных сетей

Признаки текста не задаются вручную — нейронная сеть сама учится их выделять



Классификация с помощью нейронных сетей

Полученный вектор h поступает в линейный слой:

$$z_k = w_k^T h + b_k.$$

Для каждого класса k есть свои веса w_k и смещение b_k .

Например, если есть 3 класса: $k = 1, 2, 3$, получаем три значения:

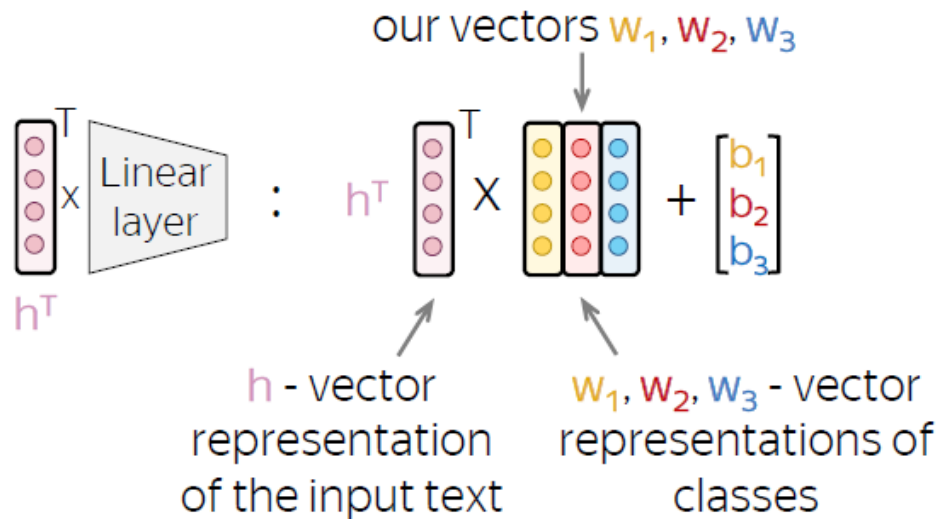
$$\begin{aligned} z_1 &= w_1^T h + b_1, \\ z_2 &= w_2^T h + b_2, \\ z_3 &= w_3^T h + b_3. \end{aligned}$$

Эти значения называются **логитами**. Затем логиты поступают в **softmax**:

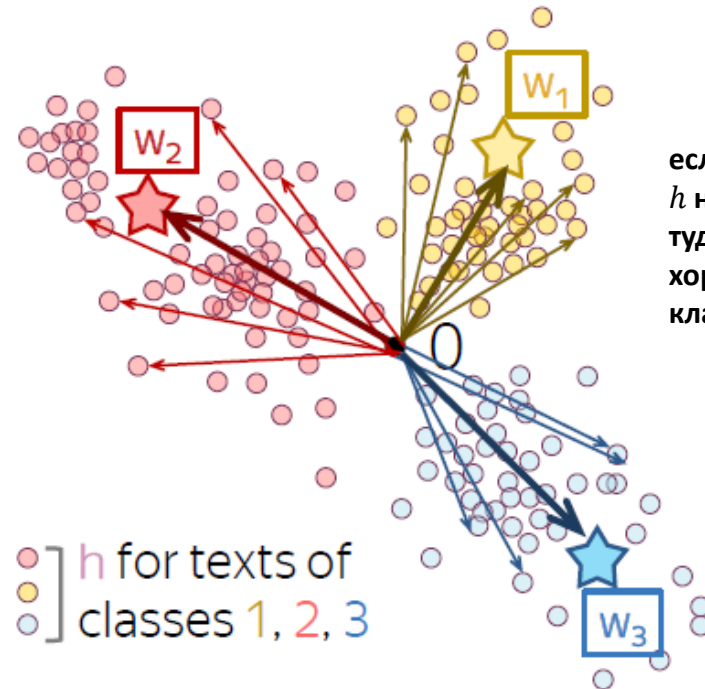
$$P(y = k | x) = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}.$$

Softmax преобразует логиты в вероятности.

Представление текста и представление классов



Векторы текста указывают в направлении соответствующих векторов класса:



если вектор текста h направлен примерно туда же, куда и w_k , то текст хорошо соответствует классу k .

Обучение: кросс-энтропия

Есть некоторый текст. Текст имеет метку k .

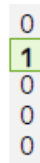
Model prediction:

$P(\text{class} = i | \text{I liked...}\langle\text{eos}\rangle)$



Target:

p^*



←

k

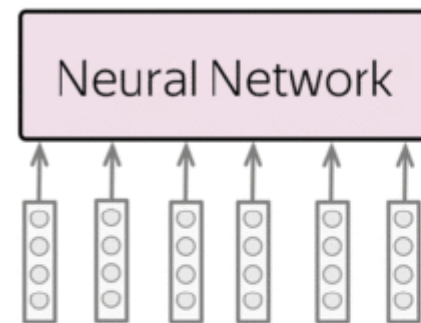
→

Cross-entropy loss:

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

For one-hot targets, this is equivalent to

$$-\log P(y = k|x) \rightarrow \min$$



Feed a text to the network

I liked the cat . $\langle\text{eos}\rangle$

Correct label: 4 ← we want the model to predict this

Обучение: кросс-энтропия

У сети есть параметры:

$$W, b.$$

Она использует их, чтобы получить предсказание:

$$x \rightarrow \text{нейросеть} \rightarrow h \rightarrow \text{Linear} \rightarrow \text{Softmax} \rightarrow P(y | x).$$

Затем считается: $L = -\log P(y = k | x)$.

После этого с помощью **backpropagation** вычисляется, как каждый параметр сети повлиял на ошибку: $\frac{\partial L}{\partial W}, \frac{\partial L}{\partial b}$.

И параметры изменяются, например градиентным спуском: $W \leftarrow W - \eta \frac{\partial L}{\partial W}$.

То есть весь цикл выглядит так:

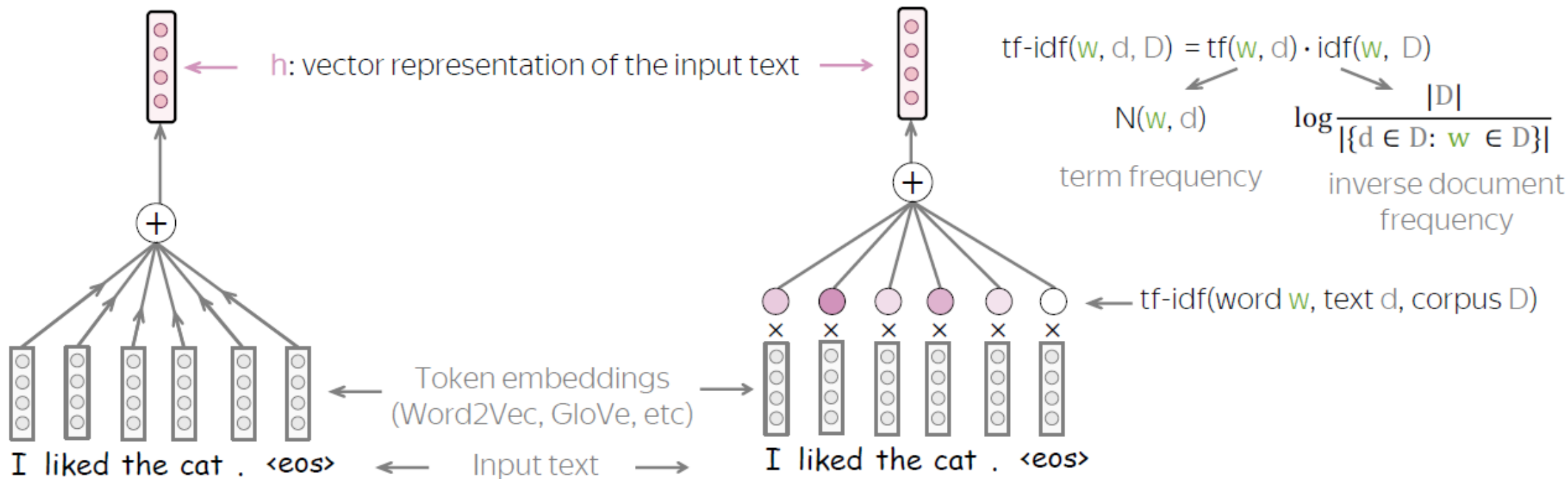
текст $\rightarrow$ предсказание $\rightarrow$ loss $\rightarrow$ градиент $\rightarrow$ изменение весов

и этот процесс повторяется много раз.

Простейшие модели: BOE и взвешенный BOE

Sum of embeddings
(BOE: Bag of Embeddings)

Weighted sum of embeddings
(e.g., using tf-idf weights)



Простейшие модели: BOE

1. BOE — Bag of Embeddings

Предположим, текст: I liked the cat

Каждое слово представлено embedding-вектором: $e_I, e_{\text{liked}}, e_{\text{the}}, e_{\text{cat}}$.

Чтобы получить один вектор всего предложения, BOE просто усредняет embeddings:

$$h = \frac{1}{n} \sum_{i=1}^n e_i$$

где n — количество слов. То есть:

$$h = \frac{e_I + e_{\text{liked}} + e_{\text{the}} + e_{\text{cat}}}{4}.$$

Получается один вектор h , который можно передать классификатору:

текст $\rightarrow$ embeddings $\rightarrow$ BOE $\rightarrow h \rightarrow$ классификатор.

