

Математические методы анализа текстов

Введение в обработку текстов.

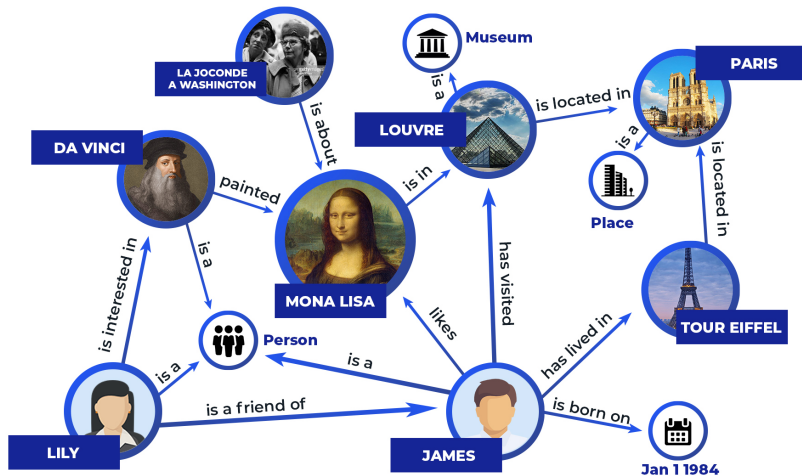
Предобработка, выделение признаков и классификация

Мурат Апишев (mel-lain@yandex.ru)

Сентябрь, 2020

Текстовые данные

- ▶ Большая часть данных в мире представлена в текстовом виде
- ▶ Текстовые данные могут быть
 - ▶ структурированными (графы знаний, базы данных)



Текстовые данные

- ▶ Большая часть данных в мире представлена в текстовом виде
- ▶ Текстовые данные могут быть
 - ▶ структурированными (графы знаний, базы данных)
 - ▶ неструктурированными (сырые тексты)

Введение в обработку естественного языка.

Под авторством Сидорова Ивана Петровича.

Вступление.

Настоящее пособие предназначено для ...

Чек

Магазин канцелярских товаров

1. Шариковая ручка (син) 23 руб.

2. Тетрадь клет (12 л) 5 руб.

...

Текстовые данные

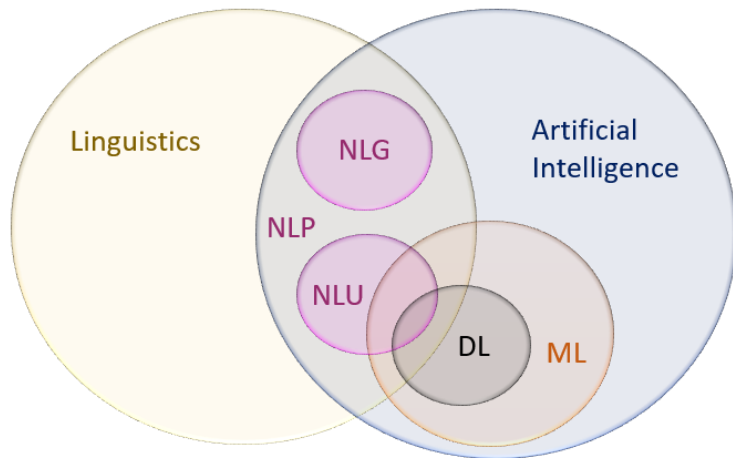
- ▶ Большая часть данных в мире представлена в текстовом виде
- ▶ Текстовые данные могут быть
 - ▶ структурированными (графы знаний, базы данных)
 - ▶ неструктурированными (сырые тексты)
 - ▶ частично структурированными (JSON, XML)

```
1      {  
2          "type": "учебник",  
3          "title": "Введение в обработку естественного языка.",  
4          "author": "Сидоров Иван Петрович",  
5          "introduction": "Настоящее пособие предназначено для \dots  
6      ",  
7          \dots  
8      }
```

Обработка текстовых данных

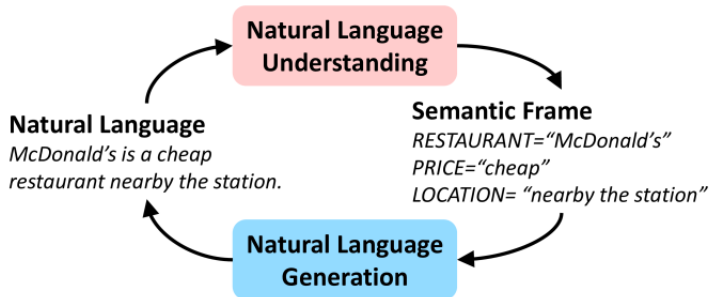
Методы **обработки естественного языка** (natural language processing, NLP) успешно используются во всех описанных

Положение NLP среди наук по анализу и обработке данных:



Структура NLP

- ▶ Внутри NLP условно выделяются два направления:
понимание языка (NLU) и генерация языка (NLG)
- ▶ текст → NLU → смысл → NLG → текст



- ▶ Граница между NLU и NLG нечёткая
 - ▶ **Пример:** задачи категоризации и анализа тональности
- ▶ Смежные области — распознавание (ASR) и генерация (TTS) речи

Пирамида NLP



Особенности обработки естественного языка

- ▶ Базовая структурная единица языка — слово
 - ▶ Даже вне контекста оно несёт полезную информацию
- ▶ В NLP обычно большие разреженные признаковые пространства
- ▶ Текст без дополнительной разметки имеет внутреннюю структуру, определяемую языком на разных уровнях:
 - ▶ текст (порядок реплик)
 - ▶ предложение (синтаксис)
 - ▶ слова и словосочетания (морфология, синтаксис)
- ▶ Наличие огромных массивов сырых текстов и структуры в них позволяет обучать **большие общезыковые модели**
 - ▶ Такие модели легко дообучать для решения конкретной задачи даже на небольшом объёме данных
- ▶ Существует много лингвистических ресурсов, которые помогают в различных задачах обработки текстов

Задачи обработки естественного языка

Можно формулировать задачи с технической и продуктовой точек зрения

Технические задачи:

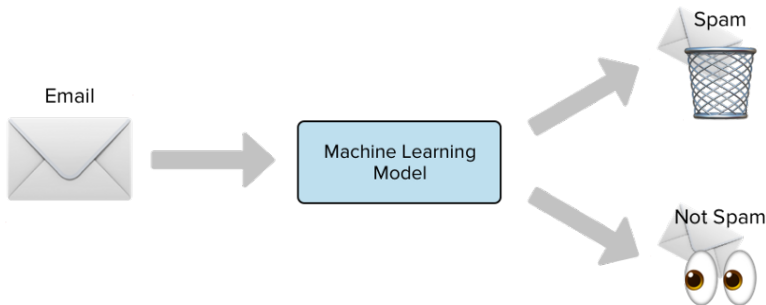
- ▶ Классификация
- ▶ POS-теггинг
- ▶ Извлечение сущностей
- ▶ Разрешение кореференции
- ▶ Лемматизация
- ▶ Сегментация
- ▶ Разметка семантических ролей
- ▶ Векторизация
- ▶ Машинный перевод
- ▶ Суммаризация
- ▶ ...

Продуктовые задачи:

- ▶ Анализ тональности
- ▶ Машинный перевод
- ▶ Генерация подписей к изображениям
- ▶ Генерация сниппетов для новостей
- ▶ Ведение диалога
- ▶ Исправление опечаток
- ▶ Поисковое и рекомендательное ранжирование
- ▶ Анализ трендов
- ▶ ...

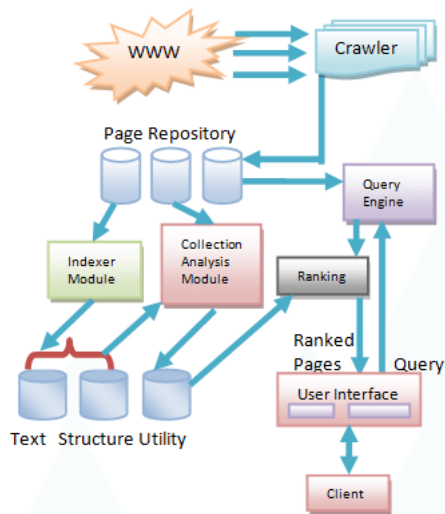
Задача классификации текстов

- ▶ Как всегда, это базовая и в общем случае наиболее простая задача
- ▶ Она лежит в основе многих продуктовых задач, **например**:
 - ▶ фильтрация почтового спама
 - ▶ анализ тональности отзывов
 - ▶ категоризация новостей и статей
 - ▶ выявление некачественных текстов и фейков



Задача ранжирования

- ▶ Ранжирование решает задачу сортировки объектов по заданному критерию полезности
 - ▶ **информационный поиск** — релевантность страницы сайта пользовательскому запросу
 - ▶ **рекомендации** — близость текстовой статьи к текущим интересам пользователя
- ▶ В таких системах ключевым сигналом является поведение пользователя
- ▶ Тексты используются для генерации дополнительных признаков
- ▶ Знание об их семантической близости помогает в ситуации «холодного старта» и при обработке редких длинных запросов



