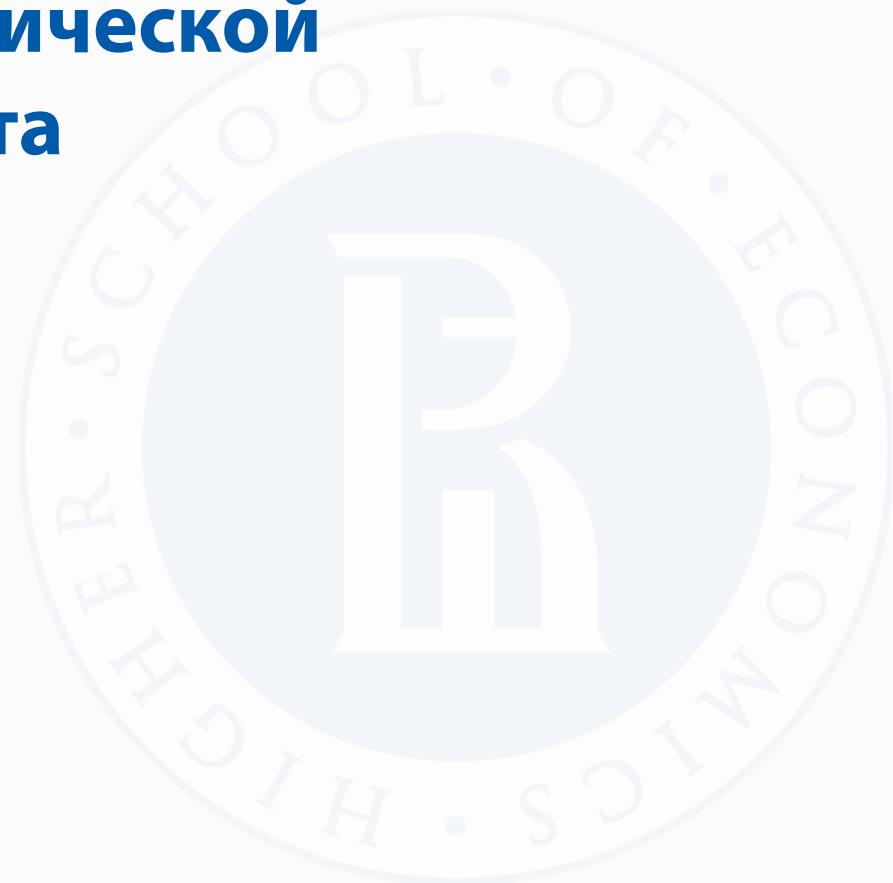


Задачи автоматической обработки текста



План



Основные задачи обработки текстов



План



Основные задачи обработки текстов



Стандартный подход в задачах обработки текстов

План



Основные задачи обработки текстов



Стандартный подход в задачах обработки текстов



Предобработка текстовых данных



Текстовая информация

→ Социальные сети, блоги, мессенджеры



Текстовая информация

- Социальные сети, блоги, мессенджеры
- Специфичные базы знаний (в том числе Википедия)



Текстовая информация

- Социальные сети, блоги, мессенджеры
- Специфичные базы знаний (в том числе Википедия)
- Отзывы и рецензии



Текстовая информация

- Социальные сети, блоги, мессенджеры
- Специфичные базы знаний (в том числе Википедия)
- Отзывы и рецензии
- Новости и статьи



Основные задачи обработки текстов

→ Классификация: по тональности, по тематике,
спам / не спам

Основные задачи обработки текстов

- Классификация: по тональности, по тематике, спам / не спам
- Ранжирование: поиск, рекомендации, диалоговые системы

Основные задачи обработки текстов

- Классификация: по тональности, по тематике, спам / не спам
- Ранжирование: поиск, рекомендации, диалоговые системы
- Разметка последовательности (sequence labeling): извлечение именованных сущностей, POS-теггинг (part-of-speech теггинг, разметка по частям речи)

Основные задачи обработки текстов

- Классификация: по тональности, по тематике, спам / не спам
- Ранжирование: поиск, рекомендации, диалоговые системы
- Разметка последовательности (sequence labeling): извлечение именованных сущностей, POS-теггинг (part-of-speech теггинг, разметка по частям речи)
- Seq2seq (sequence to sequence) задачи: машинный перевод, суммаризация, исправление опечаток

Основные задачи обработки текстов

- Классификация: по тональности, по тематике, спам / не спам
- Ранжирование: поиск, рекомендации, диалоговые системы
- Разметка последовательности (sequence labeling): извлечение именованных сущностей, POS-теггинг (part-of-speech теггинг, разметка по частям речи)
- Seq2seq (sequence to sequence) задачи: машинный перевод, суммаризация, исправление опечаток
- и многое другое!

Как обрабатывать тексты?

→ Текстовая информация чаще представлена в неструктурированном виде:



abc-mark

[рекомендации](#) | [оценки](#) | [друзья](#) | [фильмы](#) | [звезды](#)

21 сентября 2020 | 13:56

типа рецензии:  

Инверсия пройденного

Если начинать с доводов «за» просмотр нового блокбастера, на который возлагается ответственная роль спасителя загибающегося от ковидной чумы мирового кинопроката, то, несомненно, главным из них будет имя режиссера-сценариста в титрах. «Довод» чисто нолановский перфекционистский фильм, проработанный до мелочей, с фирменной непривычностью повествования, эталонным визуальным рядом, спецэффектами и добротным актерским составом. На данный момент в фильмографии режиссера нет картины, которой нельзя было бы присвоить знак качества и «Довод» в этом плане не является исключением. Однако, увы, вполне очевидна и вторичность данного фильма.

«Довод» есть качественная компиляция хорошо апробированных приемов и сюжетных ходов из предыдущих работ Нолана, которые после многократного повторения перестают удивлять. Прежде всего бросаются в глаза близкие к цитированию зеркальные заимствования из «Начала» (2010). От него «Доводу» достались и фабула, и общая структура, и отдельные мизансцены. От титров до титров фильм движется с идентичной динамикой, хронометражем и смысловыми паузами. Существенным отличием «Довода» от «Начала» является глубина, а точнее «мелководье» положенной в основу сценария драматургии. Нолан продолжает идти по пути усложнения повествования, жертвуя всем самым важным, что заставляет зрителя сопереживать истории персонажей.

Как обрабатывать тексты?



Иногда в более структурированном:

Достоинства

Цена, количество насадок

Недостатки

Один режим на всё

Комментарий

Брала его за 707р . Вот уж точно цена = качество. Измельчала овсянку в муку - на 4 из 5 , получилось. Измельчала сырое мясо вместе с сырой морковкой и луком - 5 из 5. Ну а про какие-либо коктейли вообще 10 из 10!) однозначно надо брать. Пользуюсь второй месяц.

Стандартный подход в задаче анализа текста



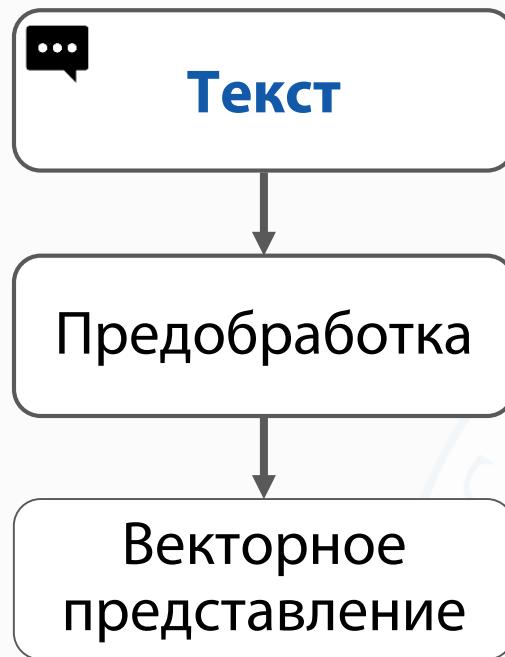
Текст



Стандартный подход в задаче анализа текста



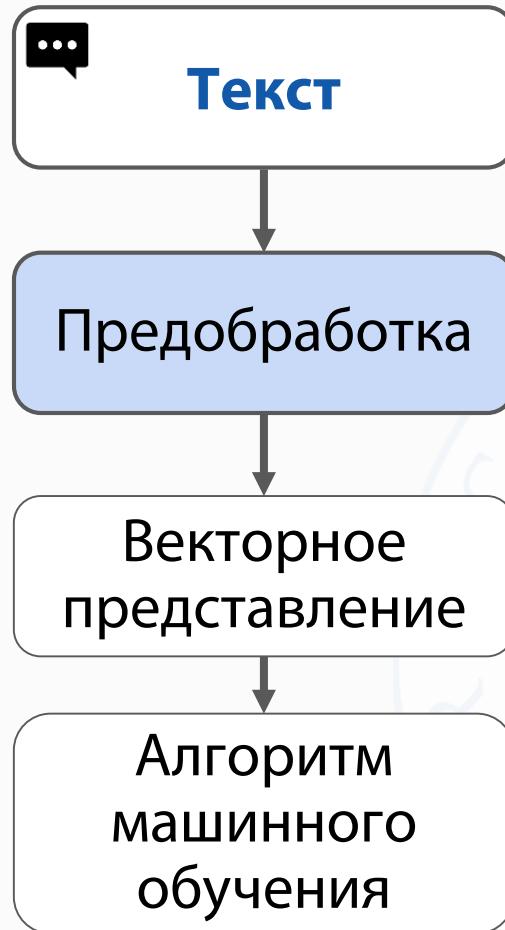
Стандартный подход в задаче анализа текста



Стандартный подход в задаче анализа текста



Предобработка текста



Предобработка текста

Основные шаги предобработки текста:

- Удаление посторонних символов
(пунктуации, HTML-разметки, ссылок и т.д.)

