ОСНОВЫ ОБРАБОТКИ ТЕКСТОВ

Лекция #5:
Векторные представления слов

Лектор: м.н.с. ИСП РАН Майоров Владимир Дмитриевич

Векторное представление слов

- Атомарные единицы текста – слова

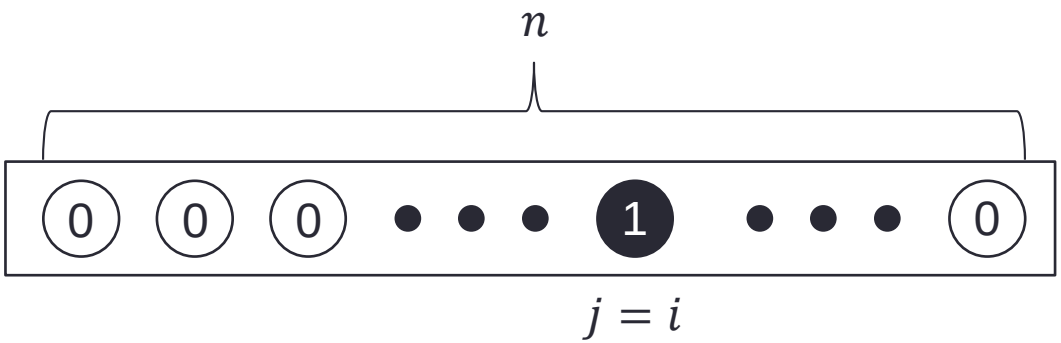
- Word embedding – вещественный вектор в пространстве с фиксированной размерностью
  - Пусть есть словарь всех слов языка $V = \{w_i\}$ размером $n = |V|$
  - Пусть задана фиксированная размерность d
  - Каждому слову $w_i \in V$ ставится в соответствие вектор из $\mathbb{R}^d$

Векторное представление слов

- Атомарные единицы текста – слова
- Word embedding – вещественный вектор в пространстве с фиксированной размерностью
  - Пусть есть словарь всех слов языка $V = \{w_i\}$ размером $n = |V|$
  - Пусть задана фиксированная размерность d
  - Каждому слову $w_i \in V$ ставится в соответствие вектор из $\mathbb{R}^d$
- На предыдущих лекциях было предложено представление слов в виде one-hot векторов

One-hot векторы

- Пусть есть словарь всех слов языка $V = \{w_i\}$ размером $n = |V|$
- Пусть задана фиксированная размерность $d = n$
- Каждому слову $w_i \in V$ ставится в соответствие вектор из $\mathbb{R}^d$, в котором $w_{ij} = \begin{cases} 1, j = i \\ 0, j \neq i \end{cases}, j = \overline{1, d}$



Синонимия в задачах NLP

- Для большинства задач NLP важен смысл слова (*лексическое значение*), а не само слово:

16 августа 1820 года Пушкин ***прибыл*** в Феодосию
приехал
пожаловал
…

- Похожесть слов (косинусная мера)

$$similarity(w_i, w_j) = \frac{(w_i, w_j)}{\|w_i\|\|w_j\|} = \frac{\sum_{k=1}^{n} w_{ik} * w_{jk}}{\sqrt{\sum_{k=1}^{n}(w_{ik})^2} * \sqrt{\sum_{k=1}^{n}(w_{jk})^2}}$$

Синонимия в задачах NLP

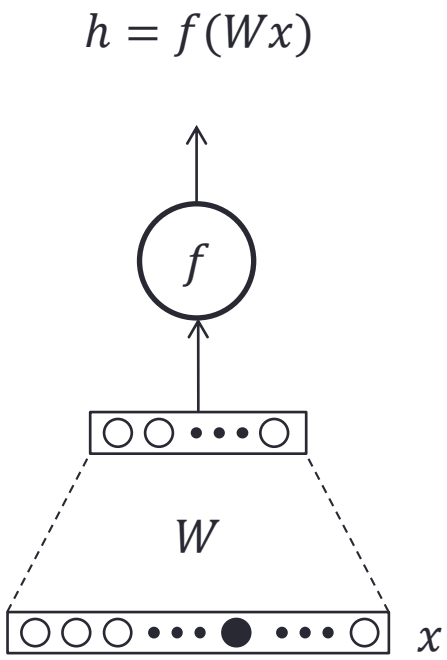
- One-hot векторы
  - Пусть есть словарь всех слов языка $V = \{w_i\}$ размером $n = |V|$
  - Пусть задана фиксированная размерность $d = n$
  - Каждому слову $w_i \in V$ ставится в соответствие вектор из $\mathbb{R}^d$, в котором $w_{ij} = \begin{cases} 1, j = i \\ 0, j \neq i \end{cases}, j = \overline{1, n}$

- все слова одинаково непохожи:

$$(w_i, w_j) = 0, \qquad i \neq j$$
$$similarity(w_i, w_j) = 0, \qquad i \neq j$$

Embedding слой в нейронной сети

- На вход сети подается слово w (one-hot вектор x)
- Перед активацией первого слоя линейное преобразование Wx – векторное представление слова w



- Проблема:
  - Для большинства задач NLP обучающих данных мало ⇒ построить «хорошую» матрицу W не удастся
- Решение:
  - Инициализировать матрицу W посчитанными заранее «хорошими» векторами

Векторное представление слова

Задача обучения без учителя (unsupervised learning):

- По коллекции объектов (обучающей выборке) определить внутренние взаимосвязи, зависимости, существующие между объектами

По коллекции неразмеченных текстов построить векторные представления слов из этих текстов (причем хочется, чтобы $similarity$ близких по значению слов была выше, чем для различных по значению)

Дистрибутивная гипотеза

- Слова, которые встречаются в схожих контекстах, имеют схожий смысл

... пил крепкий **кофе** у себя …
… подаётся чёрный **кофе** без всякого …
ведь не **кофе** пить зашёл
……

… умеренно крепкий **чай** с лимоном …
… завариваем чёрный **чай** кипящим …
… чтоб мне **чай** всегда пить …
……

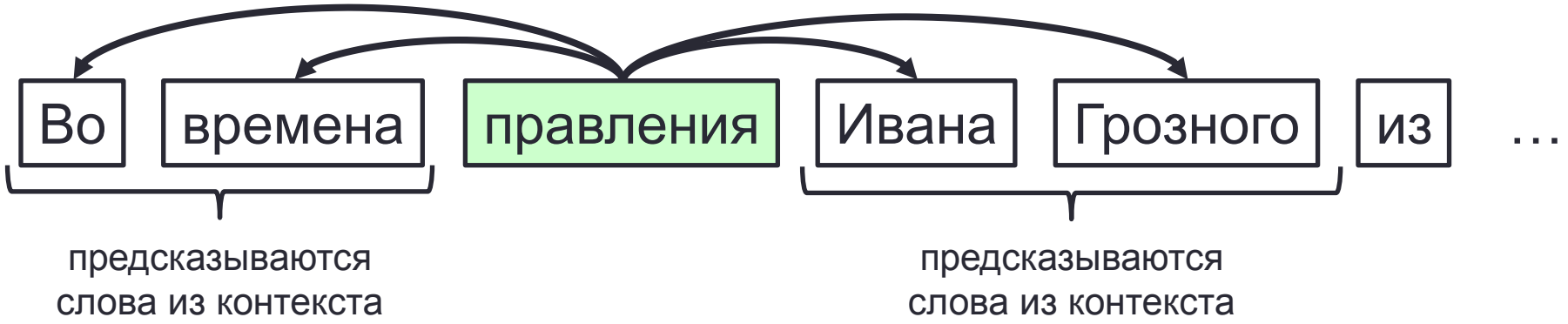
Дистрибутивная гипотеза

- Слова, которые встречаются в схожих контекстах, имеют схожий смысл

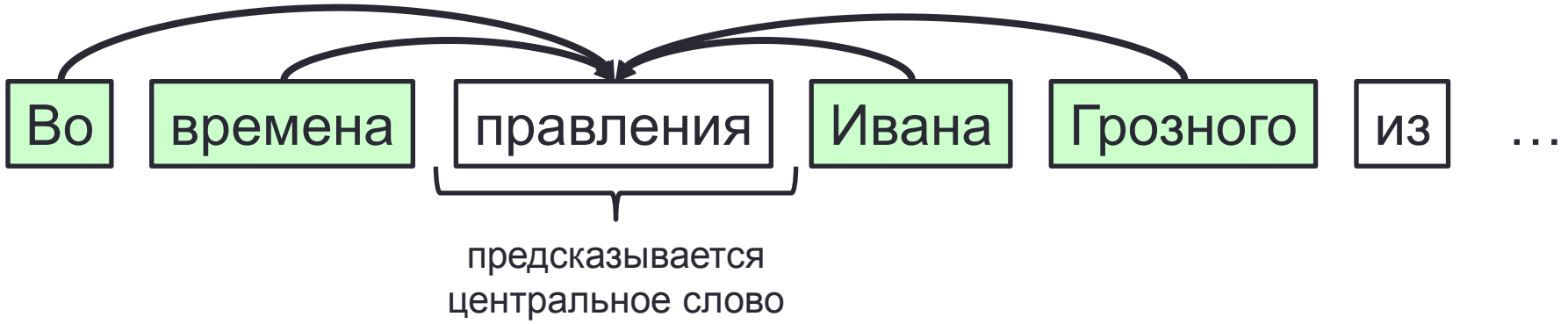
- Контекстом слова может являться:
  - Соседние слова
    - Слева
    - Справа
    - Симметрично
  - Документ (параграф, предложение)

