

Методы искусственного интеллекта

Лекция 6. Обработка текстов на естественном языке

Тема 12. Обработка текстов на естественном языке. Основы

Зачем обрабатывать тексты на естественном языке?

- **Взаимодействовать с людьми.** (Очень) часто людям удобнее использовать естественный язык для взаимодействия с машиной, нежели искусственные формальные языки (логику предикатов).
- **Обучаться.** Людьми создано огромное количество текстов и фактов, но они выражены именно естественном языке – было бы здорово, чтобы интеллектуальные агенты имели доступ к этим знаниям.
- **Получение новых научных данных о языках и их использовании.** Комбинация ИИ, лингвистики, когнитивной психологии и др.

Проблемы обработки естественного языка

- Языки разные:
 - Фиксированный и не очень порядок слов в предложении
 - Разные типы словообразования
 - Разная степень изменчивости слов
- Языки «живые»
 - Слова (и конструкции) появляются и выходят из употребления
 - Не все тексты «правильные» (соответствуют норме)
- Омонимия – смысловая, частеречная
- Следствия:
 - Ошибки практически неизбежны. Разработанные методы должны оцениваться на достаточно обширном целевом корпусе

Задачи обработки естественного языка

- Лингвистический анализ (определение структуры текста)
- Извлечение признаков – построение векторного, графового представления, сопоставление со словарями
 - Подготовка текста для машинного обучения
- Прикладные задачи (классификация, поиск, извлечение именованных сущностей и пр.)

Лингвистический анализ

- Подготовка
 - Графематический анализ
- Анализ отдельных предложений
 - Морфологический анализ
 - Определение частей речи (POS-теггинг)
 - Извлечение именованных сущностей
 - Синтаксический анализ
 - Семантический анализ
 - Извлечение отношений между сущностями внутри предложения
- Анализ целых текстов
 - Разрешение анафорических связей
 - Дискурсивный анализ

Графематический анализ (токенизация)

- Задача: разбиение на токены и предложения

- Вход:

“До станции оставалось еще с версту. Кругом было тихо, так тихо, что по жужжанию комара можно было следить за его полетом.”

- Выход:

```
[['До', 'станции', 'оставалось', 'еще', 'с', 'версту', '.'], ['Кругом',  
'было', 'тихо', ',', 'так', 'тихо', ',', 'что', 'по', 'жужжанию', 'комара',  
'можно', 'было', 'следить', 'за', 'его', 'полетом', '.']]
```

- Как:

- Регулярные выражения + специальные правила
- Вероятностные модели для разрешения неоднозначностей

Морфологический анализ

- Задача: нахождение нормальной формы и морфологических признаков (падеж, род, время, число) для каждого слова
- Вход:
`['До', 'станции', 'оставалось', 'еще', 'с', 'версту', '.']`
- Выход:
`[DocToken('До', pos='ADP'),
DocToken('станции', pos='NOUN', feats=<Inan,Gen,Fem,Sing>),
DocToken('оставалось', pos='VERB', feats=<Imp,Neut,Ind,Sing,Past,Fin,Mid>),
DocToken('еще', pos='ADV', feats=<Pos>),
DocToken('с', pos='ADP'),
DocToken('версту', pos='NOUN', feats=<Inan,Ins,Neut,Sing>),
DocToken('.', pos='PUNCT')]`

Морфологический анализ

- Сложности:
 - Не всегда по форме слова однозначно понятна его часть речи:
 - «Мама мыла раму.»
 - Разрешается на уровне предложения (с привлечением более широкого контекста)
- Методы:
 - Словари, правила
 - Вероятностные модели (учитывающие контекст)

Морфологический анализ. Граммемы и теги

- **Граммема:** Значение какой-либо грамматической характеристики слова. Например, “множественное число” или “деепричастие”. Множество всех граммем, характеризующих данное слово, образует тег.
- **Тег:** Набор граммем, характеризующих данное слово. Например, для слова “ежам” тегом может быть 'NOUN,anim,masc,plur,datv'.

```
До ADP {}
станции NOUN {
    'Animacy': 'Inan',
    'Case': 'Gen',
    'Gender': 'Fem',
    'Number': 'Sing'}
оставалось VERB {
    'Aspect': 'Imp',
    'Gender': 'Neut',
    'Mood': 'Ind',
    'Number': 'Sing',
    'Tense': 'Past',
    'VerbForm': 'Fin',
    'Voice': 'Mid'}
еще ADV {
    'Degree': 'Pos'}
с ADP {}
версту NOUN {
    'Animacy': 'Inan',
    'Case': 'Ins',
    'Gender': 'Neut',
    'Number': 'Sing'}
. PUNCT {}
```

Морфологический анализ. Граммемы и теги

- Немного разные системы морфологических описаний
 - Например: <https://opencorpora.org/dict.php?act=gram>
- Смотреть документацию по инструменту морфологического анализа

```
До ADP {}
станции NOUN {
    'Animacy': 'Inan',
    'Case': 'Gen',
    'Gender': 'Fem',
    'Number': 'Sing'}
оставалось VERB {
    'Aspect': 'Imp',
    'Gender': 'Neut',
    'Mood': 'Ind',
    'Number': 'Sing',
    'Tense': 'Past',
    'VerbForm': 'Fin',
    'Voice': 'Mid'}
еще ADV {
    'Degree': 'Pos'}
с ADP {}
версту NOUN {
    'Animacy': 'Inan',
    'Case': 'Ins',
    'Gender': 'Neut',
    'Number': 'Sing'}
. PUNCT {}
```

Лемматизация (часть морфологического анализа)

- Задача: **нахождение нормальной формы** и морфологических признаков (падеж, род, время, число) для каждого слова
- Вход:
 - Результат морфологического анализа (в том числе, с неразрешенной частеречной омонимией)
- Выход:

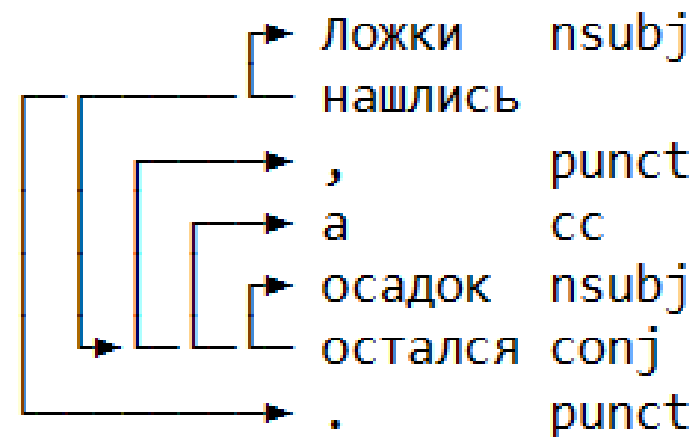
```
[DocToken('До', lemma='до'),  
 DocToken('станции', lemma='станция'),  
 DocToken('оставалось', lemma='оставаться'),  
 DocToken('еще', lemma='еще'),  
 DocToken('с', lemma='с'),  
 DocToken('версту', lemma='верста'),  
 DocToken('.', lemma='.')]
```

Извлечение именованных сущностей

- Задача: выделить блоки, соответствующие именам людей, названиям мест, организаций и пр., установив их класс
- Вход:
['Повесть', 'о', 'том', ',', 'как', 'поссорились', 'Иван', 'Иванович', 'с', 'Иваном', 'Никифоровичем', '.']
- Выход:
Повесть о том, как поссорились Иван Иванович с Иваном Никифоровичем.
PER————— PER—————
- Методы:
 - Словари, правила
 - Вероятностные модели, нейросети

Синтаксический анализ

- Задача: построение структуры предложения (дерева зависимостей)
- Вход:
 - Результаты всех предыдущих этапов
- Выход:
 - Дерево зависимостей
- Методы:
 - Алгоритмы класса сдвиг-свертка (shift-reduce), в том числе, с обучаемыми классификаторами внутри



nsubj – nominal subject (подлежащее)
conj – conjunct (что-то, соединенное союзом)
cc – coordinating conjunction (союз)

См. <https://universaldependencies.org/>

Семантический анализ

- Задача: выделение набора семантических ролей
- Вход:
 - Результаты всех предыдущих этапов
- Выход:
 - Вход разбивается на клаузы (простые предложения, высказывания).
 - Для каждой клаузы определяется главное слово (предикат).
 - Описание ситуации состоит из слотов (ролей) и аргументов им соответствующих. Например, субъект и объект.
 - Происходит выделение аргументов и сопоставление их со слотами.

Извлечение отношений между сущностями

- Вход:
 - Результаты всех предыдущих этапов
- Выход:
 - Для каждой пары сущностей устанавливается класс отношений:
 - Является частью
 - Принадлежит
 - Следует

Прикладные задачи

- Классификация
 - Тематическая классификация длинных текстов
 - Классификация коротких текстов (намерение пользователя, тональность)
- Поиск
 - Поиск по запросу
 - Поиск текста по изображению и изображений по текстам
 - Поиск похожих текстов
 - Вопросно-ответный поиск
- Извлечение структурированной информации
- Диалоговые системы
- Машинный перевод

Векторная модель текста. «Мешок слов»

- Прикладная задача:
 - Классификация (тема текста из известного набора, автор, эпоха и пр.)
- Типично решать с помощью машинного обучения:
 - Поиск функции $y = f(X)$ на основе заданного набора примеров S , исходя из минимизации эмпирической функции ошибки (функции потерь)
- Нужно как-то *представить* текст в виде набора признаков (множество X)
- Также:
 - Поиск похожих текстов – необходимо определить функцию сходства, что проще всего сделать в пространстве определенной размерности

Векторная модель текста. «Мешок слов»

Корпус

Текст
В среду ожидается снегопад.
Дракон живет в подземелье.
В Актюбинске состоится фестиваль.

Все* слова из корпуса

в	среду	ожидается	снегопад	дракон	живет	подземелье	актюбинске	состоится	фестиваль
1	1	1	1						
1				1	1	1			
1							1	1	1

Стоп-слова: и, в, во, не, что, он, на, я, с, ...