Предобработка текста

Основные шаги предобработки текста:

- Удаление посторонних символов
(пунктуации, HTML-разметки, ссылок и т.д.)
- Приведение к нижнему регистру

Предобработка текста

Основные шаги предобработки текста:

- Удаление посторонних символов
(пунктуации, HTML-разметки, ссылок и т.д.)
- Приведение к нижнему регистру
- Сплиттинг (разбиение на предложения) и токенизация
(разбиение на токены)

Предобработка текста

Основные шаги предобработки текста:

- Удаление посторонних символов
(пунктуации, HTML-разметки, ссылок и т.д.)
- Приведение к нижнему регистру
- Сплиттинг (разбиение на предложения) и токенизация
(разбиение на токены)
- Нормализация: стемминг (выделение основы слова)
или лемматизация

Предобработка текста

Основные шаги предобработки текста:

- Удаление посторонних символов
(пунктуации, HTML-разметки, ссылок и т.д.)
- Приведение к нижнему регистру
- Сплиттинг (разбиение на предложения) и токенизация
(разбиение на токены)
- Нормализация: стемминг (выделение основы слова)
или лемматизация
- Удаление стоп-слов

Резюме



Поговорили об основных задачах обработки
текстовой информации

Резюме



Поговорили об основных задачах обработки текстовой информации



Посмотрели на стандартный подход к задаче обработки текста

Резюме



Поговорили об основных задачах обработки текстовой информации



Посмотрели на стандартный подход к задаче обработки текста

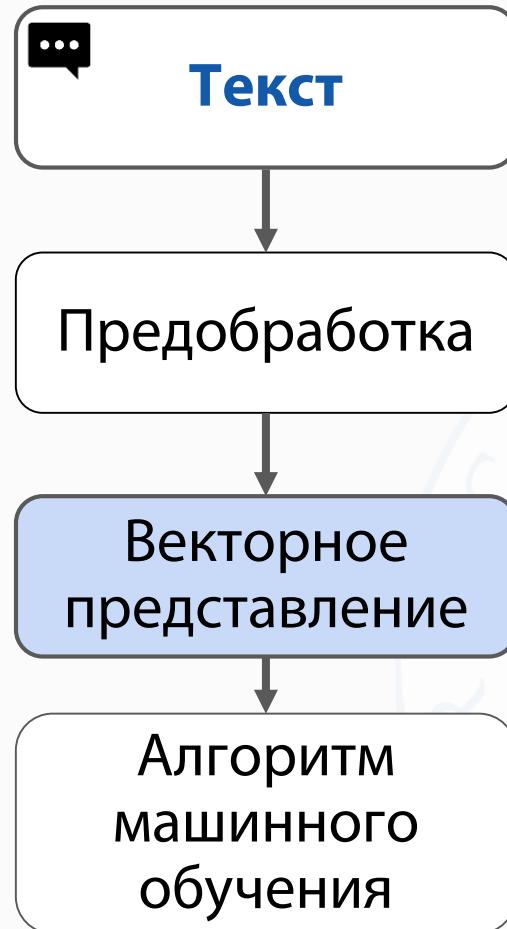


Обсудили первый этап — предобработку текста

Векторные модели представления текста



Векторные модели представления текста



Векторные модели представления текста



Счетные модели: мешок слов, TF-IDF, BPE

Векторные модели представления текста

- Счетные модели: мешок слов, TF-IDF, BPE
- Дистрибутивные модели: word2vec, fastText

Векторные модели представления текста

- Счетные модели: мешок слов, TF-IDF, BPE
- Дистрибутивные модели: word2vec, fastText
- Контекстуализированные модели: ELMo, BERT

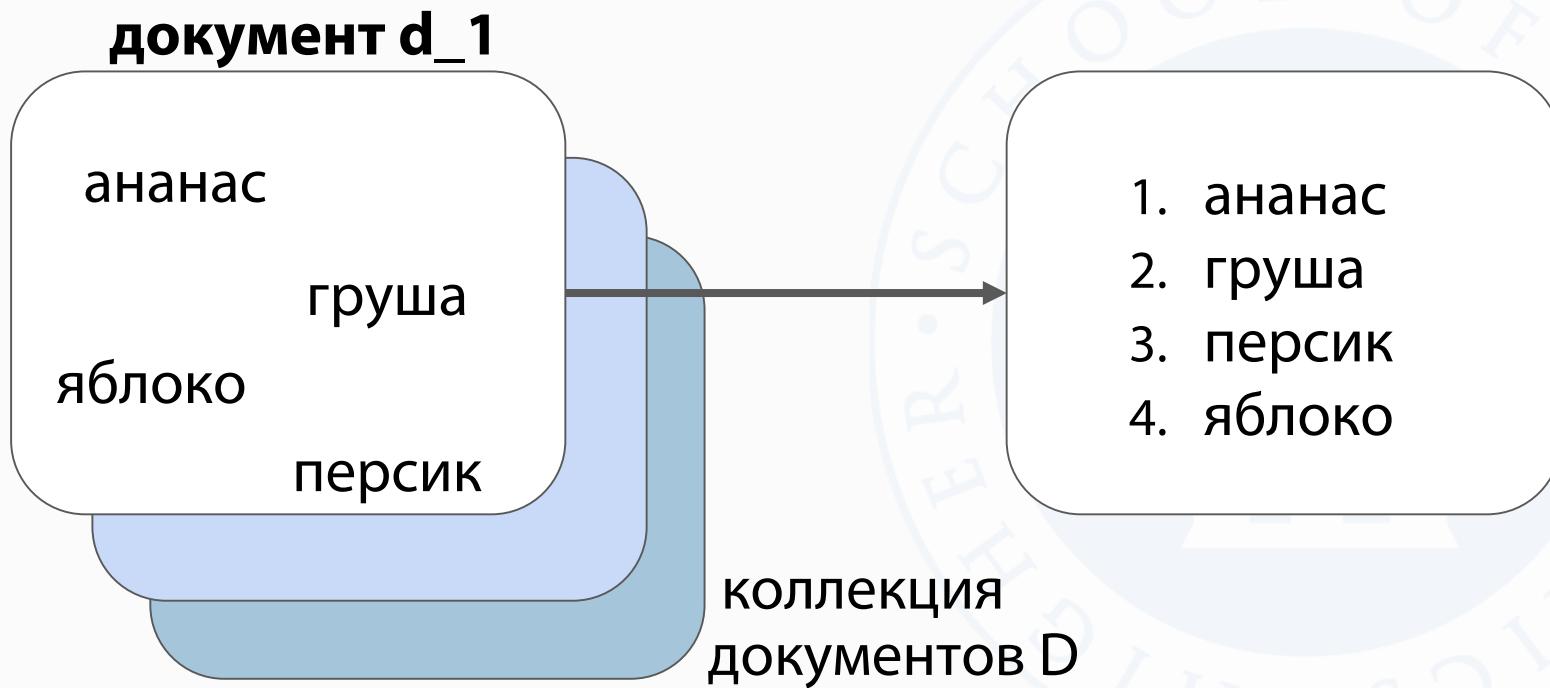
Счетные модели



Счетные модели: мешок слов

Модель «мешка слов» (bag of words, BoW):