Простейшие модели: взвешенный BOE

3. Взвешенный BOE

Идея простая: не все слова одинаково важны. Каждому слову назначается свой вес w_i :

Например: The movie was fantastic

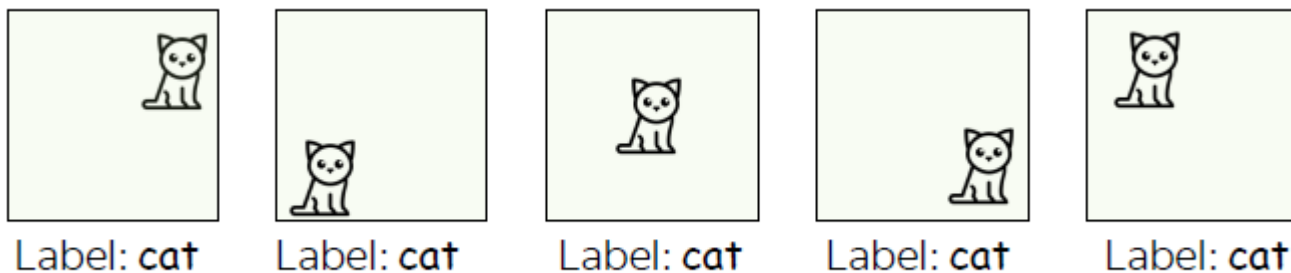
Слово the обычно менее информативно, чем fantastic.

Поэтому можно дать им разные веса: $a_{\text{the}} = 0.05$, $a_{\text{movie}} = 0.2$, $a_{\text{was}} = 0.05$, $a_{\text{fantastic}} = 0.7$.

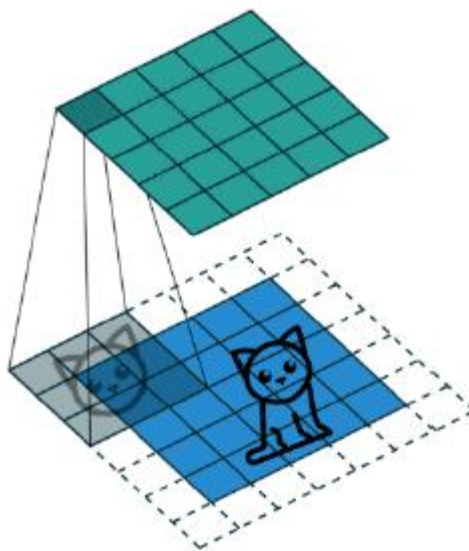
Веса узнаём с помощью TF-IDF.

Редкие и информативные слова получают больший вес, а часто встречающиеся — меньший.

Convolutional Neural Network (Свёрточная нейронная сеть)



- Применить ту же операцию к небольшим частям входных данных
- найти «совпадения» с шаблонами
- сеть узнает, какие шаблоны полезны
- шаблоны развиваются от простого к сложному



Как это использовать в текстах?

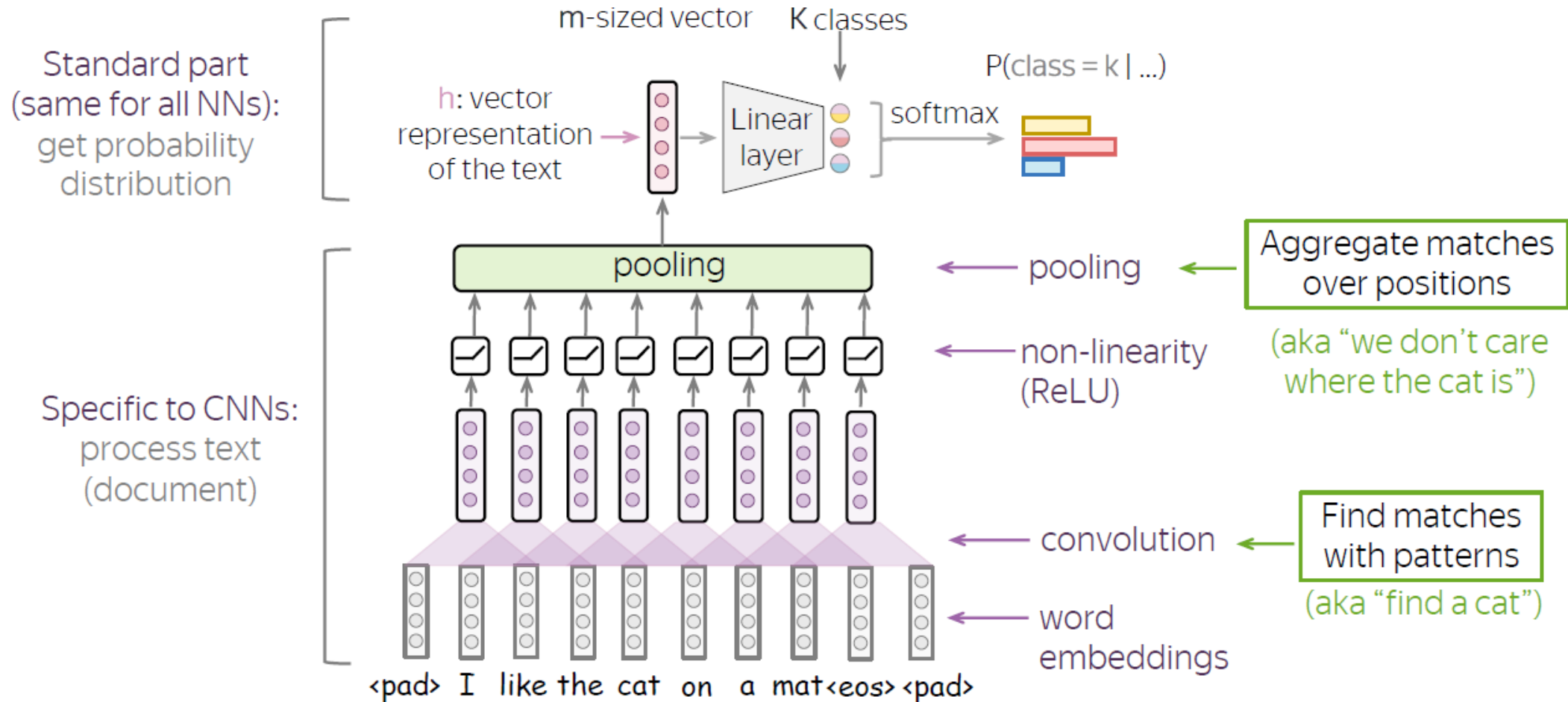
Если в тексте есть некоторая фраза, и она очень важная, то, возможно, нам не очень важно, где именно она находится.

An **absolutely great** movie! I watched the premiere with my friends.

The movie about cats was **absolutely great**, and the cats were cute.

The movie is about cats running around, and it is **absolutely great**.

TextCNN (Convolutional Neural Network for Text Classification)



Пример

Возьмём текст из картинки: «**I like the cat on a mat**»

Цель, например, определить, к какому из K классов относится этот текст.

1. Word embeddings — превращаем слова в векторы

Каждое слово представляется не просто строкой, а **вектором чисел** — embedding. Например, условно:

$$I \rightarrow \begin{bmatrix} 0.2 \\ 0.7 \\ -0.1 \\ 0.4 \end{bmatrix}; \text{ cat} \rightarrow \begin{bmatrix} 0.8 \\ 0.1 \\ 0.6 \\ -0.2 \end{bmatrix}$$

Поэтому весь текст превращается в последовательность векторов:

$$x_1, x_2, x_3, \dots, x_n$$

На картинке каждый вертикальный столбик внизу — это **вектор одного слова**.

Зачем <pad> и <eos>?

<pad> — специальный символ заполнения, чтобы все тексты имели одинаковую длину.

<eos> — **end of sentence**, конец предложения.

Пример

2. Convolution — ищем знакомые комбинации слов

Свёртка пытается найти в тексте **локальные шаблоны**.

Например, представим, что фильтр научился распознавать что-то вроде: «**very good**». Другой фильтр может распознавать: «**not good**». Третий: «**the cat**». И т.д.

Фильтр «скользит» по тексту. CNN может автоматически обнаруживать **локальные комбинации слов**, которые важны для классификации.

3. Что получается после convolution?

После свёртки мы получаем новые значения — насколько сильно каждый фильтр обнаружил свой шаблон в каждой позиции текста.

Условно:

I	like	the	cat	on	a	mat
	↑	↑				
	0.2	1.8				

Пример

4. ReLU (Rectified Linear Unit) — добавляем нелинейность

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

5. Pooling — объединяем информацию

Для классификации часто не так важно, **где именно** находится нужный шаблон. Например: «I really like the cat» и «The cat is something I really like». Pooling как раз позволяет **агрегировать информацию по позициям**.

Не так важно, **где именно** встретился шаблон.

6. Получаем вектор h

После CNN + pooling мы получаем один итоговый вектор размерности m . Каждый компонент h_i можно условно воспринимать как информацию: «Насколько сильно в тексте обнаружен определённый полезный шаблон».

Пример

7. Linear layer — классификация

Линейный слой вычисляет оценки для всех K классов:

$$z = Wh + b$$

Например, если у нас 3 класса:

$$z = \begin{bmatrix} 2.5 \\ 1.2 \\ -0.7 \end{bmatrix}$$

Это пока **не вероятности**.

Это просто оценки — насколько каждый класс подходит.

8. Softmax — превращаем оценки в вероятности

$$P(\text{class} = k \mid \text{text})$$

И сеть говорит:

текст относится к классу «кино» с вероятностью 85%.

Вывод

CNN:

Свертка ищет ключевые слова (паттерны) в любом месте текста.