*) На практике, как правило, не все:

- исключая очень редкие (это могут быть опечатки)
- исключая самые часто встречающиеся, служебные (они несут мало информации)

Векторная модель текста. Веса

- Количество вхождений слова в документ

$$n_t$$

- Нормализованное количество вхождений слова в документ

$$\frac{n_t}{\sum_k n_k}$$

- TF-IDF (Term Frequency – Inverted Document Frequency)

$$\frac{n_t}{\sum_k n_k} * \log \left(\frac{|D|}{|\{d_i \in D | t \in d_i\}|} \right)$$

Где:

n_t – количество вхождений слова t в документ

D – коллекция документов

d_i – i -тый документ из коллекции

Векторная модель текста для поиска близких текстов

- Косинусное сходство:

$$\text{sim}_{\cos}(d_i, d_j) = \frac{\sum_k d_{ik} d_{jk}}{\sqrt{\sum_k d_{ik}^2} \sqrt{\sum_k d_{jk}^2}}$$

- Это дает возможность:
 - Искать в большой коллекции документы, похожие на заданный
 - Кластеризовать документы

a	<table><tr><td>0.5</td><td>0.5</td><td>0</td><td>0</td><td>0</td></tr></table>	0.5	0.5	0	0	0	$\ a\ \approx 0.7$
0.5	0.5	0	0	0			
b	<table><tr><td>0</td><td>0</td><td>0.3</td><td>0.3</td><td>0.4</td></tr></table>	0	0	0.3	0.3	0.4	$\ b\ \approx 0.58$
0	0	0.3	0.3	0.4			
	<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	
0	0	0	0	0			

$\text{sim}_{\cos}(a, b) = 0$

a	<table><tr><td>0.5</td><td>0.5</td><td>0</td><td>0</td><td>0</td></tr></table>	0.5	0.5	0	0	0	$\ a\ \approx 0.7$
0.5	0.5	0	0	0			
b	<table><tr><td>0</td><td>0.4</td><td>0.3</td><td>0.3</td><td>0</td></tr></table>	0	0.4	0.3	0.3	0	$\ b\ \approx 0.58$
0	0.4	0.3	0.3	0			
	<table><tr><td>0</td><td>0.2</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0.2	0	0	0	
0	0.2	0	0	0			

$\text{sim}_{\cos}(a, b) = \frac{0.2}{0.7 * 0.58} \approx 0.49$

Векторная модель текста для классификации

Текст	Целевая метка
В среду ожидается снегопад.	1
Дракон живет в подземелье.	0
В Актюбинске состоится фестиваль.	0

Логистическая регрессия:
 $\arg \min_{\theta} L(h(X; \theta), y)$

X										y
в	среду	ожидается	снегопад	дракон	живет	подземелье	актюбинске	состоится	фестиваль	Целевая метка
1	1	1	1							1
1				1	1	1				0
1							1	1	1	0

Ограничения векторной модели

- Не учитывает контекст употребления слова
 - Как следствие, дает очень грубое представление о смысле текста, «перемалывает» его
- Не учитывает синонимы и близкие по смыслу слова
- Очень много признаков
 - Доступны только простейшие модели – логистическая регрессия, наивный байесовский классификатор

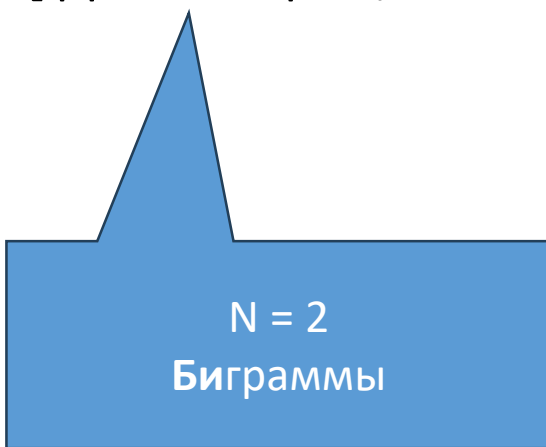
Контекст употребления: N-граммы

- Биграммы, триграммы и т.д.

['До', 'станции', 'оставалось', 'еще', 'с', 'версту', '. ']

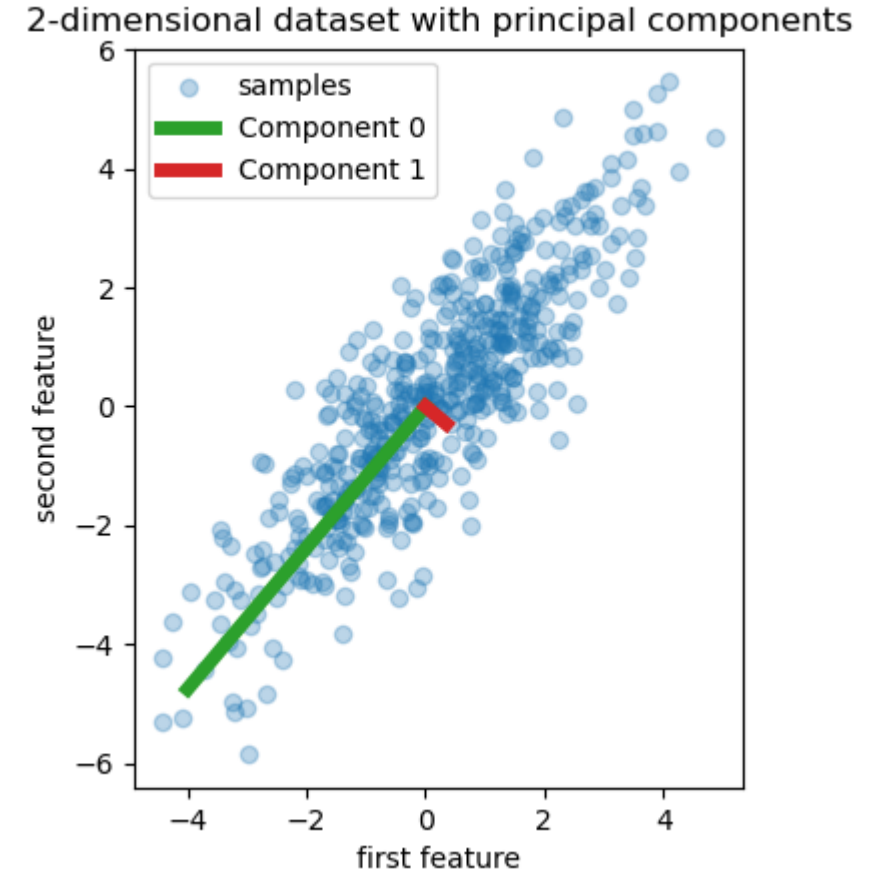


['до станции', 'станции оставалось', 'оставалось еще', 'еще с', 'с версту']




Количество признаков: сокращение размерности

- Метод главных компонент (PCA)
- Латентно-семантический анализ (LSA)
- И др.

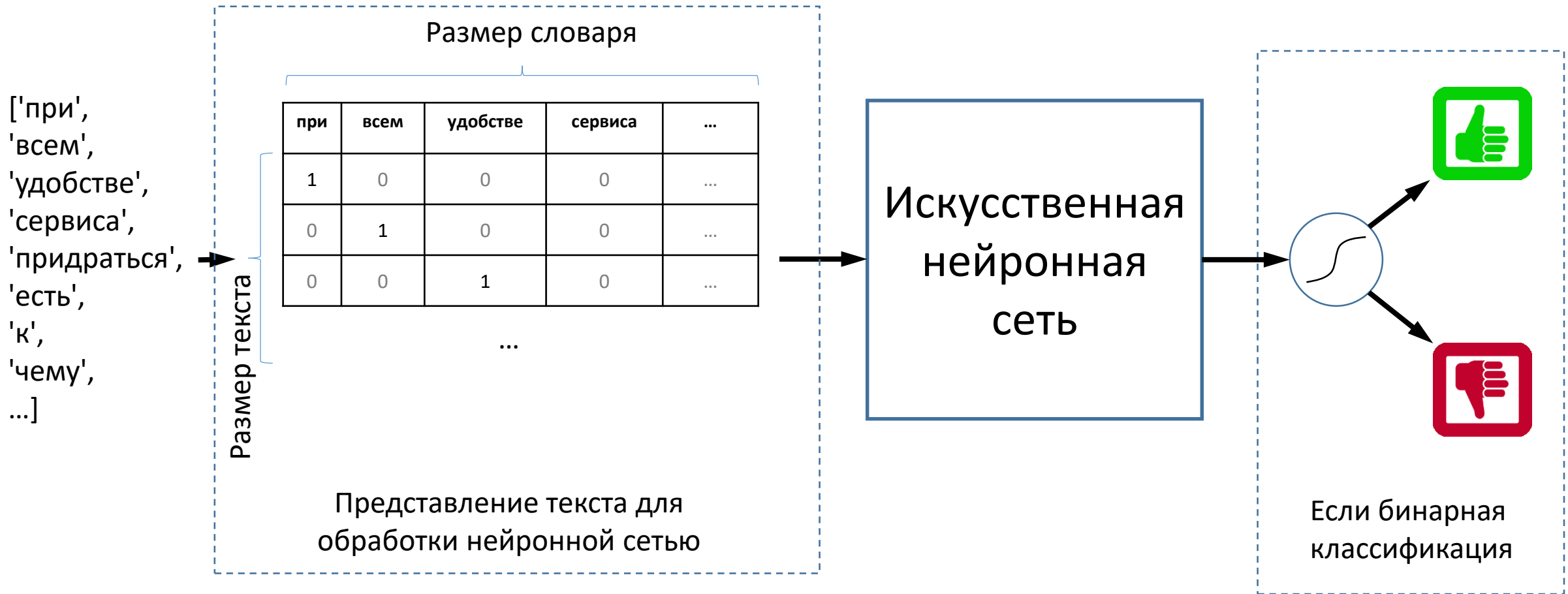


Тема 13. Обработка текстов на естественном языке. Нейросетевые методы

A blue callout box with a triangular pointer pointing towards the word 'языке' in the title above.

И других символьных
последовательностей!

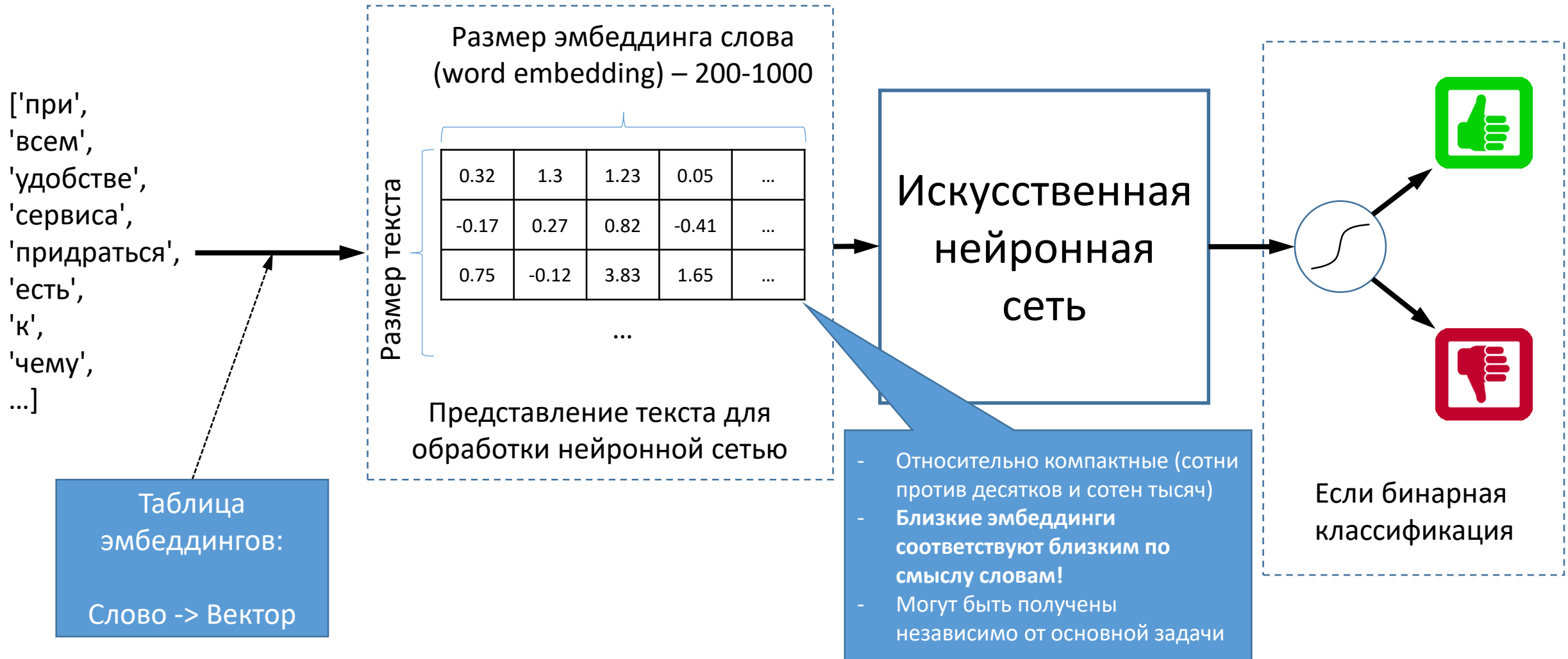
Обобщенная схема нейросетевой модели для обработки текста (неправильная)



