



# **Введение. Предобработка текстов. Линейные модели классификации.**

**Попов Артём, осень 2022**

**Natural Language Processing**

# Информация о лекторе

## Лаборатория машинного интеллекта МФТИ (2017-2019)

- система планирования медиаконтента, система обработки жалоб клиентов, автоматизация создания выборки для обучения чат-бота

## Медицинский отдел компании Алгомост (2019-2020)

- поиск информации в научных статьях, детектирование опасностей по медкарте пациента, суммаризация научных статей

## AI отдел / Research отдел компании JetBrains (2020-2022)

- автодополнение кода, система text-to-SQL

## Преподаватель курсов

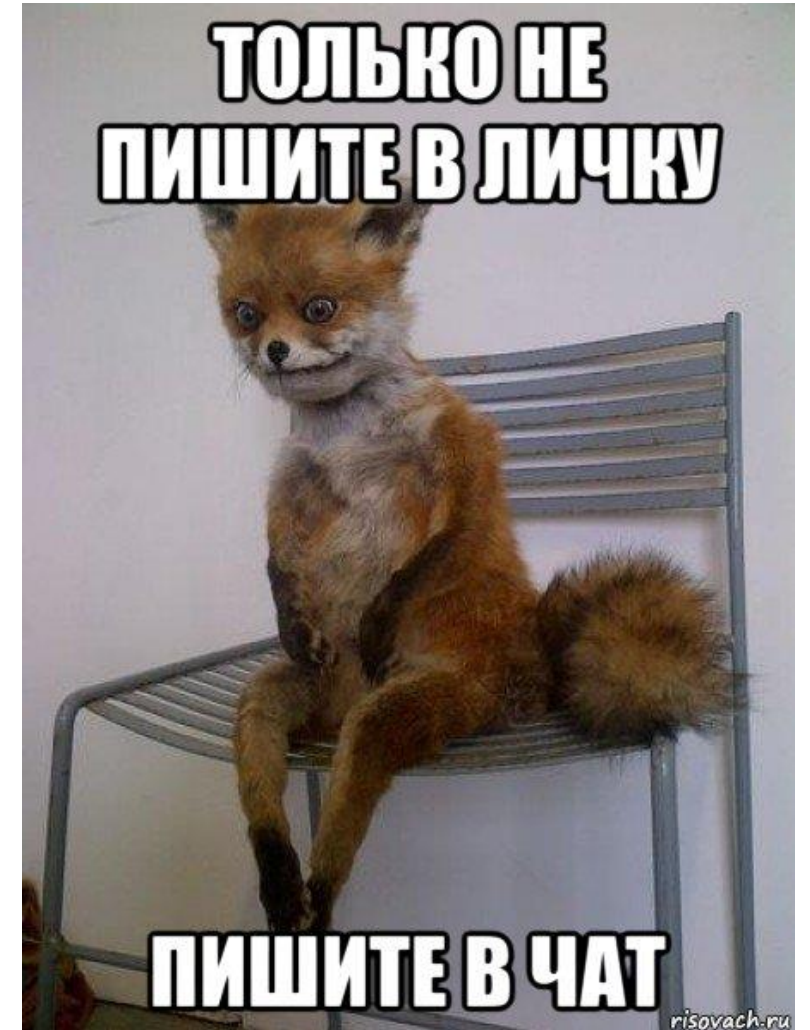
- Практикум по Python (ВМК МГУ, 2016-2020)
- Практикум по машинному обучению, NLP (OzonMasters, 2019-2022)
- Математические методы анализа текстов (ВМК МГУ, ФПМИ МФТИ, 2018-н.в.)

# Информация о курсе

Курс состоит из 13-14 лекций:

- Попов Артём  
([artems-07@mail.ru](mailto:artems-07@mail.ru), @arti\_lehtonen)
- Мурат Апишев
- Кирилл Хрыльченко
- Константин Воронцов

Все вопросы, касающиеся курса,  
пишите в телеграм-чат курса!



# Формат курса

- 13-14 лекций
- 4 практических задания, за каждое можно получить 10 баллов
- Мягкий дедлайн, день просрочки – 1 балл штрафа
- Теоретические опросы на лекциях (до 1 балла за один опрос)
- В конце курса экзамен (по 10-ти балльной шкале)