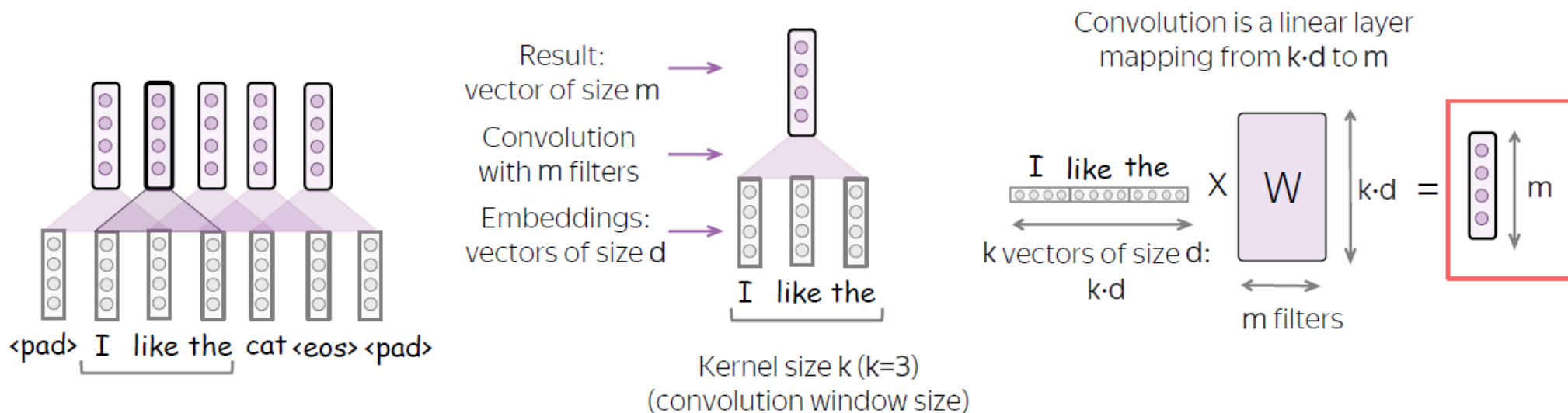
Пулинг оставляет только сам факт наличия этих слов, игнорируя их позицию.

Классификатор на основе этих фактов выдает вердикт.

Главное отличие от обычных нейросетей:

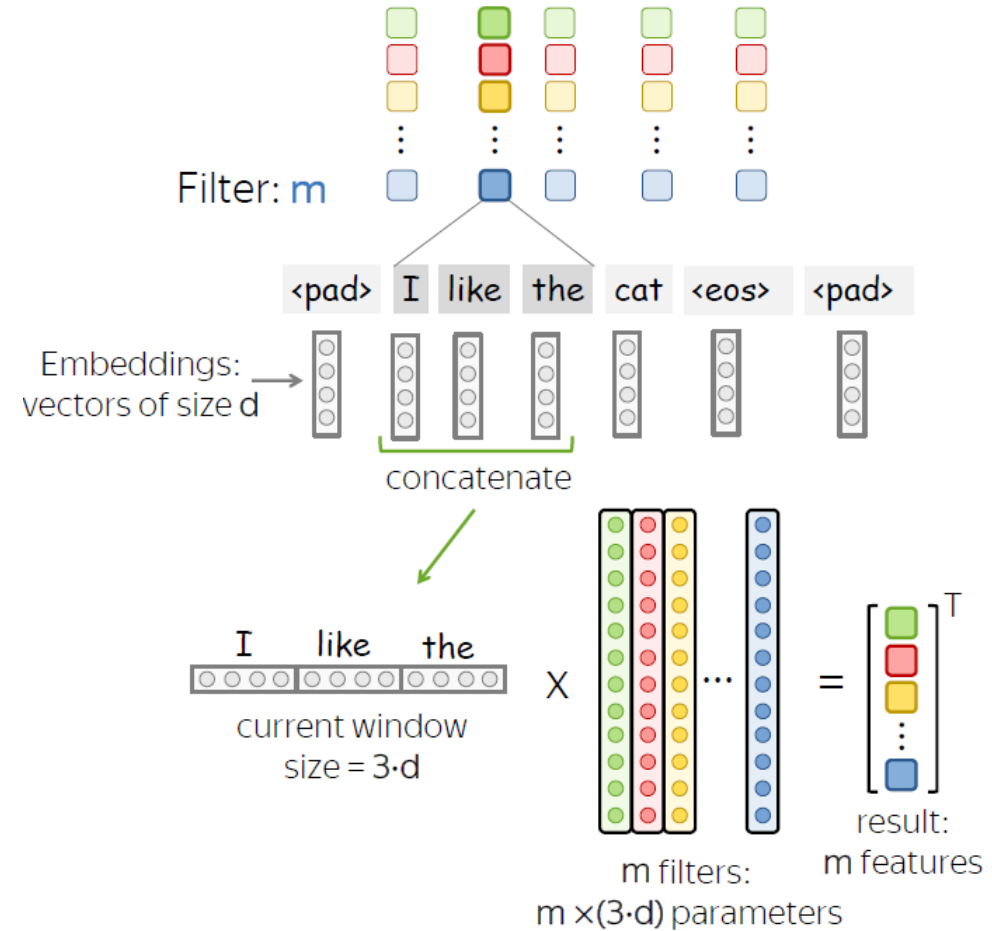
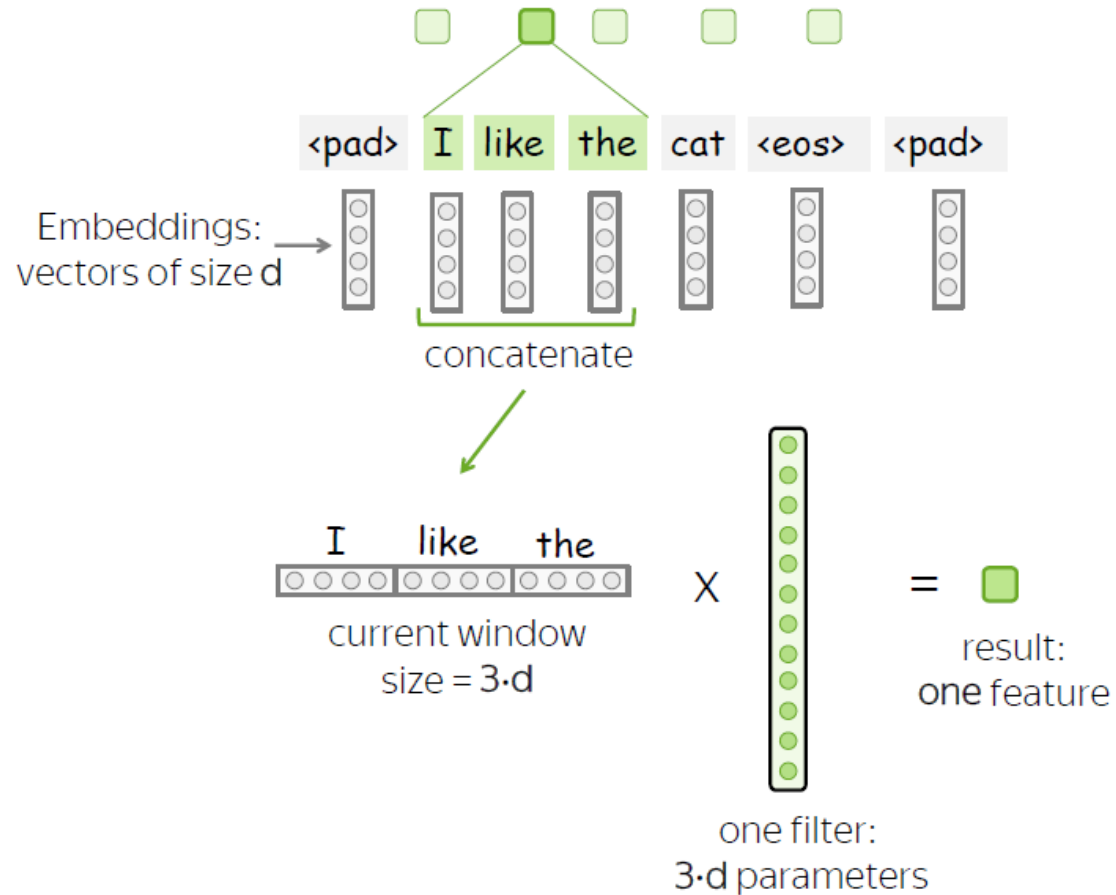
Обычная сеть смотрит на текст как на мешок слов и теряет порядок. CNN же смотрит на **последовательности** (окна по 2-3 слова), поэтому она понимает контекст, например, разницу между «не плохо» и «плохо не».

Сверточный слой (Convolutional Layer)



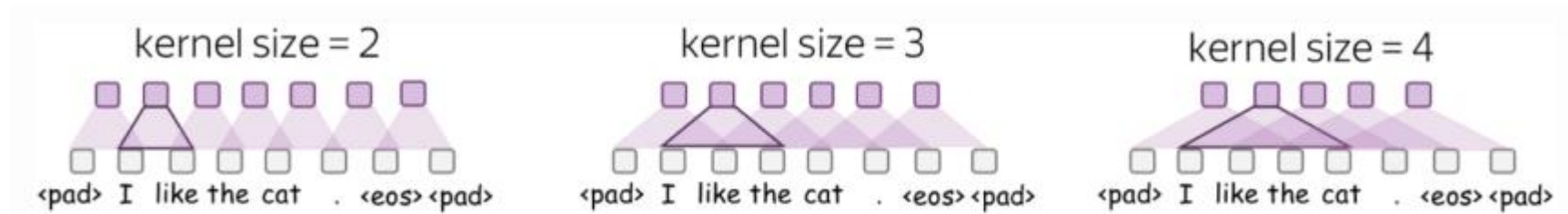
- $x_1, x_2, \dots, x_n$ - input vectors (e.g., word embeddings)
 - d (input channels) – input vector size
 - k (kernel size) – conv. window length
 - m (output channels) – number of filters
- $u_i = [x_i, \dots, x_{i+k-1}] \in \mathbb{R}^{k \cdot d}$ - concatenate representations in the i -th window
 $W \in \mathbb{R}^{(k \cdot d) \times m}$ - convolution
 $F_i = u_i \times W$ - convolution applied to the i -th window