Задача машинного перевода

- ▶ Для текста на одном языке нужно получить перевод на другой язык
- ▶ Недостаточно просто дословно перевести фразу
- ▶ Нужно учитывать обороты речи, устойчивые выражения, семантику фразы, кореференцию

it is raining cats and dogs

✗ идет дождь из кошек и собак

✓ льет как из ведра

Задача анализа корректности текста

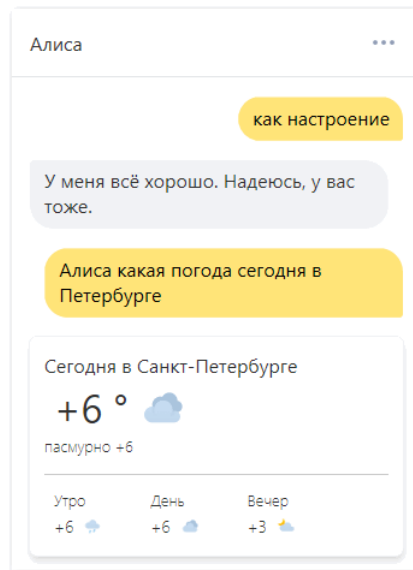
- ▶ Тексты, которые пишут люди, надо проверять на наличие
 - ▶ опечаток в словах
 - ▶ ошибки согласования и употребления слов
 - ▶ синтаксических ошибок, ошибок управления в предложении
- ▶ Оценить стиль текста и качество изложения, дать рекомендации по его улучшению

✗ Заваривая кофе, у Сергея упал кофейник

✓ Заваривая кофе, Сергея уронил кофейник

Задача ведения диалога

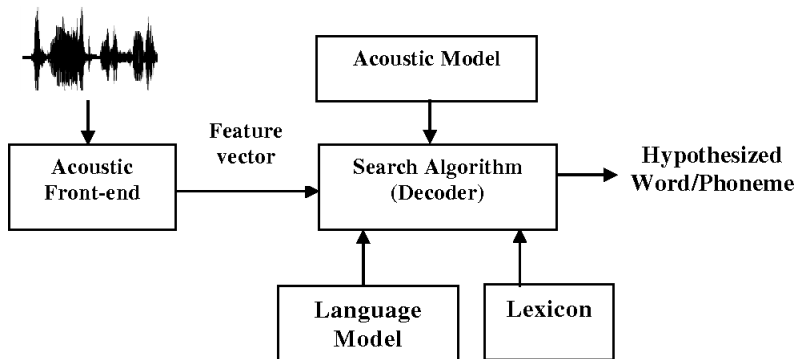
- ▶ Диалоговые системы общаются с человеком на естественном языке
- ▶ Производится анализ входных реплик
- ▶ На основе анализа генерируются адекватные ответа или действия
- ▶ Система может быть строго детерминированной
- ▶ А может работать в режиме свободного диалога



Задача распознавания речи


- ▶ Часто диалоговые системы используют голосовой ввод
- ▶ Парсинг голоса требует знания о языке для выбора наиболее вероятного результата транскрибации

Speech Utterance



Задача поиска ответов на вопросы

- ▶ **Вопросно-ответные системы** (QA-система) используются для поиска ответов на вопросы, заданные на естественном языке
- ▶ Вопросы могут быть разнообразными: фактологическими, сравнительными, задаваемыми по тексту
- ▶ QA-системы часто используются в качестве элементов поисковых систем
- ▶ **Пример: «что такое луна?»**



Луна
Естественный спутник

Луна — единственный естественный спутник Земли. Самый близкий к Солнцу спутник планеты, так как у ближайших к Солнцу планет их нет. Второй по яркости объект на земном небосводе после Солнца и пятый по величине естественный спутник планеты Солнечной системы. Среднее расстояние между центрами Земли и Луны — 384 467 км. [Википедия](#)

Расстояние до Земли: 384 400 км

Ускорение свободного падения: 1,62 м/с²

Радиус: 1 737,1 км

Возраст: 4,53Е9 лет

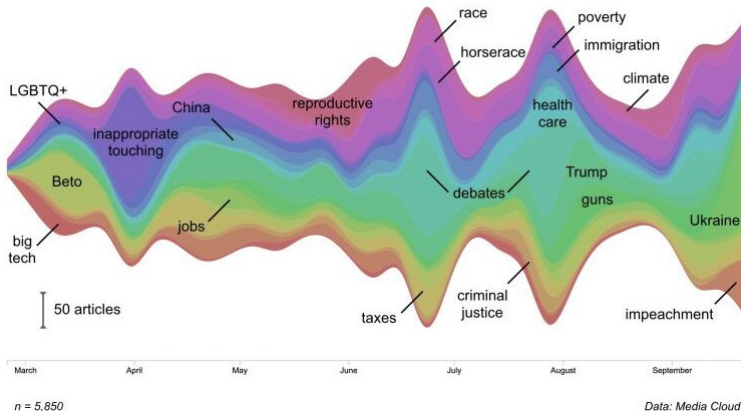
Период обращения: 27 дней

Обращается вокруг: [Земля](#)

Задача выделения и отслеживания трендов

- ▶ Для многих прикладных задачах полезно уметь извлекать из данных соцсетей и медиа обсуждаемые темы и упоминания сущностей/брендов
- ▶ Затем изменения тем отслеживаются в динамике