**Итоговая оценка за курс:**  $\text{round}(0.7 * D / 4 + 0.3 * E) * [E \geq 3]$

round — математическое округление

D — баллы за задания и опросы

E — баллы за экзамен

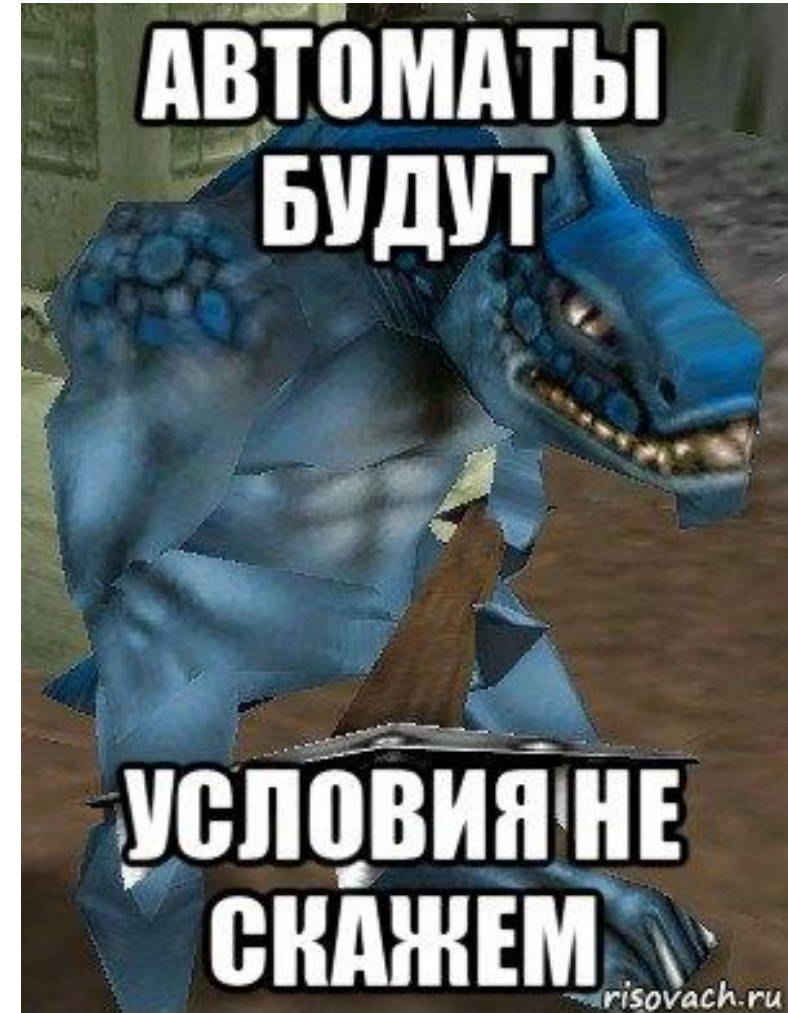


# Будут ли автоматы?

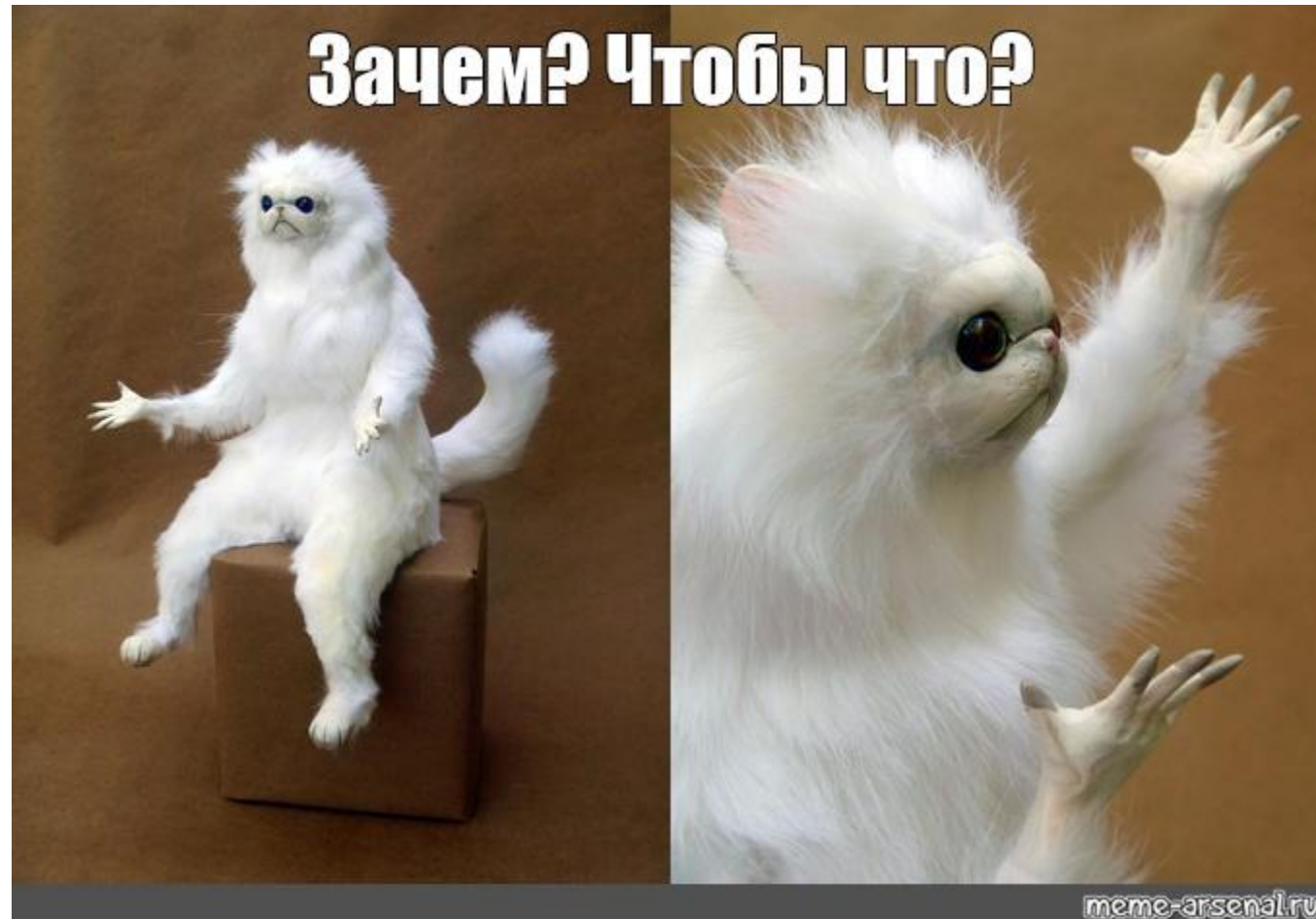
Будут :)

Но условия мы скажем ближе к концу курса.

Прошлые года: автомат за > 40 баллов по домашним заданиям.



**Зачем нужен курс по NLP, если я уже прослушал курсы по машинному и глубинному обучению?**



# О чём будет курс?

- Какие постановки задач встречаются в NLP?
- Как нужно решать стандартные задачи в NLP в 2022, имея определённый набор ресурсов?
- Как новую задачу свести к известным хорошо-решаемым задачам?
- Как комбинировать разные подходы и получать лучшее качество для конкретной задачи?

**Первая половина курса:** must have для любого, кто хочет в NLP

**Вторая половина курса:** нестандартные концепции, жизненный опыт, постановки задач, встречающиеся на практике

# **Natural Language Processing – обработка естественного языка / обработка текстов**

Область на стыке искусственного интеллекта и лингвистики,  
изучающая проблемы анализа и синтеза текстов на  
естественном языке.

Нет никакой связи с НЛП, нейролингвистическим  
программированием...



# Что такое текстовые данные?

## Типы текстовых данных

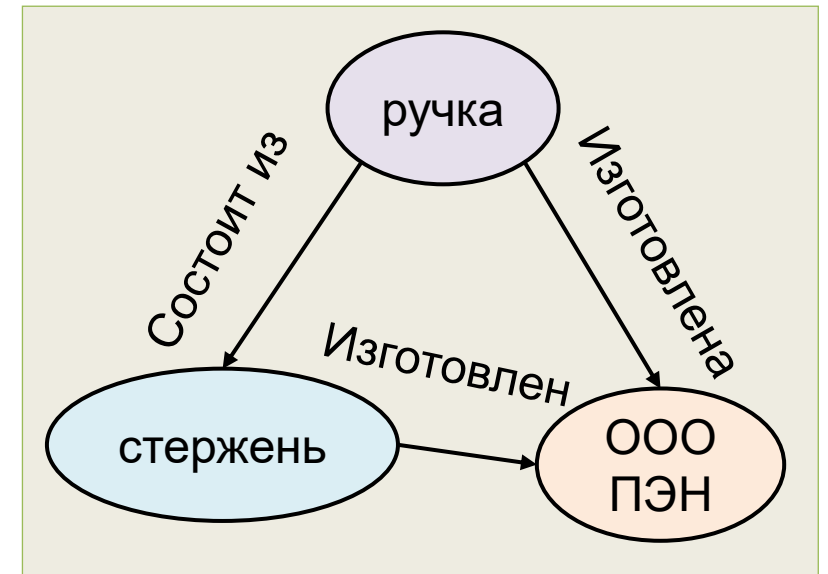
Неструктурированные  
(сырые тексты)

Чек  
Магазин канцелярии  
Шариковая ручка (синяя) – 15  
рубля  
Тетрадь (12 листов) – 10  
рублей