Обобщенная схема нейросетевой модели для обработки текста (неправильная)



Обобщенная схема нейросетевой модели для обработки текста (правильная)



Эмбединги и дистрибутивная семантика

- **Дистрибутивная семантика** — это область лингвистики, которая занимается вычислением *степени семантической близости* между лингвистическими единицами на основании их *распределения* в больших массивах лингвистических данных.
- **Дистрибутивная гипотеза** — лингвистические единицы, встречающиеся в схожих контекстах, имеют близкие значения.

Например:

Выпей еще *чашку* **чая**.

Выпей еще **кофе**.

Выпей *стакан* **воды**.


GloVe. Матрица совместной встречаемости

['выпью', 'чаю', 'выпью', 'кофе', 'сьем', 'пончик', ...]

$X_{ij} =$

	выпью	чаю	кофе	сьем	пончик
выпью					
чаю					
кофе					
сьем					
пончик					

GloVe. Матрица совместной встречаемости

 ['выпью', 'чаю' 'выпью', 'кофе', 'съем', 'пончик', ...]

$X_{ij} =$

	выпью	чаю	кофе	съем	пончик
выпью		1			
чаю					
кофе					
съем					
пончик					


GloVe. Матрица совместной встречаемости

▼
'выпью', 'чаю', 'выпью' 'кофе', 'съем', 'пончик', ...]

$X_{ij} =$

	выпью	чаю	кофе	съем	пончик
выпью		1			
чаю	2				
кофе					
съем					
пончик					

GloVe. Матрица совместной встречаемости

 ['выпью', 'чаю', 'выпью', 'кофе', 'съем', 'пончик', ...]

$X_{ij} =$

	выпью	чаю	кофе	съем	пончик
выпью		2	1		
чаю	2				
кофе					
съем					
пончик					

GloVe. Матрица совместной встречаемости

['выпью', 'чаю', 'выпью', 'кофе', 'съем', 'пончик', ...]

$X_{ij} =$

	выпью	чаю	кофе	съем	пончик
выпью		2	1		
чаю	2				
кофе	1			1	
съем					
пончик					

GloVe. Матрица совместной встречаемости

['выпью', 'чаю', 'выпью'  'кофе', 'съем', 'пончик' ...]

$X_{ij} =$

	выпью	чаю	кофе	съем	пончик
выпью		2	1		
чаю	2				
кофе	1			1	
съем			1		1
пончик					

Имеет смысл делать только на
ОЧЕНЬ больших корпусах:
 $6 \cdot 10^9$, $42 \cdot 10^9$, $840 \cdot 10^9$ токенов

GloVe. Сглаживание

Цель: уменьшение разброса значений, борьба с выбросами.

$$\log(1 + X_{ij})$$

х	$\log(1 + x)$
0	0
10	2.4
100	4.62
1000	6.91
10000	9.21

GloVe. Аппроксимация

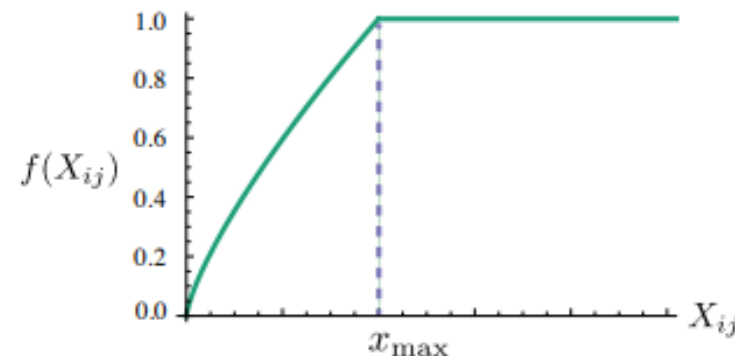
Цель: подбор значений параметров для каждого слова, чтобы минимизировать ошибку реконструкции сглаженной матрицы совместной встречаемости.

Метод: градиентный спуск.

Особенности: дополнительный акцент на значимые совместные употребления.

$$\arg \min_w \sum_{i \neq j} f(X_{ij}) (w_i w_j^T - \log(1 + X_{ij}))^2$$

$$f(x) = \begin{cases} (x/x_{max})^\alpha & \text{если } x < x_{max} \\ 1 & \text{иначе} \end{cases}$$



GloVe. Общий алгоритм

1. Построить матрицу совместной встречаемости в контексте определенной ширины W (в оригинале – $W=1$), X_{ij} – количество раз, когда слово i оказывается не дальше W от слова j .
2. Определить гиперпараметры, регулирующие обработку редких пар: α и X_{max} .
3. Используя градиентный спуск, подобрать такие значения параметров w_i, w_j (для всех пар слов i, j), чтобы минимизировать следующую функцию потерь:

$$\sum_{i \neq j} f(X_{ij}) (w_i w_j^T - \log(1 + X_{ij}))^2$$

4. Значения параметров w_i, w_j и есть искомые эмбединги!

Word2Vec

Идея: подобрать такие вектора слов, чтобы минимизировать ошибку предсказания:

- слова по контексту (CBOW);
- окружающих слов по известному (Skip-grams).

Метод: градиентный спуск.

Word2Vec. CBOW – Continuous Bag of Words (предсказание слова по контексту)

▼
'выпью', 'чаю', 'выпью' 'кофе', 'съем', 'пончик', ...]


Размер контекста

Размер контекста		
Слово 1	Слово 2	Целевое слово
выпью	выпью	чаю

Признаки НЕ
упорядочены
(поэтому Bag of Words)

Word2Vec. CBOW – Continuous Bag of Words (предсказание слова по контексту)


['выпью', 'чаю', 'выпью', 'кофе' 'съем', 'пончик', ...]



Слово 1	Слово 2	Целевое слово
выпью	выпью	чаю
чаю	кофе	выпью

Word2Vec. CBOW – Continuous Bag of Words (предсказание слова по контексту)

['выпью', 'чаю', 'выпью', 'кофе', 'съем', 'пончик', ...]



Слово 1	Слово 2	Целевое слово
выпью	выпью	чаю
чаю	кофе	выпью
выпью	съем	кофе

Word2Vec. CBOW – Continuous Bag of Words (предсказание слова по контексту)

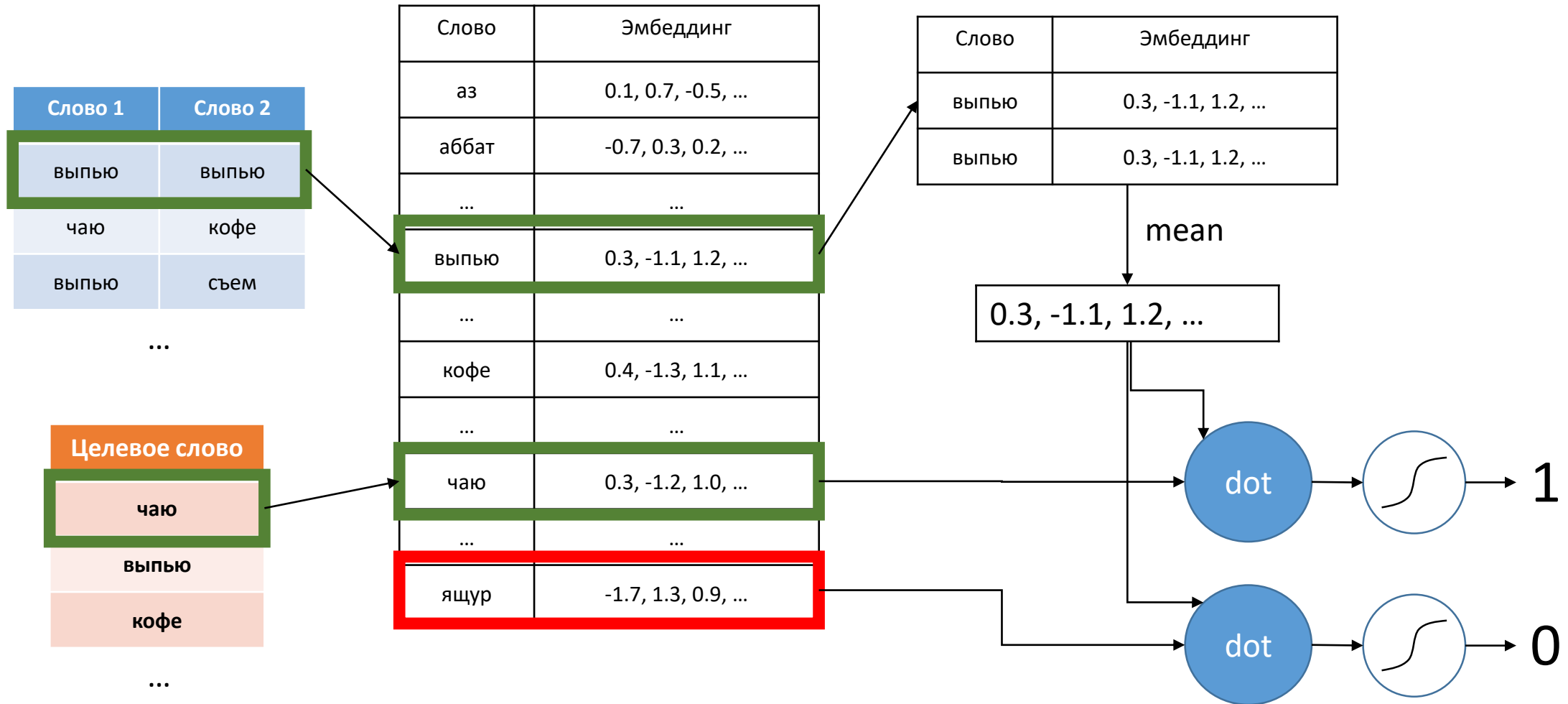
Слово 1	Слово 2	Целевое слово
выпью	выпью	чаю
чаю	кофе	выпью
выпью	съем	кофе

...