Фильтр

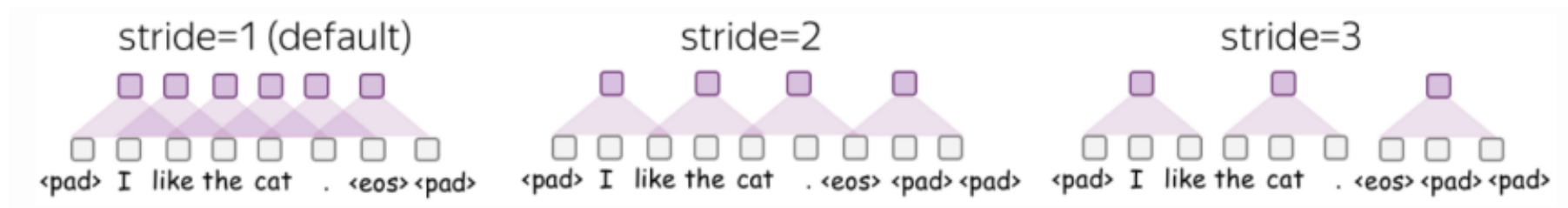


Свёртка: параметры

Размер ядра: сколько токенов за один раз вы смотрите



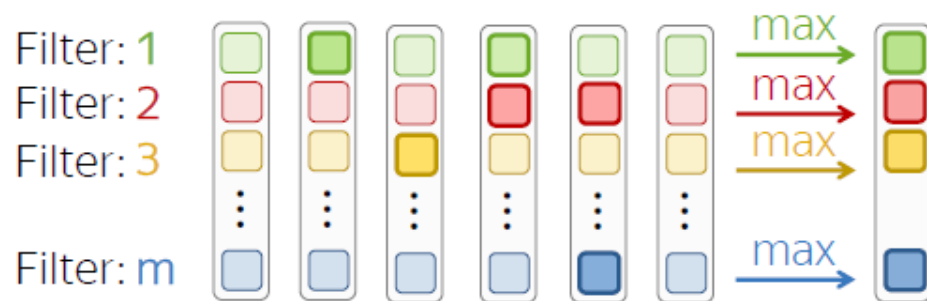
Шаг: насколько нужно переместить фильтр на каждом шаге



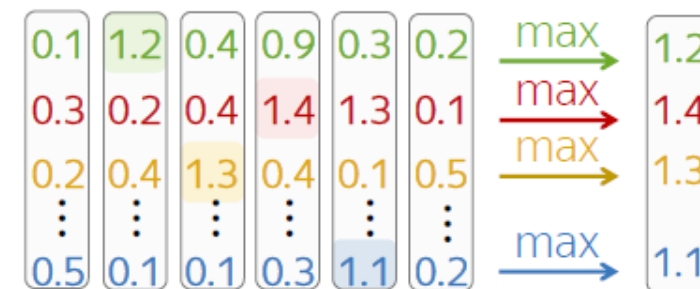
Pooling (max, mean, etc) (объединение)

Max pooling: максимальное значение для каждого измерения (признака)

Mean pooling: среднее значение для каждого измерения (признака) (не игнорирует частоту)



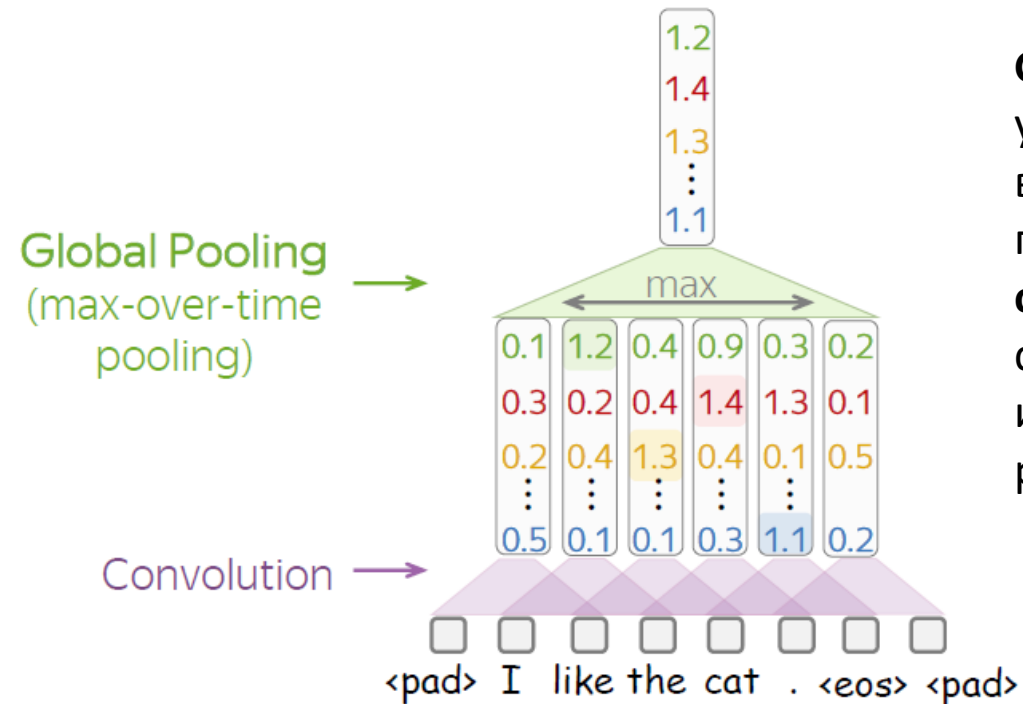
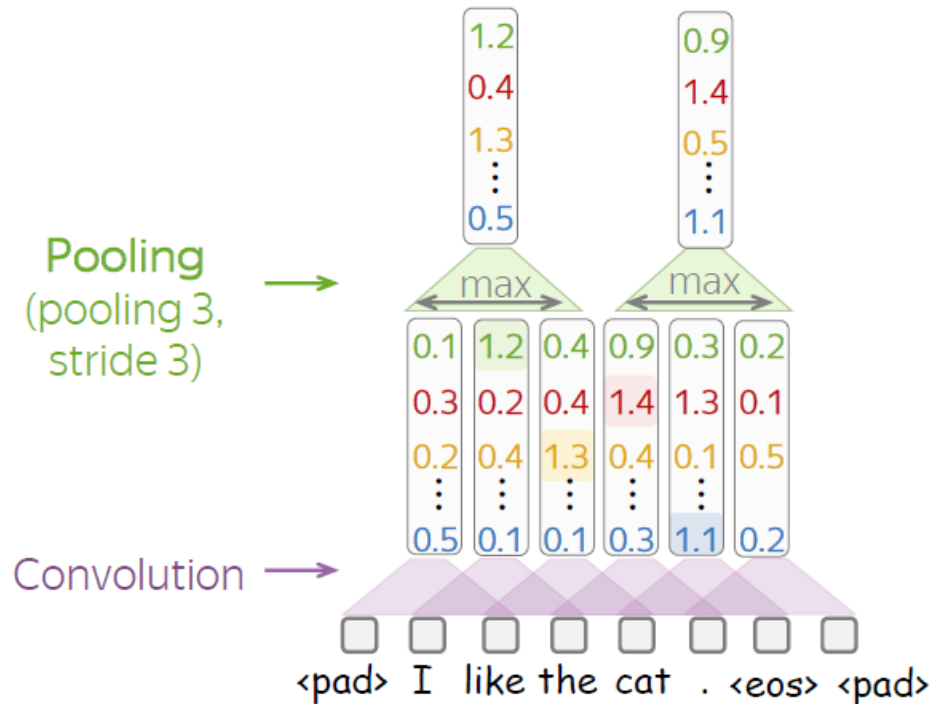
Max pooling:
maximum for each
dimension (feature)



Pooling and Global Pooling

Pooling: агрегация признаков в некоторой области

Global pooling: агрегация признаков по всем входным данным

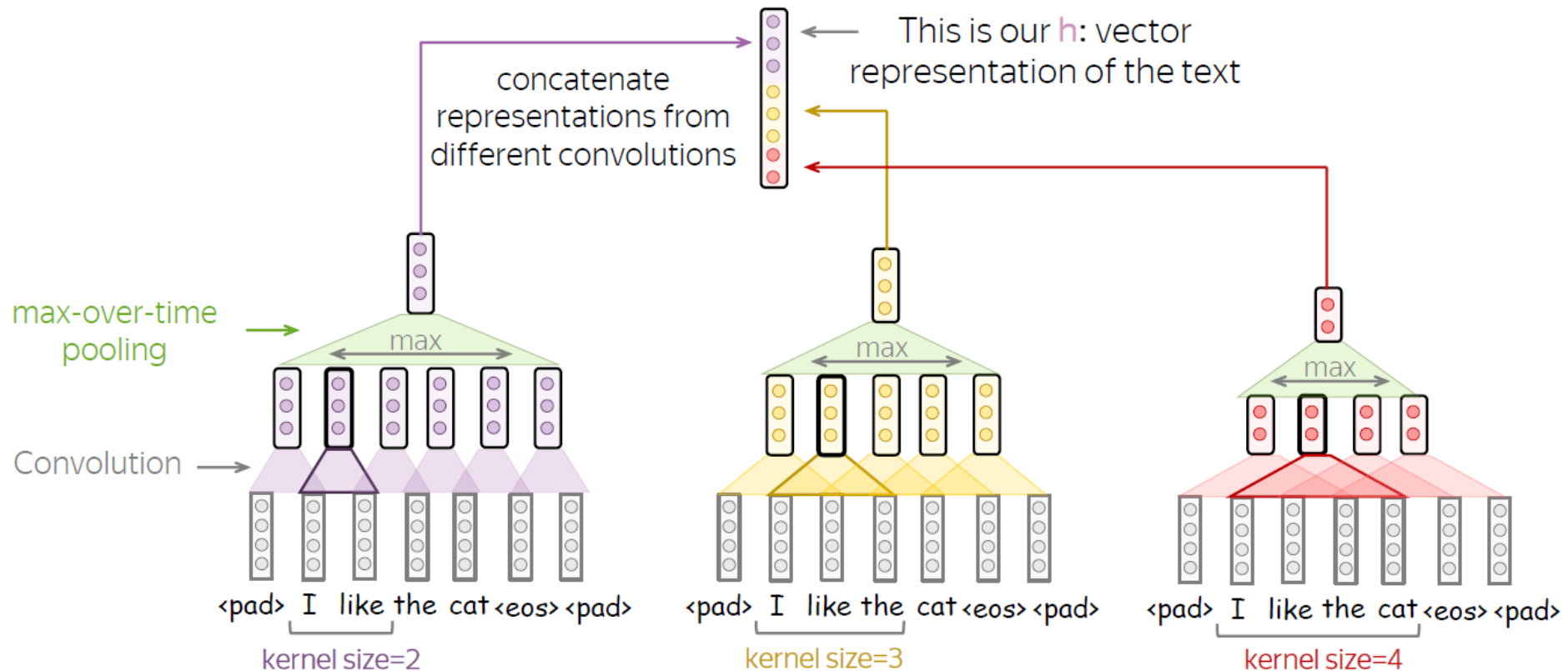


Global Pooling - учитывает только выраженность признака;