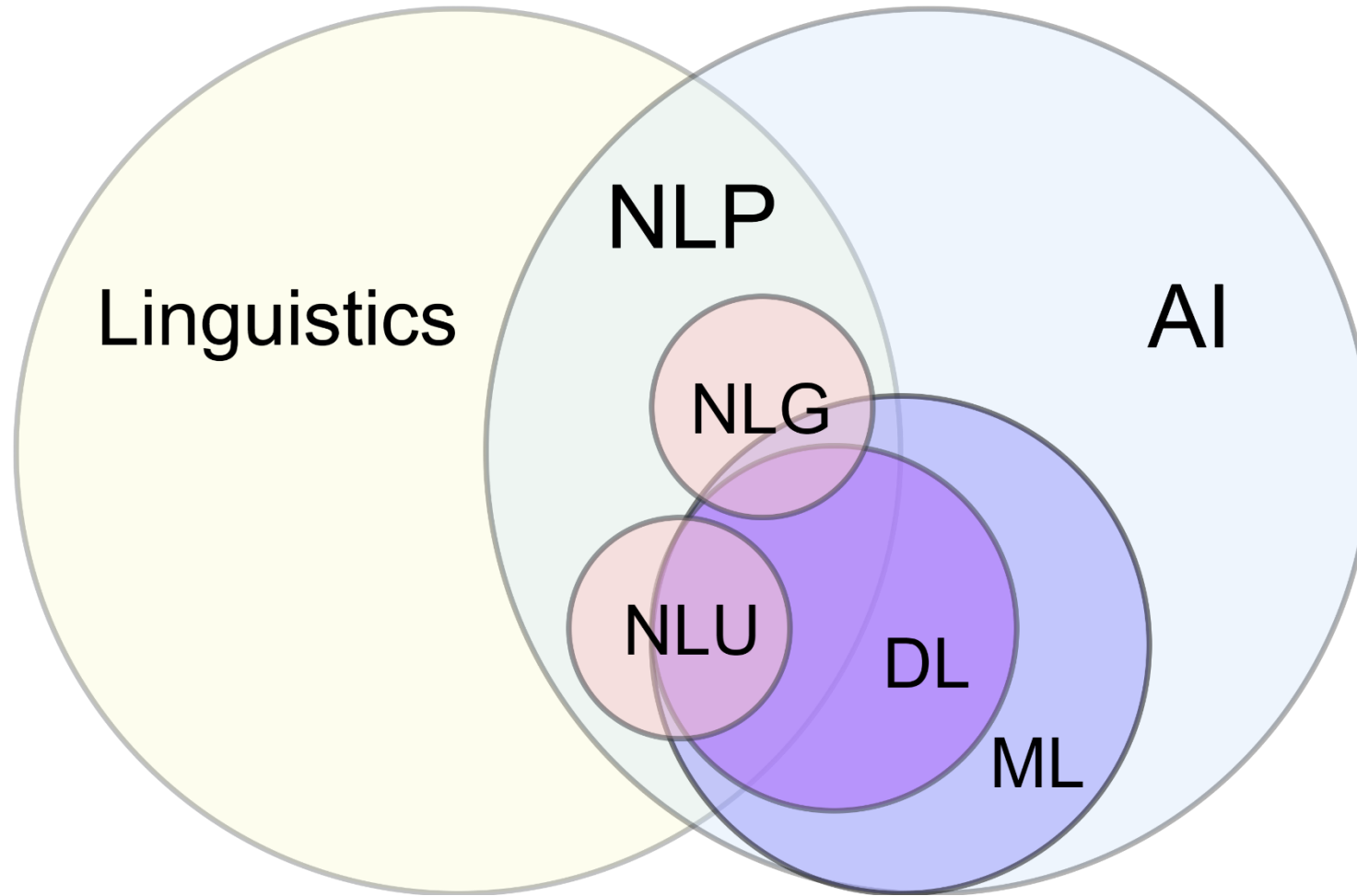
Частично  
структурированные  
(файлы разметки)

```
[{  
  "name": "Шариковая ручка",  
  "type": "ручка",  
  "quantity": "1",  
  "price": "10",  
},  
...  
]
```

Структурированные  
(графы знаний,  
базы данных)



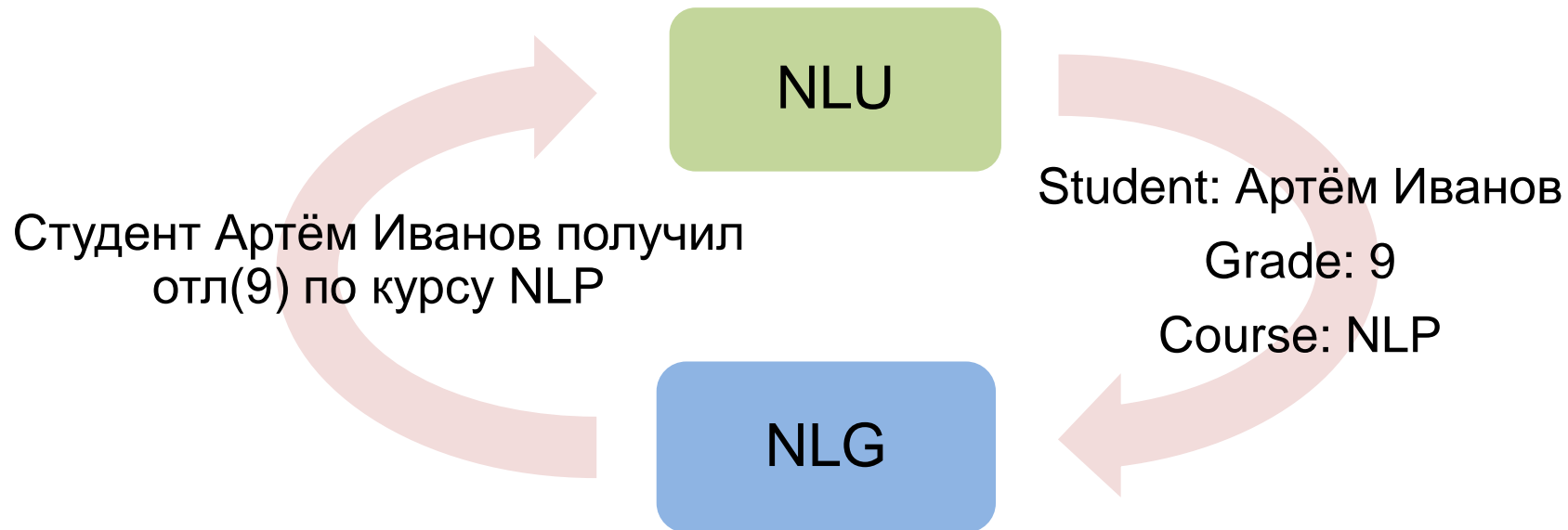
# Как NLP связано с тем, что вы изучали до этого



# NLU и NLG

Внутри NLP условно выделяются два направления:

- Natural Language Understanding (“понимание”), текст -> структура
- Natural Language Generation (генерация), структура -> текст



Почему понимание указано в кавычках?

# Лингвистика

Лингвистика — наука, изучающая языки.

Какие проблемы интересны с точки зрения лингвистики:

- Языки А и В произошли от одного языка?
- Что такое падеж? Сколько падежей в русском языке?
- Анализ изменения языка во времени / в пространстве
- Составление актуального словаря языка

Корпусная лингвистика – создание больших размеченных текстовых датасетов (например, НКРЯ).

# Пирамида NLP – связь NLP с лингвистикой





# Как ещё пересекаются лингвистика и NLP?

- Правилловые системы (rule-based).  
Используются всё реже с появлением глубинного обучения!
- Интерпретация моделей.
- Создание нестандартных задач для предобучения глубоких нейросетей.
- Создание моделей, решающих определённые лингвистические задачи (research).

# Особенности NLP как области анализа данных

Базовая структурная единица языка — слово.

Даже вне контекста слово несёт огромную информацию.

**Пример.** Если предложение содержит слово “посредственность”, скорее всего оно эмоционально негативно окрашено.

**Почему это полезно.** Можем в некоторых ситуациях пренебречь структурой текста.

Что можно сказать о картинке, если мы знаем цвет одного из пикселей?

# Особенности NLP как области анализа данных

Текст без дополнительной разметки имеет внутреннюю структуру, определяемую языком на разных уровнях:

- Текст – последовательность предложений
- Предложение – последовательность слов (связанных синтаксическими связями)
- Слова – последовательность букв / слогов

**Почему это полезно.** Имея большой массив структурированных данных, легко составлять большие выборки и задачи для предобучения глубоких нейросетей.

# Особенности NLP как области анализа данных

- Большие разреженные признаковые пространства.
- Мультиязычность. Не для всех языков одинаково хорошо работают одни и те же алгоритмы.
- Наличие большого объёма как неразмеченных, так и размеченных текстов

# Какими задачами занимается NLP?

Какие продукты, основанные на NLP, существуют на рынке?