word2vec

- Continuous skip-gram

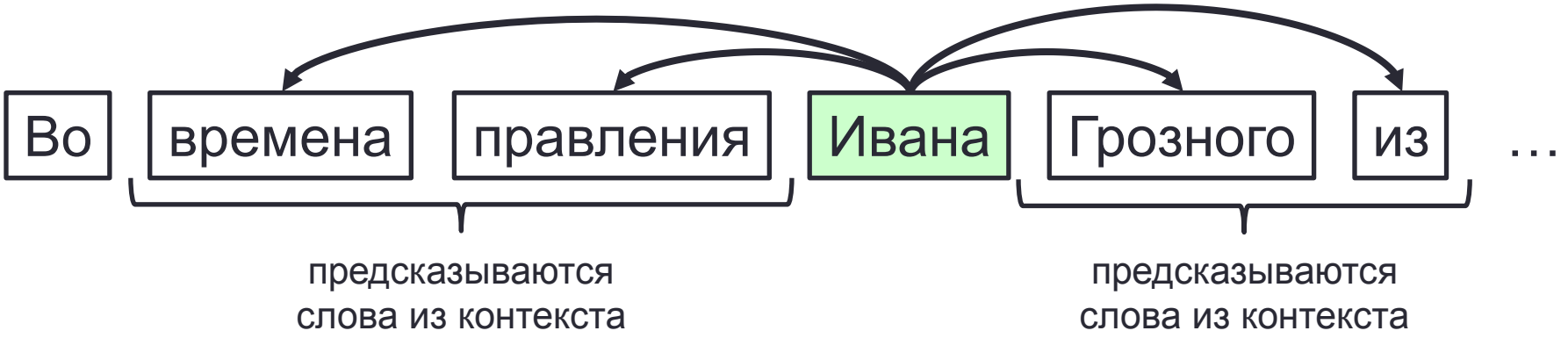


- Continuous bag of words

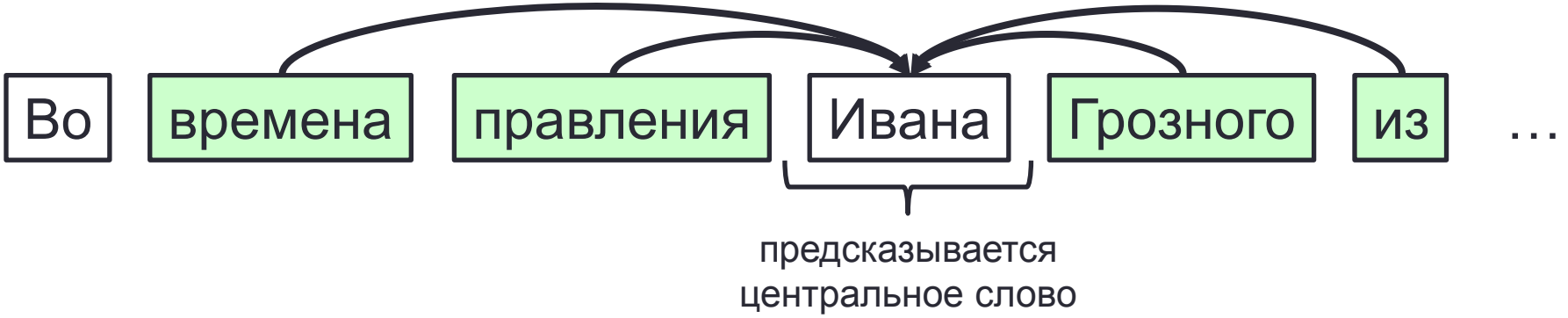


word2vec

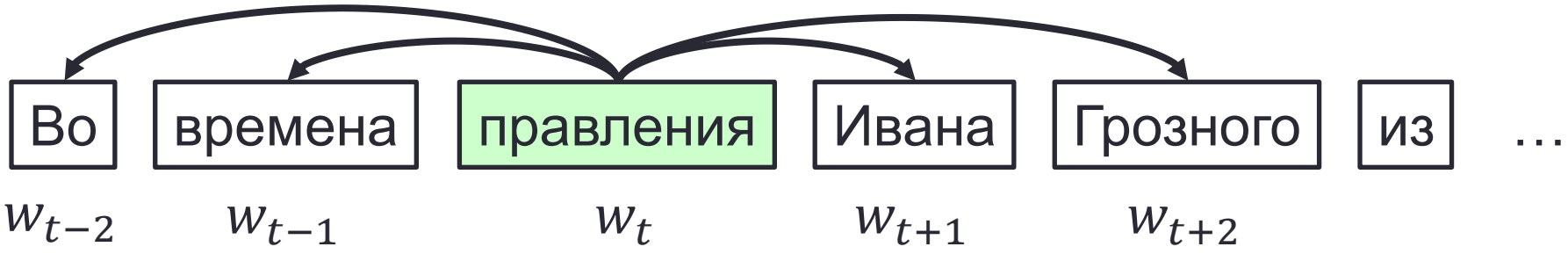
- Continuous skip-gram



- Continuous bag of words



word2vec skip-gram

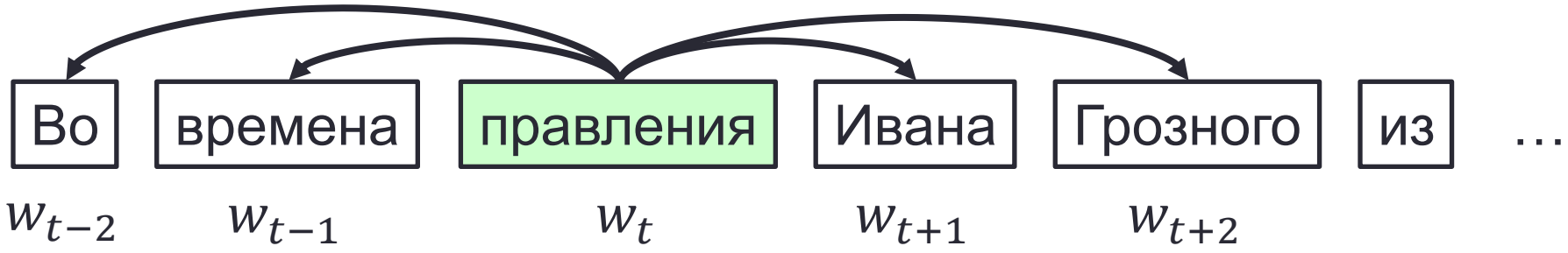


- Цель: максимизировать вероятность всех контекстных слов при данном центральном слове

$$J'(\theta) = \prod_{t=1}^{T} \prod_{-m \le j \le m, j \neq 0} p(w_{t+j} | w_t, \theta)$$

- θ – оптимизируемые параметры

word2vec skip-gram



- Цель: максимизировать логарифм вероятности всех контекстных слов при данном центральном слове

$$J(\theta) = -\log J'(\theta) = -\sum_{t=1}^{T} \sum_{-m \le j \le m, j \ne 0} \log p_\theta(w_{t+j} | w_t)$$

- θ – оптимизируемые параметры – векторы слов

word2vec skip-gram

$$J(\theta) = -\sum_{t=1}^{T} \sum_{-m \le j \le m, j \neq 0} \log p_\theta(w_{t+j} | w_t)$$

- $\theta = \{V, U\}$;
- V – векторы центрального слова
- U– векторы слова из контекста

- $p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{n} \exp(u_w^T v_c)}$;

word2vec skip-gram

$$J(\theta) = -\sum_{t=1}^{T} \sum_{-m \le j \le m, j \neq 0} \log p_\theta(w_{t+j} | w_t)$$

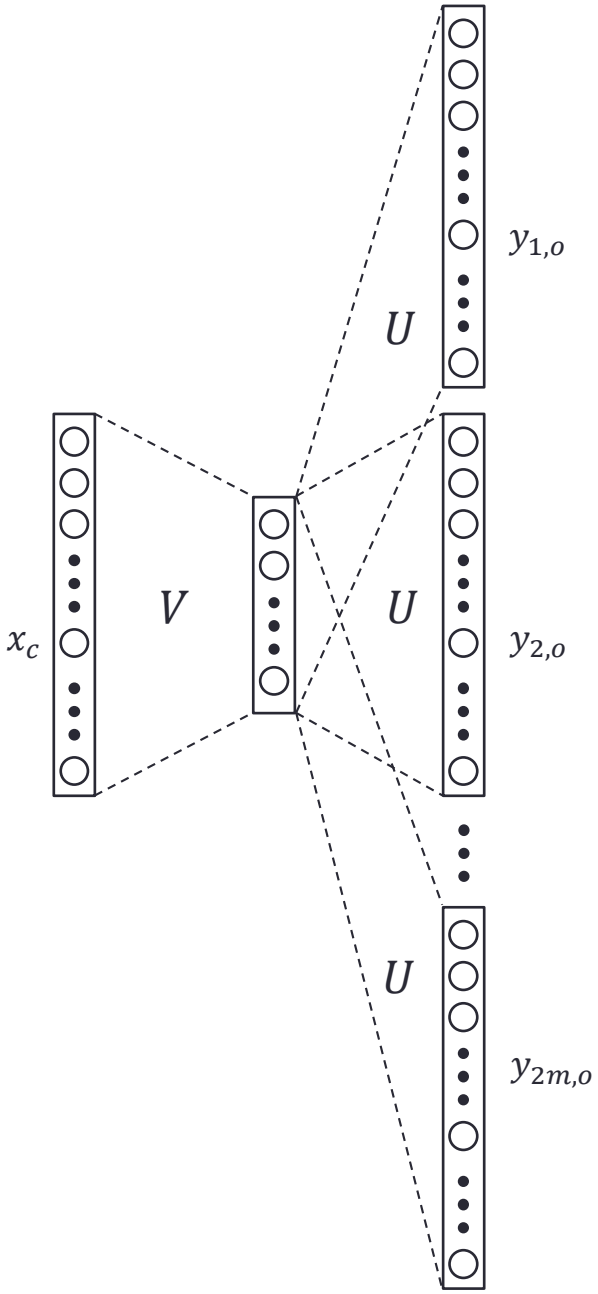
- Для каждого окна минимизируем

$$-\log p(o|c) = -\log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{n} \exp(u_w^T v_c)};$$

word2vec skip-gram

Модель может быть представлена в виде нейронной сети:

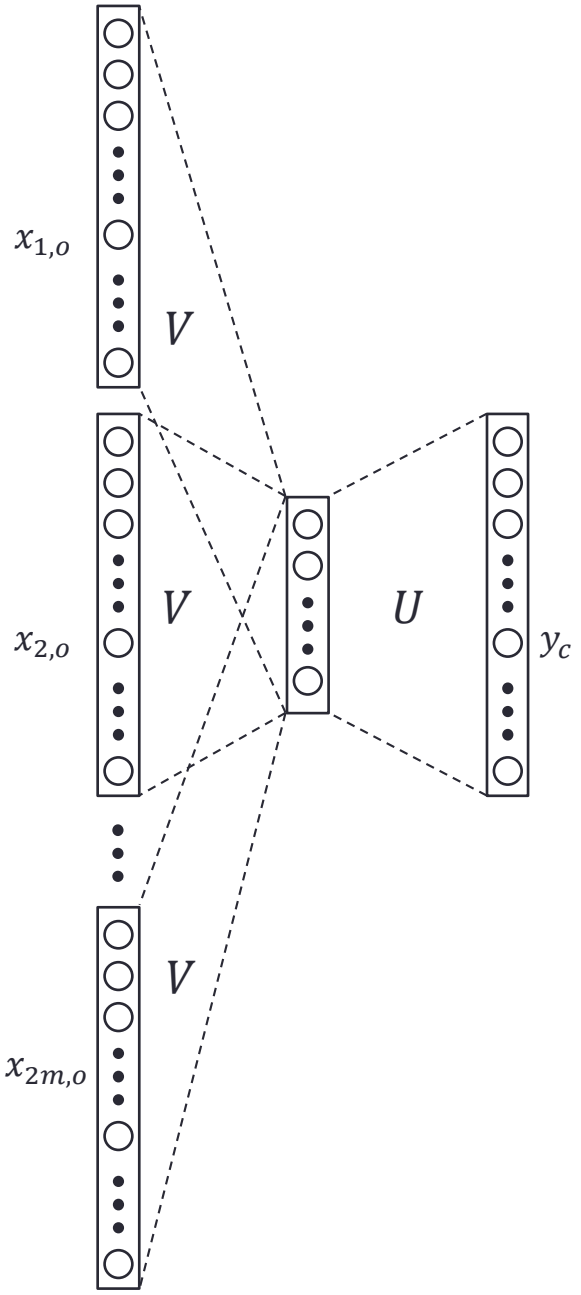
- **Вход**: one-hot центрального слова
- **Скрытый слой**: линейный
- **Выходной слой**: softmax
- **Функция ошибки**: cross-entropy



word2vec CBOW

Модель может быть представлена в виде нейронной сети:

- **Вход**: one-hot контекстных слов
- **Скрытый слой**: линейный
- **Выходной слой**: softmax
- **Функция ошибки**: cross-entropy



word2vec skip-gram

$$J(\theta) = -\sum_{t=1}^{T} \sum_{-m \le j \le m, j \neq 0} \log p_\theta(w_{t+j} | w_t)$$

- Для каждого окна минимизируем

$$-\log p(o|c) = -\log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{n} \exp(u_w^T v_c)};$$