A topic analysis of news articles published by 28 outlets since March 2019 mentioning Joe Biden, Bernie Sanders, Elizabeth Warren, Kamala Harris, Pete Buttigieg, Beto O'Rourke, Cory Booker, Kirsten Gillibrand, Amy Klobuchar, or Tulsi Gabbard



Задача суммаризации

- ▶ Для текстового документа нужно сгенерировать краткое изложение
- ▶ Важна не только передача смысла, но и сохранение важных фактов

В Японии зафиксировали падение неизвестного небесного тела

Падение неизвестного небесного тела было зафиксировано в ряде мест в Токио и других префектурах района Канто, сообщает издание "Иомури". Очевидцы заметили светящийся объект в пятницу около 22.30 по местному времени. [В источнике](#)

Пилот F-16 проведет воздушный бой против искусственного интеллекта

Американские военные запланировали на 20 августа знаковый воздушный бой пилота ВВС США, который имеет солидный опыт полетов на F-16, с необычным соперником. Им станет искусственный интеллект. Правда, как пишет Defence Talk, пока речь идет о виртуальном бое. [В источнике](#)

Этапы решения NLP-задачи

Всё так же, как и при обработке других типов данных:

- ▶ Выбор верной метрики качества
- ▶ Сбор обучающих и тестовых данных
- ▶ Предобработка данных
- ▶ Формирование признакового описания текст
- ▶ Выбор подхода и класса моделей
- ▶ Обучение моделей и настройка решения

Предположим, что данные есть в некотором подходящем для работы формате

Инструменты для работы с текстами

- ▶ В обработке текстов часто полезны библиотеки общего назначения:
 1. `re/regex` — модули для работы с регулярными выражениями
 2. `numpy/pandas/scipy/sklearn` — базовые библиотеки для анализа данных и ML
 3. `pytorch` — одна из основных библиотек для обучений нейросетей
 4. `codecs` — модуль для работы с кодировками в Python 2.*
- ▶ По ходу курса будем узнаем про много библиотек, созданных конкретно для решения разных задач обработки текстов
- ▶ Прежде всего познакомимся с **Natural Language Toolkit (nltk)**:
 1. токенизация и сегментация предложений, поиск коллокаций
 2. выделение именованных сущностей, разметка на части речи
 3. визуализация синтаксических зависимостей
 4. ...

Регулярные выражения

- ▶ Регулярные выражения появились от т.н. регулярных автоматов (классификация грамматик по Хомскому)
- ▶ По факту это некоторый строковый шаблон, на соответствие которому можно проверить текст
- ▶ Для работы с регулярными выражениями есть множество инструментов и онлайн-сервисов, в Python есть два основных модуля: `re` и `regex`
- ▶ С синтаксисом можно ознакомиться на странице выбранного инструмента, но основные правила одинаковы, например:
 - ▶ `.` — означает наличие одного любого символа
 - ▶ `[a-zA-Z0-9]` — означает множество символов из заданного диапазона
 - ▶ `+`, `*` — показывают, что следующий перед ними символ или последовательность символов должны повториться ≥ 1 раз (`r+`) или > 0 раз (`(ха-)*`)

Пример из жизни

Регулярные выражения, служащие для определения заголовков публикаций, быстро теряющих актуальность:

```
. *онлайн-|-трансляция.*
```

```
. *в эт(у?и?) минут.*
```

```
. *в эт((от)?и?) час.*
```

```
. *в этот день.*
```

```
. *[0-9]{2}[0-9]{2}( . *| : *| : . *| ; . *| )
```

```
. *[0-9]{1,2}[0-9]{2}([0-9]{2}|[0-9]{4})( . *| : *| : . *| ; . *| )
```

```
. *(от|за|на) [0-9]{1,2} (январ|феврал|март|апрел  
|ию(н|л)|август|сентябр|ноябр|октябр|декабр|ма(я|й)) . *
```

```
. *(от|за|на) [0-9]{1,2}[0-9]{2}( . *| : *| : . *| ; . *| )
```

Предобработка текстов

- ▶ Пусть дана коллекция текстовых документов D
- ▶ Текст представляет собой одну строку и алфавитных и неалфавитных символов
- ▶ Обработать его в таком виде неудобно, сперва нужно выделить числовые признаки
- ▶ Для этого данные надо привести к удобному виду и нормализовать
- ▶ Базовые шаги предобработки:
 1. токенизация
 2. приведение к нижнему регистру
 3. удаление стоп-слов
 4. удаление пунктуации
 5. фильтрация слов по частоте/длине/регулярному выражению
 6. лемматизация или стемминг

Токенизация

- ▶ Токенизацию можно производить по словам и/или предложениям
- ▶ Используются как подходы, основанные на правилах, так и модели ML
- ▶ В nltk есть много различных токенизаторов, например RegexpTokenizer и sent_tokenize

```
1 from nltk.tokenize import sent_tokenize
2 text = 'Это предложение. И это тоже предложение и т.д.'
3 print(sent_tokenize(text, language='russian'))
4 #['Это предложение.', 'И это тоже предложение и т.д.']
```