обычный pooling - сохраняет некоторую информацию о расположении

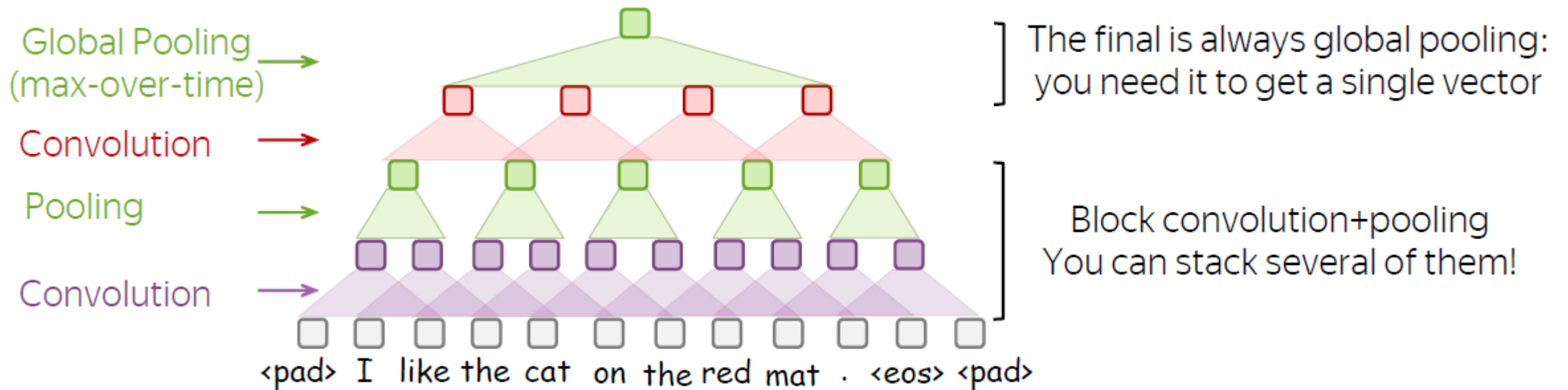
Сверточные модели для классификации текстов

Несколько сверток с разными размерами ядра запускаются параллельно

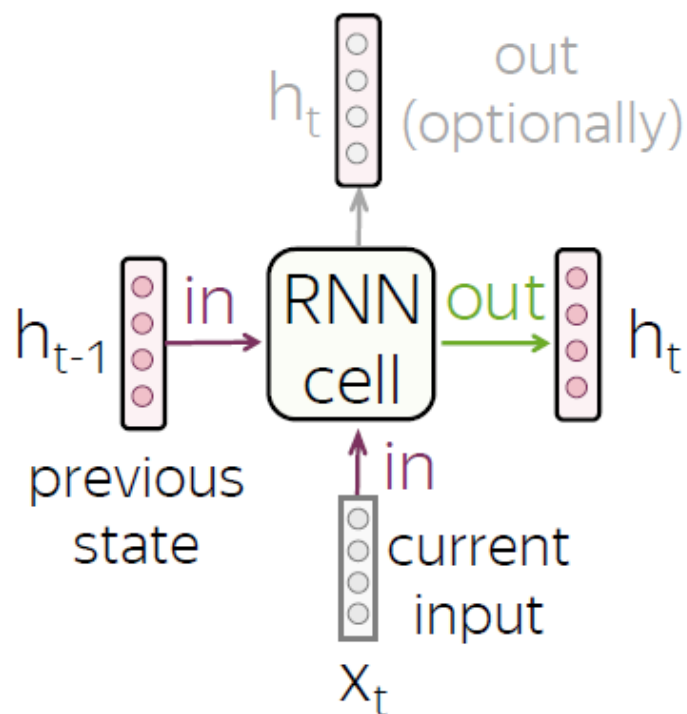


Сверточные модели для классификации текстов

Стек из нескольких блоков: свертка+пулинг. Главная идея - **каждый следующий блок может строить более сложные признаки из признаков предыдущего блока.**



Рекуррентные нейронные сети



Рекуррентная нейронная сеть (RNN, Recurrent Neural Network) читает текст последовательно, слово за словом и сохраняет информацию о предыдущих словах.

Рекуррентные нейронные сети

1. Главная идея RNN

Возьмём предложение: «Этот фильм был очень интересным»

RNN читает его последовательно:

Этот → фильм → был → очень → интересным

На каждом шаге у неё есть **скрытое состояние** h_t , которое можно понимать как некоторую «память» о том, что сеть уже прочитала.

Схематично:

$$\begin{aligned}x_1 &\rightarrow h_1 \\x_2 + h_1 &\rightarrow h_2 \\x_3 + h_2 &\rightarrow h_3\end{aligned}$$

и так далее.

Здесь:

x_t — вектор текущего слова (обычно embedding);

h_t — скрытое состояние после прочтения слова t ;

h_{t-1} — информация, накопленная до текущего слова.

Рекуррентные нейронные сети

2. Как происходит вычисление

На каждом шаге RNN получает два входа: текущее слово x_t ; предыдущее состояние h_{t-1} .

И вычисляет новое состояние: $h_t = f(W_x x_t + W_h h_{t-1} + b)$

где f — функция активации (как ReLU в CNN).

3. Что в итоге используется для классификации?

Для классификации часто используют **последнее скрытое состояние** h_T . Оно должно содержать информацию обо всём прочитанном тексте. Затем:

$$h_T \rightarrow \text{Linear} \rightarrow \text{Softmax}$$

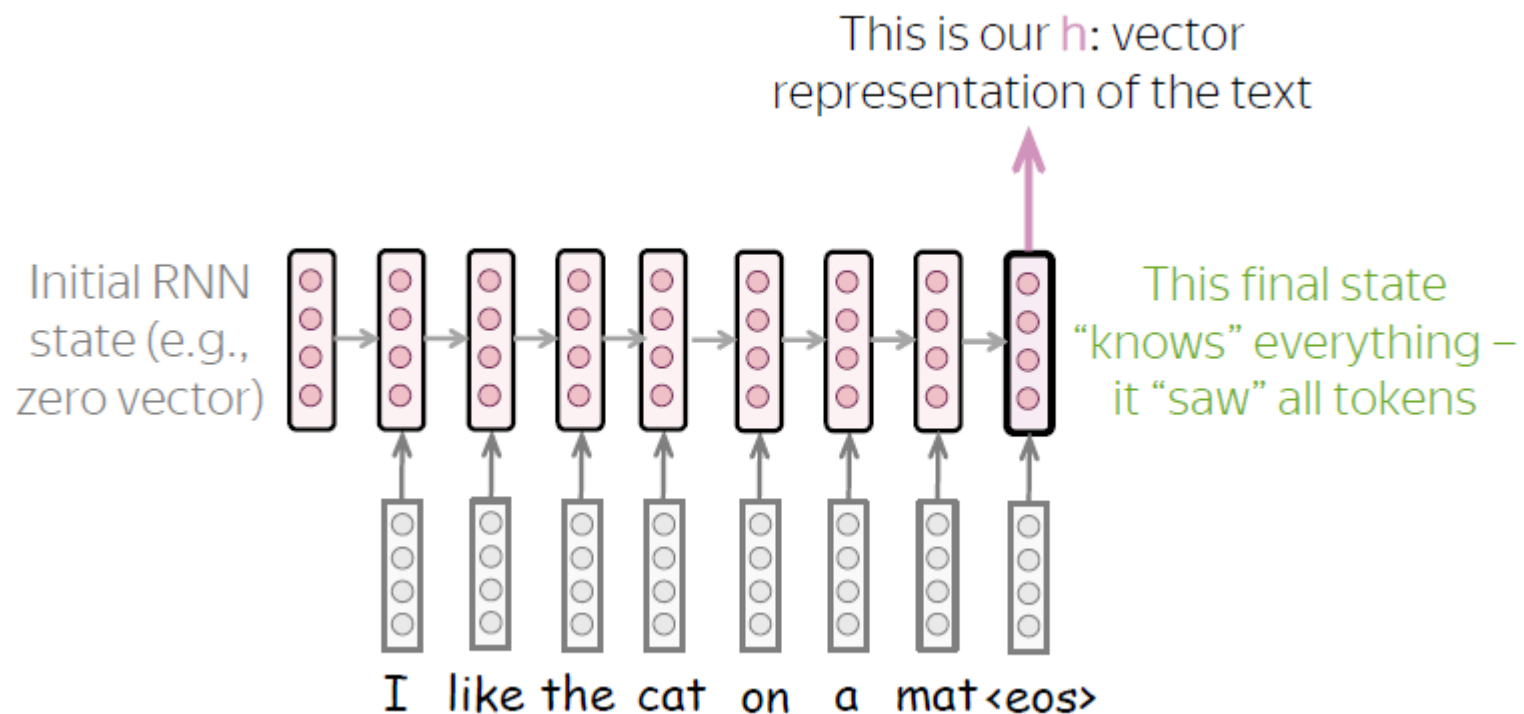
Например:

$$h_T \rightarrow W h_T + b \rightarrow \begin{cases} P(\text{положительный}) = 0.92 \\ P(\text{отрицательный}) = 0.08 \end{cases}$$

4. Почему RNN подходит для текста? Потому что **порядок слов имеет значение**. Например: «Я люблю этот фильм» и «Этот фильм любит меня» содержат примерно те же слова, но смысл разный.