- Метод градиентного спуска

$$\frac{\partial(-\log p(o|c))}{\partial v_c} = -u_o + \sum_{i=1}^{n} \frac{\exp\left(u_i^T v_c\right)}{\sum_{w=1}^{n} \exp(u_w^T v_c)} u_i =$$

$$= -u_o + \sum_{x \in V} p(x|c) u_x$$

word2vec skipgram

- Проблемы:
  - На каждом шаге градиентного спуска вычисляется

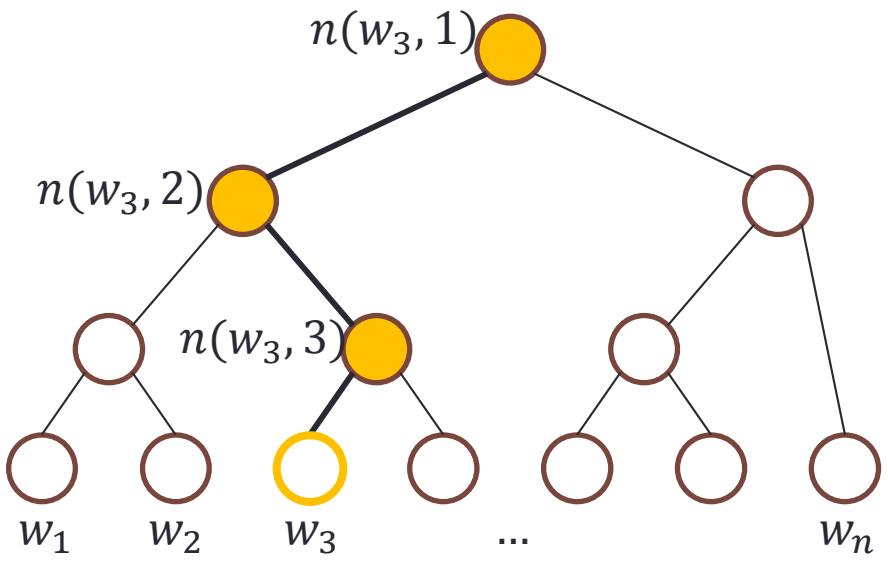
$$\sum_{w=1}^{n} \exp(u_w^T v_c)$$

    словарь большой ⇒ долго

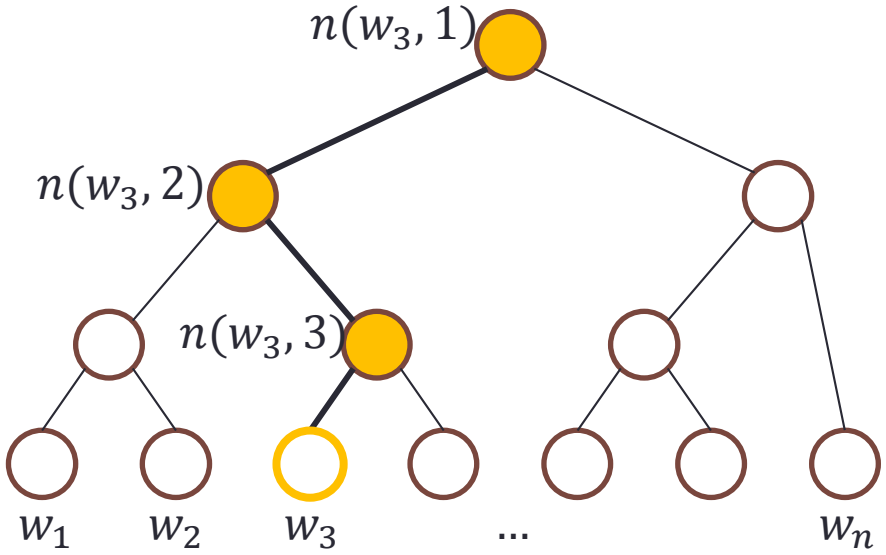
- Решения:
  - Hierarchical softmax
  - Negative sampling

Hierarchical softmax

- Идея:
  - Составить из словаря бинарное дерево
  - Предсказывать путь в дереве вместо слова из словаря



Hierarchical softmax



$$p(o|c) = \prod_{j=1}^{L(o)-1} \sigma(\llbracket n(o, j+1) = ch\big(n(o, j)\big)\rrbracket \, u_{n(o,j)}^T v_c)$$

$ch(n)$ – левый потомок узла n, $\llbracket x \rrbracket = \begin{cases} 1, & \text{если } x - \text{истина} \\ -1, & \text{если } x - \text{ложь} \end{cases}$

Negative sampling

- Решаем более простую задачу бинарной классификации:

$$z = \begin{cases} 1, & (\mathrm{c}, o) \in D \\ 0, & (\mathrm{c}, o) \notin D \end{cases}$$

- $p(z = 1|o, \mathrm{c}) = \frac{1}{1 + \exp(-v_\mathrm{c}^T u_o)} = \sigma(v_c^T u_o)$

- На каждый положительный пример берем K отрицательных:
  - Небольшие наборы данных – 5-20 примеров
  - Большие наборы данных – 2-5 примеров

Negative sampling

- Для каждого окна максимизируем

$$J_t(\theta) = \log p(z = 1|o, c) + \sum_{j \sim P(w)} \log p(z = 0|j, c) =$$

$$\log \sigma(v_c^T u_o) + \sum_{j \sim P(w)} \log \sigma(-v_c^T u_j)$$

- $P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{n} f(w_j)^{3/4}}$

Negative sampling

- Набор данных:
  - Положительные примеры: пары слов из окон наших текстов
  - Отрицательные примеры: случайные слова из текстов

o	c	z
времена	правления	1
из	правления	0
слово	правления	0
Москвы	правления	0
	…	

Проблема частых слов

- Слишком частые слова (предлоги, союзы, пунктуация)
  - часто встречаются в корпусе ⇒ вносят большое влияние на векторы слов
  - встречаются во всевозможных контекстах ⇒ векторы не отражают значение слова

- Решение:
  - Выкидывать слишком частые слова из корпуса

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}},$$

  - t – порог частоты (обычно около 10^{-5})

Матрица совместной встречаемости

- мама мыла раму.
- раму мыла мама.
- мыла мылом раму.

word-word

	мама	**мыла**	**раму**	**мылом**	**.**
мама	0	2	0	0	1
мыла	2	0	2	1	0
раму	0	2	0	1	2
мылом	0	1	1	0	0
.	1	0	2	0	0

Матрица совместной встречаемости

- Понижение размерности (SVD)
  - $A = U\Sigma V^T$;
    - Σ – матрица сингулярных значений(m*m)
  - $\hat{A} = \hat{U}\hat{\Sigma}\hat{V}^T$; (теорема Эккарта-Янга)
    - $\hat{U}$ –первые k столбцов матрицы U
    - $\hat{\Sigma}$ – k первых столбцов и строк матрицы Σ
    - $\hat{V}$ –первые k столбцов матрицы V

Матрица совместной встречаемости

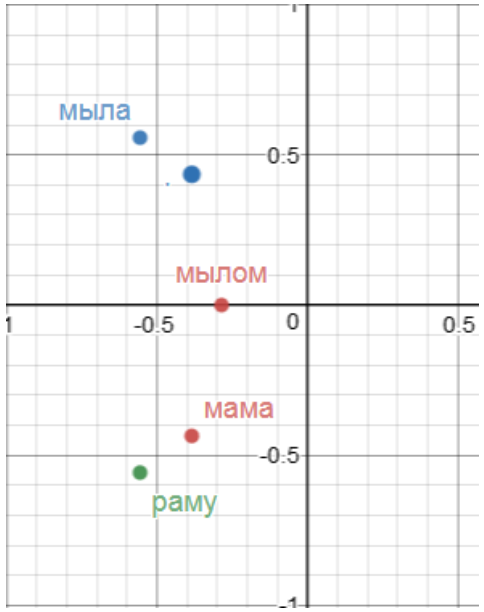
- Понижение размерности (SVD)
  - $A = U\Sigma V^T$;
    - Σ – матрица сингулярных значений(m*m)

$$A = \begin{bmatrix} 0 & 2 & 0 & 0 & 1 \\ 2 & 0 & 2 & 1 & 0 \\ 0 & 2 & 0 & 1 & 2 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 \end{bmatrix}$$

$$\underset{U}{\begin{bmatrix} -0.385 & -0.435 & -0.359 & -0.557 & -0.472 \\ -0.557 & 0.557 & 0.411 & -0.435 & 0.142 \\ -0.557 & -0.557 & 0.411 & 0.435 & 0.142 \\ -0.286 & 0 & -0.636 & 0 & 0.716 \\ -0.385 & 0.435 & -0.359 & 0.557 & -0.472 \end{bmatrix}} * diag \underset{\Sigma}{\begin{pmatrix} 3.895 \\ 3.562 \\ 1.292 \\ 0.562 \\ 0.397 \end{pmatrix}} * \underset{V^T}{\begin{bmatrix} -0.385 & 0.435 & 0.359 & -0.557 & -0.472 \\ -0.557 & -0.557 & -0.411 & -0.435 & 0.142 \\ -0.557 & 0.557 & -0.411 & 0.435 & 0.142 \\ -0.286 & 0 & 0.636 & 0 & 0.716 \\ -0.385 & -0.435 & 0.359 & 0.557 & -0.472 \end{bmatrix}^T}$$

Матрица совместной встречаемости

- Понижение размерности (SVD)
  - $\hat{A} = \hat{U}\hat{\Sigma}\hat{V}^T$;
    - $\hat{U}$ –первые k столбцов матрицы U
    - $\hat{\Sigma}$ – k первых столбцов и строк матрицы Σ
    - $\hat{V}$ –первые k столбцов матрицы V



$$\hat{A} = \begin{bmatrix} -0.10 & 1.70 & -0.02 & 0.43 & 1.25 \\ 1.70 & 0.10 & 2.32 & 0.62 & -0.03 \\ -.03 & 2.32 & 0.10 & 0.62 & 1.70 \\ 0.43 & 0.62 & 0.62 & 0.32 & 0.43 \\ 1.25 & -0.03 & 1.70 & 0.43 & -0.10 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} -0.385 & -0.435 & -0.359 & -0.557 & -0.472 \\ -0.557 & 0.557 & 0.411 & -0.435 & 0.142 \\ -0.557 & -0.557 & 0.411 & 0.435 & 0.142 \\ -0.286 & 0 & -0.636 & 0 & 0.716 \\ -0.385 & 0.435 & -0.359 & 0.557 & -0.472 \end{bmatrix}}_{U} * diag\underbrace{\begin{pmatrix} 3.895 \\ 3.562 \\ 1.292 \\ 0.562 \\ 0.397 \end{pmatrix}}_{\Sigma} * \underbrace{\begin{bmatrix} -0.385 & 0.435 & 0.359 & -0.557 & -0.472 \\ -0.557 & -0.557 & -0.411 & -0.435 & 0.142 \\ -0.557 & 0.557 & -0.411 & 0.435 & 0.142 \\ -0.286 & 0 & 0.636 & 0 & 0.716 \\ -0.385 & -0.435 & 0.359 & 0.557 & -0.472 \end{bmatrix}^T}_{V^T}$$

$\hat{U}$ $\hat{\Sigma}$ $\hat{V}$

GloVe

- Строится на основе word-word матрицы совместной встречаемости

- X_{ij} – количество употреблений слова w_j встретилось в контексте слова w_i

- Функция потерь:

$$J(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{n} f\left(X_{ij}\right)\left(v_i^T u_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

$$f(x) = \begin{cases} (x/x_{max})^{\alpha}, x < x_{max} \\ 1, \quad x \geq x_{max} \end{cases}, \qquad \alpha < 1$$

Оценка качества

- Intrinsic (in vitro)
  - Пытаемся оценить результат решения задачи сравнением с эталонным результатом
- Extrinsic (in vivo)
  - Пытаемся оценить результат решения задачи как подзадачи более сложной задачи

Оценка качества (Intrinsic)