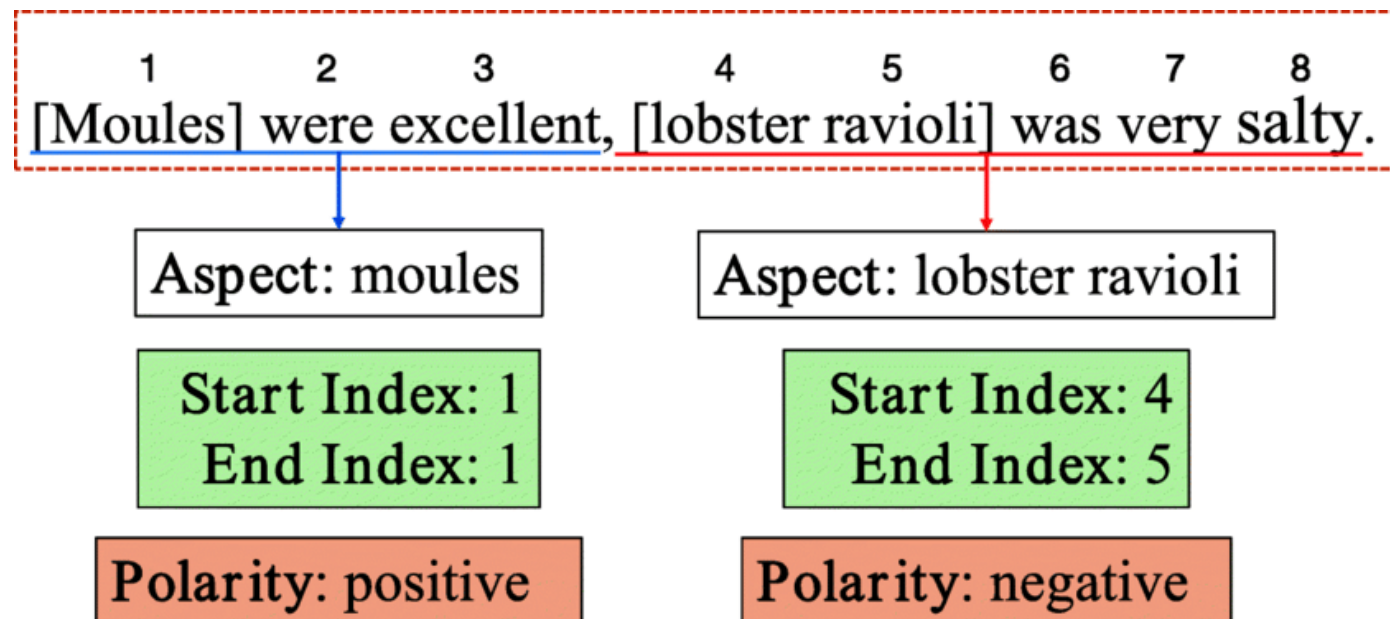
Какие технические задачи нужно решить для создания этих  
продуктов?



# Анализ тональности (sentiment analysis)

Анализ тональности – определение оценки, которую несёт текст.

- бинарный (позитивный / негативный)
- многоклассовый (оценка от пользователя)
- аспектный (тональность отдельных фраз текста)



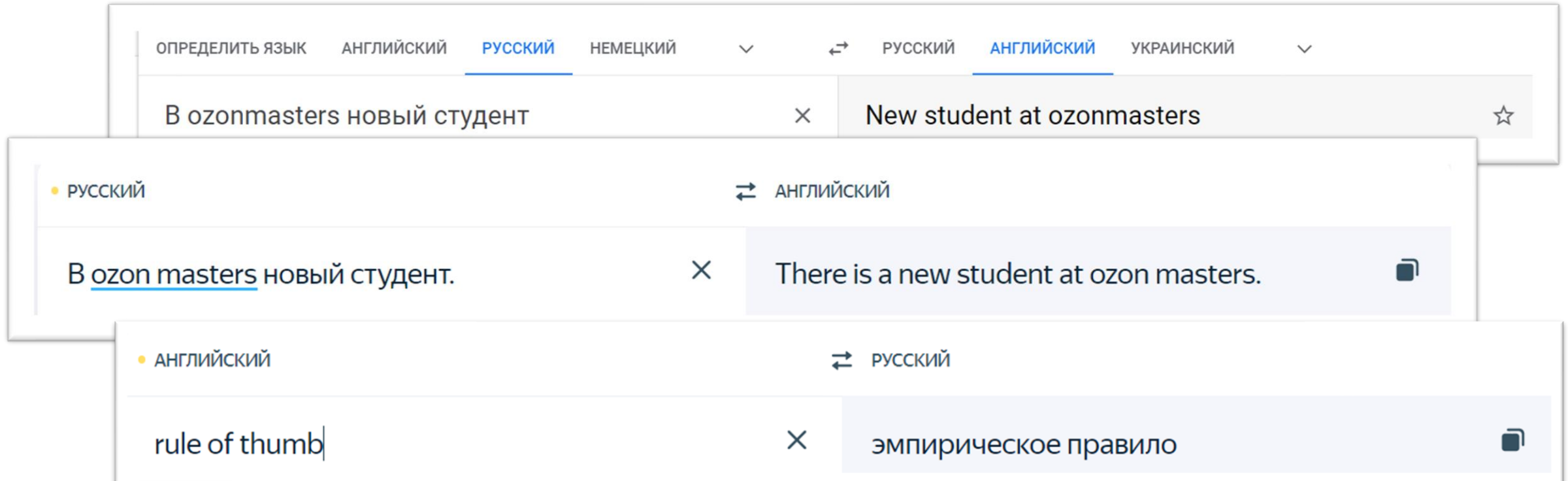
# Задачи, основанные на классификации

- Анализ тональности отзывов
- Фильтрация почтового спама
- Категоризация статей / новостей / жалоб клиентов
- Выявление фейковой информации
- Детектирование опасности пациенту по истории заболеваний

# Задача машинного перевода

Для текста на одном языке нужно получить перевод на другой.

Нужно учитывать особенности языка, обороты речи, устойчивые выражения, семантику фразы, кореференцию.