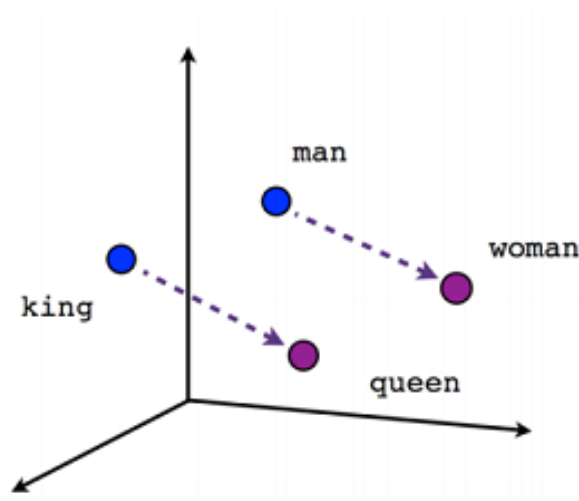
аз	аббат	...	выпью	...	кофе	...	чаю	...	ящур
0	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0

...

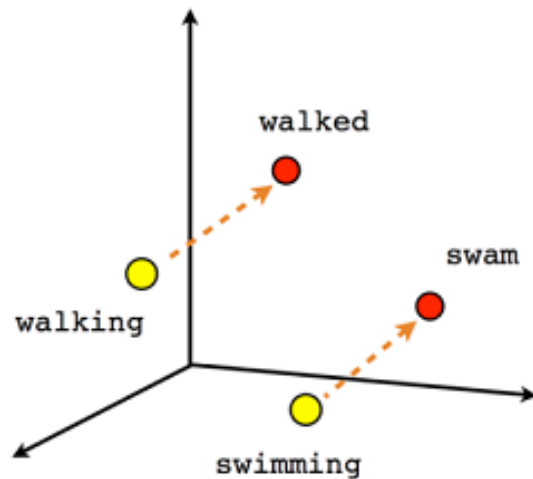
Word2Vec. CBOW – Continuous Bag of Words (предсказание слова по контексту)



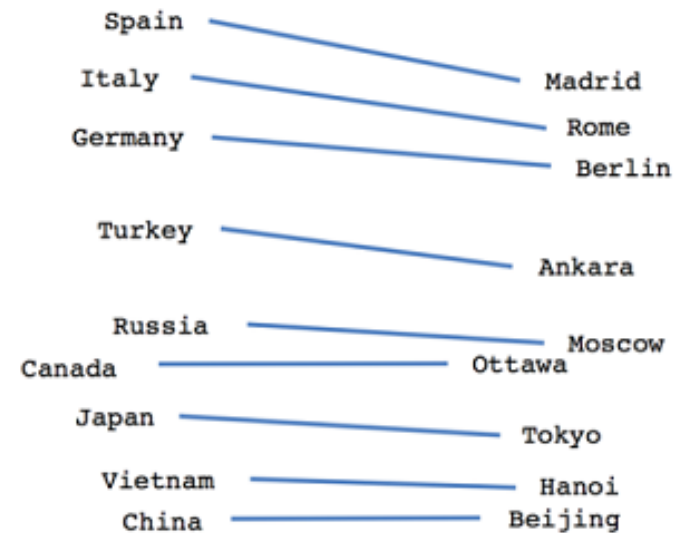
Интересные свойства эмбедингов



Male-Female



Verb tense



Country-Capital

king + (woman - man) = queen
Portugal + (Moscow - Russia) = Lisbon

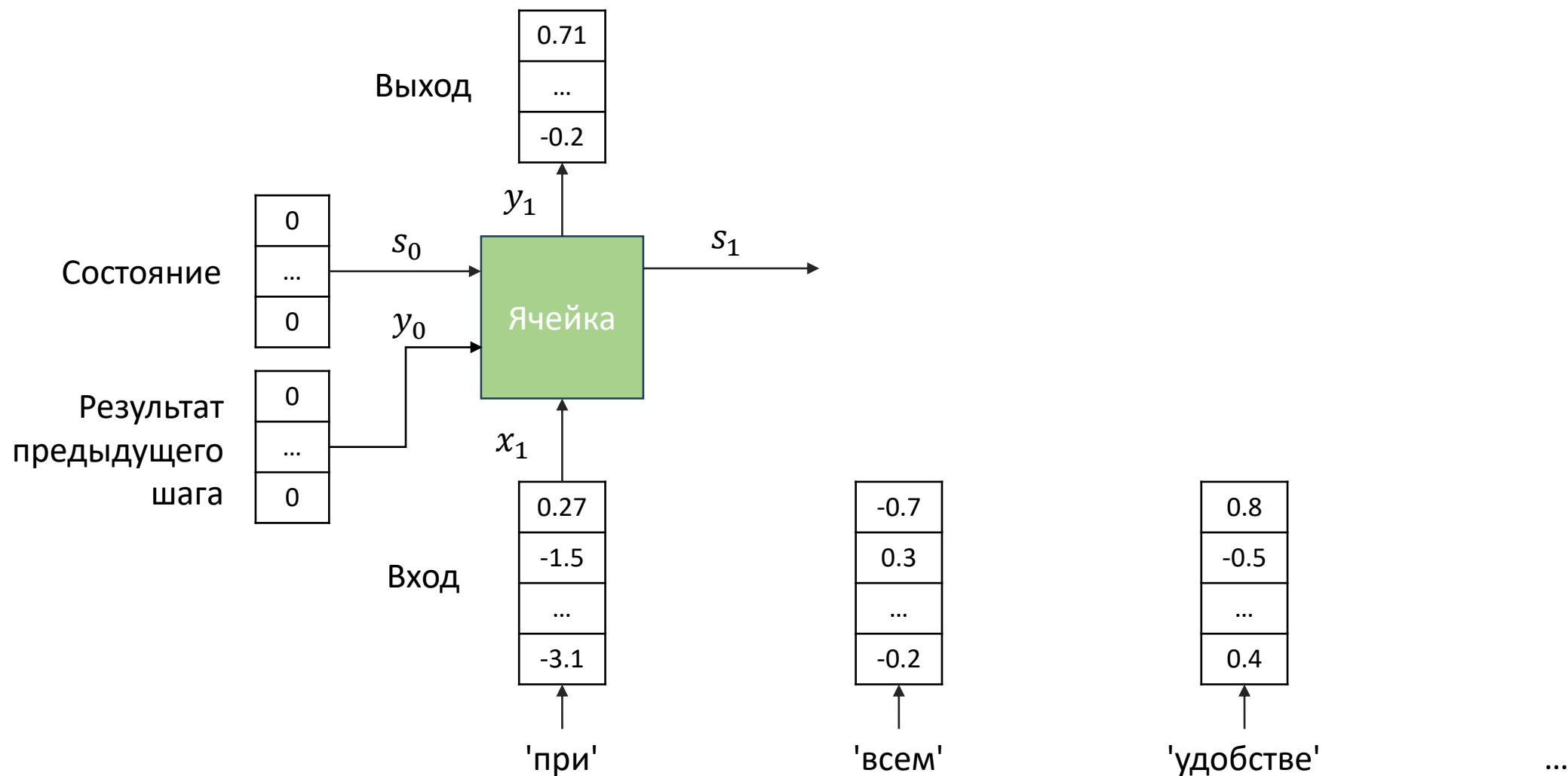
Не обязательно слова!

- История действий пользователя:
 - Действие – слово
 - Последовательность действий – предложение или текст
 - Внутренняя логика этой последовательности так же определяется неким явно не доступным, но присутствующим у автора замыслом
 - Используя схожие (или непосредственно эти) техники можно получить вложения для действий, обладающие тем же качеством – схожие действия обладают близкими векторами вложений – и дальше использовать их при решении прикладных задач (предсказание следующего элемента, рекомендации и пр.)

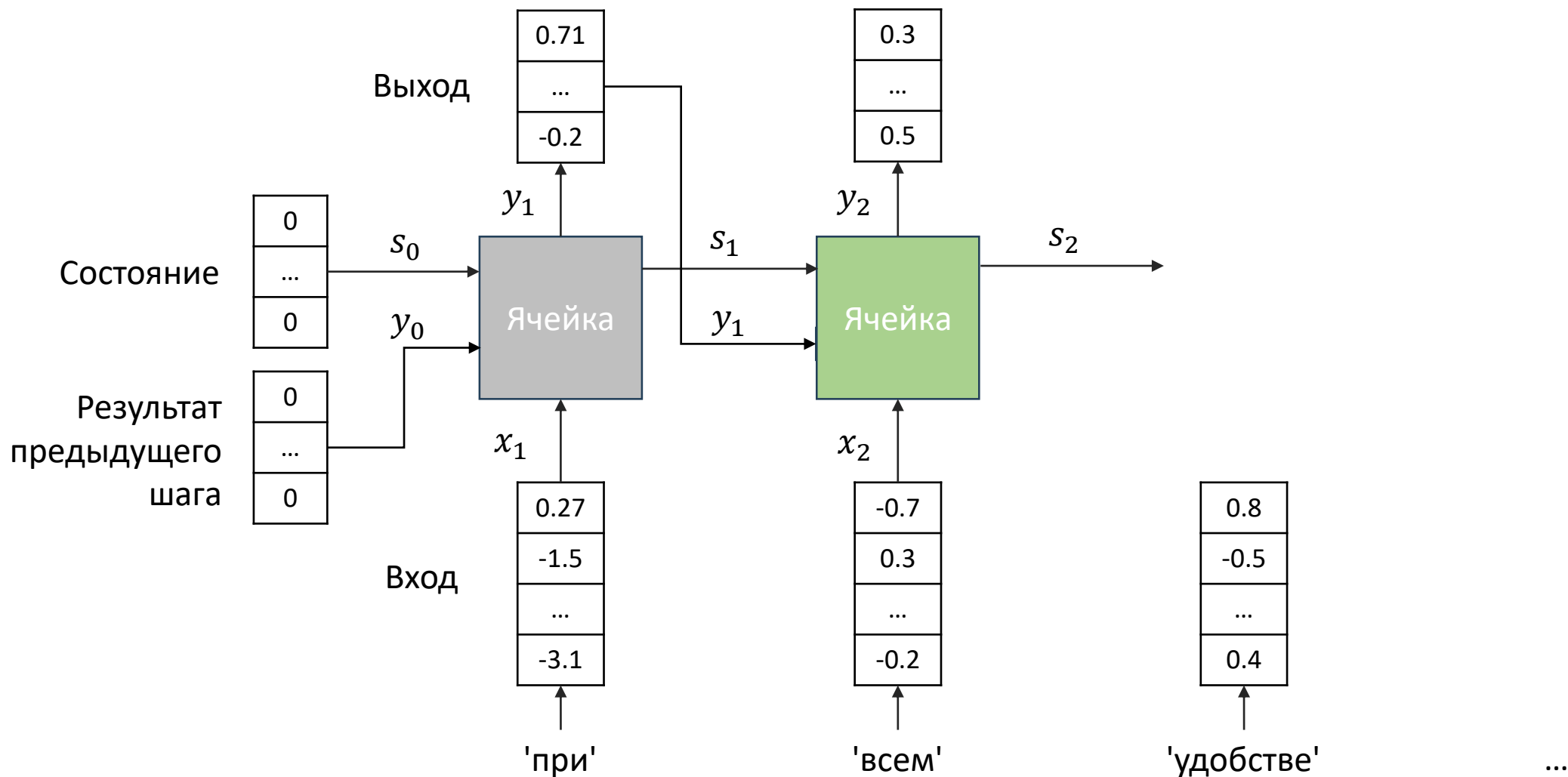
Обработка последовательностей нейронной сетью