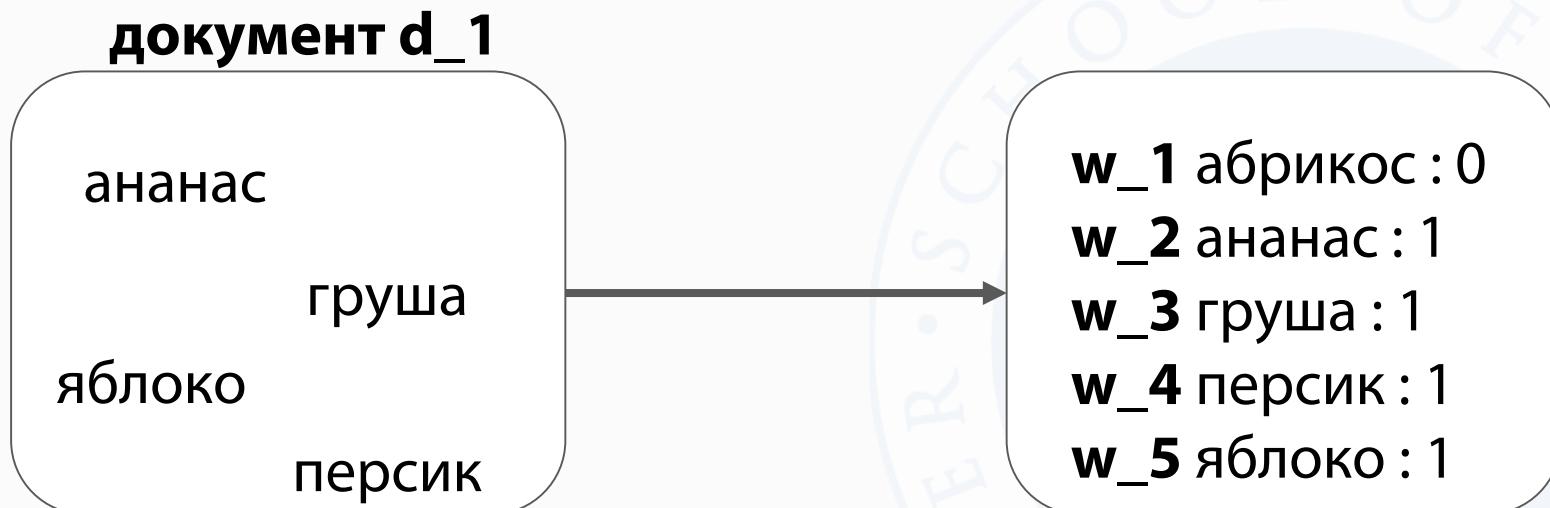
- Ранжируем **все** уникальные слова из коллекции по какому-то признаку (например, по алфавиту)



Счетные модели: мешок слов

Модель «мешка слов» (bag of words, BoW):

- Каждый документ представляем вектором, элементами которого является количество вхождений каждого слова в документ



Счетные модели: мешок слов

Модель «мешка слов» (bag of words, BoW):

Номер документа	Количество вхождений слова w_1	Количество вхождений слова w_2	...	Количество вхождений слова w_n
d_1	0	2	...	6
d_2	3	0	...	0
...
d_m	4	0	...	0

Счетные модели: мешок слов

Модель «мешка слов» (bag of words, BoW):

Номер документа	Количество вхождений слова w_1	Количество вхождений слова w_2	...	Количество вхождений слова w_n
d_1	0	2	...	6
d_2	3	0	...	0
...
d_m	4	0	...	0

вектор документа d_m

вектор слова w_1: [1, 0, 0, ..., 0]

вектор слова w_n: [0, 0, 0, ..., 1]

Счетные модели: мешок слов



Преимущества:

- Просто и быстро вычисляется



Недостатки:



Счетные модели: мешок слов



Преимущества:

- Просто и быстро вычисляется



Недостатки:

- Не учитывает порядок слов



Счетные модели: мешок слов



Преимущества:

- Просто и быстро вычисляется



Недостатки:

- Не учитывает порядок слов
- Размерность:
длина вектора равна
числу уникальных слов
в коллекции

Счетные модели: мешок слов



Преимущества:

- Просто и быстро вычисляется



Недостатки:

- Не учитывает порядок слов
- Размерность: длина вектора равна числу уникальных слов в коллекции
- Все векторы слов взаимно ортогональны

Счетные модели: TF-IDF

Интуитивное объяснение:

- Если слово **часто встречается в документе**, скорее всего, оно важное для этого документа => **tf (term frequency)**

Счетные модели: TF-IDF

Интуитивное объяснение:

- Если слово **часто встречается в документе**, скорее всего, оно важное для этого документа => **tf (term frequency)**

$$tf(w, d) = \frac{n_w}{|d|}$$

w – текущее слово

d – текущий документ

n_w – количество раз, которое слово w встречается
в документе d

$|d|$ – количество слов в документе d

Счетные модели: TF-IDF



Если слово **часто встречается во всех документах коллекции**, скорее всего, оно не поможет отличить один документ от другого => **idf (inverse document frequency)**

$$idf(w, D) = \log\left(\frac{|D|}{|d_i \in D : w_i \in d_i|}\right)$$

Счетные модели: TF-IDF

→ Если слово часто встречается во всех документах коллекции, скорее всего, оно не поможет отличить один документ от другого => **idf** (inverse document frequency)

$$idf(w, D) = \log\left(\frac{|D|}{|d_i \in D : w_i \in d_i|}\right)$$

w – текущее слово

D – коллекция документов,

$|D|$ – число документов в коллекции

$|d_i \in D : w_i \in d_i|$ – количество документов, в которых встречается слово

Счетные модели: TF-IDF

→ Итого:

$$tfidf(w, d, D) = tf(w, d) \cdot idf(w, D)$$

→ Чем выше вес tf-idf, тем «важнее» слово в данном документе

Счетные модели: TF-IDF

→ Итого:

$$tfidf(w, d, D) = tf(w, d) \cdot idf(w, D)$$

- Чем выше вес tf-idf, тем «важнее» слово в данном документе
- Частоупотребимые слова имеют низкий вес во всех документах

Счетные модели: TF-IDF



Представление документа:

Номер документа	w_1	...	w_i	...	w_n
d_1
...
d_j	tf-idf (w_i, d_j, D)
...
d_m

Счетные модели: TF-IDF



Представление документа:

Номер документа	w_1	...	w_i	...	w_n
d_1
...
d_j	tf-idf (w_i, d_j, D)
...
d_m

Вектор документа d_m

Счетные модели: TF-IDF

→ Преимущества:



Счетные модели: TF-IDF

→ Преимущества:

- Относительно быстро вычисляется

Счетные модели: TF-IDF

→ Преимущества:

- Относительно быстро вычисляется
- В отличие от мешка слов, «важные» слова учитываются с большим весом в векторе документа

Счетные модели: TF-IDF



Преимущества:

- Относительно быстро вычисляется
- В отличие от мешка слов, «важные» слова учитываются с большим весом в векторе документа



Недостатки:

- Не учитывает порядок слов

Счетные модели: TF-IDF



Преимущества:

- Относительно быстро вычисляется
- В отличие от мешка слов, «важные» слова учитываются с большим весом в векторе документа



Недостатки:

- Не учитывает порядок слов
- Размерность: длина вектора равна числу уникальных слов в коллекции

Счетные модели: TF-IDF



Преимущества:

- Относительно быстро вычисляется
- В отличие от мешка слов, «важные» слова учитываются с большим весом в векторе документа



Недостатки:

- Не учитывает порядок слов
- Размерность: длина вектора равна числу уникальных слов в коллекции
- Вычислительно тяжелое добавление нового слова или документа в коллекцию

Счетные модели: N-граммы



Единицей текста не всегда является одно слово.

N-граммы — это n идущих подряд в тексте слов, например, биграммы ($n = 2$), триграммы ($n = 3$) и т.д.

«На Капитолийском холме состоялась инаугурация президента США»

Счетные модели: N-граммы

→ Единицей текста не всегда является одно слово.
N-граммы — это n идущих подряд в тексте слов,
например, биграммы ($n = 2$), триграммы ($n = 3$) и т.д.

$n = 2$

«На Капитолийском холме состоялась инаугурация президента США»

Счетные модели: N-граммы

- Использование n-грамм позволяет учитывать, например, устойчивые словосочетания в качестве единицы текста

Счетные модели: N-граммы

- Использование n-грамм позволяет учитывать, например, устойчивые словосочетания в качестве единицы текста
- Как это использовать?
Например, считать мешок слов не на словах, а на n-граммах

Счетные модели: N-граммы

- Использование n-грамм позволяет учитывать, например, устойчивые словосочетания в качестве единицы текста
- Как это использовать?
Например, считать мешок слов не на словах, а на n-граммах
- Преимущества:

Счетные модели: N-граммы

→ Использование n-грамм позволяет учитывать, например, устойчивые словосочетания в качестве единицы текста

→ Как это использовать?