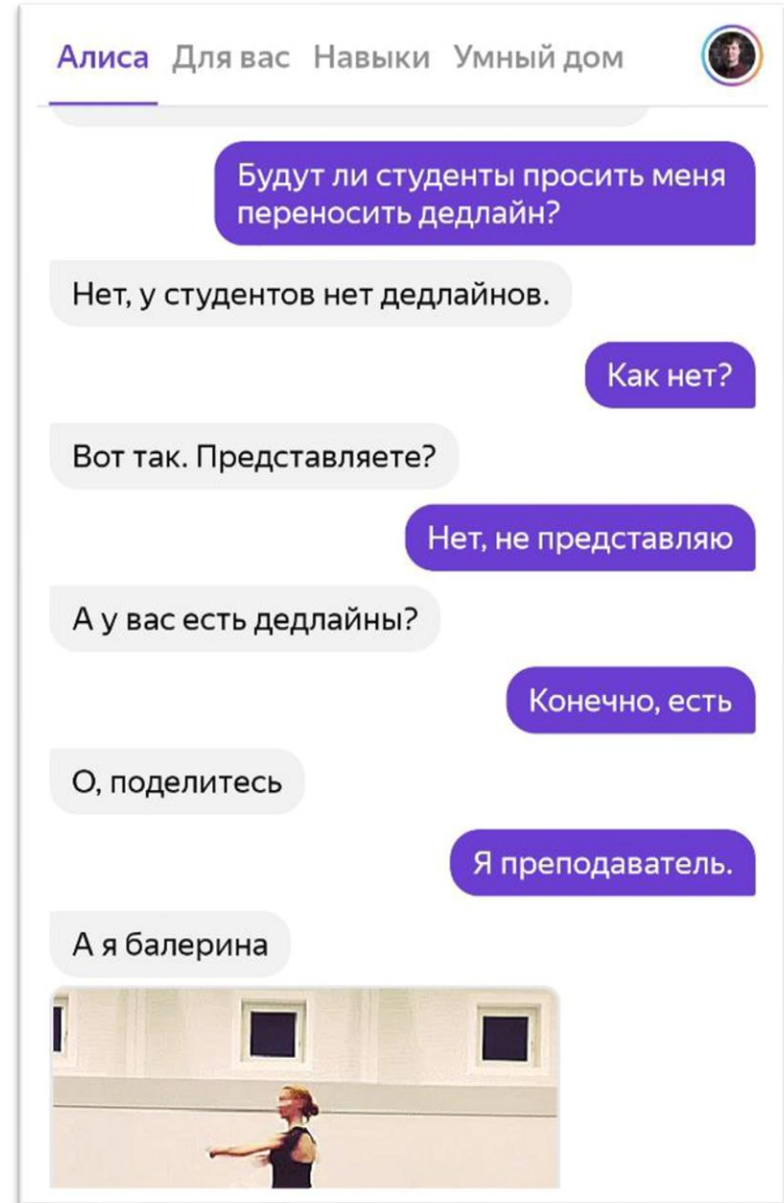


# Задача ведения диалога

Голосовые помощники общаются с человеком на естественном языке.

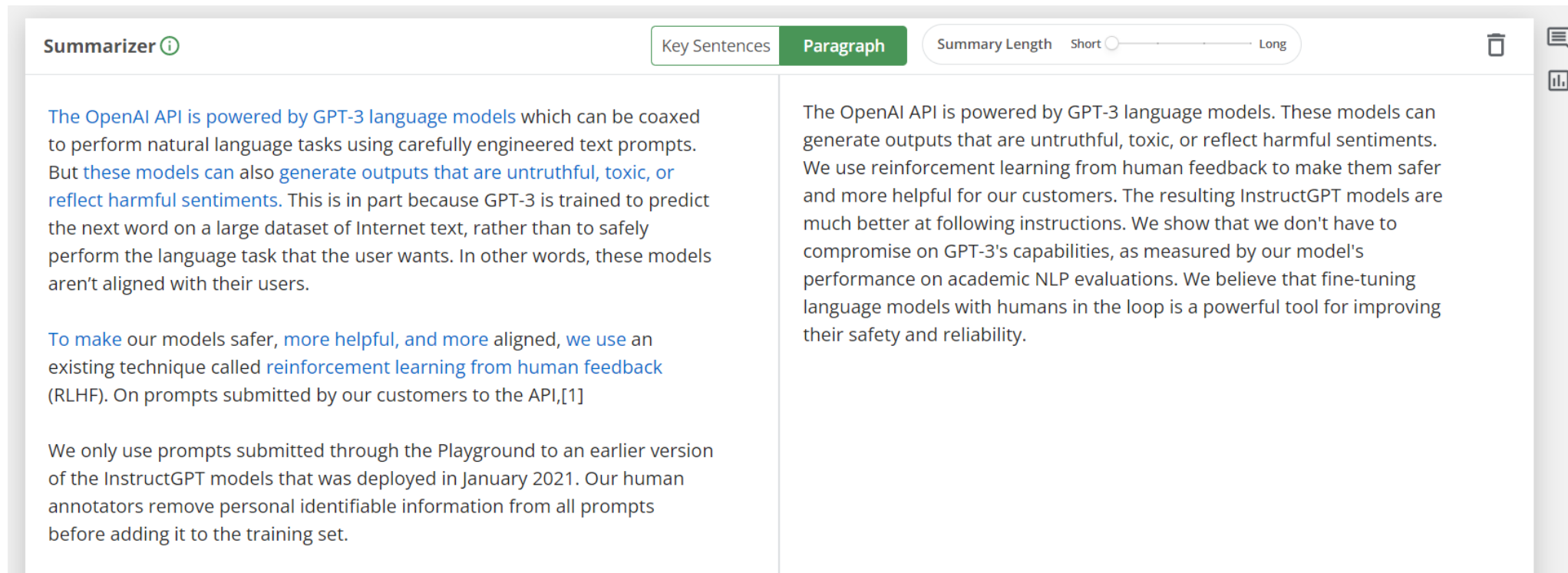
Необходимо проанализировать входную реплику, сгенерировать ответ или действие.

Система может работать детерминировано или в режиме свободного диалога.



# Задача суммаризации

Для текстового документа нужно сгенерировать краткое изложение. Необходимо передать основной смысл, без искажения фактов, в как можно более кратком виде.



The screenshot displays a web-based summarization tool. At the top, there's a header with the title "Summarizer" and an information icon. Below the header, there are two tabs: "Key Sentences" and "Paragraph", with "Paragraph" currently selected. To the right of the tabs is a "Summary Length" slider, set between "Short" and "Long". On the far right, there are icons for a trash can, a list, and a bar chart. The main content area is split into two columns. The left column contains the original text, which discusses the OpenAI API's use of GPT-3 models and the challenges of aligning them with user expectations. The right column shows the generated summary, which is a concise paragraph capturing the main points of the original text.

**Summarizer** ⓘ

Key Sentences Paragraph

Summary Length Short  Long

The OpenAI API is powered by GPT-3 language models which can be coaxed to perform natural language tasks using carefully engineered text prompts. But these models can also generate outputs that are untruthful, toxic, or reflect harmful sentiments. This is in part because GPT-3 is trained to predict the next word on a large dataset of Internet text, rather than to safely perform the language task that the user wants. In other words, these models aren't aligned with their users.

To make our models safer, more helpful, and more aligned, we use an existing technique called reinforcement learning from human feedback (RLHF). On prompts submitted by our customers to the API,[1]

We only use prompts submitted through the Playground to an earlier version of the InstructGPT models that was deployed in January 2021. Our human annotators remove personal identifiable information from all prompts before adding it to the training set.