- 1D CNN
- Рекуррентные сети
 - RNN
 - LSTM
 - GRU
- Трансформер

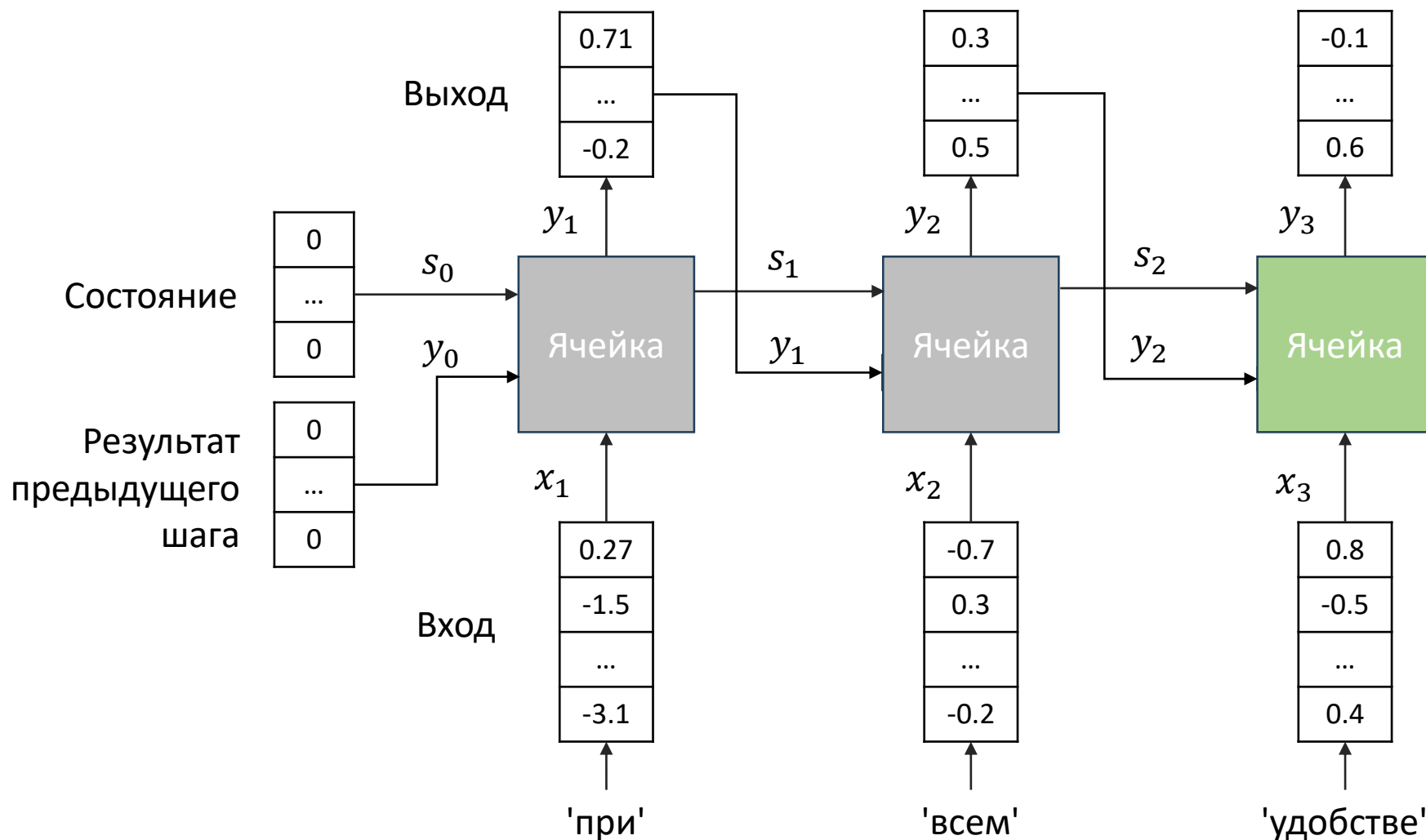
Рекуррентная сеть. Общая идея



Рекуррентная сеть. Общая идея



Рекуррентная сеть. Общая идея



Ключевые моменты:

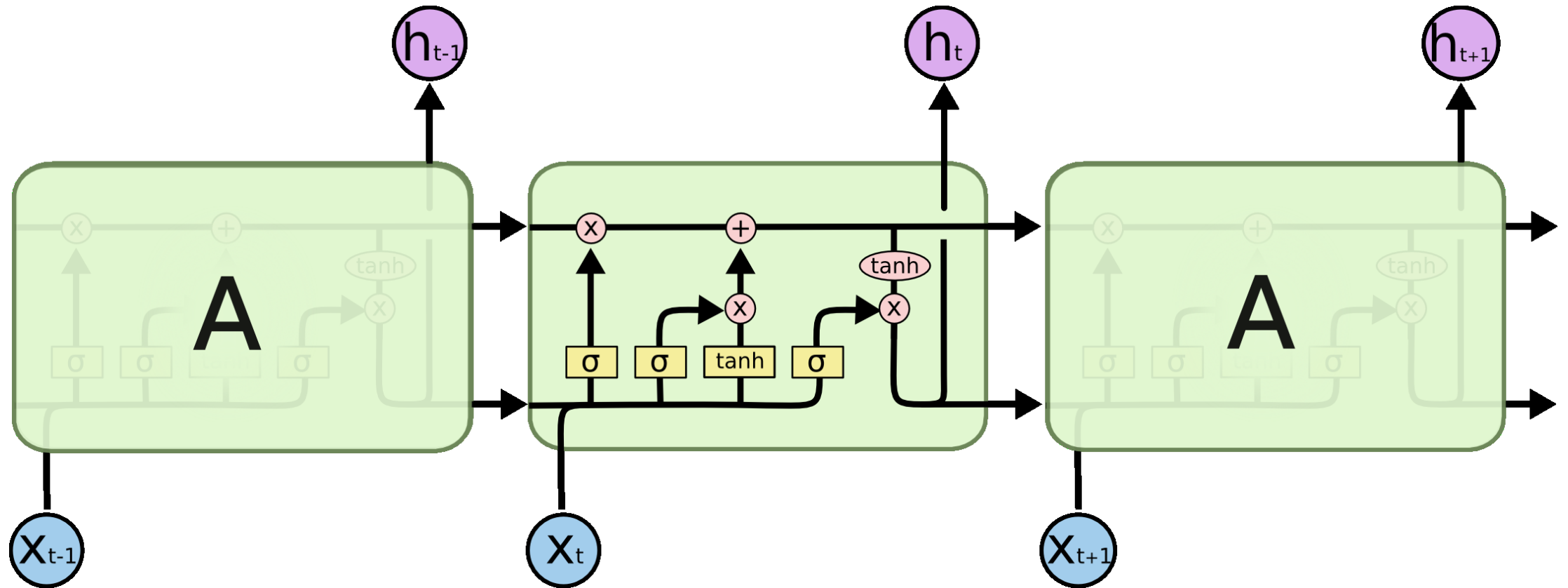
- последовательная обработка
- одни и те же коэффициенты для всех элементов последовательности
- результат зависит от элемента последовательности (x_i), состояния (s_{i-1}) и результата предыдущего шага (y_{i-1})

...

LSTM (Long Short-Term Memory)

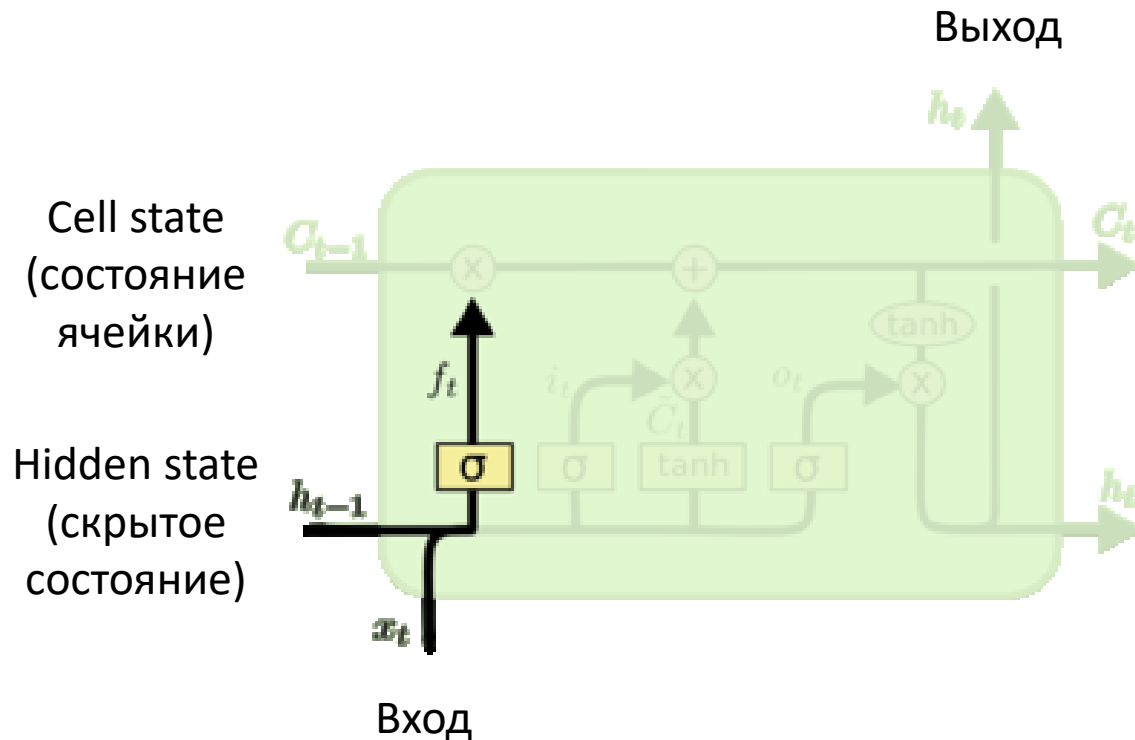
- Предложена еще в 1997 году как улучшение простых рекуррентных ячеек, у которых были существенные проблемы с обучаемостью и долговременной памятью
- Была основным инструментом решения задач, связанных с обработкой последовательностей (в том числе, языковых) примерно до 2017 года

