

# Stepik. Neural networks and NLP. 1.

## Introduction

### 1.2 В общих чертах: естественный язык и текст

Всем привет! Сейчас мы поговорим о том, что такое [естественный язык](#) вообще, и как с ним работают, в самых общих чертах. Формально, язык можно определить как множество цепочек символов из некоторого алфавита. Не всех цепочек, а только тех, которые удовлетворяют некоторым правилам. Отдельно взятая цепочка, построенная по этим правилам — это и есть текст. Алфавит — это множество символов, которые можно использовать для построения текстов. По идеи, каждая цепочка должна нести какую-то информацию, но на практике — нет. Самое интересное в этом всём — это правила построения цепочек. Если мы хотим работать с текстами на естественном языке, то общий алгоритм сводится к двум глобальным шагам. Сначала, вполне понятно, нам нужно выучить правила, то есть понять их самим, запрограммировать, или обучить какой-нибудь классификатор, а затем нам дают какой-нибудь текст, и мы применяем наши правила к этому тексту, чтобы понять — как именно было построено каждое конкретное предложение.

SAMSUNG  
Research  
Russia

- это множество допустимых цепочек символов из некоторого алфавита
- цепочки строятся по некоторым правилам
- текст — одна такая цепочка
- алфавит — множество символов, из которых строятся тексты языка
- каждая цепочка должна нести какую-то информацию (нет)

Правила построения цепочек — и есть самое интересное!

Анализ языка сводится к двум глобальным задачам:

- ① Обучение: понять правила языка (выучить, запрограммировать и т.п.)
- ② Применение (inference, вывод): для некоторого заданного текста понять по каким именно правилам он построен



Давайте рассмотрим пример, который вы все, скорее всего, знаете. Он был придуман советским лингвистом [Львом Владимировичем Щербой](#) в тридцатых годах. Звучит он так: "Глокая куздра штеко бодланула бокра и кудрячит бокрёнка". (Что?..) Что здесь происходит? Давайте попробуем разобраться. Явно здесь есть три действующих лица: это какая-то "куздра", это "бокр" и "бокрёнок". Ещё, что мы можем из этого предложения понять — это то, что первое лицо что-то делает с остальными двумя, а ещё про эти действующие лица мы можем понять, что "куздра" — какая? — "глокая" и "бокрёнок" — там есть суффиксы, которые говорят нам о том, что это как "бокр" только маленький. Есть и другие примеры наподобие этого, но мы не будем на них подробно останавливаться сейчас — вы можете остановить видео и прочитать. Давайте, попробуем разобраться, что сейчас произошло. В этих текстах много незнакомых слов, поэтому мы не можем детально нарисовать картину, описываемую в этих текстах, но мы можем до какой-то степени понять, что же там происходит — кто с чем что делает, какая последовательность действий, и так далее. Почему же мы можем это понять? Потому, что тексты написаны на русском языке, правила которого мы знаем.

Пример

«Глокая куздра штеко будланула бокра и курдячит бокрёнка» (Л.В. Щерба, 1930-е)

- Как по Вашему, что тут происходит?
- Есть три действующих лица (куздра, бокр и бокрёнок)
- Первое лицо что-то делает с остальными двумя.



Людмила Петрушевская. Лингвистические сказочки

Сяпала Калуша с Калушатами по напушке. И увазила Бутявку, и волит:

- Калушата! Калушаточки! Бутявка!

Калушата присяпали и Бутявку стрямкали. И подудонились.

А Калуша волит:

- Оее! Оее! Бутявка-то некузявая!

Калушата Бутявку вычучили.

Бутявка вздребезнулась, сопритюкнулась и усяпала с напушки.

А Калуша волит калушатам:

- Калушаточки! Не трямкайте бутявок, бутявки дюбые и зюмо-зюмо некузяые.

От бутявок дудонятся.

А Бутявка волит за напушкой:

- Калушата подудонились! Зюмо некузяые! Пуськи бятые!