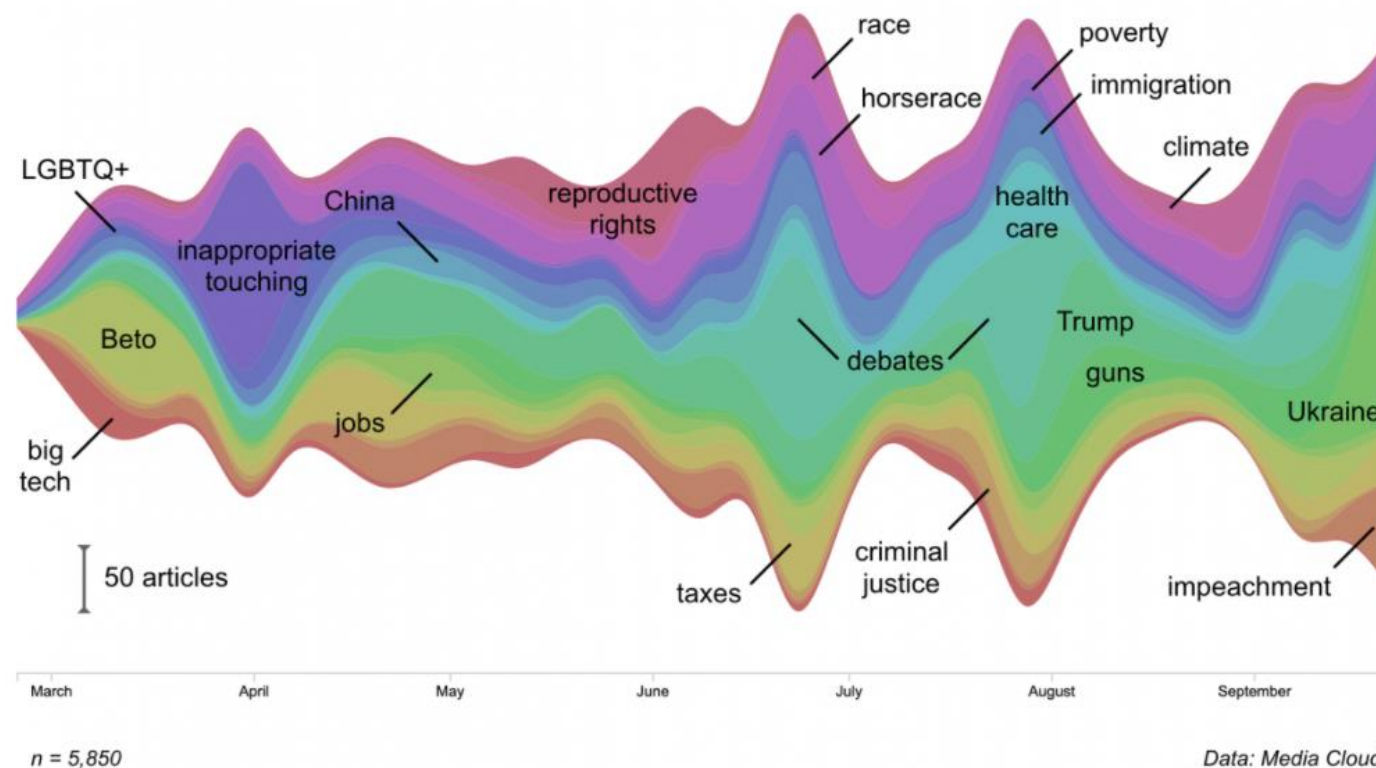
The OpenAI API is powered by GPT-3 language models. These models can generate outputs that are untruthful, toxic, or reflect harmful sentiments. We use reinforcement learning from human feedback to make them safer and more helpful for our customers. The resulting InstructGPT models are much better at following instructions. We show that we don't have to compromise on GPT-3's capabilities, as measured by our model's performance on academic NLP evaluations. We believe that fine-tuning language models with humans in the loop is a powerful tool for improving their safety and reliability.



# Выделение и отслеживание трендов

Извлечение из данных соцсетей обсуждаемых темы и анализ их изменения с течением времени.

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



# Смежные с другими областями задачи

## Компьютерное зрение

- image-to-caption – построение описания по картинке

## Обработка звука:

- speech-to-text – транскрибация речи в текст

## Обработка программного кода:

- text-to-sql – построение по текстовому описанию SQL запроса

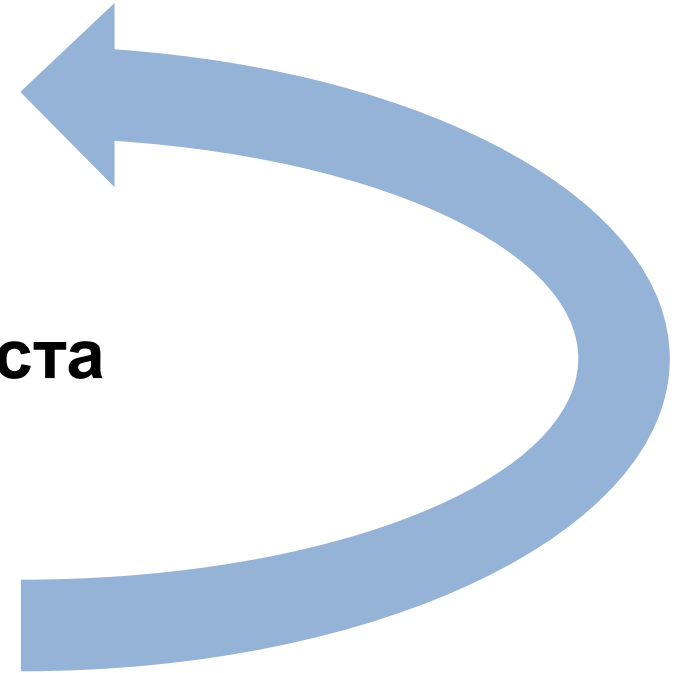
В области обработки кода, биоинформатики используются NLP алгоритмы с некоторыми модификациями.

# Этапы решения NLP задачи

Предобработка данных

# Этапы решения NLP-задачи

1. Выбор верной метрики качества
2. Сбор обучающих и тестовых данных
- 3. Предобработка данных**
- 4. Формирование признакового описания текста**
5. Выбор подхода и класса моделей
6. Обучение моделей и настройка решения
7. Анализ модели и интерпретация ошибок



На сегодняшней лекции рассмотрим пункты 3 и 4.

# Этапы предобработки текстов

Основные этапы предобработки:

- токенизация документа на предложения
- приведение к нижнему регистру
- токенизация предложений на слова
- фильтрация слов (частота, длина, паттерн, стоп-слова)
- фильтрация символов (удаление знаков препинания)
- нормализация слов (лемматизация, стемминг)

**Не все этапы нужны в любой задаче!**

**Не всегда порядок этапов такой как на слайде!**



# Основные средства при предобработке текста

- **re** – библиотека для работы с регулярными выражениями
- **nltk** (Natural Language Toolkit) – библиотека, содержащая большое количество классических алгоритмов NLP, в том числе для предобработки текста
- **spacy** – набор разных NLP-средств для работы с разными языками (но для предобработки часто избыточен)
- **natasha** – набор разных NLP-средств для работы с русским языком (*natasha*, *razdel*, *naves* и т.д.)
- **pymorphy2** – базовый лемматизатор для русского языка

# Регулярные выражения

Регулярное выражение – некоторый строковый шаблон, на соответствие которому можно проверить текст.

Основной синтаксис библиотеки **re**:

- **compile** – компиляция шаблона (для ускорения использования)
- **search** – поиск подстроки, соответствующей шаблону
- **sub** – замена непересекающихся подстрок, соответствующих шаблону, на выбранную строку
- **split** – разбиение предложения на слова по разделителю, заданному шаблоном
- **findall** – поиск всех непересекающихся подстрок, соответствующих шаблону

# Примеры регулярных выражений: базовые

Здесь и далее примеры использования команды `re.findall`:

<code>aba</code>	точное совпадение	<code>ababaabaхaba</code>
<code>[xyzA-D]</code>	один из множества	<code>xhelloy camelCase</code>
<code>[^xyzA-D]</code>	всё кроме множества	<code>xhelloy camelCase</code>
<code>.</code>	любой символ (один)	<code>anything, all</code>
<code>ab AB</code>	любое из двух	<code>abaaBaABaBa</code>
<code>^word</code>	в начале строки	<code>word and word</code>
<code>word\$</code>	в конце строки	<code>word and word</code>
<code>\</code> например, <code>\[</code>	задание служебных символов	<code>[word]</code>

# Примеры регулярных выражений: сокращения

\d	[0-9]	123456789,000*1e-10
\D	обратный к \D, [^0-9]	123456789,000*1e-10
\w	символы слов из юникода если выставлен флаг ASCII: [a-zA-Z0-9_]	\$999
\W	обратный к \w	\$999
\s	пробельные символы [ \t\n\r\f\v]	Word and word
\S	обратный к \s	Word and word

# Примеры регулярных выражений: модификаторы

Результат применения шаблона к строке “www.ru”:

{n,m}	предыдущий шаблон от n до m раз	w{1,2}	[“w”, “ww”]
*	предыдущий шаблон любое число раз ~ {0}	w*	[“www”, “”, “”, “”, “”]
+	предыдущий шаблон ненулевое число раз ~ {1}	w+	[“www”]
?	предыдущий шаблон ноль или один раз ~ {0,1}	w?	[ 'w', 'w', 'w', '', '', '', '' ]
+? или *? или ??	не жадный вариант модификаторов	w+?	[w, w, w]

# Фильтрация телефонных номеров

Напишите регулярное выражение, по которому можно будет выделять телефонные номера в тексте.