LSTM (Long Short-Term Memory)



LSTM (Long Short-Term Memory). Forget Gate

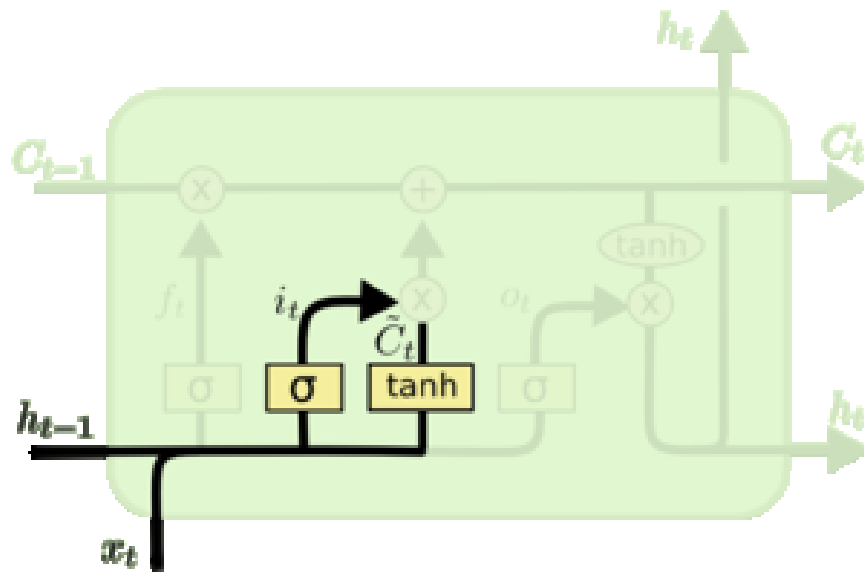
Forget gate
(вентиль, фильтр) забывания



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

LSTM (Long Short-Term Memory). Input Gate

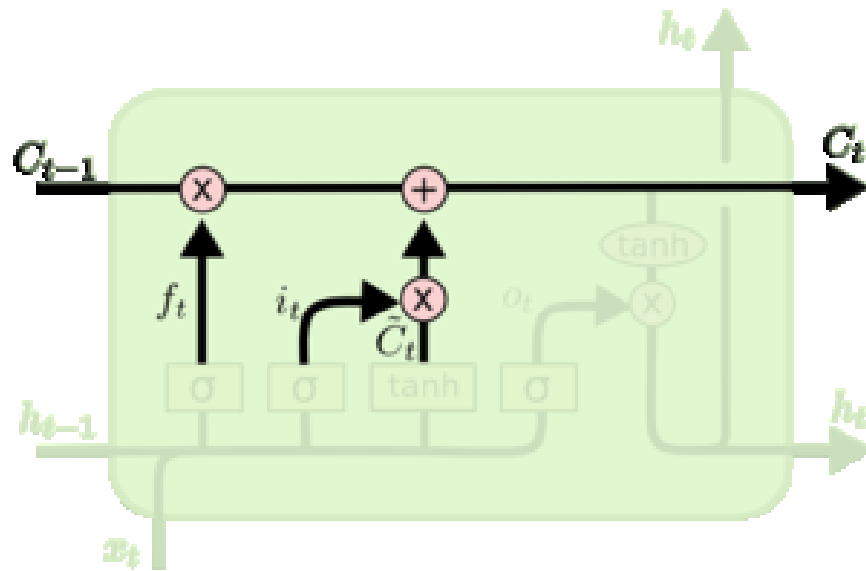
Input gate
(вентиль, фильтр) входа



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{c}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

LSTM (Long Short-Term Memory)

Обновление состояния ячейки

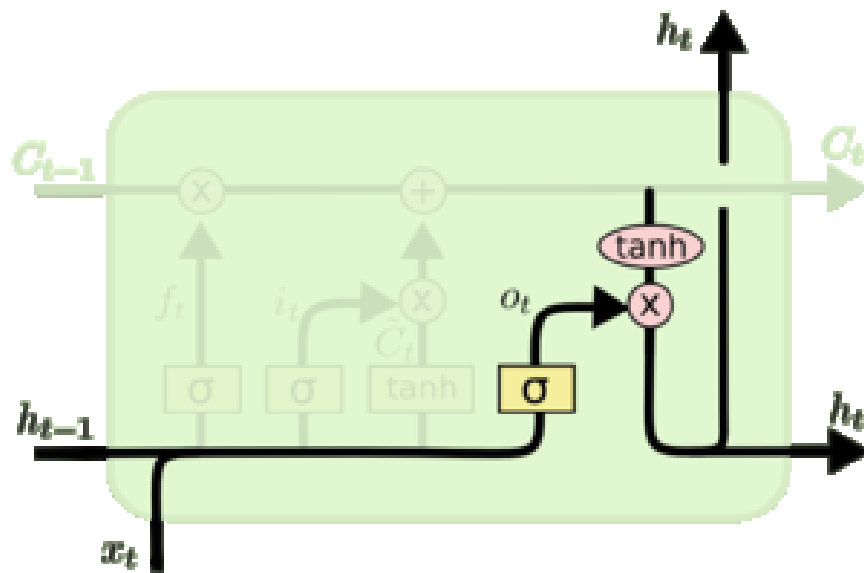


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM (Long Short-Term Memory). Output Gate

Output gate

(вентиль, фильтр) выхода



$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

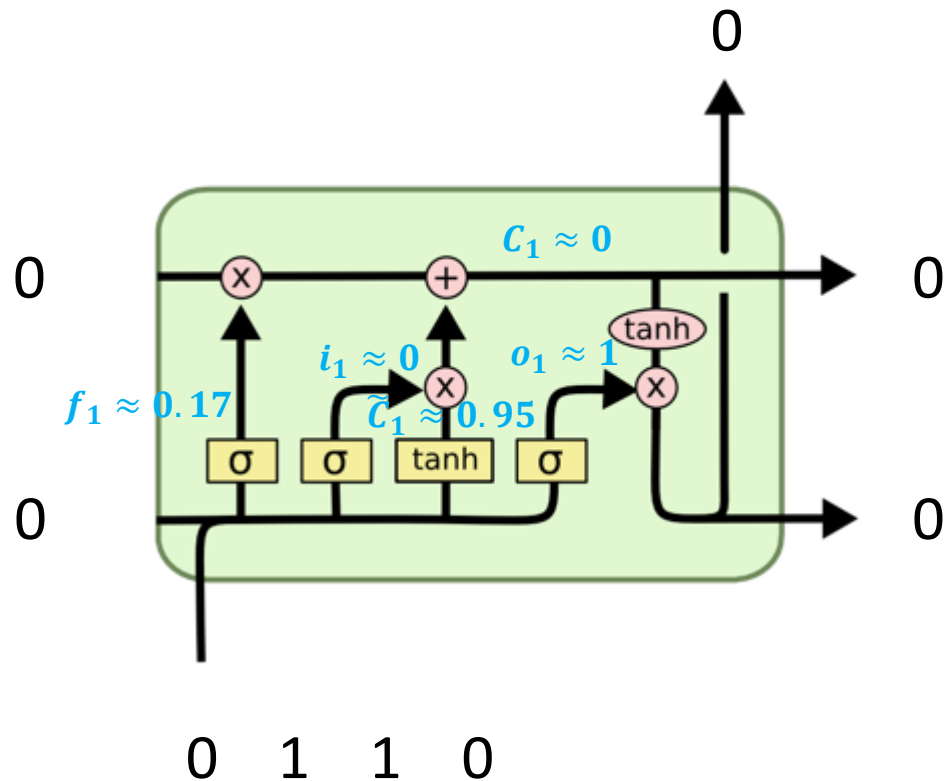
LSTM. Наглядный (но очень упрощенный) пример

Задача: определить четность количества единиц в последовательности (0 – четное, 1 – нечетное)

Вход: 0 1 0 1 0 0 1 0

Выход: 0 1 1 0 0 0 1 1

LSTM. Наглядный (но очень упрощенный) пример



Параметры:

$$W_f = (3.5; -2.3); b_f = -1.6$$

$$W_i = (4.8; 6.6); b_i = -3.1$$

$$W_c = (-4.3; -1.9); b_c = 5$$

$$W_o = (2; 3.3); b_o = 4.6$$

Вычисления:

$$\begin{aligned} f_1 &= \sigma(W_f \cdot [h_0; x_1] + b_f) = \\ &= \sigma\left((3.5; -2.3) \cdot \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 1.6\right) \approx 0.17 \end{aligned}$$

$$i_1 = \sigma\left((4.8; 6.6) \cdot \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 3.1\right) \approx 0$$

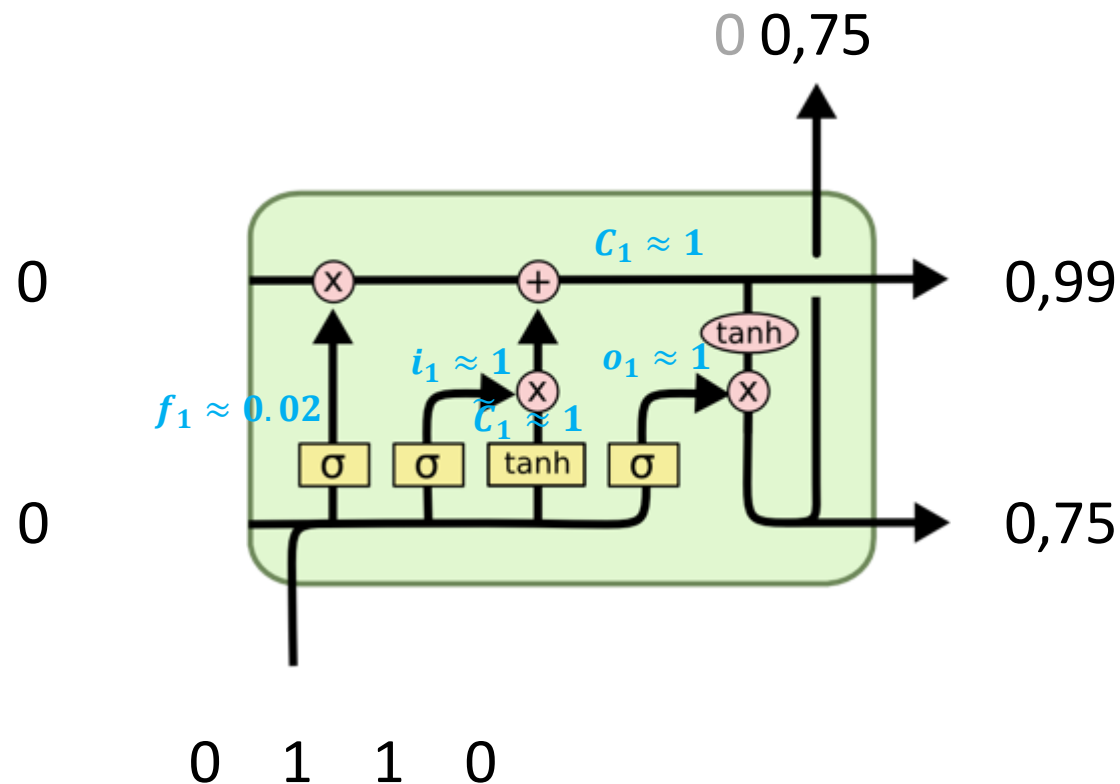
$$\tilde{c}_1 = \tanh\left((-4.3; -1.9) \cdot \begin{pmatrix} 0 \\ 0 \end{pmatrix} + 5\right) \approx 0.95$$