- Задача аналогии слов:
  - Слово a относится к слову b также, как слово c относится к слову ___.
  - $d = \arg\max_i similarity(x_b - x_a + x_c, x_i)$
- Метрика:
  - $Accuracy = \frac{correct}{total}$;
- Синтаксические аналогии:

a	b	c	___
лекция	лекции	семинар	***семинары***
бежать	бегущий	лежать	***лежащий***

Оценка качества (Intrinsic)

- Задача аналогии слов:
  - Слово a относится к слову b также, как слово c относится к слову ___.
  - $d = \arg\max_i similarity(x_b - x_a + x_c, x_i)$
- Метрика:
  - $Accuracy = \frac{correct}{total}$;
- Семантические аналогии:

a	b	c	___
лететь	плыть	самолет	***корабль***
Россия	Москва	Франция	***Париж***

Оценка качества (Intrinsic)

- Задача аналогии слов
  - Найти слово по аналогии с другими
- Задача похожести пар слов
  - Отсортировать пары слов в соответствии со смысловой близостью
- Задача поиска синонимов
  - Для заданного слова найти синонимы среди заданного множества слов
- Задача поиска лишнего слова
  - В множестве слов найти лишнее

Проблема редких слов

- Для слов, которые встречаются слишком редко, невозможно построить хорошие векторы

- В обучающем корпусе могут отсутствовать редкие слова $\Rightarrow$ такие слова не попадут в словарь V

Проблема редких слов

- Редкие слова заменяются на специальную константу OOV (out of vocabulary)
  - В процессе обучения вычисляется вектор для OOV
  - Этот вектор используется для слов, не вошедших в V

Проблема редких слов

- Редкие слова заменяются на специальную константу OOV (out of vocabulary)
  - В процессе обучения вычисляется вектор для OOV
  - Этот вектор используется для слов, не вошедших в V

- Слова рассматриваются не как атомарные единицы, а как последовательности символов

fasttext

- Каждое слово w представим как мультимножество символьных n-gram

Грозного

n=2 <Гр> <ро> <оз> <зн> <но> <ог> <го>

n=3 <Гро> <роз> <озн> <зно> <ног> <ого>

- Составим словарь слов V и словарь n-gram G

fasttext

- В word2vec:

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{n} \exp(u_i^T v_c)} = \frac{\exp(s(w_o, w_c))}{\sum_{i=1}^{n} \exp(s(w_i, w_c))},$$

$s(w_o, w_c) = u_o^T v_c$ - score функция

- В fasttext:

$$s(w_o, w_c) = u_o^T v_c + \sum_{g \in \Gamma(w_c)} u_o^T z_g ,$$

$\Gamma(w_c)$– n-gram’ы слова w_c

Сверточные сети (CNN)

- Архитектура нейронной сети, нацеленная на эффективное распознавание образов

- Состоит из:
  - Слой свертки
  - Слой активации
  - Pooling слой

Сверточные сети (CNN)

- Слой свертки
  - На входе матрица (изображение) $X \in \mathbb{R}^{m \times n}$
  - Задана матрица весов (фильтр, ядро свертки) $K \in \mathbb{R}^{h \times w}$
  - Строим выходное изображение, «двигая» фильтр по матрице

$$Y \in \mathbb{R}^{(m-h+1)\times(n-w+1)}$$

$$y_{i,j} = \sum_{q=1}^{h}\sum_{r=1}^{w} X_{i+q-1,j+r-1} * K_{q,r}, i = \overline{1, m-h+1}, j = \overline{1, n-w+1}$$

Сверточные сети (CNN)

- Слой свертки

$K \in \mathbb{R}^{3\times3}$

2	-1	-1
-1	0	0
-1	0	2

$Y \in \mathbb{R}^{?\times?}$

$X \in \mathbb{R}^{4\times5}$

2	1	3	1	2
3	4	2	0	1
2	0	1	2	3
1	3	2	4	0

Сверточные сети (CNN)

- Слой свертки

$K \in \mathbb{R}^{3\times3}$

2	-1	-1
-1	0	0
-1	0	2

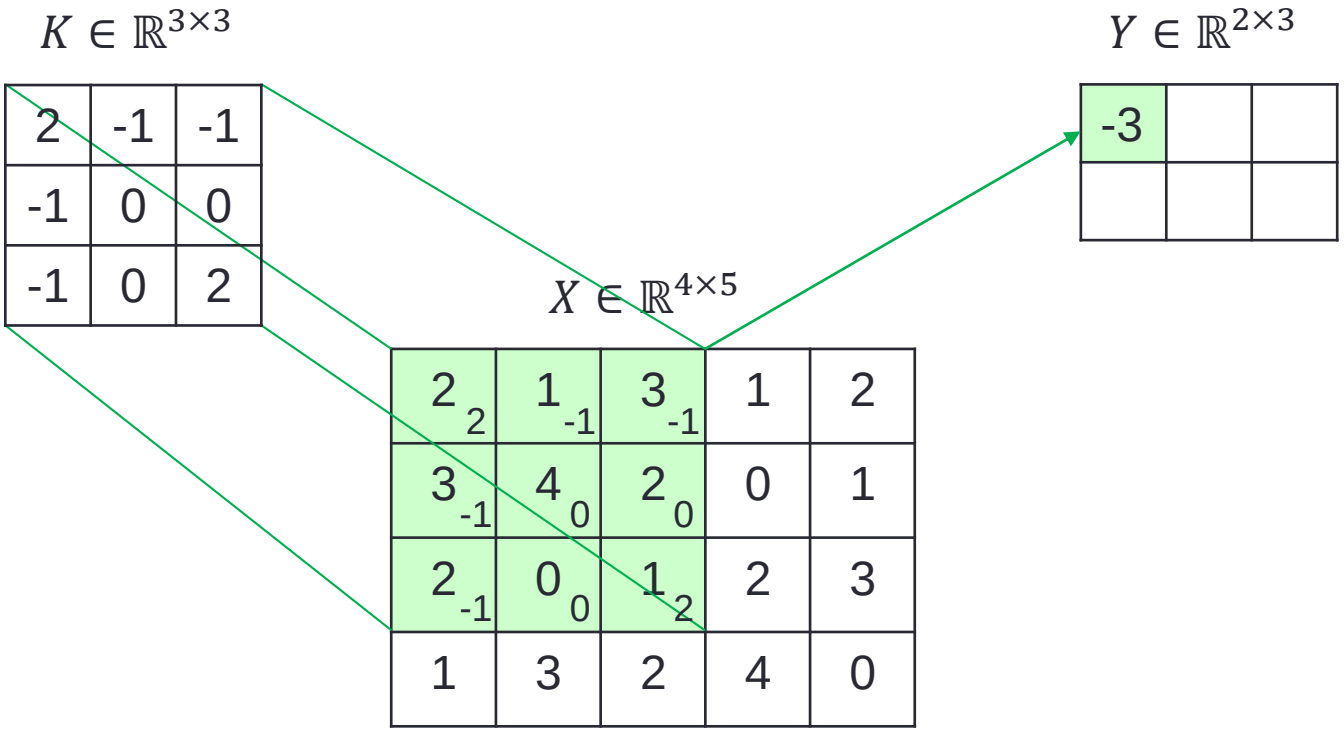
$Y \in \mathbb{R}^{2\times3}$

$X \in \mathbb{R}^{4\times5}$

2	1	3	1	2
3	4	2	0	1
2	0	1	2	3
1	3	2	4	0

Сверточные сети (CNN)

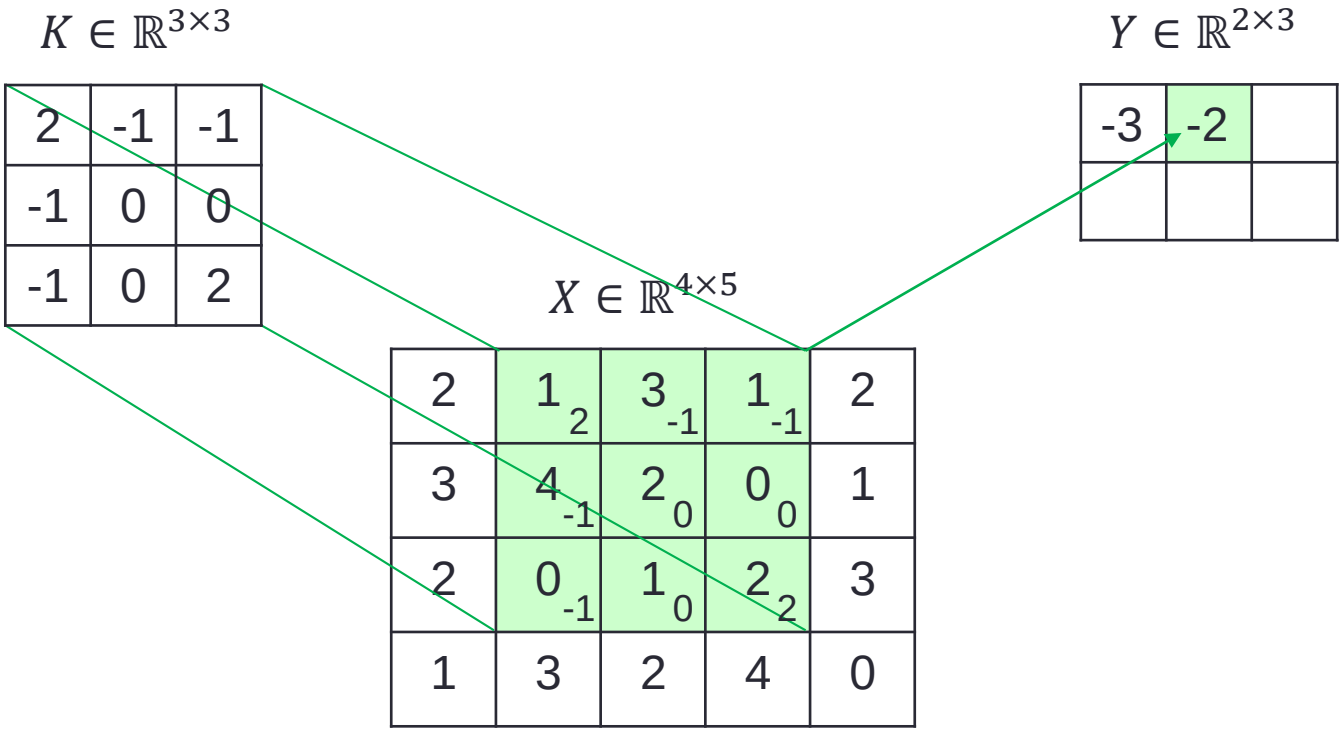
- Слой свертки



$$y_{1,1} = 2 * 2 + (-1) * 1 + (-1) * 3 + (-1) * 3 + 0 * 4 + 0 * 2 + (-1) * 2 + 0 * 0 + 2 * 1 = -3$$

Сверточные сети (CNN)