Например, считать мешок слов не на словах, а на n-граммах

→ Преимущества:

- Можно учитывать связи между соседними словами

Счетные модели: N-граммы

→ Использование n-грамм позволяет учитывать, например, устойчивые словосочетания в качестве единицы текста

→ Как это использовать?

Например, считать мешок слов не на словах, а на n-граммах

→ Преимущества:

- Можно учитывать связи между соседними словами

→ Недостатки:

Счетные модели: N-граммы

→ Использование n-грамм позволяет учитывать, например, устойчивые словосочетания в качестве единицы текста

→ Как это использовать?

Например, считать мешок слов не на словах, а на n-граммах

→ Преимущества:

- Можно учитывать связи между соседними словами

→ Недостатки:

- Вырастает размерность пространства

Счетные модели: N-граммы

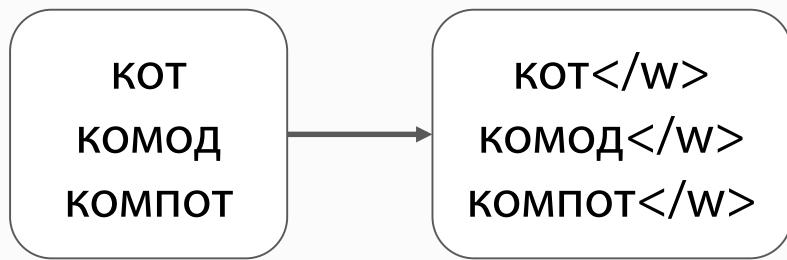
→ N-граммы могут быть не только для слов, но и для символов (символьные n-граммы)

- бутыль => **бут**, **уты**, **ыль**
- бутылка => **бут**, **уты**, **ылк**, **лка**
- собутыльник => **соб**, **обу**, **бут**, **уты**, **ыль**, **льн**, **ьни**, **ник**

Счетные модели: BPE

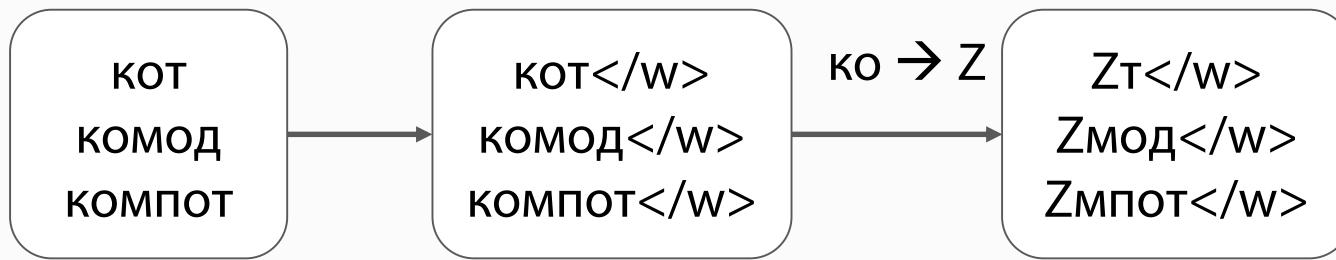
- BPE (byte pair encoding) — это алгоритм кодирования (сжатия) данных без потерь, который хорошо работает и для сжатия текстовых данных

Счетные модели: ВРЕ



→ К каждому слову добавляем символ окончания слова </w>

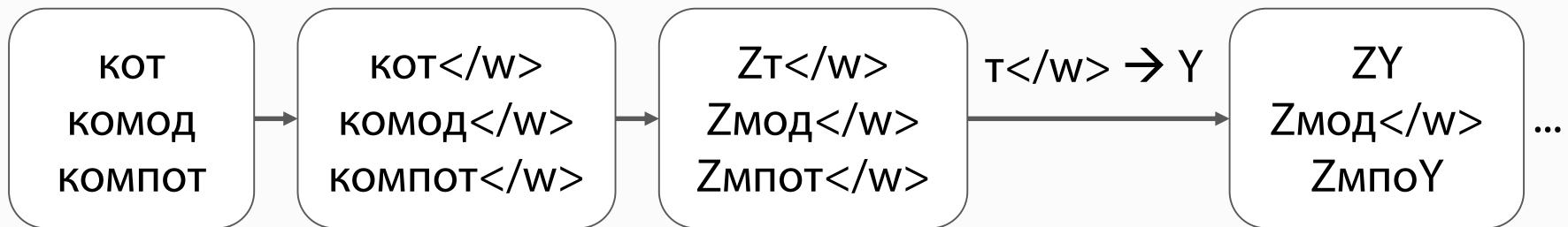
Счетные модели: ВРЕ



ко : 3
т</w> : 2
от : 2
...
мп : 1

- К каждому слову добавляем символ окончания слова </w>
- Считаем, сколько раз каждая символьная биграмма (пара подряд идущих символов) встретилась в данных, заменяем самую частую биграмму на новый символ (например, Z)

Счетные модели: ВРЕ



- К каждому слову добавляем символ окончания слова </w>
- Считаем, сколько раз каждая символьная биграмма (пара подряд идущих символов) встретилась в данных, заменяем самую частую биграмму на новый символ (например, Z)
- Повторяем шаг 2 до тех пор, пока не достигнем желаемого размера словаря или не исчерпаем число итераций

Счетные модели: ВРЕ

→ Преимущества:



Счетные модели: ВРЕ

→ Преимущества:

- Можно регулировать размерность итогового словаря

Счетные модели: ВРЕ

→ Преимущества:

- Можно регулировать размерность итогового словаря
- Использование символьных n-грамм позволяет справляться даже с редкими или OOV (out of vocabulary) словами

Счетные модели: ВРЕ

→ Преимущества:

- Можно регулировать размерность итогового словаря
- Использование символьных n-грамм позволяет справляться даже с редкими или OOV (out of vocabulary) словами

→ Недостатки:



Счетные модели: ВРЕ

→ Преимущества:

- Можно регулировать размерность итогового словаря
- Использование символьных n-грамм позволяет справляться даже с редкими или OOV (out of vocabulary) словами

→ Недостатки:

- Модель «сжимает» размерность словаря, но дальше надо применять алгоритм векторизации

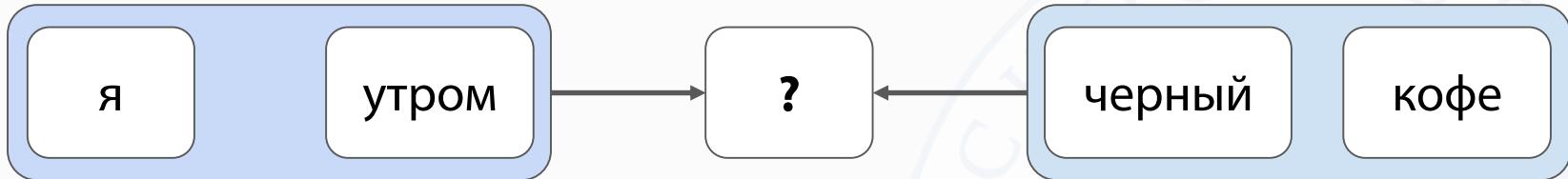
Дистрибутивные модели



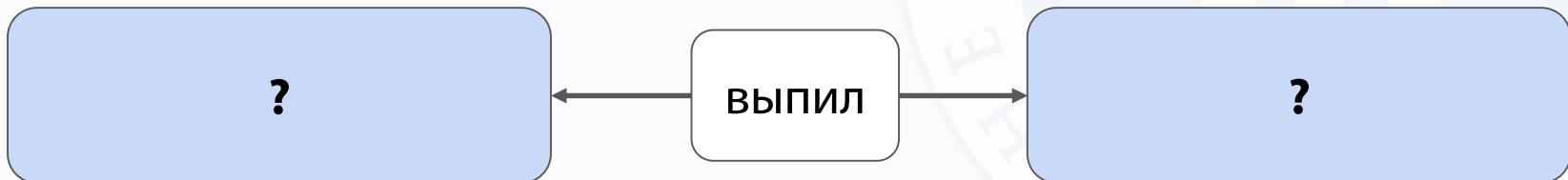
Дистрибутивные модели: word2vec

→ Гипотеза дистрибутивной семантики:
слова со схожим смыслом встречаются в схожих контекстах

→ Поэтому мы будем обучать модель предсказывать слово по контексту (CBOW)