- ▶ Часто слова грубо выделяют разделением по пробелам с помощью метода split:

```
1 text = 'Набор слов, составляющий какое-то предложение.'
2 print(text.split(' '))
3 #['Набор', 'слов,', 'составляющий', 'какое-то', 'предложение.']
```


Удаление пунктуации

- ▶ В ряде задач пунктуация полезна, но часто от неё лучше избавляться
- ▶ Удалять пунктуацию можно напрямую, полезен модуль `string`:

```
1 import string
2 text = 'Набор из 7 слов, составляющий какое-то предложение.'
3 for c in string.punctuation:
4     text = text.replace(c, '')
5 print(text)
6 # Набор из 7 слов составляющий какое-то предложение
```

- ▶ Если нужно точно удалить все символы, кроме алфавитных, лучше использовать регулярные выражения:

```
1 import re
2 text = 'Набор из 7 слов, составляющий какое-то предложение.'
3 print(re.sub('[^а-яА-ЯёЁйЙ ]+', '', text))
4 # Набор из 7 слов составляющий какое-то предложение
```

Регистр и пунктуация

- ▶ Есть задачи, в которых пунктуация и регистр несут важную информацию
- ▶ Это важно для определения границ предложений, для решения задачи выделения именованных сущностей

в комнату вошёл **лев** и, потянувшись, достал из кармана сигару

лев обитает в саванне, в арктике не обитает

- ▶ В задаче анализа тональности существенное значение имеют **смайлы** (текстовые или символы в Unicode)

✗ Одежда у вас в магазине очень своеобразная: /

✓ Одежда у вас в магазине очень своеобразная:)

Нормализация слов

- ▶ Слова в тексте могут иметь различную формы, часто такая информация скорее мешает, чем помогает анализу
- ▶ Для нормализации применяется один из подходов:
 - ▶ лемматизация (pymorphy2, pymystem3)
 - ▶ стемминг (реализации в nltk)
- ▶ Лемматизация приводит слова к нормальной форме
- ▶ Стемминг приводит слова к псевдооснове (убирает окончания и формообразующие суффиксы)

```
1 import pymorphy2
2 text = 'я хотел бы поговорить с вами'.split(' ')
3 lemmatizer = pymorphy2.MorphAnalyzer()
4 print([lemmatizer.parse(t)[0].normal_form for t in text])
5 # ['я', 'хотеть', 'бы', 'поговорить', 'с', 'вы']
```

Фильтрация словаря

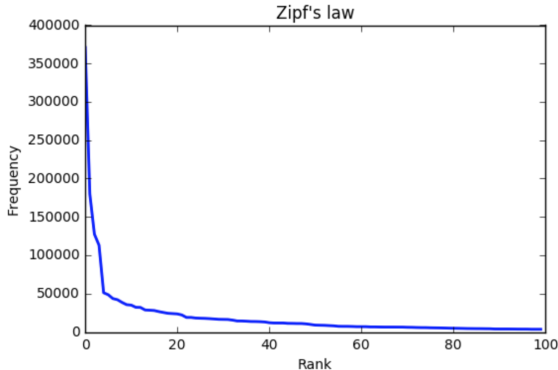
- ▶ Часто из текстов нужно удалять лишние слова
- ▶ Обычно это стоп-слова, очень редкие и очень частые слова
- ▶ К стоп-словам относятся союзы, предлоги, модальные глаголы, местоимения, вводные слова
- ▶ Большой набор стоп-слов есть в nltk:

```
1 from nltk.corpus import stopwords
2 sw = set(stopwords.words('russian'))
3
4 for w in ['я', 'хотеть', 'бы', 'поговорить', 'с', 'вы']:
5     if w not in sw:
6         print(w, end=' ')
7     # хочет поговорить
```

- ▶ При необходимости можно урезать или расширять этот список

Фильтрация словаря

- ▶ Слишком частые или редкие слова тоже могут оказаться вредными
- ▶ Такие слова могут и мешать обучению модели, и увеличивать затраты ресурсов памяти и времени счёта
- ▶ При обработке коллекции стоит проверить выполнение **закона Ципфа**:



- ▶ Имеет смысл обрезать это распределение по порогам с обеих сторон

Признаковые описания документов

- Обычно в ML данные представляют собой матрицу «объекты-признаки»:

Номер автомобиля	Тип топлива	Мощность двигателя	...	Масса
1	Бензин	120	...	1700
...
N	Дизель	160	...	2100

- Для текстов тоже нужно как-то получить такую матрицу
- Базовым элементами текста являются слова
- Можно считать признаком наличие слова в тексте:

Номер текста	Содержит «хорошо»
1	1
...	...
N	0

- Такой простой признак может быть полезен при анализе тональности

Признаковые описания документов

- ▶ Очевидно, что информации о наличии одного слова недостаточно
- ▶ Можно проверять наличие всех возможных слов из некоторого словаря:

Номер текста	Содержит «абрикос»	...	Содержит «яблоко»
1	0	...	1
...
N	1	...	0

- ▶ Пусть документ 1 связан с яблоками, в нём это слово частое
- ▶ Текущее представление эту информацию не отражает
- ▶ Пусть значением признака будет не наличие слова, а число его вхождений в документ («мешок слов»):

Номер текста	Вхождений «абрикос»	...	Вхождений «яблоко»
1	0	...	23
...
N	2	...	0

Представление TF-IDF

- ▶ Представление «мешка слов» часто используется при обработке текстов
- ▶ Но частота встречаемости слова не самый информативный признак
- ▶ **Идея:** хотим выделить слова, которые часто встречаются в данном тексте, и редко — в других текстах
- ▶ Используем **значения TF-IDF**:

$$v_{wd} = tf_{wd} \times \log \frac{N}{df_w}$$

- ▶ tf_{wd} — доля слова w в словах документа d
 - ▶ df_w — число документов, содержащих w
 - ▶ N — общее число документов
- ▶ С помощью TF-IDF можно также выделять **ключевые слова**

«Мешок слов» и TF-IDF в Python

- ▶ Оба представления можно легко построить в sklearn:

```
1 from sklearn.feature_extraction.text import CountVectorizer
2 from sklearn.feature_extraction.text import TfidfVectorizer
3
4 c_vectorizer, t_vectorizer = CountVectorizer(), TfidfVectorizer()
5 corpus = [
6     'This is the first document.',
7     'This is the second second document.',
8     'And the third one.',
9     'Is this the first document?',
10 ]
11 X_c = c_vectorizer.fit_transform(corpus)
12 X_t = t_vectorizer.fit_transform(corpus)
```

- ▶ Формула для TF-IDF в sklearn отличается от классической наличием двух нормализаций

Какие ещё могут быть признаки

- ▶ Мы рассмотрели два вида векторов признаков для текстов, основанные на счётчиках встречаемости слов
- ▶ Эти векторы имеют большую размерность и сильно разрежены
- ▶ Можно строить **сжатые векторные представления**, которые используют информацию о совместной встречаемости слов
- ▶ Можно строить любые иные векторы признаков (с учётом, например, синтаксиса или произвольной внешней информации)
- ▶ Векторы «мешка слов» и TF-IDF мы формировали для слов (**униграмм**)
- ▶ Эти и другие признаки можно строить для сочетаний слов (**коллокаций** или **N-грамм**)

Коллокации

- ▶ **N-граммы** — устойчивые последовательности из N слов, идущих подряд («**машина опорных векторов**»)
- ▶ **Коллокация** — устойчивое сочетание слов, не обязательно идущих подряд («Он **сломал** своему противнику **руку**»)
- ▶ Часто коллокациями бывают именованные сущности (но не всегда)
- ▶ Методы получения N-грамм:
 - ▶ на основе частот встречаемости (sklearn, nltk)
 - ▶ на основе морфологических шаблонов (**Томита**, **YARGY-парсер**)
 - ▶ с помощью ассоциации и статистических критериев на основе частот совместных встречаемостей (nltk, **TopMine**)
 - ▶ иные подходы (**RAKE**, **TextRank**)

<https://yandex.ru/dev/tomita>

<https://github.com/natasha/yargy>

El-Kishky, Ahmed, et al. Scalable topical phrase mining from text corpora, 2014

Rose, Engel, et al. Automatic keyword extraction from individual documents, 2010

Beliga, Mestrovic, et al. An Overview of Graph-Based Keyword Extraction Methods and Approaches, 2015

Меры ассоциации биграмм

- ▶ Поточечная совместная информация (Pointwise Mutual Information, PMI):

$$\text{PMI}(w_1, w_2) = \log \frac{f(w_1, w_2)}{f(w_1)f(w_2)}$$

- ▶ T-Score (по сути — тест Стьюдента):

$$T_{\text{score}}(w_1, w_2) = \frac{f(w_1, w_2) - f(w_1)f(w_2)}{\sqrt{f(w_1, w_2)/N}}$$

- ▶ w_i — слово
- ▶ $f(\cdot)$ — частота слова или биграммы
- ▶ В обоих случаях проверяется гипотеза независимости появления пары токенов (слов или N-грамм)
- ▶ Чем выше значение критерия, тем скорее пара токенов является устойчивым сочетанием