Как минимум, необходимо учесть варианты:

- +79005553535
- +7-800-555-35-35
- 8-800-555-35-35
- +7(800)-555-35-35

# Фильтрация телефонных номеров

Напишите регулярное выражение, по которому можно будет выделять телефонные номера в тексте.

Как минимум, необходимо учесть варианты:

- +79005553535
- +7-800-555-35-35
- 8 800 555 35 35
- +7(800)-555-35-35

Пример ответа: `\+?[0-9]?\(?[0-9]{1,4}\)?[-\s0-9]*`

# Пример: категоризация заголовков

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}( . \*|` . \*|:. \*|;. \*|)



# Что ещё нужно знать про регулярные выражения?

\1 - \9	ссылка на сгруппированное раннее выражение	(\w+) \1	off off off uff
(?=regex)	Положительный просмотр вперёд (далее идёт regex)	[^. ]+(?= .ru)	www.nlp.ru
(?!regex)	Отрицательный просмотр вперёд (далее не идёт regex)	[^. ]+(?! .ru)	www.nlp.ru
(?<=regex)	Положительный просмотр назад (ранее идёт regex)	(?<=www. )[^. ]+	www.nlp.ru
(?<!regex)	Отрицательный просмотр назад (ранее не идёт regex)	(?<!www. )[^. ]+	www.nlp.ru

# Токенизация текста

Деление текста на буквенные n-граммы, слова или предложения.

Обычно используются правилые подходы, но может использоваться и ML (например, свёрточные нейросети).

```
1 from nltk.tokenize import sent_tokenize
2 from razdel import sentenize
```

```
1 text = "Это предложение. И это тоже предложение и т.д. А А.С. Пушкин родился 06.06.1799"
```

```
1 for elem in razdel.sentenize(text):
2     print(elem.text, end=" | ")
```

Это предложение. | И это тоже предложение и т.д. | А А.С. Пушкин родился 06.06.1799 |

```
1 for elem in sent_tokenize(text):
2     print(elem, end=" | ")
```

Это предложение. | И это тоже предложение и т.д. | А А.С. | Пушкин родился 06.06.1799 |

# Регистр и пунктуация

Почти всегда следует удалять пунктуацию и приводить текст к нижнему регистру (метод `.lower()`), но есть исключения:

- определение границ предложения
- анализ тональности (капс это агрессия, смайлы важны)
- выделение именованные сущности
- генеративные модели

✓ одежда у вас в магазине очень своеобразная :)

✗ одежда у вас в магазине очень своеобразная :/

как только лев зашёл в комнату, он взял сигарету и закурил

лев обитает в саванне, в арктике не обитает

# Нормализация слов

Слова в тексте могут иметь различную формы, часто такая информация скорее мешает, чем помогает анализу.

Для нормализации применяется один из двух подходов:

- лемматизация – приведение слова к нормальной форме (pymorphy2, pymystem3)
- стемминг – отбрасывание окончания у слова (реализации в nltk)

Для русского языка обычно используют лемматизацию.

Для английского языка используют стемминг или вообще обходятся без нормализации.

# Фильтрация словаря

Часто из текстов нужно удалять лишние слова для понижения размерности признаков или уменьшения размера модели.

Обычно удаляют:

- стоп-слова – вводные слова, предлоги, местоимения и т.п.
- редкие слова (встречаются меньше чем в 5 документах)
- частые слова (встречаются в 99% документах)

Базовый набор стоп-слов есть в nltk (`nltk.corpus.stopwords`).

# Фильтрация словаря: закон Ципфа

**Закон Ципфа.** Если все слова языка упорядочить по убыванию их частотности, то частотность  $n$ -го слова будет примерно обратно пропорциональна его порядковому номеру  $n$ .

Сильные отклонения от закона Ципфа – один из сигналов, что что-то пошло не так.

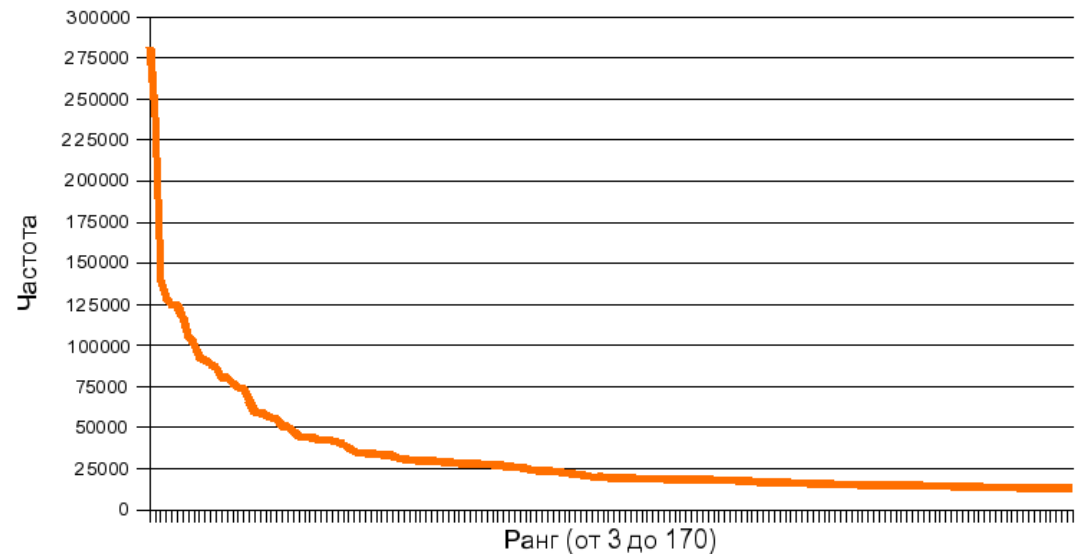


График для частотностей слов из статей русской Википедии с рангами от 3 до 170

# Этапы решения NLP задачи

Разреженные признаковые описания  
документов

Линейные модели классификации

# Основные обозначения

$W$  – словарь коллекции, множество уникальных токенов

$w \in W$  – один токен коллекции (слово, n-грамма, символ и т.д.)

$D$  – коллекция документов

$d \in D$  – один документ коллекции

$n_d$  – длина документа  $d$  в токенах

$d = [w_1, w_2, \dots, w_{n_d}], w_i \in W$

$v_d$  – признаковое описание документа  $d$



# Представление мешка слов (Bag of words)

Наличие слова в документе само по себе несёт информацию => можем пренебречь структурой текста для его представления:

$$v_d = [n_{wd}]_{w \in W}$$

$n_{wd}$  – частота появления слова  $w$  в документе  $d$

## Свойства представления:

- длина вектора  $v_d$  –  $|W|$ , количество уникальных токенов
- обычно  $v_d$  – разреженный (большое количество нулей)

В каких случаях частота слова неинформативный признак?

# Представление TF-IDF

**Идея.** Важны не частотные слова, а слова, которые часто встречаются в данном документе и редко в других документах.

$$v_d = [tf(w, d) \times idf(w, D)]_{w \in W}$$

$$tf(w, d) = \frac{n_{wd}}{n_d}$$

$$idf(w, D) = \log \frac{|D|}{|\{d \mid n_{wd} > 0, d \in D\}|}$$

По TF-IDF можно выделять ключевые слова в документе.