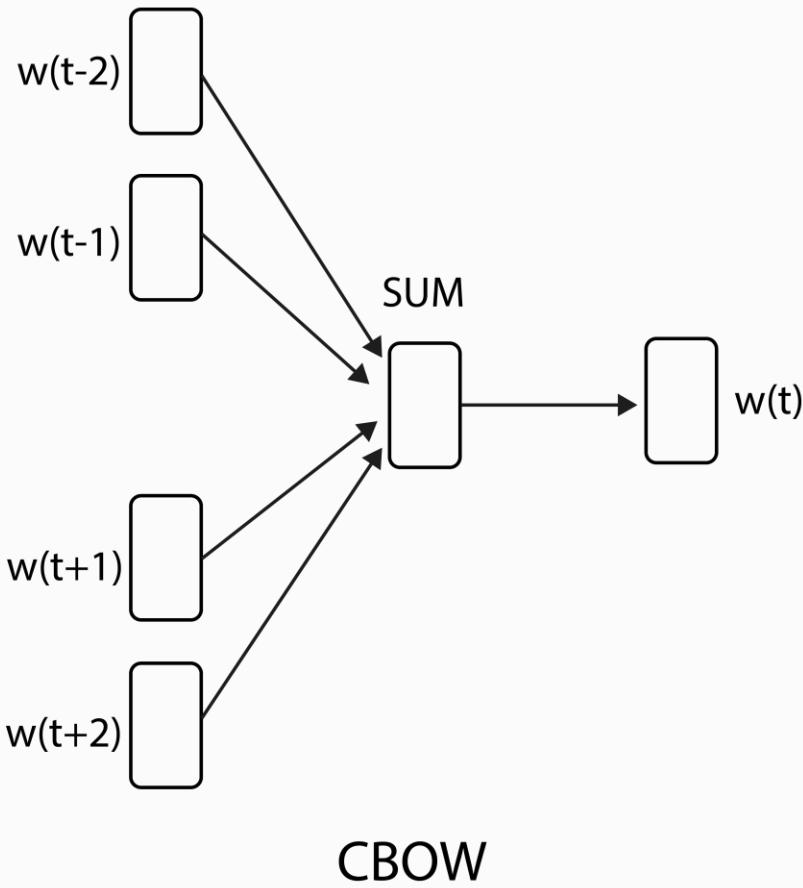


→ или контекст по слову (skip-gram)

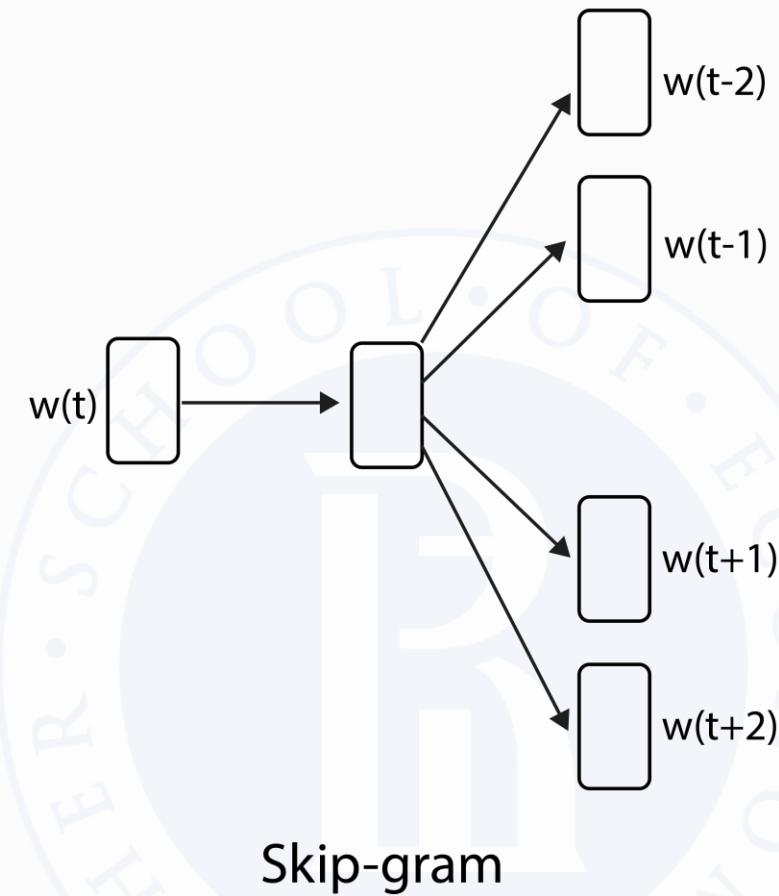


Дистрибутивные модели: word2vec

Входной вектор Проекция Выходной вектор



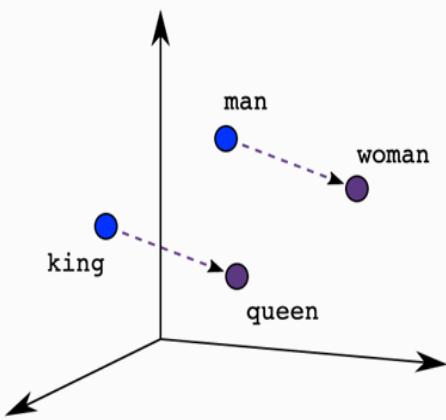
Входной вектор Проекция Выходной вектор



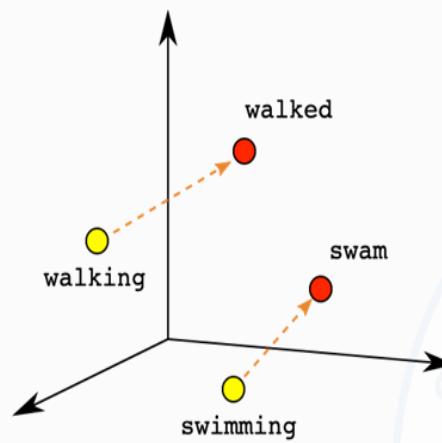
Дистрибутивные модели: word2vec



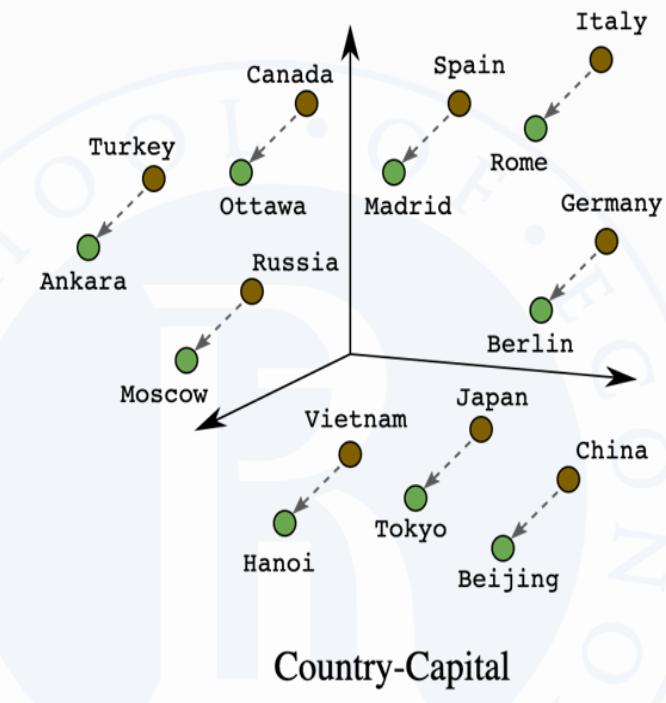
Полученные представления позволяют работать со словами, как с полноценными векторами:



Male-Female



Verb Tense



Country-Capital

Дистрибутивные модели: word2vec

→ Преимущества:



Дистрибутивные модели: word2vec

→ Преимущества:

- Получаем векторы, которые отражают смысл слов

Дистрибутивные модели: word2vec

→ Преимущества:

- Получаем векторы, которые отражают смысл слов
- Размерность итоговых векторов — параметр модели, но, в любом случае, модели менее разреженные, чем счетные (BoW, TF-IDF)

Дистрибутивные модели: word2vec

→ Преимущества:

- Получаем векторы, которые отражают смысл слов
- Размерность итоговых векторов — параметр модели, но, в любом случае, модели менее разреженные, чем счетные (BoW, TF-IDF)

→ Недостатки:



Дистрибутивные модели: word2vec

→ Преимущества:

- Получаем векторы, которые отражают смысл слов
- Размерность итоговых векторов — параметр модели, но, в любом случае, модели менее разреженные, чем счетные (BoW, TF-IDF)

→ Недостатки:

- Получаем векторы слов, не документов

Дистрибутивные модели: word2vec

→ Преимущества:

- Получаем векторы, которые отражают смысл слов
- Размерность итоговых векторов — параметр модели, но, в любом случае, модели менее разреженные, чем счетные (BoW, TF-IDF)

→ Недостатки:

- Получаем векторы слов, не документов
- Обучается дольше, чем счетные модели

Дистрибутивные модели: word2vec



Преимущества:

- Получаем векторы, которые отражают смысл слов
- Размерность итоговых векторов — параметр модели, но, в любом случае, модели менее разреженные, чем счетные (BoW, TF-IDF)



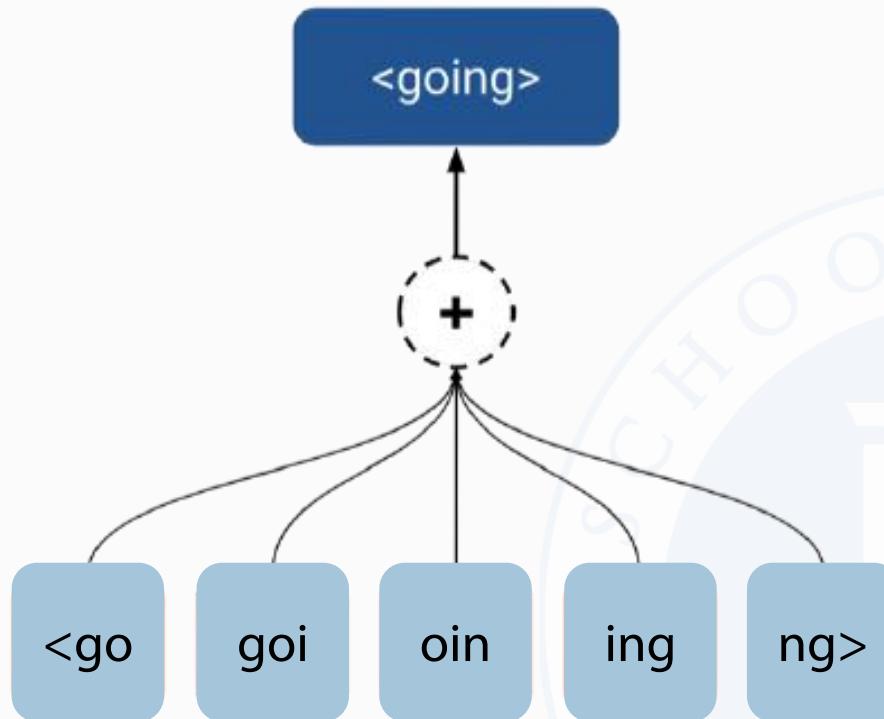
Недостатки:

- Получаем векторы слов, не документов
- Обучается дольше, чем счетные модели
- Есть проблема OOV слов

Дистрибутивные модели: fastText



Идея: используем тот же подход, что в word2vec,
но векторы учим не для слов, а для символьных n-грамм



Символьные триграммы

Дистрибутивные модели: fastText



Hashing trick:

Дистрибутивные модели: fastText



Hashing trick:

- фиксируем максимальное число векторов, которые хотим получить

Дистрибутивные модели: fastText

→ Hashing trick:

- фиксируем максимальное число векторов, которые хотим получить
- этим векторам сопоставляем хеш-таблицу

Дистрибутивные модели: fastText



Hashing trick:

- фиксируем максимальное число векторов, которые хотим получить
- этим векторам сопоставляем хеш-таблицу
- n-граммы — элементы этой таблицы

Дистрибутивные модели: fastText



Hashing trick:

- фиксируем максимальное число векторов, которые хотим получить
- этим векторам сопоставляем хеш-таблицу
- n-граммы — элементы этой таблицы



Как результат, количество параметров модели не растет с ростом параметра n в n-граммах!

Дистрибутивные модели: fastText

→ Преимущества:



Дистрибутивные модели: fastText

→ Преимущества:

- Получаем векторы, которые отражают смысл слов

Дистрибутивные модели: fastText

→ Преимущества:

- Получаем векторы, которые отражают смысл слов
- Нет проблемы OOV слов

Дистрибутивные модели: fastText

→ Преимущества:

- Получаем векторы, которые отражают смысл слов
- Нет проблемы OOV слов
- Обучается быстрее, чем word2vec, за счет hashing trick

Дистрибутивные модели: fastText

→ Преимущества:

- Получаем векторы, которые отражают смысл слов
- Нет проблемы OOV слов
- Обучается быстрее, чем word2vec, за счет hashing trick

→ Недостатки:



Дистрибутивные модели: fastText

→ Преимущества:

- Получаем векторы, которые отражают смысл слов
- Нет проблемы OOV слов
- Обучается быстрее, чем word2vec, за счет hashing trick

→ Недостатки:

- Получаем векторы слов, не документов

Контекстуализированные модели



Контекстуализированные модели: идея



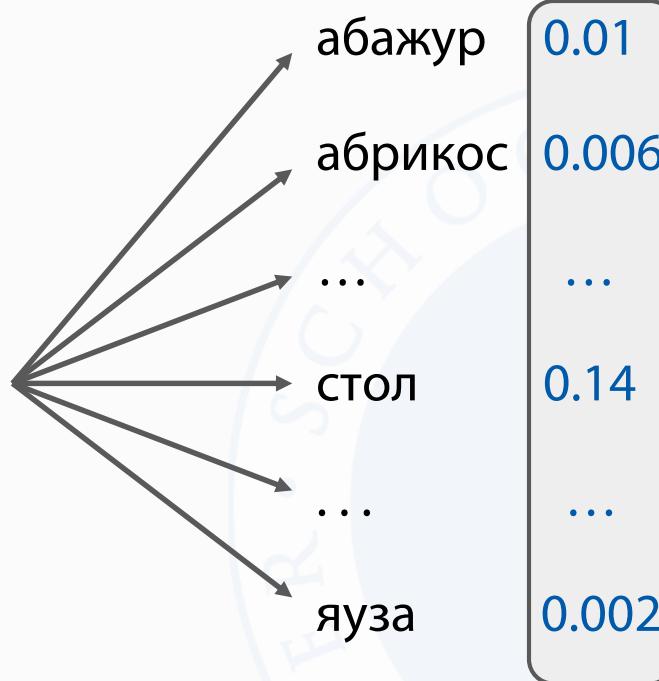
Смысл слов часто зависит от контекста

- Поток реки огибает долину
- На потоке трое студентов из Китая
- Производство поставили на поток

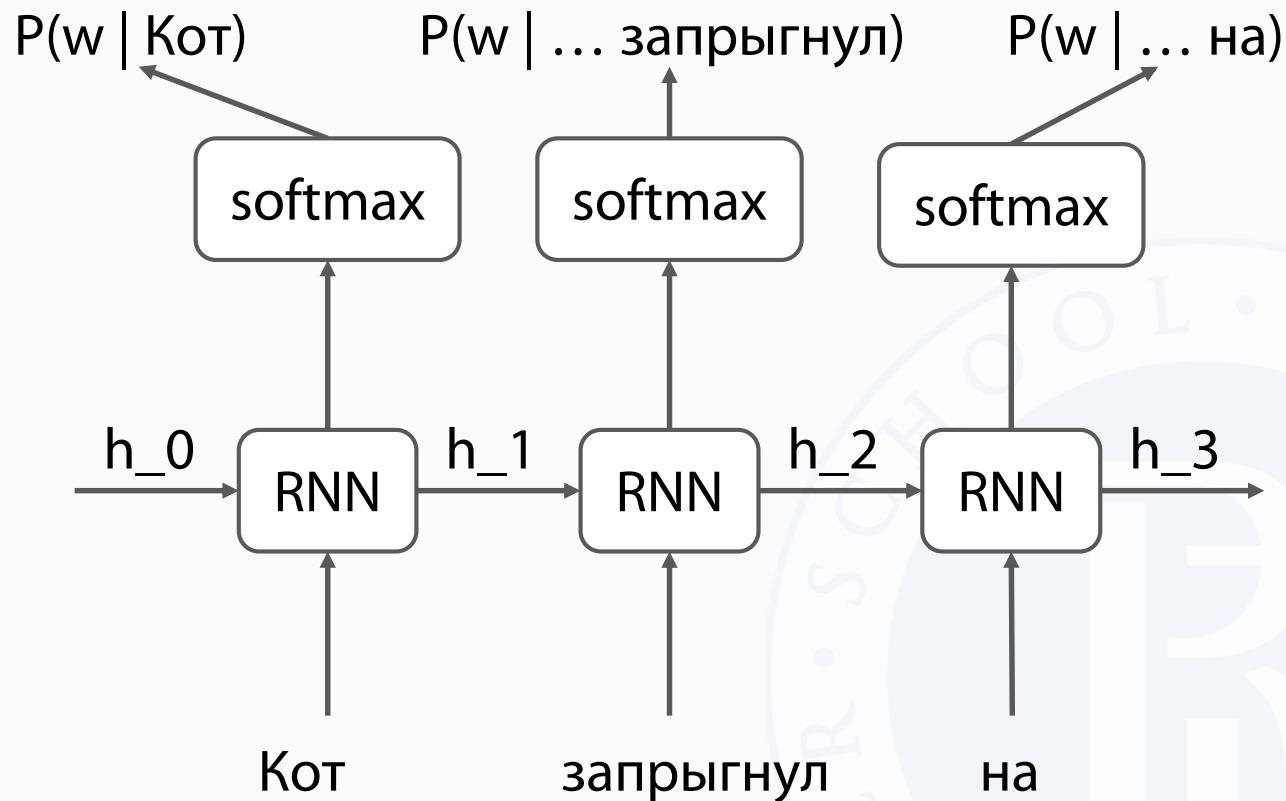
Контекстуализированные модели: идея

→ Будем обучать **языковую модель**: задача — предсказание следующего слова по левому контексту

Кот запрыгнул на ...