Рекуррентные модели для классификации текстов

просто: прочитайте текст,
определите конечное состояние



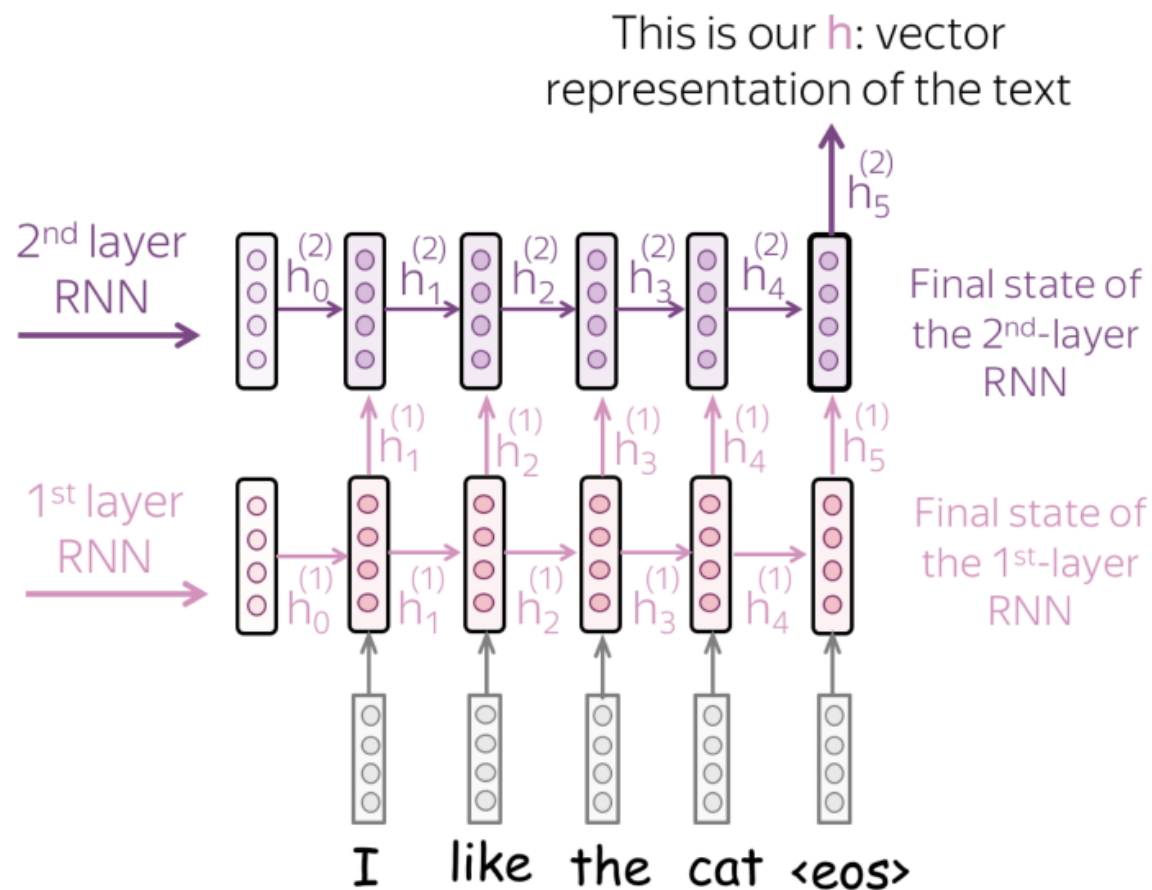
Рекуррентные модели для классификации текстов

Обычная (однослойная) RNN читает текст и запоминает его в одном векторе состояний. Но для сложных задач одного слоя может не хватать. Поэтому используют «глубокие» RNN — они состоят из нескольких слоев, наложенных друг на друга.

Как это работает:

Первый слой (1st layer RNN) получает на вход слова. Он обрабатывает их последовательно и на каждом шаге выдает скрытое состояние $h^{(1)}$.

Второй слой (2nd layer RNN) получает на вход не исходные слова, а выходы первого слоя. Он строит свою собственную последовательность состояний $h^{(2)}$, которая представляет собой более абстрактное представление текста.



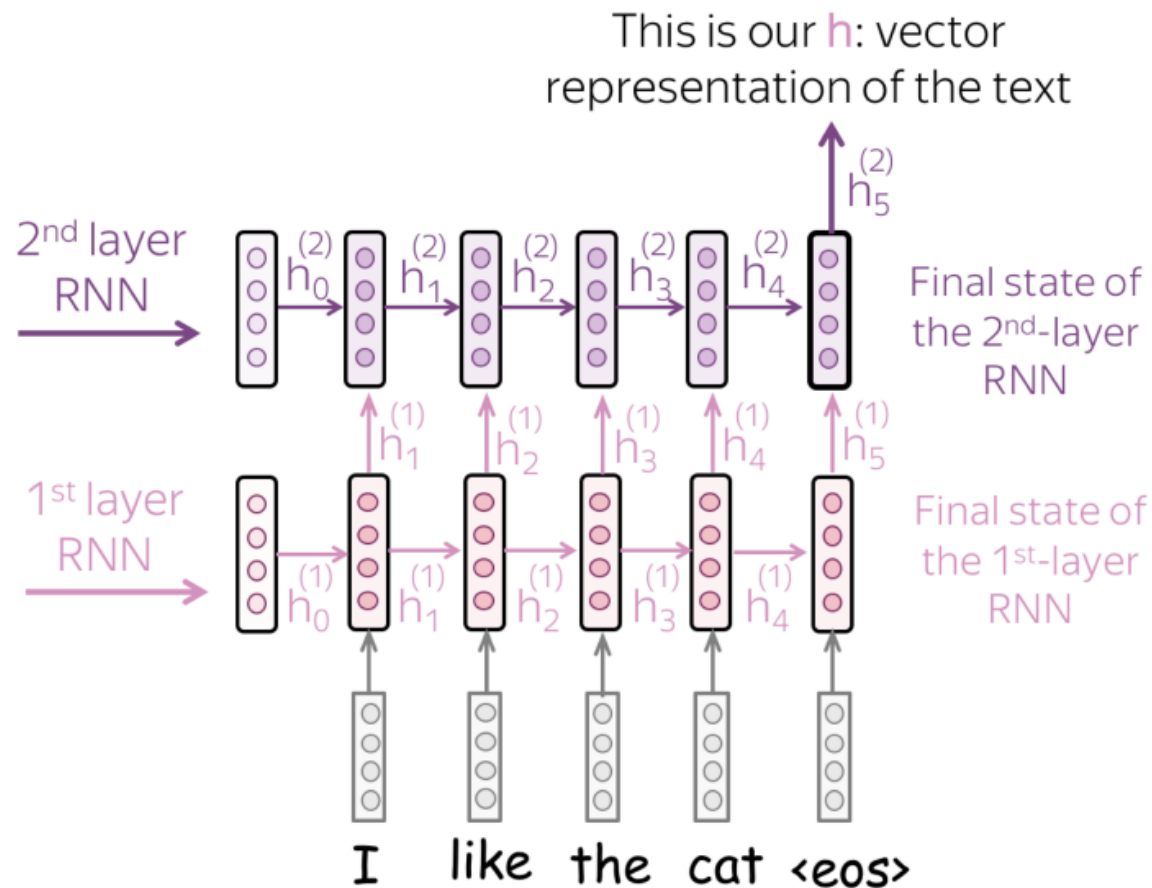
Рекуррентные модели для классификации текстов

Проблема!

1. В математике RNN это работает так: состояние памяти (h) на каждом шаге умножается на матрицу весов (W) и пропускается через функцию активации. Если веса маленькие (меньше 1), то при перемножении много раз они стремятся к нулю.