Пример использования

t	$C(w^1)$	$C(w^2)$	$C(w^1 w^2)$	w^1	w^2
4.4721	42	20	20	Ayatollah	Ruhollah
4.4721	41	27	20	Bette	Midler
4.4720	30	117	20	Agatha	Christie
4.4720	77	59	20	videocassette	recorder
4.4720	24	320	20	unsalted	butter
2.3714	14907	9017	20	first	made
2.2446	13484	10570	20	over	many
1.3685	14734	13478	20	into	them
1.2176	14093	14776	20	like	people
0.8036	15019	15629	20	time	last

- ▶ Описанные меры реализованы в `nlk.collocations.bigram_measures`:
 - ▶ `pmi`, `student_t`
- ▶ Есть и другие критерии, например
 - ▶ `chi_sq`, `likelihood_ratio`

Поиск разрывных коллокаций

- ▶ Часто устойчивые словосочетания находятся не рядом
- ▶ Примеры:
 - ▶ She *knocked* on his *door*
 - ▶ They *knocked* on his heavy *door*
 - ▶ A man *knocked* on the metal front *door*
- ▶ Что делаем:
 - ▶ Рассмотрим все пары слов в некотором окне
 - ▶ Посчитаем расстояние между словами
- ▶ Что измеряем:
 - ▶ Матожидание — показывает, насколько часто слова встречаются вместе
 - ▶ Дисперсия — вариабельность позиции
- ▶ Важно провести лемматизацию

Пример поиска разрывных коллокаций

s	\bar{d}	Count	Word 1	Word 2
0.43	0.97	11657	New	York
0.48	1.83	24	previous	games
0.15	2.98	46	minus	points
0.49	3.87	131	hundreds	dollars
4.03	0.44	36	editorial	Atlanta
4.03	0.00	78	ring	New
3.96	0.19	119	point	hundredth
3.96	0.29	106	subscribers	by
1.07	1.45	80	strong	support
1.13	2.57	7	powerful	organizations
1.01	2.00	112	Richard	Nixon
1.05	0.00	10	Garrison	said

$$\bar{d} = \frac{\sum_{i=1}^n d_i}{n}$$

$$s^2 = \frac{\sum_{i=1}^n (d_i - \bar{d})^2}{n - 1}$$

- ▶ n — число раз, когда два слова встретились.
- ▶ d_i — смещение между словами (может быть < 0).
- ▶ \bar{d} — выборочное среднее смещений.

Чем больше дисперсия, тем менее интересно сочетание слов

Постановка задачи классификации

- ▶ Предобрабатывать тексты и выделять признаки научились, теперь попробуем построить модель
- ▶ Классификация — одна из наиболее часто решаемых задач в NLP
- ▶ Многие задачи (разметка последовательности, анализ тональности) могут быть сведены к классификации
- ▶ Данные:
 - ▶ $d \in D$ — множество документов (объектов)
 - ▶ $c \in C$ — множество меток классов
- ▶ Типы задач:
 - ▶ Бинарная классификация: $|C| = 2, \quad \forall d \in D \leftrightarrow c \in C$
 - ▶ Многоклассовая классификация [multiclass]:
 $|C| = K, K > 2, \quad \forall d \in D \leftrightarrow c \in C$
 - ▶ Многотемная классификация [multi-label]:
 $|C| = K, K > 2, \quad \forall d \in D \leftrightarrow \tilde{C} \subseteq C$

Метрики качества классификации

- ▶ Доля правильных ответов (**accuracy**): $acc = \#(correct) / \#(total)$
- ▶ Чаще всего используются точность (**precision**) и полнота (**recall**):

$$Pr = \frac{tp}{tp + fp}, \quad R = \frac{tp}{tp + fn}$$

$$F_1 = \frac{2}{\frac{1}{Pr} + \frac{1}{R}} = \frac{2 \cdot Pr \cdot R}{Pr + R}$$

		Верный ответ	
		+1	-1
Ответ модели	+1	tp	fp
	-1	fn	tn

- ▶ В многоклассовом случае используется **микро-усреднение** (нивелируется влияние маленьких классов):

$$Pr_{micro} = \frac{\sum tp_i}{\sum tp_i + \sum fp_i}, \quad R_{micro} = \frac{\sum tp_i}{\sum tp_i + \sum fn_i}$$

- ▶ Или **макро-усреднение** (все классы учитываются одинаково):

$$Pr_{macro} = \sum Pr_i / |C|, \quad R_{macro} = \sum R_i / |C|$$

Модель для классификации

- ▶ Для классификации текстов используются многие модели ML:
 - ▶ Наивный байесовский классификатор
 - ▶ Линейные модели
 - ▶ Полносвязные нейросети
 - ▶ Свёрточные нейросети
 - ▶ Рекуррентные нейросети
- ▶ У всех подходов есть свои преимущества и недостатки, которые обсудим в дальнейшем
- ▶ Сейчас рассмотрим **лог-регрессию** — простое и эффективное решение для большинства задач классификации текстов