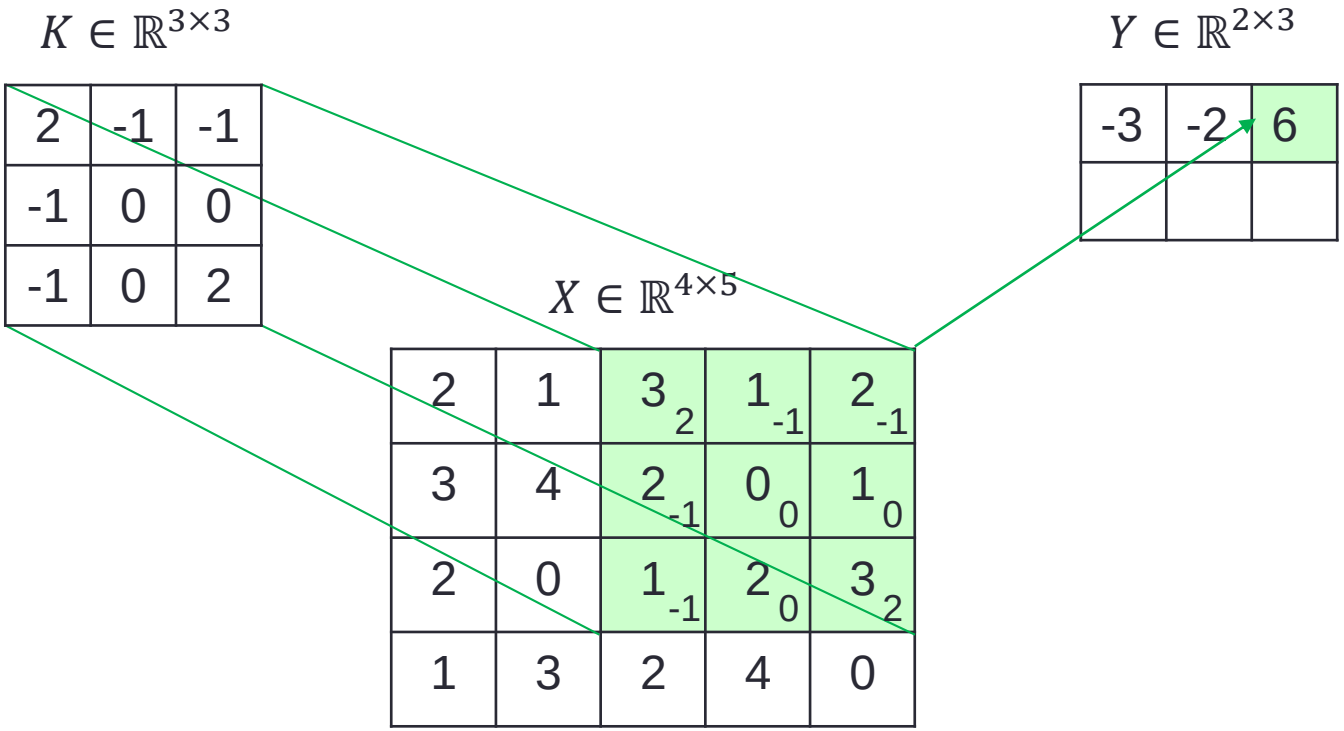
- Слой свертки



$$y_{1,2} = 2 * 1 + (-1) * 3 + (-1) * 1 + (-1) * 4 + 0 * 2 + 0 * 0 + (-1) * 0 + 0 * 1 + 2 * 2 = -2$$

Сверточные сети (CNN)

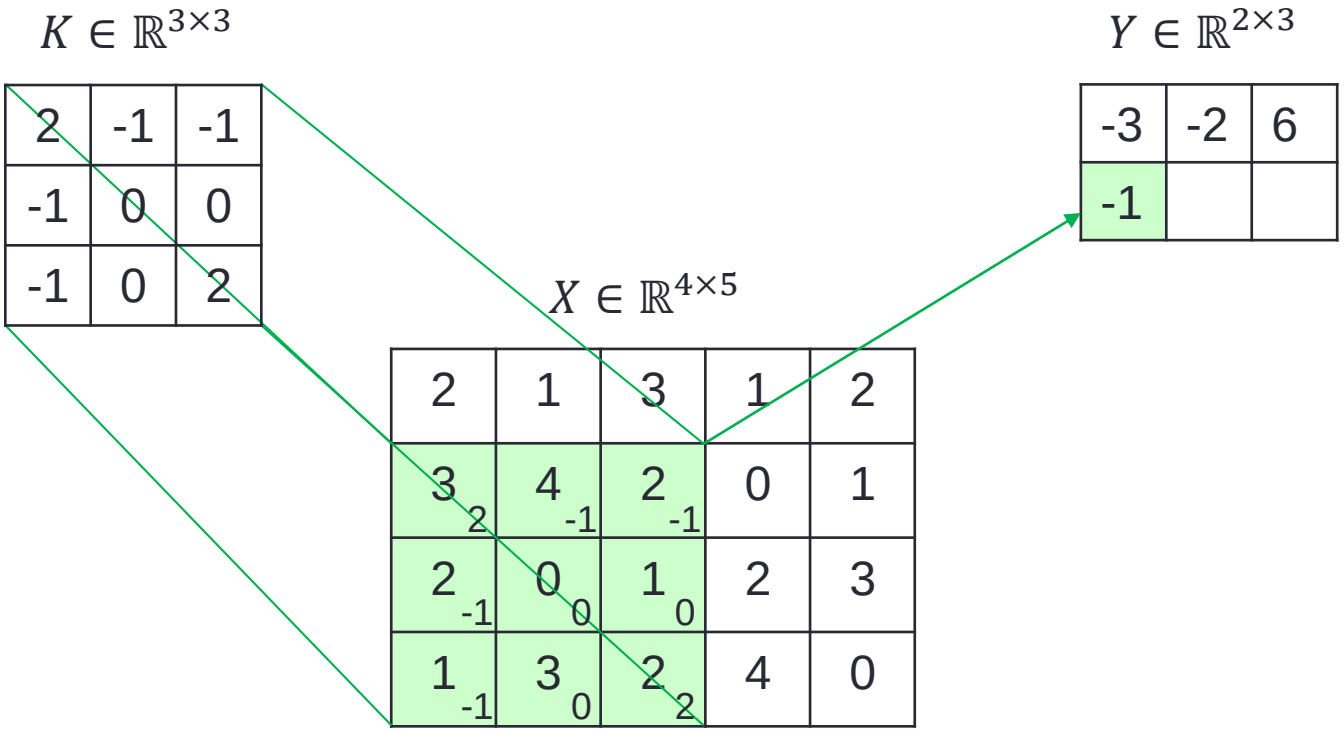
- Слой свертки



$$y_{1,3} = 2 * 3 + (-1) * 1 + (-1) * 2 + (-1) * 2 + 0 * 0 + 0 * 1 + (-1) * 1 + 0 * 2 + 2 * 3 = 6$$

Сверточные сети (CNN)

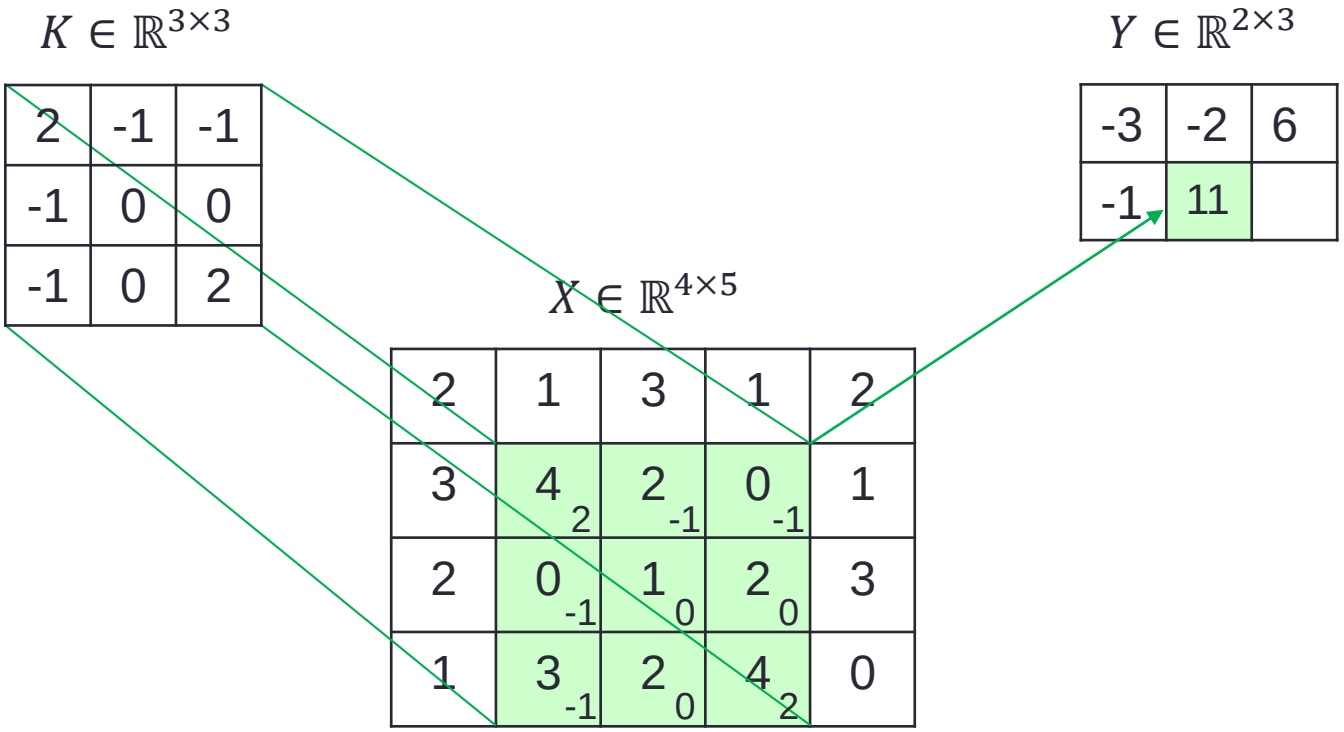
- Слой свертки



$$y_{2,1} = 2 * 3 + (-1) * 4 + (-1) * 2 + (-1) * 2 + 0 * 0 + 0 * 1 + (-1) * 1 + 0 * 3 + 2 * 2 = -1$$

Сверточные сети (CNN)

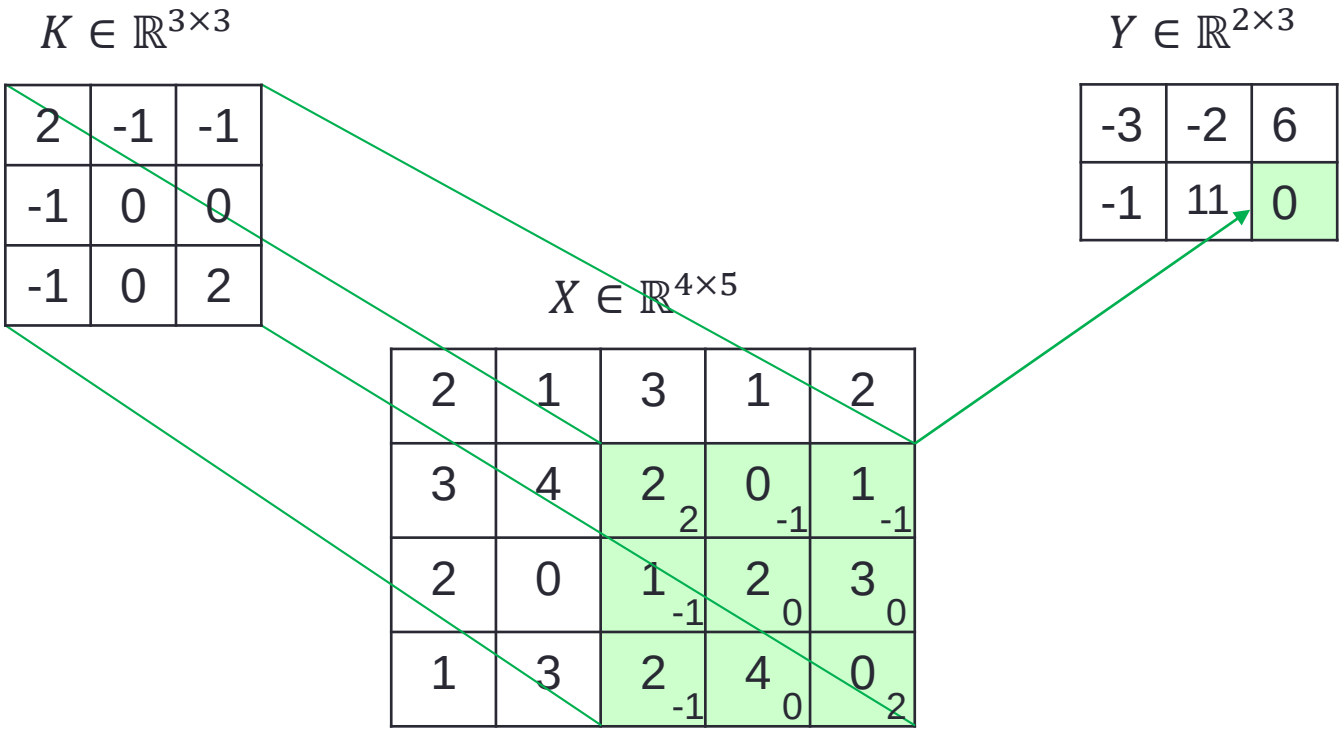
- Слой свертки



$$y_{2,2} = 2 * 4 + (-1) * 2 + (-1) * 0 + (-1) * 0 + 0 * 1 + 0 * 2 + (-1) * 3 + 0 * 2 + 2 * 4 = 11$$