$$0.9 \times 0.9 \times 0.9 \dots \approx 0$$

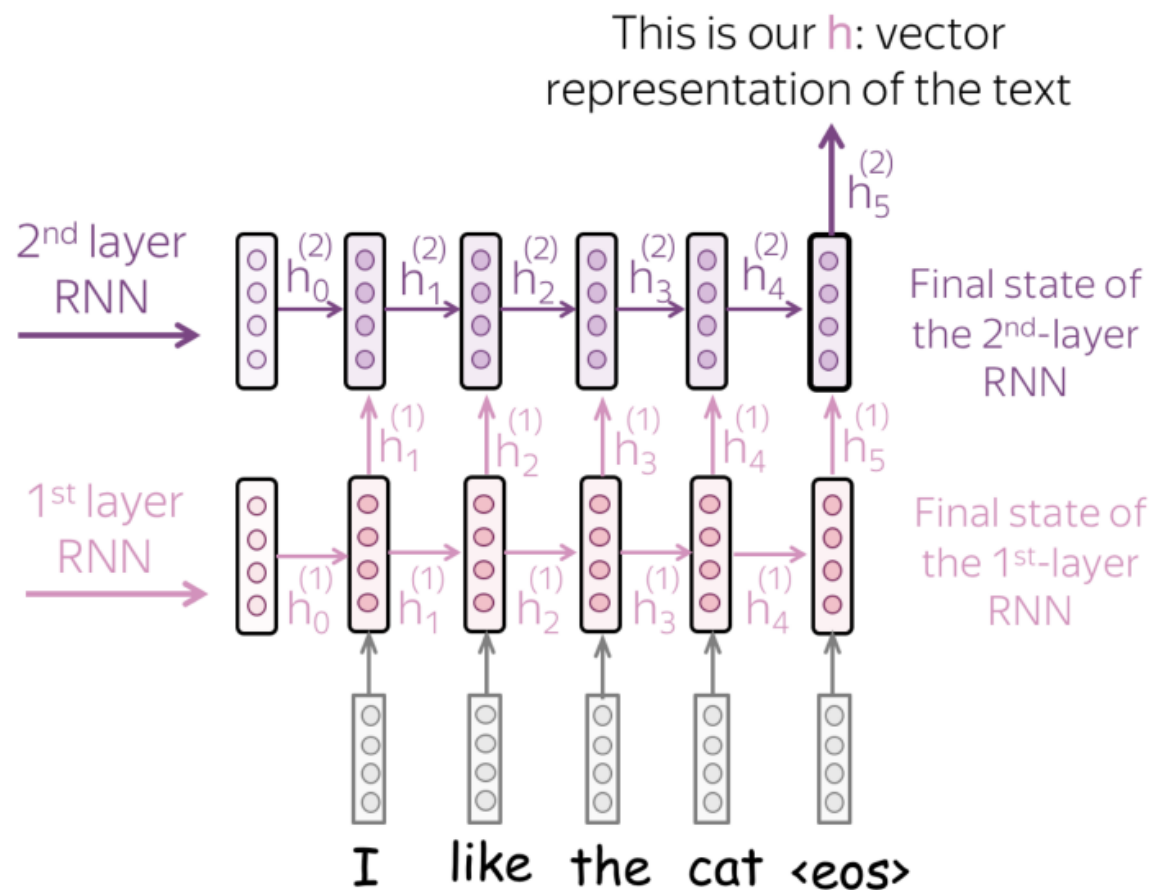
Это называется **затуханием градиента**. Сигнал об ошибке просто не доходит до первых слов, и сеть не понимает, что их нужно учитывать.



Рекуррентные модели для классификации текстов

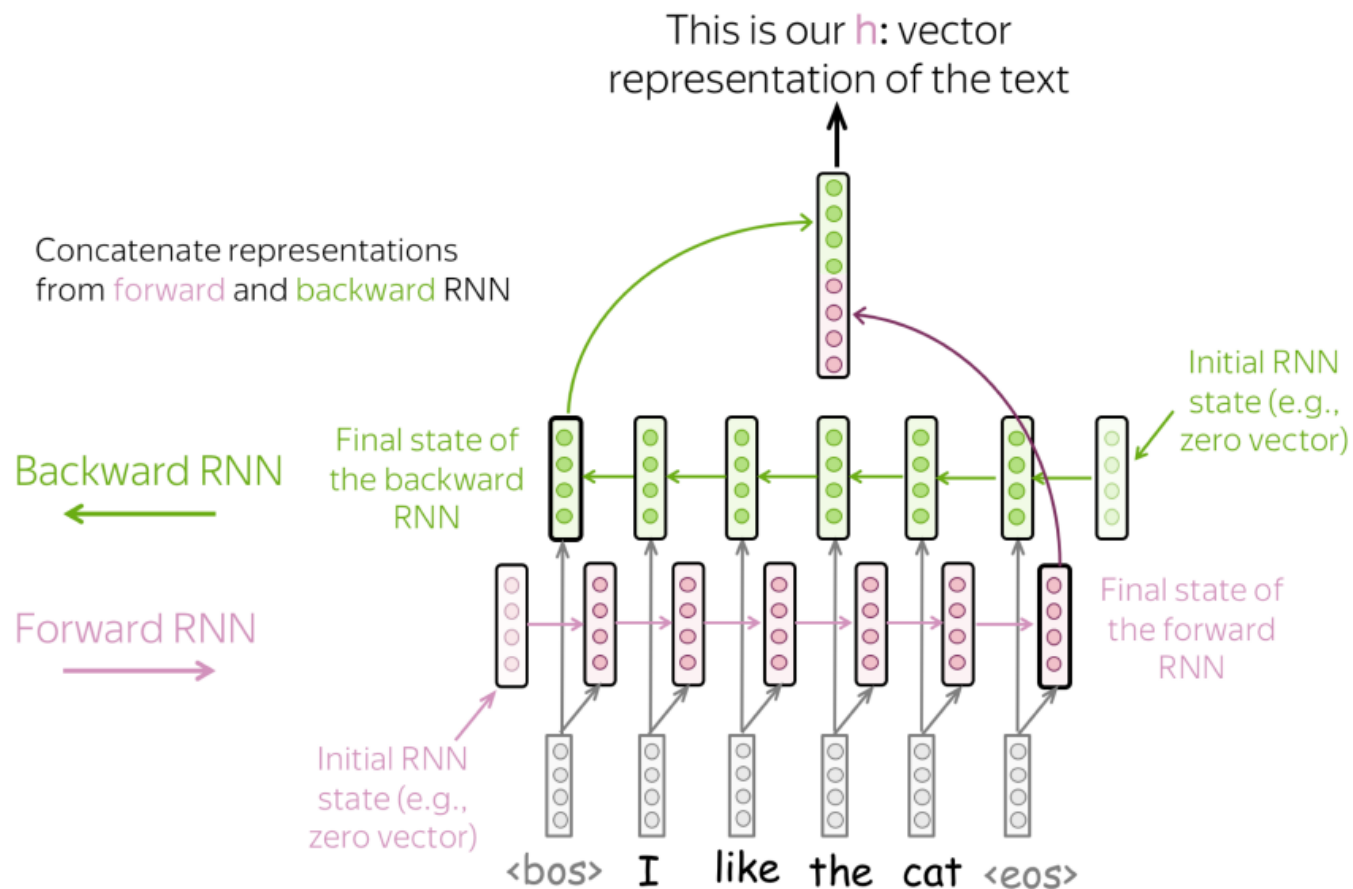
Проблема!

2. У обычной RNN есть только один вектор состояния (h_t), который передается по цепочке. Этот вектор имеет фиксированный размер. На каждом шаге сеть **полностью перезаписывает** этот вектор, смешивая старую информацию с новой.



Рекуррентные модели для классификации текстов

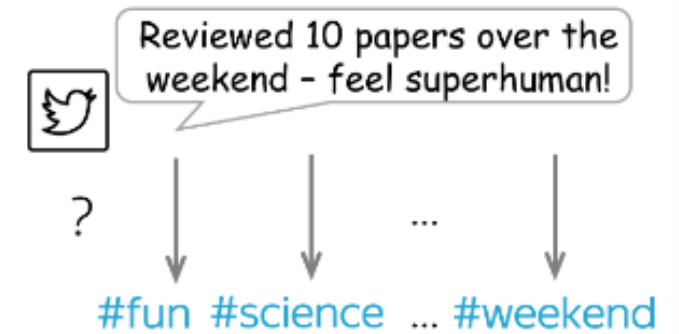
Два направления: использовать конечные состояния из прямых и обратных RNN



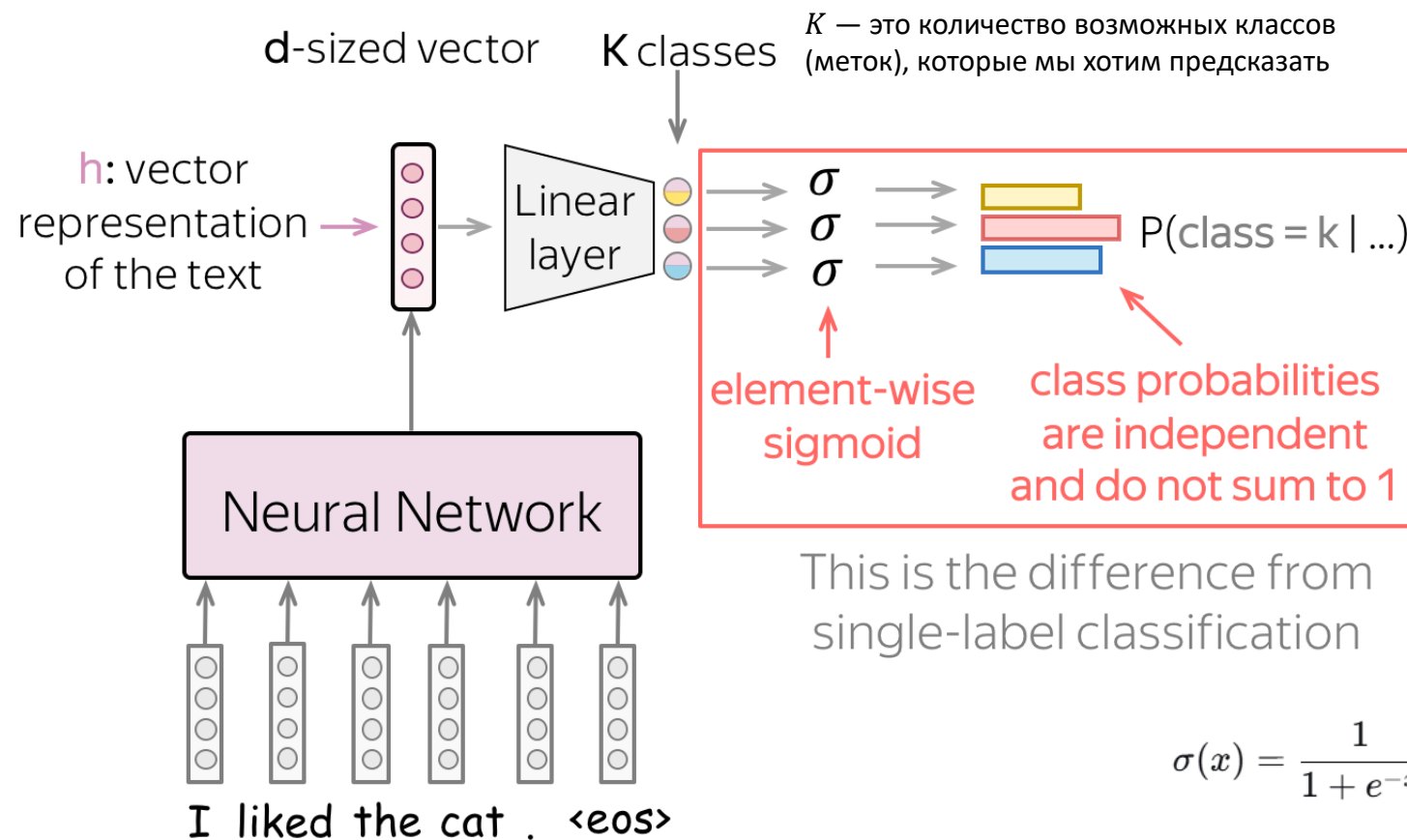
Многометочная классификация

Для задачи с несколькими метками нам нужно изменить два элемента в схеме с одной меткой:

- модель (способ оценки вероятностей классов);
- функцию потерь



Многометочная классификация



Функция потерь: бинарная кросс-энтропия для каждого класса

Training example: I liked the cat on the mat <eos>

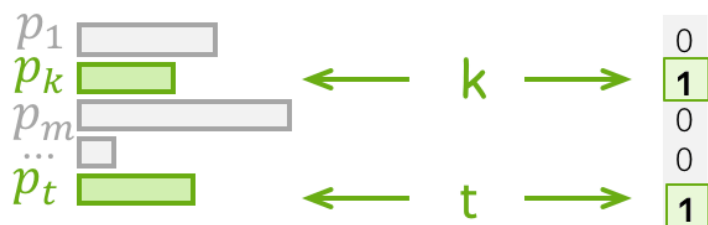
Labels: k, t
↑
target

Model prediction:

$P(y_i = 1 | \dots \text{<eos>}), i = 1..K$

Target:

$p_i^*, i = 1..K$



Binary cross-entropy loss for each class:

$$-\sum_{i=1}^K [p_i^* \cdot \log P(y_i = 1|x) + (1 - p_i^*) \cdot \log P(y_i = 0|x)] \rightarrow \min,$$

where $P(y_i = 0|x) = 1 - P(y_i = 1|x)$

binary classifier loss for class i

Практические советы. Что делать с эмбедингами?

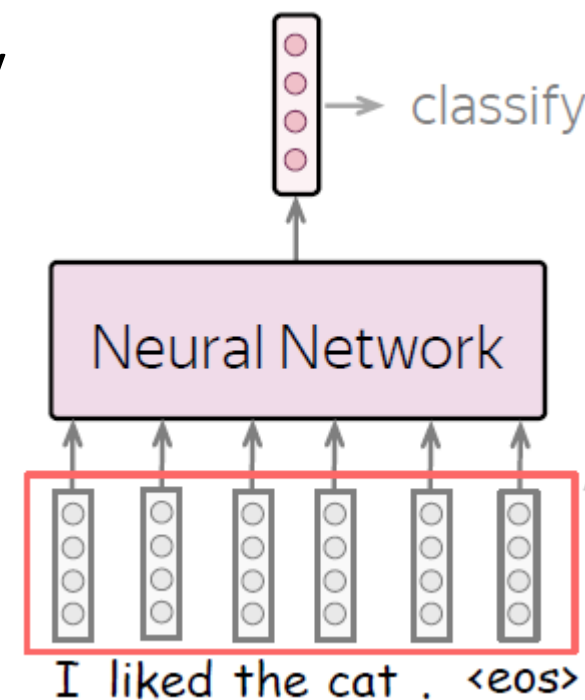
Входные векторные представления слов:

- Обучение с нуля (создаём таблицу, где каждому слову словаря соответствует случайный вектор чисел.

В процессе обучения нейросеть сама подстраивает эти числа, чтобы они наилучшим образом решали вашу задачу)

- Использование предварительно обученных данных (Word2Vec, GloVe)

- Инициализация с использованием предварительно обученных данных, затем тонкая настройка (сочетание первого и второго подходов)



Что делать с эмбедингами?

- обучение с нуля (данных может быть недостаточно, чтобы изучить взаимосвязи между словами)
- предобученные данные (Word2Vec, GloVe) (знают взаимосвязи между словами, но не знают специфику задачи)
- инициализация с помощью предобученных данных, затем тонкая настройка (знают взаимосвязи между словами и адаптируются к задаче)

Получение дополнительных данных (Data Augmentation)

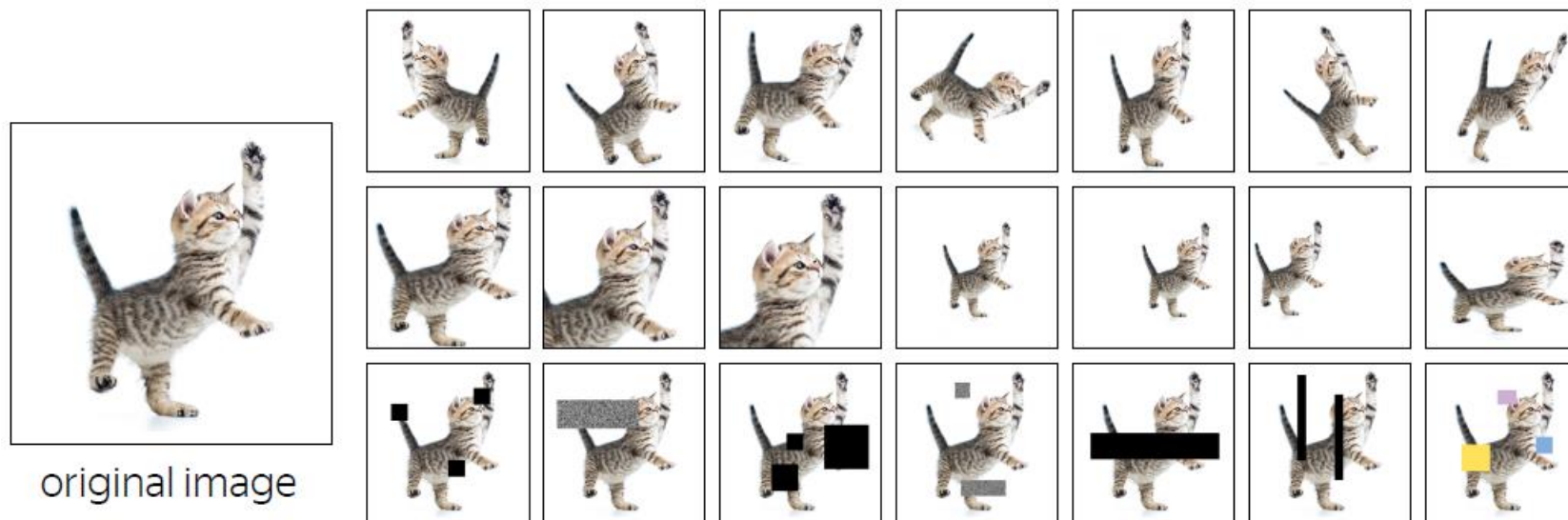
Аугментация данных изменяет ваш набор данных различными способами, чтобы получить альтернативные версии одного и того же обучающего примера.

Она может увеличить:

- объем данных
- разнообразие данных

Data Augmentation для изображений

- переворот изображения
- геометрические преобразования (вращение, растяжение, увеличение/уменьшение масштаба)
- скрывание отдельных участков



Data Augmentation для текста

- word dropout (исключение слов)

The movie **about** cats was absolutely **great**, and the **cats** were cute.

replace with UNK

pick several words randomly

replace with random words

The movie **UNK** cats was absolutely **UNK**, and the **UNK** were cute.

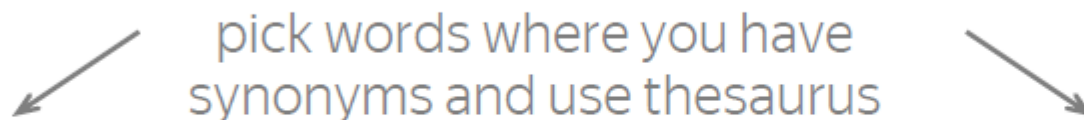
The movie **mejorate** cats was absolutely **fellows**, and the **mak** were cute.

Data Augmentation для текста

- использование внешних ресурсов (например, тезауруса)

The **movie** about cats was **absolutely** great, and the cats were **cute**.

pick words where you have
synonyms and use thesaurus

A diagram illustrating text augmentation. It shows a central instruction 'pick words where you have synonyms and use thesaurus' with two arrows pointing downwards to two modified sentences. The first arrow points to the left, and the second points to the right.

The **film** about cats was **certainly** great, and the cats were **nice**.

The **video** about cats was **completely** great, and the cats were **charming**.

Data Augmentation для текста

- использование отдельных моделей для перефразирования

The movie about cats was absolutely great, and the cats were cute.



Задание 5 (2 балла)

1. В чем главное преимущество сверточных нейронных сетей (CNN) перед простыми моделями на основе «мешка слов» (Bag of Words) при работе с текстом?
2. Что означают параметры «размер ядра» (kernel size) и «шаг» (stride) в сверточном слое для текста? Как изменение размера ядра влияет на то, какие фразы «видит» сеть?
3. В чем принципиальное отличие рекуррентных нейронных сетей (RNN) от сверточных (CNN) с точки зрения обработки последовательности слов?
4. С какими двумя главными проблемами сталкивается обычная (однослойная) RNN при обработке длинных текстов?