Логистическая регрессия

- ▶ Лог-регрессия — линейная модель с логистической функцией потерь

$$\log(1 + \exp(-y\langle x, w \rangle))$$

- ▶ $x \in \mathbb{R}^n$ — вектор признаков объекта
- ▶ $w \in \mathbb{R}^n$ — вектор весов линейной модели
- ▶ $y \in \{+1, -1\}$ — метка класса объекта
- ▶ При обучении может использоваться регуляризация:
 - ▶ L_1 (Lasso)
 - ▶ L_2 (Ridge)
 - ▶ $L_1 + L_2$ (ElasticNet)
- ▶ Выход модели определяется с помощью сигмоиды по формуле

$$p(c = +1|x; w) = \frac{1}{1 + \exp(-\langle x, w \rangle)}$$

Многоклассовая классификация

- ▶ Лог-регрессия обобщается на многоклассовый случай заменой сигмоиды на функцию **softmax**:

$$p(c = c_i | x; W) = \frac{\exp(\langle x, w_i \rangle)}{\sum_{j=1}^m \exp(\langle x, w_j \rangle)}$$

- ▶ c_i — метка класса
 - ▶ m — число классов
 - ▶ $W \in \mathbb{R}^{m \times n}$ — набор весов m линейных моделей
 - ▶ w_i — i -я строка матрицы W
- ▶ Любой бинарный классификатор можно использовать для многоклассового случая с помощью одной из стратегий:
 - ▶ **One-vs-One**: обучаем по классификатору на каждую пару классов
 - ▶ **One-vs-Rest**: обучаем по классификатору на каждый класс

Пример обучения лог-регрессии на «мешке слов»

```
1  from sklearn.metrics import f1_score
2  from sklearn.datasets import fetch_20newsgroups
3  from sklearn.model_selection import train_test_split
4  from sklearn.linear_model import LogisticRegression
5
6  X, y = data.data, data.target  # data = fetch_20newsgroups()
7  X = [text for text, label in zip(X, y) if label <= 1]
8  y = [label for label in y if label <= 1]
9  X_tr, X_test, y_tr, y_test = train_test_split(X, y, test_size=0.25)
10
11 vect = CountVectorizer()
12 X_tr = vect.fit_transform(X_tr)
13 X_test = vect.transform(X_test)
14
15 model = LogisticRegression()  # L2 reg by default
16 model.fit(X_tr, y_tr)
17 f1_score(y_test, model.predict(X_test))  # near 0.99
```

На что стоит обращать внимание

- ▶ Порог-бинаризации:
 - ▶ Лог-регрессия возвращает вероятность отнесения к классу, по ней решаем, к какому классу окончательно отнести объект
 - ▶ Для этого выставляется порог бинаризации, метод `predict` по-умолчанию использует значение 0.5
 - ▶ Порог лучше подбирать на валидации (используя метод `predict_proba`)
- ▶ Кросс-валидация: лог-регрессия обучается быстро, можно использовать кросс-валидацию для более честной оценки качества модели
- ▶ TF-IDF: обычно даёт более хороший результат, чем просто «мешок слов»
- ▶ Регуляризация: часто помогает, коэффициент лучше подбирать
- ▶ Сокращение размерности:
 - ▶ Изученные признаковые описания имеют большую размерность
 - ▶ Иногда полезно перейти в пространство меньшей размерности
 - ▶ Можно использовать **усечённое сингулярное разложение** (truncated SVD)

Пример многоклассовой классификации

```
1 from sklearn.decomposition import TruncatedSVD
2 from sklearn.feature_extraction.text import TfidfVectorizer
3
4 X, y = data.data, data.target # data = fetch_20newsgroups()
5 X_tr, X_test, y_tr, y_test = train_test_split(X, y, test_size=0.25)
6
7 vect = TfidfVectorizer() # +2% vs. bag-of-words
8 X_tr = vect.fit_transform(X_tr)
9 X_test = vect.transform(X_test)
10
11 #svd = TruncatedSVD(n_components=200, n_iter=5)
12 #X_tr = svd.fit_transform(X_tr)
13 #X_test = svd.transform(X_test)
14
15 model = LogisticRegression()
16 model.fit(X_tr, y_tr)
17 f1_score(y_test, model.predict(X_test), average='macro') # near 0.9
```

Итоги занятия

- ▶ NLP — очень востребованная и активно развивающаяся область на стыке машинного обучения, анализа данных и лингвистики
- ▶ Существуют разнообразные постановки задач обработки текстов, технические и бизнесовые
- ▶ Работа с текстами почти всегда требует тщательного изучения и аккуратной предобработки данных
- ▶ Можно использовать разнообразные признаковые описания, базовыми являются представления «мешка слов» и TF-IDF
- ▶ Самая распространённая задача NLP — классификация текстов, многие практические задачи сводятся к ней
- ▶ Лог-регрессия — полезный, простой и быстрый инструмент для классификации, хороший бейзлайн, который бывает сложно обойти