# BoW и TF-IDF в scikit-learn

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

c_vectorizer, t_vectorizer = CountVectorizer(), TfidfVectorizer()
corpus = [
    "This is the first document.",
    "This is the second second document.",
    "And the third one.",
    "Is this the first document?",
]
X_c = c_vectorizer.fit_transform(corpus)
X_t = t_vectorizer.fit_transform(corpus)
```

**Важно.** По умолчанию TfidfVectorizer проводит L2 нормировку  $v_d$

# Коллокации и n-граммы

Токеном может быть не только слово, но и n-грамма:

- n-грамма – частотная последовательность из n слов, встречающихся подряд в тексте
- коллокация – устойчивое сочетание слов, связанных синтаксически и семантически

## Отличие n-граммы от коллокации:

- граница между коллокациями и n-граммами достаточно размыта
- n-граммы выделять проще, а коллокации сложнее
- коллокации могут быть разрывными, n-граммы нет
- n-граммы больше про практику, а коллокации про теорию

# Коллокации и n-граммы

*Он поставил нам условие: даже если идёт дождь, мы всё равно тренируемся.*

**2-граммы (без отсева по частоте): ???**

**коллокации (из 2 слов): ???**

# Коллокации и n-граммы

*Он поставил нам условие: даже если идёт дождь, мы всё равно тренируемся.*

**2-граммы (без отсева по частоте):** он поставил, поставил нам, нам условие, условие даже, даже если, если идёт, идёт дождь, дождь мы, мы всё, всё равно, равно тренируемся

**коллокации (из 2 слов):** поставил условие, идёт дождь, всё равно

# Методы получения n-грамм / коллокаций

- на основе частот встречаемости (nltk, scikit-learn)
- на основе морфологических шаблонов (Томита, YARGY-парсер)

***Пример.** Частотные последовательности, в которых первое и третье слово существительные, а второе предлог.*

- с помощью ассоциации и статистических критериев (nltk, TopMine, Gensim)
- графовые подходы (TextRank)

# Меры ассоциации биграмм

**PMI** (pointwise mutual information, поточечная взаимная информация):

$$PMI(w, u) = \log \frac{p(w, u)}{p(w)p(u)} \approx \log \frac{n_{wu}}{n_w n_u} + Const(D)$$

$n_w$  – частота появления токена  $w$  в коллекции  $D$

$n_{wu}$  – частота появления последовательности  $[w, u]$  в коллекции  $D$

**T-score** (на основе теста стьюдента):

$$T_{score}(w, u) = \frac{f(w, u) - f(w)f(u)}{\sqrt{f(w_1, w_2)}}$$

$f$  – статистика на основе частоты появления



# Меры ассоциации биграмм

Хотим итеративно проверять гипотезу независимости появления пары токенов в коллекции.

для  $k$  от 1 до  $N$ :

считаем по коллекции попарные и униграммные частоты токенов

для всех соседних токенов:

проверяем гипотезу независимости

если верна, объединяем два токена в один

# Разрывные коллокации

Что делать если мы хотим извлекать разрывные коллокации?

- подсчитывая попарные совстречаемости, учитываем токены, находящиеся на небольшом расстоянии
- после проверки гипотезы независимости, анализируем дисперсию разброса расстояний

Другой способ – использовать синтаксические деревья:

- подсчитывая попарные совстречаемости, учитываем токены, находящиеся в соседних узлах дерева

На практике такая задача возникает редко.

# Постановка задачи классификации текстов

**Дана** коллекция документов  $D$ , для каждого документа  $d \in D$  известна метка класса  $y_d \in C$  – множеству классов

**Найти** по документу  $d$  предсказать его метку класса  $y_d$

## Метрики качества:

- accuracy (точность) классификации
- **бинарная**: precision (точность), recall (полнота), f-score (f-мера)
- **многоклассовые**: микро/макро-усреднения
- **бинарная (скоринг)**: AUC ROC, logloss

# Алгоритмы классификации текстов

- наивный байесовский классификатор
- линейные модели (логистическая регрессия, SVM)
- полносвязные нейросети
- свёрточные нейросети
- рекуррентные нейросети
- трансформеры

У всех подходов есть преимущества и недостатки!

В качестве бейзлайна хорошо подходит логистическая регрессия.

# Логистическая регрессия

Обучение модели – максимизация логарифма правдоподобия методом стохастического градиентного спуска (его аналогами).

Вероятности класса считаются при помощи сигмоиды:

$$\sum_{d \in D} p(y_d | d) \rightarrow \max_{a, b}$$

$$p(y = 1 | d) = \sigma(\langle v_d, a \rangle + b) = 1 - p(y = 0 | d)$$

$$\sigma(x) = 1 / (1 + \exp(-x))$$

$$\text{где } y \in \{-1, 1\}, a \in \mathbb{R}^{|W|}, b \in \mathbb{R}$$

При обучении можно использовать регуляризацию:

- L1 (Lasso), L2 (Ridge), L1 + L2 (Elastic-Net)

# Многоклассовая классификация

Мультиномиальная регрессия – обобщение логистической регрессии на многоклассовый случай.

Вероятности считаются при помощи функции *softmax*:

$$p(y|d) = \text{softmax}_y(Av_d + b)$$

$$\text{softmax}_y(x) = \frac{\exp(x_y)}{\sum_j \exp(x_j)}$$

где  $y \in K$ ,  $A \in \mathbb{R}^{|K| \times |W|}$ ,  $b \in \mathbb{R}^{|K|}$

**Другой способ:** обобщение бинарной классификации на многоклассовый случай методами One-vs-All и One-vs-One.

# На что стоит обращать внимание

## Порог-бинаризации (бинарная классификация):

- модель возвращает вероятность принадлежности классу, необходимо выбрать порог по которому выбирается метка
- порог лучше всего подбирать по валидационной выборке

## Сокращение размерности:

- можно понизить размерность признакового описания, используя SVD разложение или более продвинутые методы
- часто это даёт улучшение в качестве и ускоряет обучение

**Регуляризация:** стоит использовать всегда, L2 стабильнее L1

# Итоги занятия

- NLP – область на стыке искусственного интеллекта и лингвистики, занимающаяся анализом и синтезом текстов
- NLP имеет свои особенности, отличающие её от классического машинного обучения или CV
- Предобработка данных можно улучшить вашу модель, поэтому к ней нужно подходить ответственно
- Простейшие признаковые представления документа – Bag of Words и TF-IDF
- Логистическая регрессия – хороший бейзлайн при решении задачи классификации



# Материалы по курсу (учебники)

## Основной учебник:

- Dan Jurafsky and James H. Martin [Speech and Language Processing](#) (3rd ed. draft)

## Заслуживают внимания:

- Stewen Bird et. al. [Natural Language Processing with Python](#). 2-nd edition. 2016.
- Большакова Е.И и другие. [Автоматическая обработка текстов на естественном языке и анализ данных](#). НИУ ВШЭ, 2017.
- Yoav Goldberg et. al. Neural Network Methods in Natural Language Processing
- Elena Voita. [NLP Course | For You](#)

# Материалы по курсу (другие курсы)

**Один из лучших курсов по NLP в мире:**

- [CS224N: Natural Language Processing with Deep Learning](#)

**Заслуживают внимания:**

- [YSDA Natural Language Processing course](#)
- [CS224U: Natural Language Understanding](#)
- [Natural Language Processing \(coursera, HSE\)](#)
- [Введение в обработку естественного языка \(stepik\)](#)
- [Нейронные сети и обработка текстов \(stepik, Samsung\)](#)
- [Введение в обработку естественного языка \(CS center\)](#)