$$c_1 = 0 * 0.17 + 0 * 0.95 = 0$$

$$o_1 = \sigma\left((2; 3.3) \cdot \begin{pmatrix} 0 \\ 0 \end{pmatrix} + 4.6\right) \approx 1$$

$$h_1 = 1 * \tanh(0) \approx 0$$

LSTM. Наглядный (но очень упрощенный) пример



Параметры:

$$W_f = (3.5; -2.3); b_f = -1.6$$

$$W_i = (4.8; 6.6); b_i = -3.1$$

$$W_c = (-4.3; -1.9); b_c = 5$$

$$W_o = (2; 3.3); b_o = 4.6$$

Вычисления:

$$f_1 = \sigma(W_f \cdot [h_0; x_1] + b_f) = \\ = \sigma\left((3.5; -2.3) \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} - 1.6\right) \approx 0.02$$

$$i_1 = \sigma\left((4.8; 6.6) \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} - 3.1\right) \approx 0.97$$

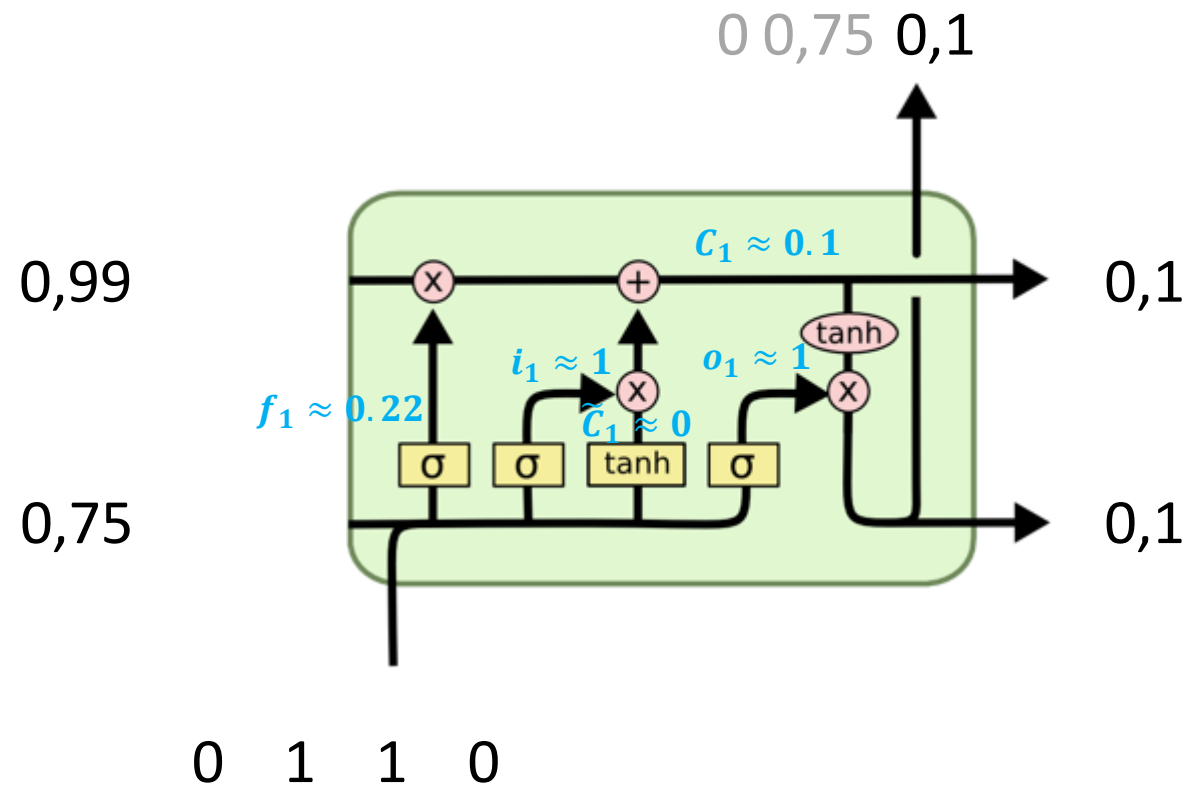
$$\tilde{c}_1 = \tanh\left((-4.3; -1.9) \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} + 5\right) \approx 0.99$$

$$c_1 = 0 * 0.02 + 1 * 0.99 = 0.99$$

$$o_1 = \sigma\left((2; 3.3) \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} + 4.6\right) \approx 1$$

$$h_1 = 1 * \tanh(0.99) \approx 0.75$$

LSTM. Наглядный (но очень упрощенный) пример



Параметры:

$$W_f = (3.5; -2.3); b_f = -1.6$$

$$W_i = (4.8; 6.6); b_i = -3.1$$

$$W_c = (-4.3; -1.9); b_c = 5$$

$$W_o = (2; 3.3); b_o = 4.6$$

Вычисления:

$$f_1 = \sigma(W_f \cdot [h_0; x_1] + b_f) = \\ = \sigma\left((3.5; -2.3) \cdot \begin{pmatrix} 0.75 \\ 1 \end{pmatrix} - 1.6\right) \approx 0.22$$

$$i_1 = \sigma\left((4.8; 6.6) \cdot \begin{pmatrix} 0.75 \\ 1 \end{pmatrix} - 3.1\right) \approx 0.97$$

$$\tilde{c}_1 = \tanh\left((-4.3; -1.9) \cdot \begin{pmatrix} 0.75 \\ 1 \end{pmatrix} + 5\right) \approx -0.12$$

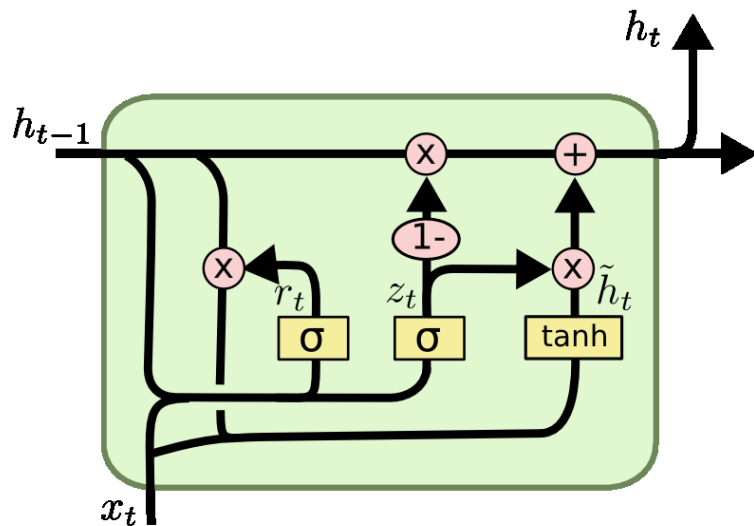
$$c_1 = 0.99 * 0.22 + 1 * (-0.12) = 0.1$$

$$o_1 = \sigma\left((2; 3.3) \cdot \begin{pmatrix} 0.75 \\ 1 \end{pmatrix} + 4.6\right) \approx 1$$

$$h_1 = 1 * \tanh(0.1) \approx 0.1$$

GRU (Gated Recurrent Unit)

- Предложена в 2014 году
- Быстрее (меньше операций)
- Чуть меньшие выразительные возможности
- Особенности:
 - Состояние ячейки объединено со скрытым состоянием (только h вместо C, h)
 - Вентили забывания и входа объединены в один – обновления (update gate)
 - Вентиль сброса (reset gate), схожий по назначению с вентилем забывания (forget gate) LSTM
 - Выходного вентиля нет



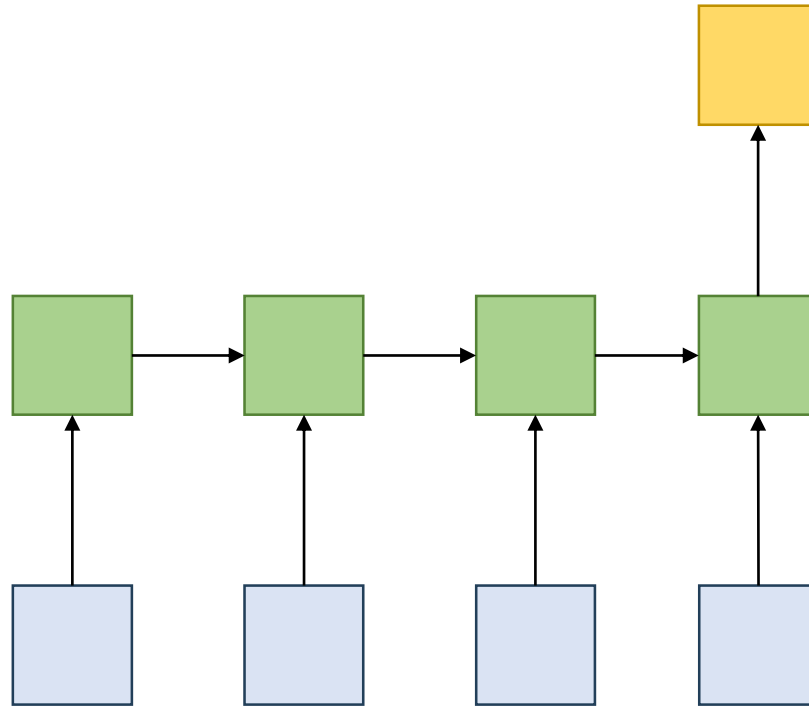
$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