Сверточные сети (CNN)

- Слой свертки



$$y_{2,3} = 2 * 2 + (-1) * 0 + (-1) * 1 + (-1) * 1 + 0 * 2 + 0 * 3 + (-1) * 2 + 0 * 4 + 2 * 0 = 0$$

Сверточные сети (CNN)

- Pooling слой
  - Агрегирует выходы группы нейронов в один выход
  - На входе матрица (признаки) $X \in \mathbb{R}^{m \times n}$
  - Задан размер окна ($h \times w$)
  - Строим выходное признаки, «двигая» окно по матрице и выполняя операцию агрегации

$$Y \in \mathbb{R}^{\frac{m}{h} \times \frac{n}{w}};$$

$$y_{i,j} = f\left(X_{(i-1)*h:i*h,(j-1)*w:j*w}\right)$$

  - В качестве f обычно используется функция max

Сверточные сети (CNN)

- Pooling слой

$X \in \mathbb{R}^{2\times 3}$

-3	-2	6
-1	11	0

Окно $(h \times w)$: $h = 2; w = 1$

$Y \in \mathbb{R}^{?\times ?}$

Сверточные сети (CNN)

- Pooling слой

$X \in \mathbb{R}^{2\times 3}$

-3	-2	6
-1	11	0

Окно $(h \times w)$: $h = 2; w = 1$

$Y \in \mathbb{R}^{1\times 3}$

Сверточные сети (CNN)

- Pooling слой

$X \in \mathbb{R}^{2\times 3}$

-3	-2	6
-1	11	0

Окно $(h \times w)$: $h = 2; w = 1$

$Y \in \mathbb{R}^{1\times 3}$

-1		

Сверточные сети (CNN)

- Pooling слой

$X \in \mathbb{R}^{2\times 3}$

-3	-2	6
-1	11	0

Окно $(h \times w)$: $h = 2; w = 1$

$Y \in \mathbb{R}^{1\times 3}$

-1	11	

Сверточные сети (CNN)

- Pooling слой

$X \in \mathbb{R}^{2\times 3}$

-3	-2	6
-1	11	0

Окно $(h \times w)$: $h = 2; w = 1$

$Y \in \mathbb{R}^{1\times 3}$

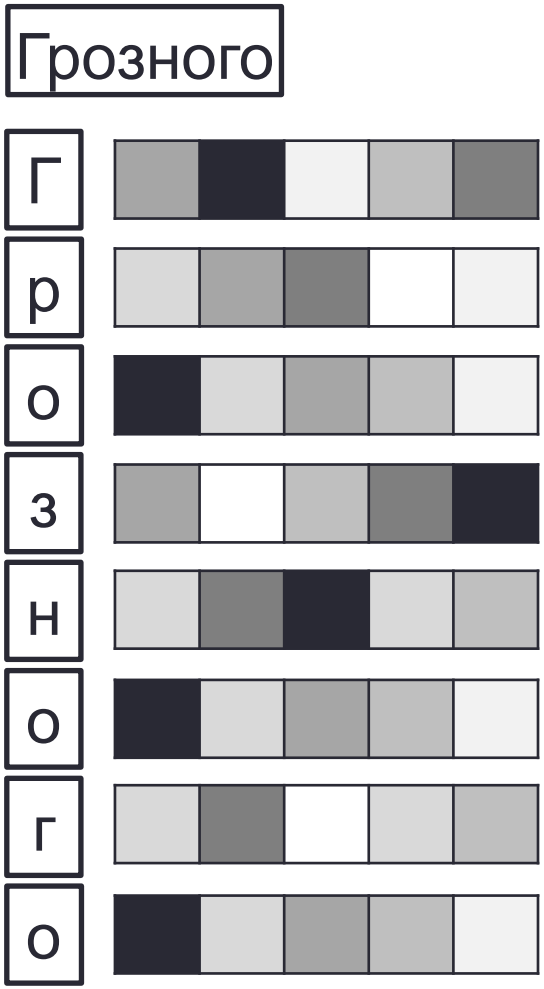
-1	11	6

CharCNN

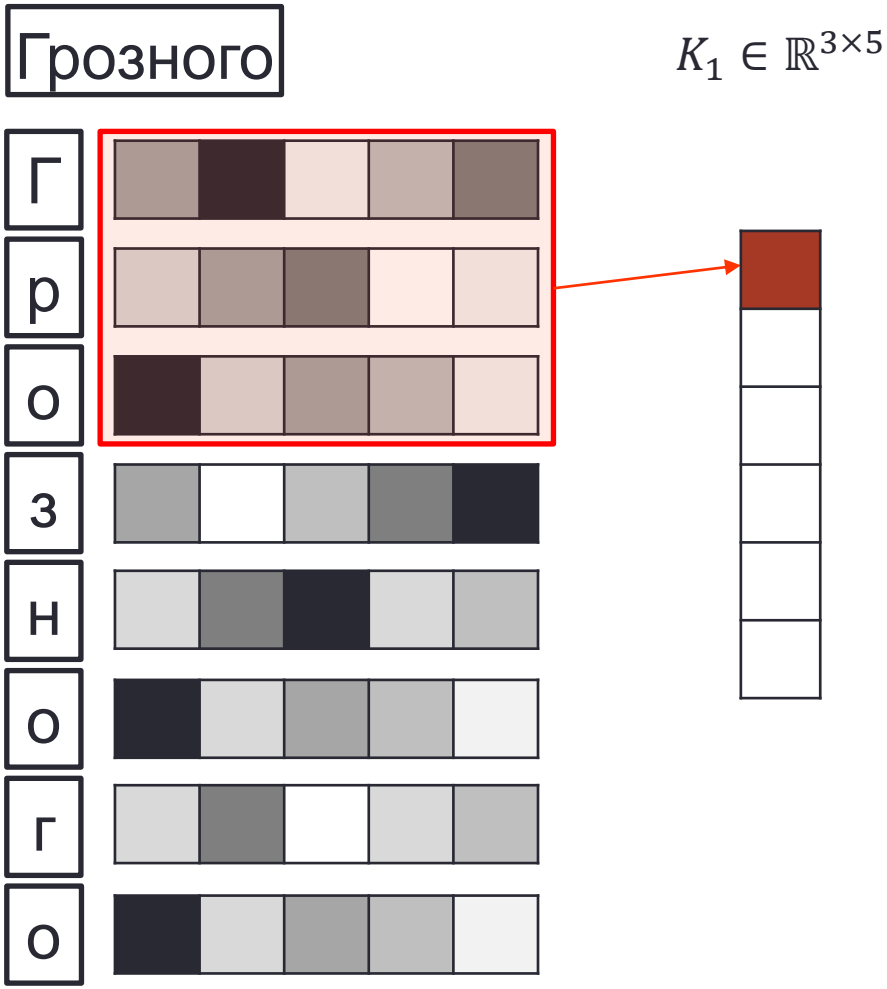
- Собираем словарь символов C по обучающему корпусу
- Каждому символу $c \in C$ ставим в соответствие вектор $v_c \in \mathbb{R}^d$
- Слово w рассматриваем как последовательность символов $[c_1, c_2, \ldots, c_n]$
- Подставляя для каждого символа c_i вектор v_{c_i} получаем матрицу $X \in \mathbb{R}^{n \times d}$
- Задаем m фильтров $K_i \in \mathbb{R}^{k_i \times d}$

- Применяем к X свертки с фильтрами K_i с max pooling и получаем вектор $y \in \mathbb{R}^m$ для слова w