Контекстуализированные модели: RNN



Контекстуализированные модели: ELMo

→ Модель — двухслойная
дву направленная LSTM сеть



Контекстуализированные модели: ELMo

- Модель — двухслойная
дву направленная LSTM сеть
- На вход модели подаются слова



Контекстуализированные модели: ELMo

- Модель — двухслойная двунаправленная LSTM сеть
- На вход модели подаются слова
- Выходы скрытых слоев конкатенируются и складываются с весами для получения итогового вектора

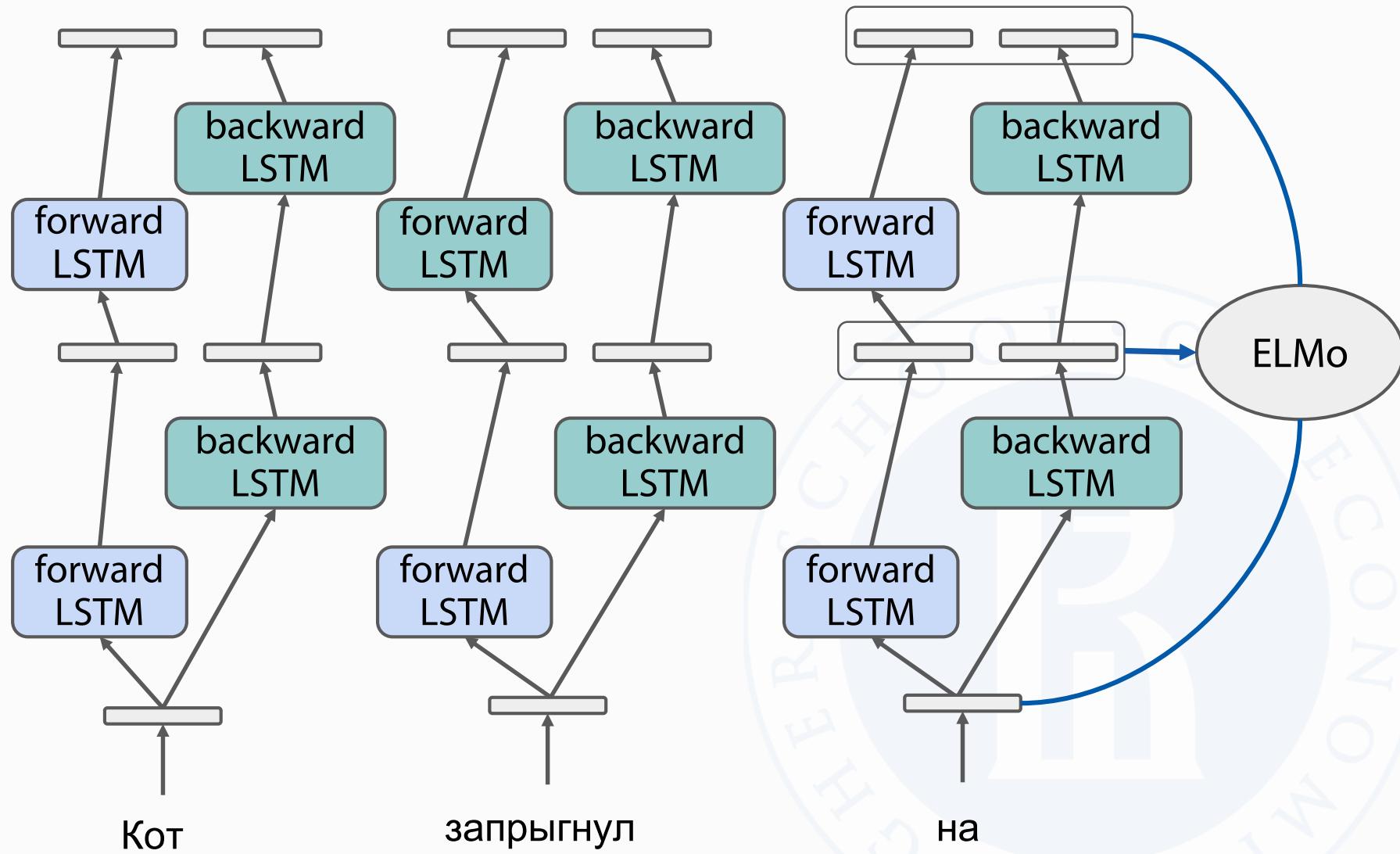


Контекстуализированные модели: ELMo

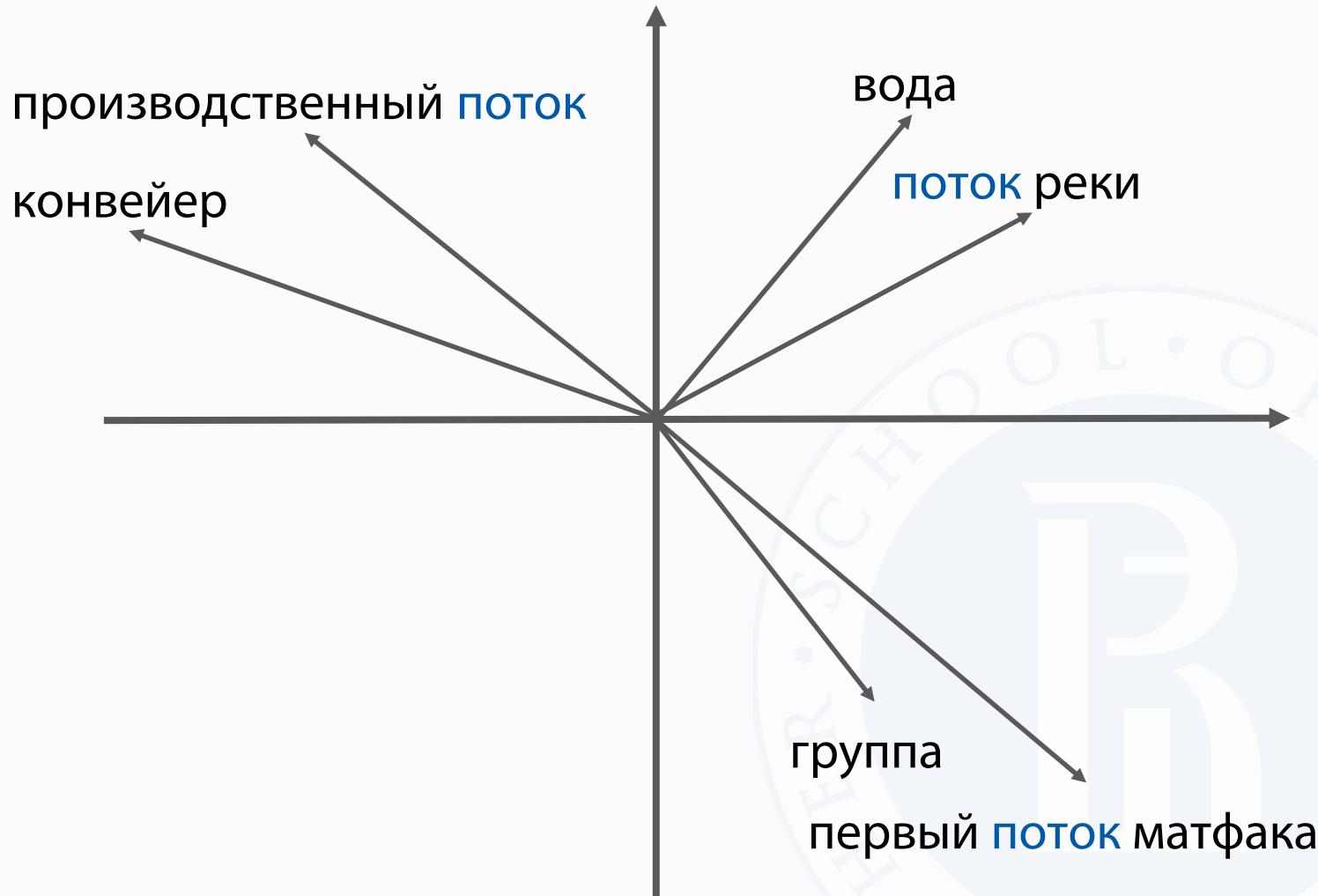
- Модель — двухслойная двунаправленная LSTM сеть
- На вход модели подаются слова
- Выходы скрытых слоев конкатенируются и складываются с весами для получения итогового вектора
- Итоговый вектор — контекстно-зависимый вектор слова