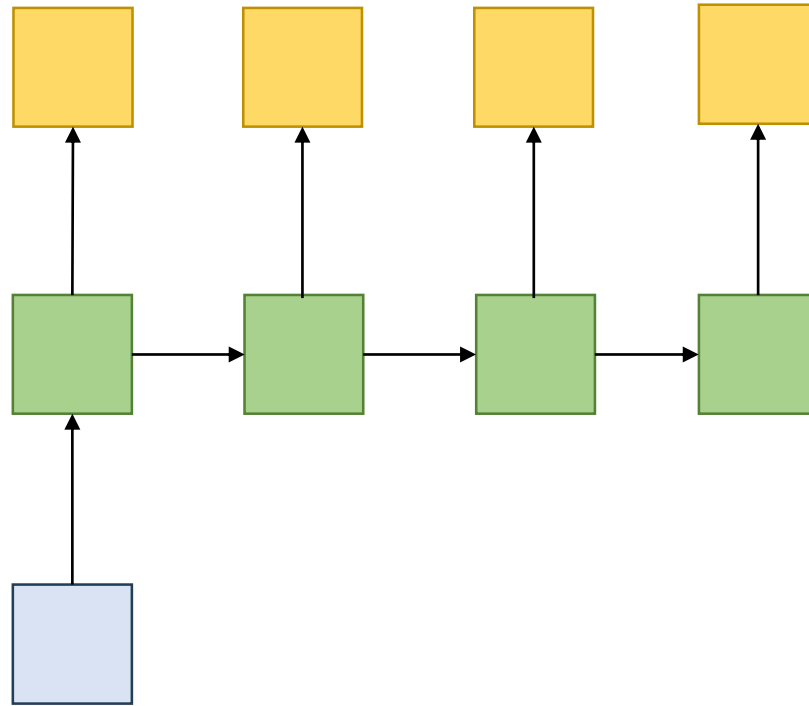
$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

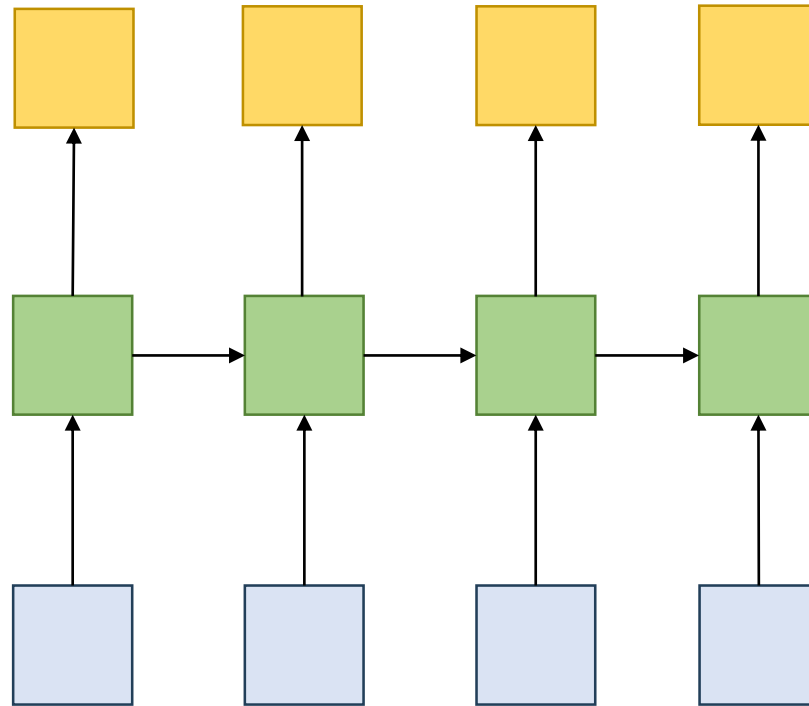
Виды архитектурных решений на основе рекуррентных сетей. Много-к-одному



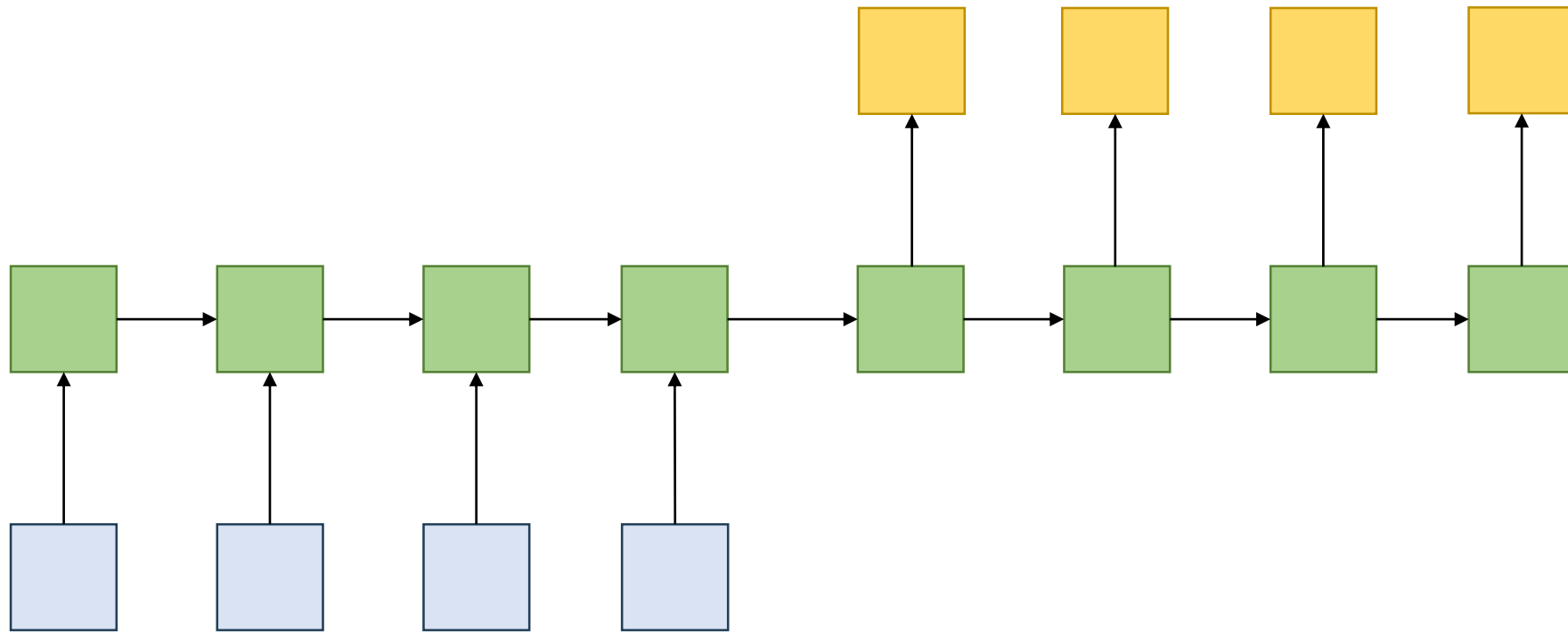
Виды архитектурных решений на основе рекуррентных сетей. Один-ко-многим



Виды архитектурных решений на основе рекуррентных сетей. Много-ко-многим

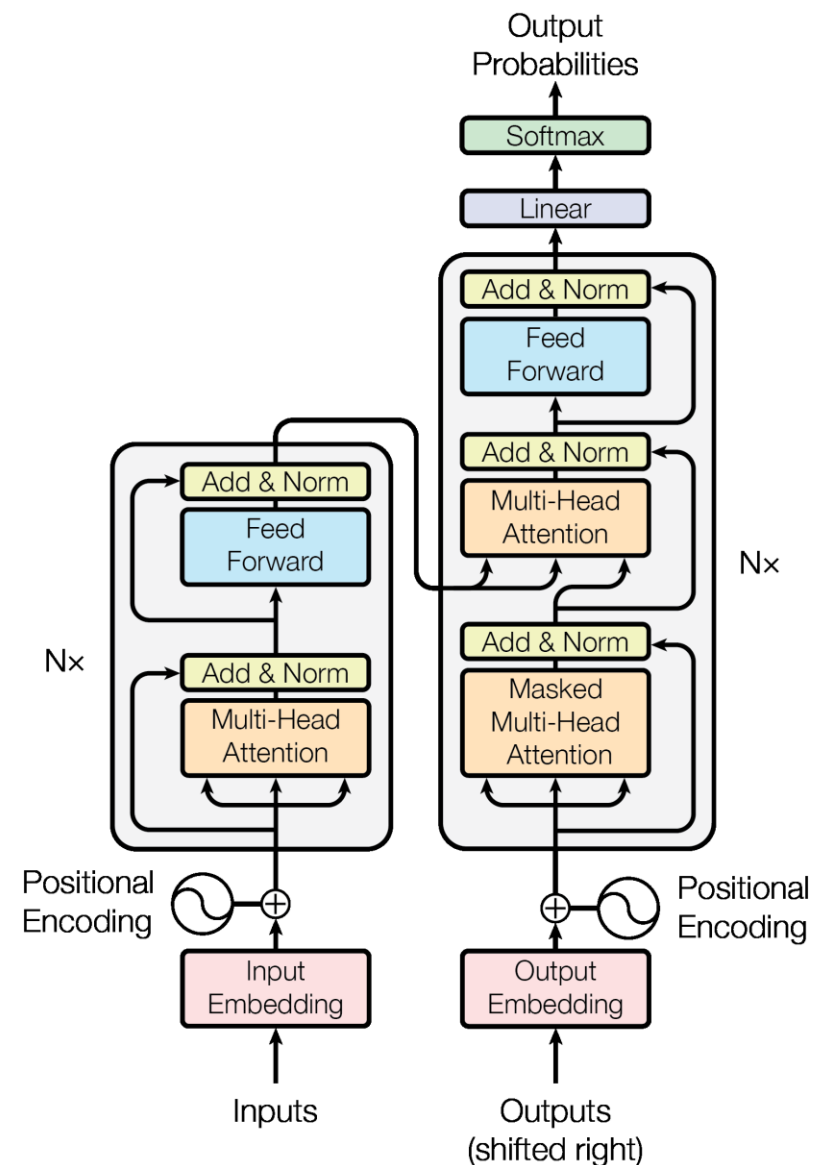


Виды архитектурных решений на основе рекуррентных сетей. Много-ко-многим



Пара слов об архитектуре трансформер

- Предложена в 2017 году
- Данные обрабатываются параллельно (нет последовательного применения блока к элементам, как это происходит в рекуррентных сетях)
 - Вместо рекуррентных ячеек – механизм внимания (attention)
- Лучшие результаты в большинстве задач, связанных с обработкой языка (и последовательностей в целом)



Резюме

- Есть несколько типовых стадий обработки текста, применимость и необходимость каждой стадии сильно зависят от задачи
- Векторная модель документа («мешок слов»)
 - Несмотря на простоту, неплохо работает в ряде задач (тематическая классификация больших текстов)
 - Разновидность: N-граммы
- Эмбединги (вложения) слов. Близкие вектора – близкие слова:
 - GloVe
 - Word2Vec
- Построение нейросетевых моделей обработки текста (последовательностей):
 - 1D свёртки
 - **Рекуррентные сети**
 - LSTM, GRU
 - Трансформеры

Литература

- Основное:
 - Dive into Deep Learning. <https://d2l.ai/>
- Дополнительно:
 - Нейронные сети и обработка текста. Stepik: <https://stepik.org/course/54098>
 - CS224N: Natural Language Processing with Deep Learning <https://web.stanford.edu/class/cs224n/> , (YouTube) <https://www.youtube.com/playlist?list=PLoROMvovdv4rOSH4v6133s9LFPRHjEmbmJ>
 - Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. <https://nlp.stanford.edu/projects/glove/>
 - Mikolov T. et al. Efficient Estimation of Word Representations in Vector Space. <https://arxiv.org/abs/1301.3781>