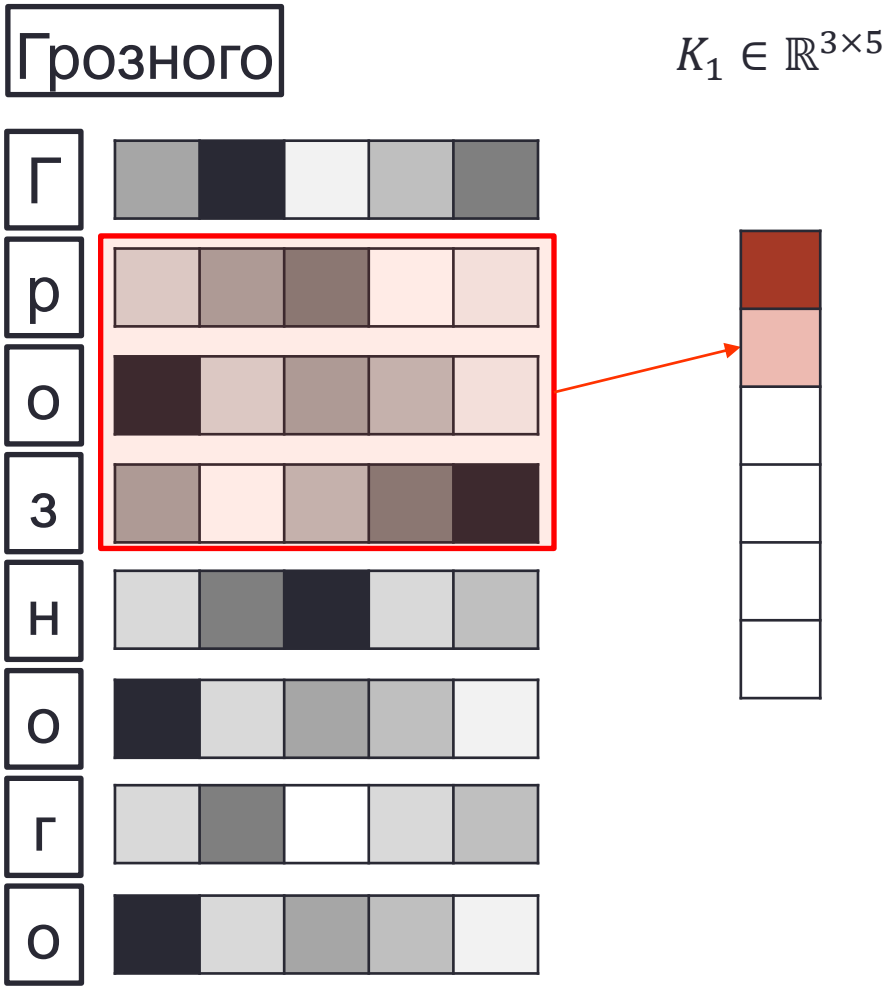
CharCNN



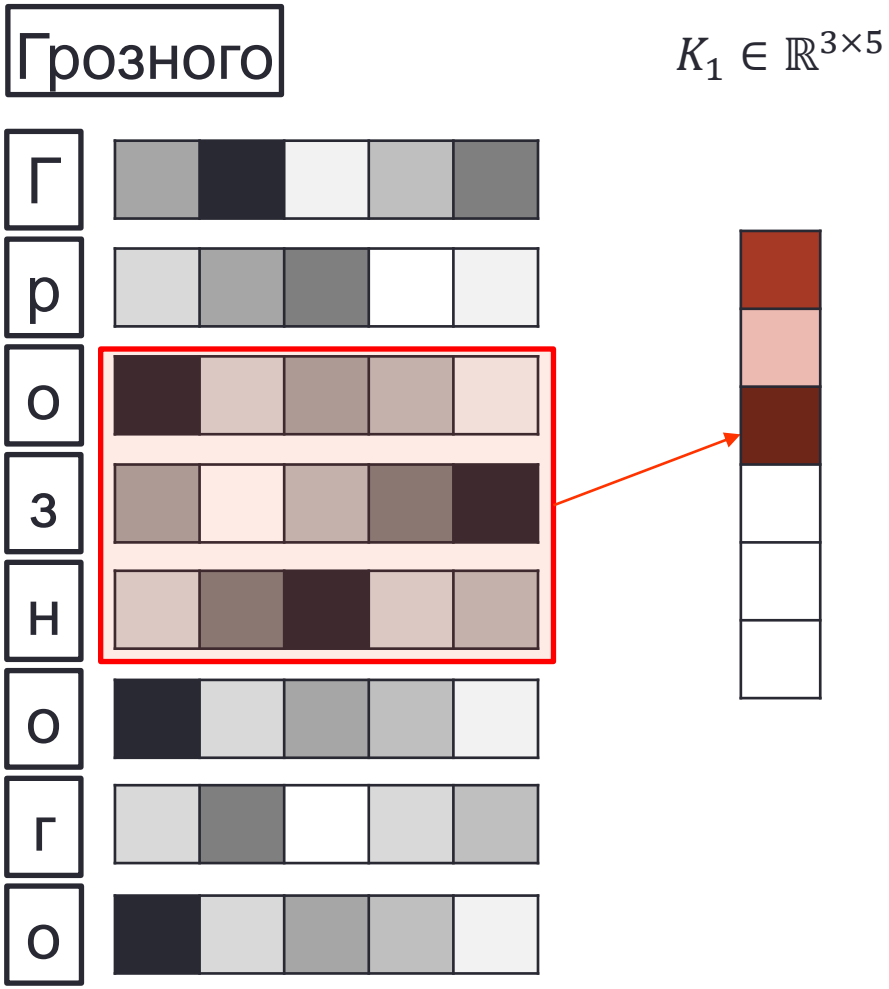
CharCNN



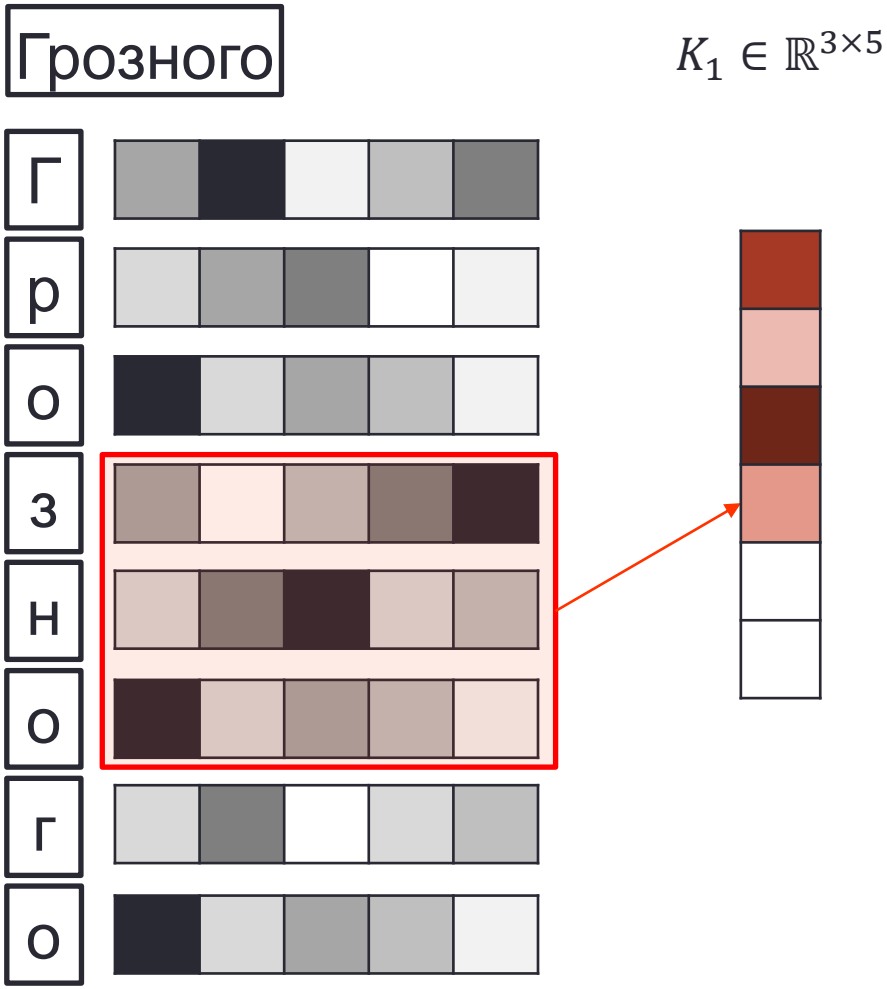
CharCNN



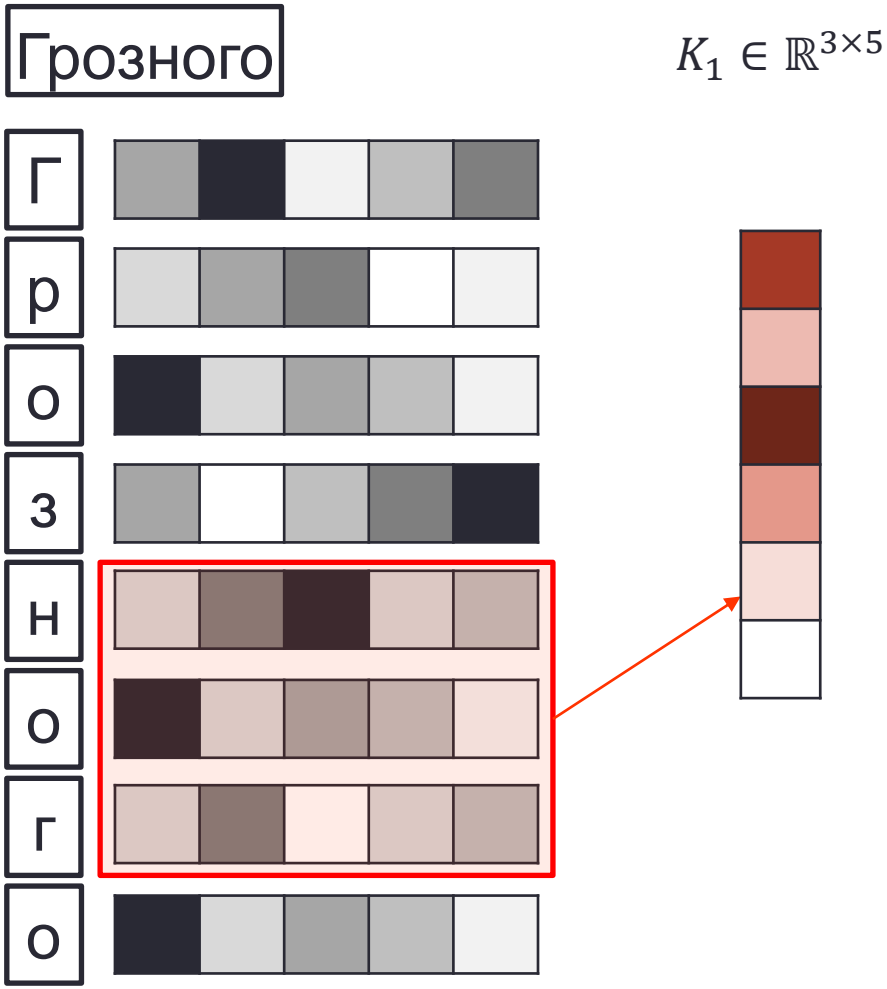
CharCNN



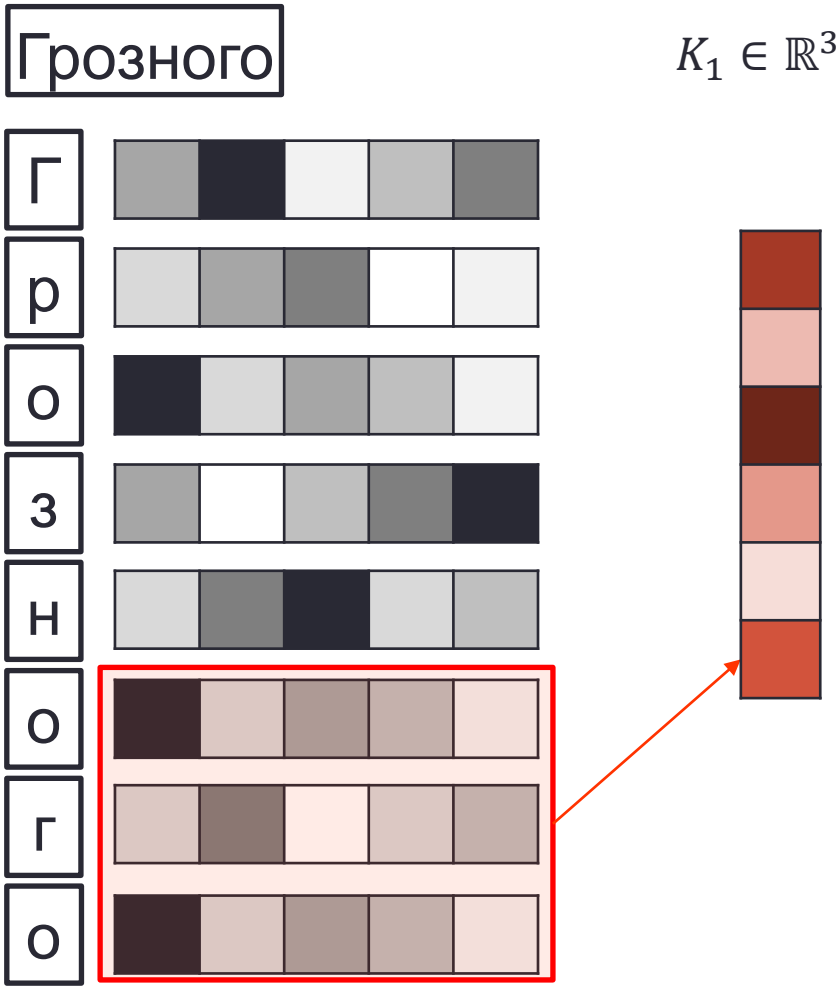
CharCNN



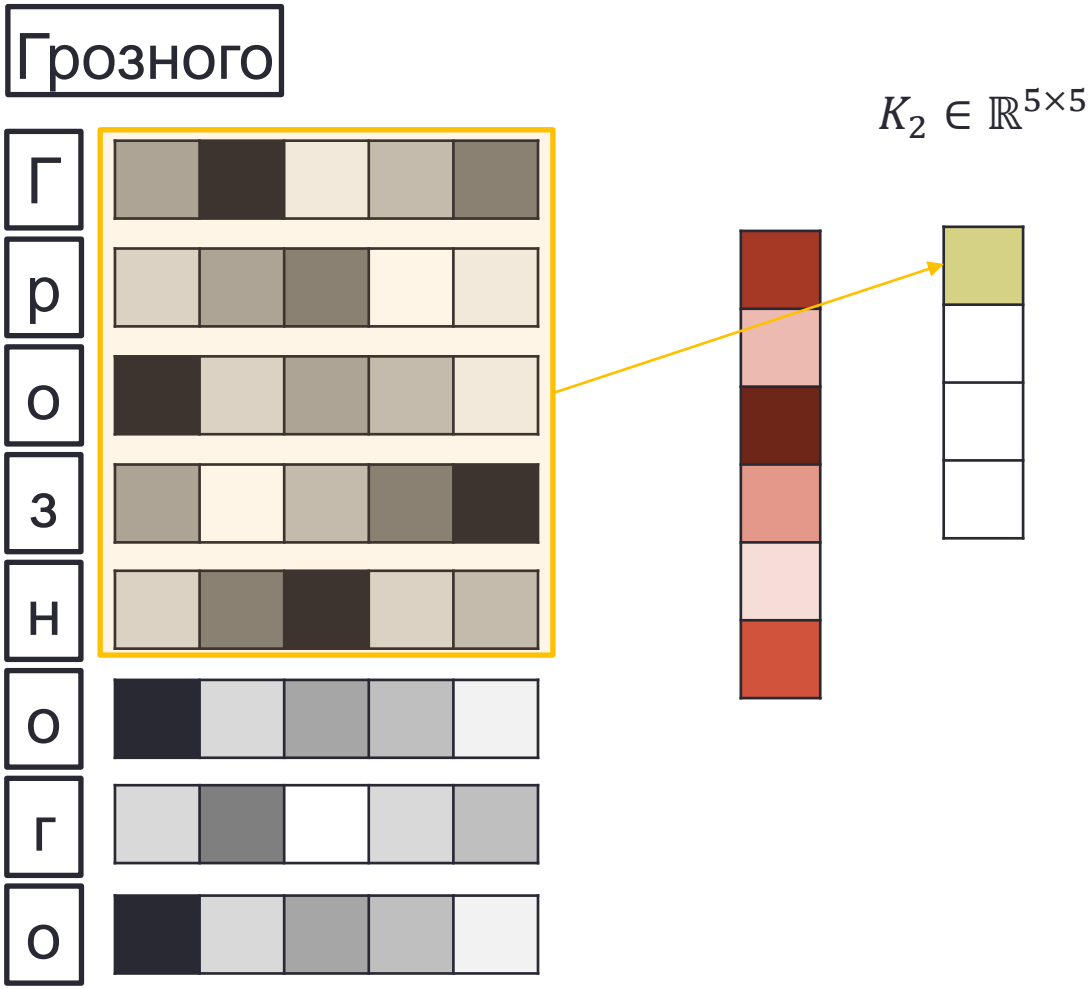
CharCNN



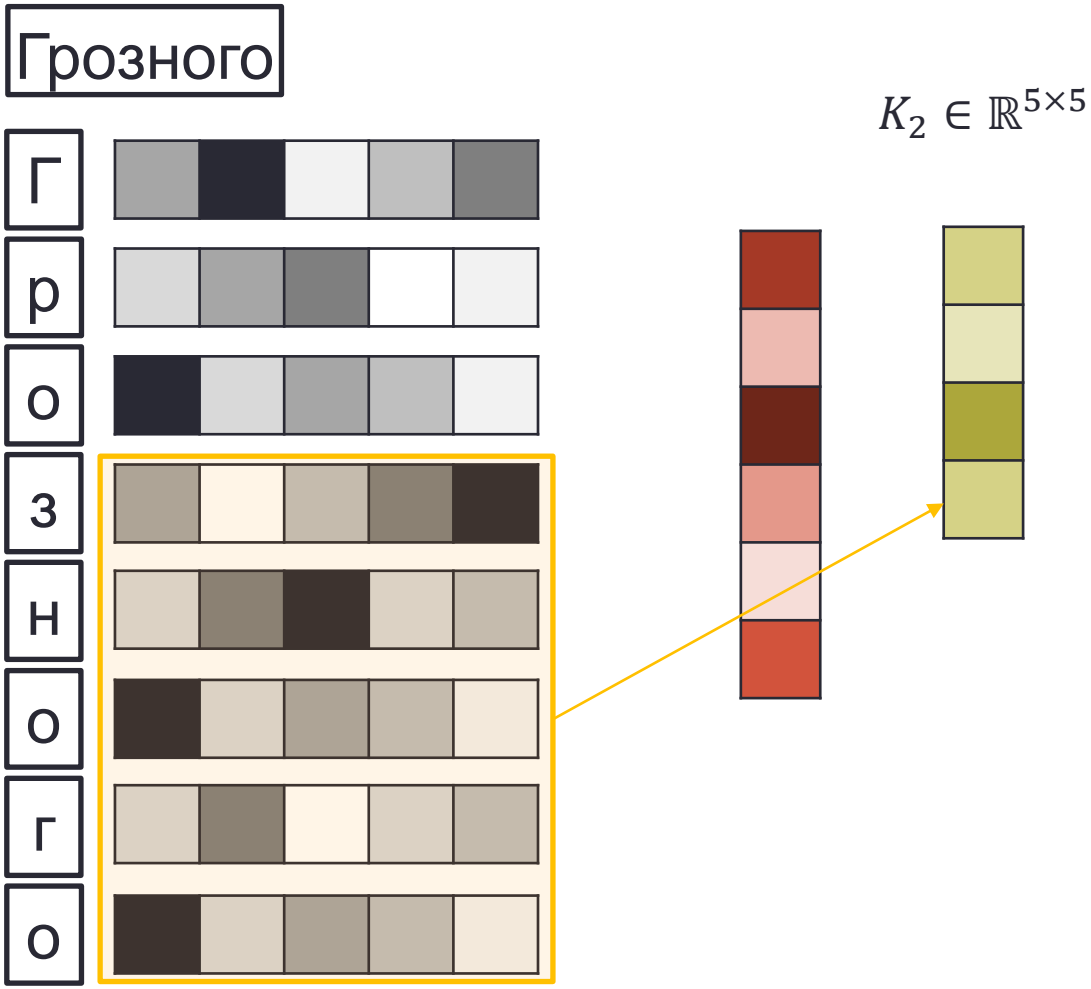