Контекстуализированные модели: ELMo



Контекстуализированные модели: ELMo



Контекстуализированные модели: ELMo

→ Преимущества:



Контекстуализированные модели: ELMo

→ Преимущества:

- Вектор слова зависит от контекста

Контекстуализированные модели: ELMo



Преимущества:

- Вектор слова зависит от контекста



Недостатки:

Контекстуализированные модели: ELMo



Преимущества:

- Вектор слова зависит от контекста



Недостатки:

- LSTM-архитектура долго обучается, обучение нельзя распараллелить



Контекстуализированные модели: BERT

→ Модель основана
на энкодер-декодер архитектуре
(трансформер-архитектуре)



Контекстуализированные модели: BERT

- Модель основана на энкодер-декодер архитектуре (трансформер-архитектуре)
- Качественный прорыв в языковых моделях за счет механизма внимания (attention)



Контекстуализированные модели: BERT

- Модель основана на энкодер-декодер архитектуре (трансформер-архитектуре)
- Качественный прорыв в языковых моделях за счет механизма внимания (attention)
- Обучается на двух задачах: предсказание маскированного токена и предсказание следующего предложения

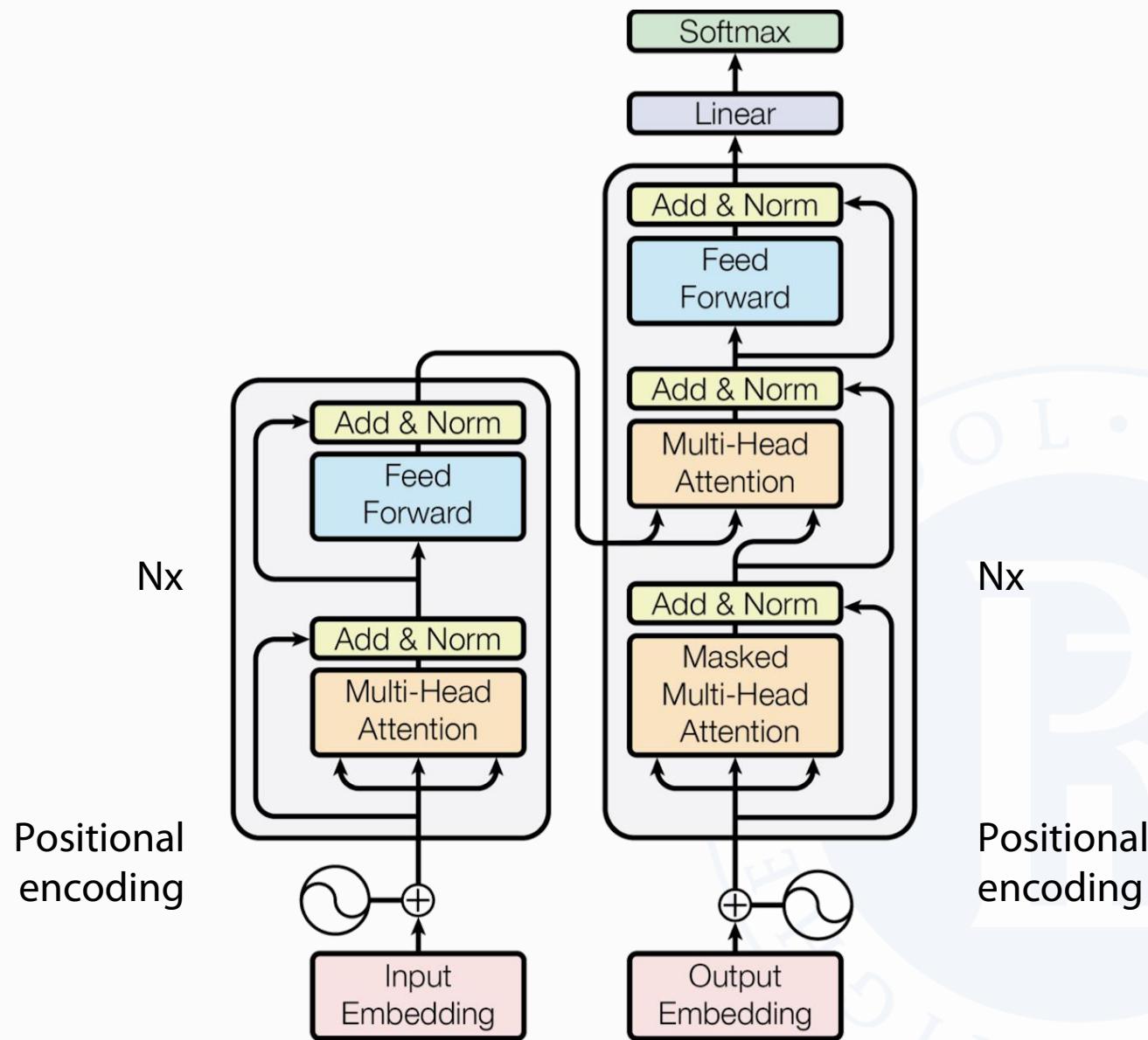


Контекстуализированные модели: BERT

- Модель основана на энкодер-декодер архитектуре (трансформер-архитектуре)
- Качественный прорыв в языковых моделях за счет механизма внимания (attention)
- Обучается на двух задачах: предсказание маскированного токена и предсказание следующего предложения
- Есть две конфигурации: base и large



Контекстуализированные модели: attention



Контекстуализированные модели: BERT

- С практической точки зрения:
 - Берем предобученную модель BERT (или аналогичную)

Контекстуализированные модели: BERT

- С практической точки зрения:
 - Берем предобученную модель BERT (или аналогичную)
 - Если данные обладают спецификой, дообучаем BERT на этих данных (не нужна разметка!)

Контекстуализированные модели: BERT

- С практической точки зрения:
 - Берем предобученную модель BERT (или аналогичную)
 - Если данные обладают спецификой, дообучаем BERT на этих данных (не нужна разметка!)
 - Добавляем и дообучаем последний слой (или несколько слоев) для нашей задачи, например, классификации или разметки последовательности

Контекстуализированные модели: BERT

→ Преимущества:



Контекстуализированные модели: BERT

→ Преимущества:

- Контекстуализированные вектора слов

Контекстуализированные модели: BERT

→ Преимущества:

- Контекстуализированные вектора слов
- Архитектура позволяет распараллелить процесс обучения

Контекстуализированные модели: BERT

→ Преимущества:

- Контекстуализированные вектора слов
- Архитектура позволяет распараллелить процесс обучения
- Есть предобученные модели, которые можно относительно быстро дообучить для конкретной задачи

Контекстуализированные модели: BERT



Преимущества:

- Контекстуализированные вектора слов
- Архитектура позволяет распараллелить процесс обучения
- Есть предобученные модели, которые можно относительно быстро дообучить для конкретной задачи



Недостатки:



Контекстуализированные модели: BERT



Преимущества:

- Контекстуализированные вектора слов
- Архитектура позволяет распараллелить процесс обучения
- Есть предобученные модели, которые можно относительно быстро дообучить для конкретной задачи



Недостатки:

- Несмотря на возможность распараллеливания обучения, и сам процесс обучения, и использование модели требуют больших ресурсов

Векторное представление документов

→ На практике нам редко нужны векторы отдельных слов, чаще — **векторы предложений или документов**

→ Как получить из векторов слов вектор предложения?
Например, **усреднить**

Коты → $W_1 = [w_{11}, w_{12}, \dots, w_{1n}]$

любят → $W_2 = [w_{21}, w_{22}, \dots, w_{2n}] \rightarrow (W_1 + W_2 + W_3) / 3$

сметану → $W_3 = [w_{31}, w_{32}, \dots, w_{3n}]$

Заключение



Для работы с текстом почти всегда требуется стадия предобработки, шаги которой зависят от задачи и от самих данных



Алгоритмы машинного обучения, как правило, работают с векторным представлением текста



По-прежнему широко используются стандартные алгоритмы машинного обучения поверх этих векторных представлений



State of the art подход сейчас — предобученные языковые модели, но их использование требует больших ресурсов