CharCNN



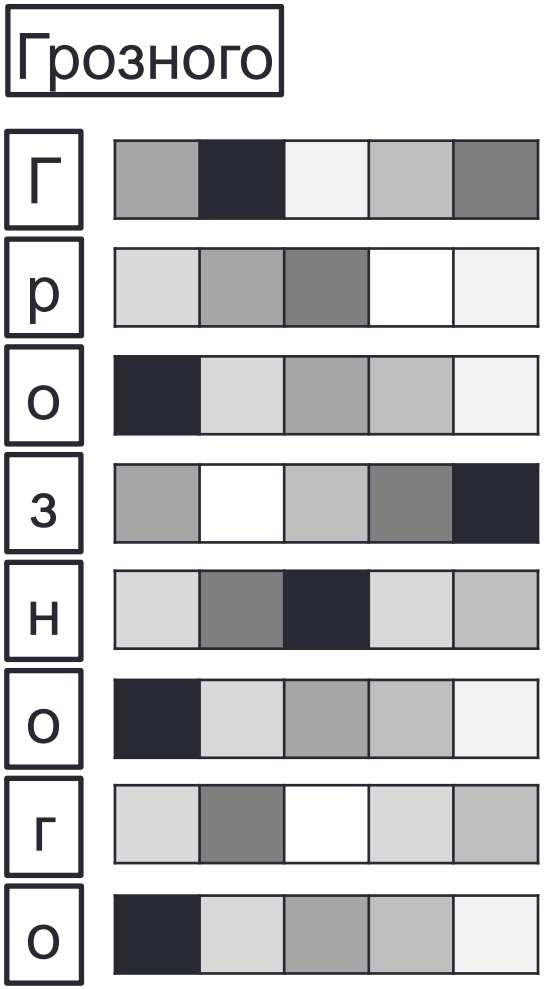
CharCNN

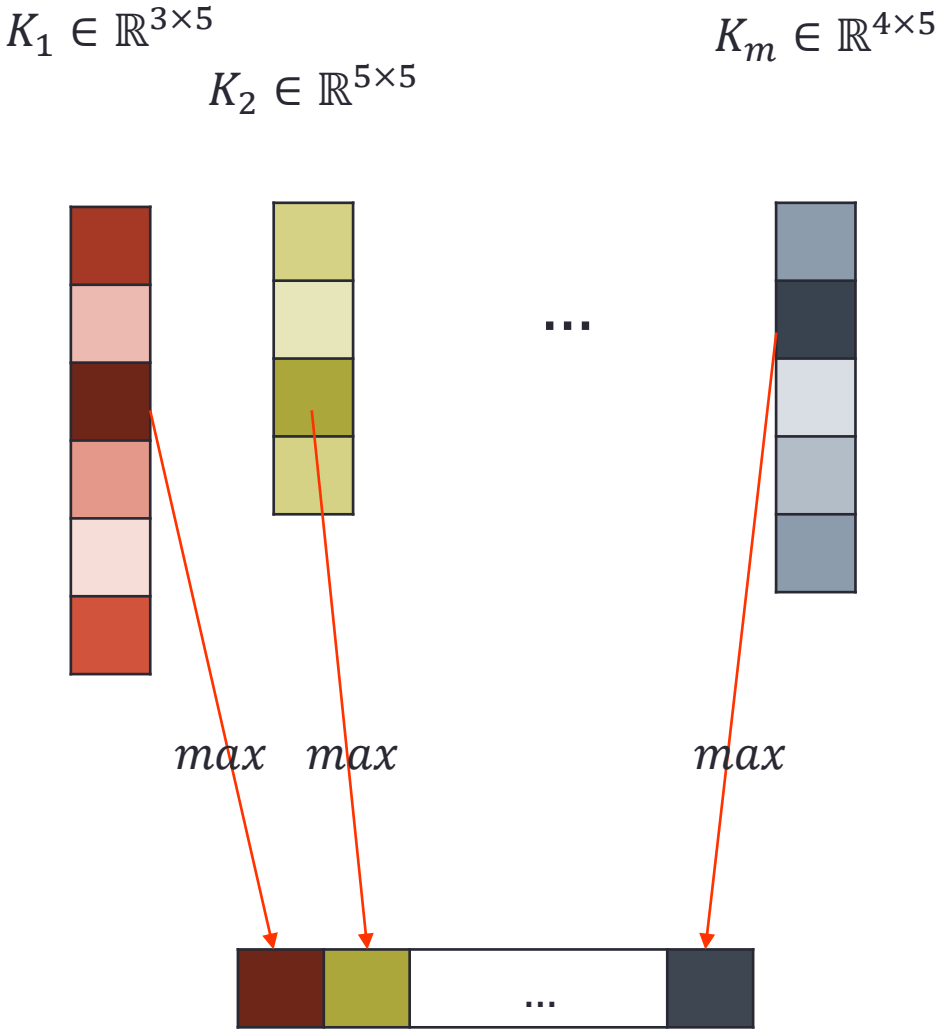


CharCNN



CharCNN


Грозного
Г
р
о
з
н
о
г
о


$K_1 \in \mathbb{R}^{3\times5}$
$K_2 \in \mathbb{R}^{5\times5}$
$K_m \in \mathbb{R}^{4\times5}$
...
max
max
max
...

Следующая лекция

Базовые задачи обработки текстов