Правила бывают разные. Обычно их разбивают на уровни — от низкоуровневых до высокоуровневых. Графематические правила — это правила разбиения текста на отдельные единицы: на предложения и слова. Морфологические правила управляют процессом словообразования и словоизменения. Синтаксические правила управляют согласованием слов, чтобы было понятно, что к чему относится. И семантические правила управляют процессом

передачи смысла посредством правильного использования всех вышеперечисленных правил. Ну и наконец — стилистические правила говорят нам, что уместно в данной конкретной ситуации, что неуместно и как мы должны говорить, если хотим вызвать какие-то определённые эмоции у слушателя. Надо заметить, что существуют и другие виды правил, но сейчас это не так важно. Язык — это неотделимая часть культуры, в которой он используется и развивается. И, вроде бы, один и тот же язык может иметь различия на разных уровнях правил — например, "испанский" испанский и "южно-американский" испанский имеют общий синтаксис, но стилистика употребления слов них разная. OK, а при чем же тут собственно компьютеры и искусственный интеллект? Давайте сделаем шаг в эту сторону, рассмотрев общий алгоритм обработки текстов в программе. Как правило, процесс строится снизу вверх, от графематики к семантике, и дальше. И, как правило, на каждом шаге используются результаты всех предыдущих шагов. Итак, в самом начале у нас есть сырой текст, который представляется просто цепочкой символов — в питоне это строковый тип (например, такой). На выходе графематического анализа мы уже получаем "список списков", каждый список представляет предложение, а внутри предложения есть отдельные "токены" или "[лексемы](#)". Следующий уровень — это морфологический анализ. Он обогащает каждую лексему информацией о словоформе, то есть о морфологических признаках и начальной форме слова. Вот — для нашего примера мы видим, что простые строки заменены уже на структуры и для каждого слова указана начальная форма и часть речи. Например, "мама" — это существительное. На выходе [синтаксического анализа](#) мы получаем дерево, характеризующее подчинение слов. В структуре данных оно может задаваться, например, ссылкой на родительский узел. В нашем примере, синтаксическое дерево может иметь следующую конструкцию: главное слово в предложении — глагол. Глаголу подчиняются главные слова в словосочетаниях — как правило, это существительные. А существительным подчиняются определения. Таким образом, мы получили дерево для нашего примера, и в структуре данных оно задаётся ссылкой на номер токена в предложении, например.

- В этих текстах много незнакомых слов
- Мы можем понять, что происходит
- Правила русского языка в этих примерах соблюдаются.
- Правила бывают разных уровней:
  - графематические (как разделять слова и предложения между собой)
  - морфологические (как строить и изменять слова)
  - синтаксические (как согласовывать словоформы друг с другом, чтобы не было "красная велосипед")
  - семантические (как нужно применить все предыдущие виды правил, чтобы сообщить необходимую информацию)
  - стилистические ("уместность" словоупотребления в данной конкретной ситуации, эмоциональное окрашивание)
  - и т.п.
- Многие правила, особенно высокоуровневые - семантика и стилистика, - неотделимы от культуры, в которой язык используется  
например: европейский испанский и южноамериканский испанский



Порядок обработки: низкоуровневые -> высокоуровневые.  
На выходе каждого этапа обработки получается новая структура данных.

- ① "Сырой" текст - цепочка символов

#### Сырой текст

"Мама мыла раму. Потом мы пошли гулять."

- ② Графематический анализ - список предложений. Предложение - список лексем (токенов).

#### Токенизованный текст

[["Мама", "мыла", "раму", "."], ["Потом", "мы", "пошли", "гулять."]]

- ③ Морфологический анализ - набор меток для каждого токена и ссылка на начальную форму

#### Текст с морфологическими тегами

[[Token("Мама", init="мама", pos="сущ"), Token("мыла", init="мыть", pos="глаг"), Token("раму", init="рама", pos="сущ"), ...], ...]



- ④ Синтаксический анализ - дерево синтаксического подчинения слов. Ссылки на управляющий токен (на "родителя" в дереве).



Семантический анализ строит ещё более многосвязную структуру — граф или семантическую сеть. Этот граф описывает ситуацию на самом верхнем уровне: что происходит, какие есть участники и какие роли они выполняют. Давайте вернёмся к нашему примеру: главное слово, которое задаёт структуру ситуации — опять же, глагол. Вместо глагола может использоваться также отглагольное существительное, также такие слова называют [предикатными словами](#). Кроме предикатного слова в предложении есть какие-то аргументы, "мама", "оконная рама" и "тряпка". Аргументы в контексте глагола получают какие-то роли. Например, мама это — субъект, "оконная рама" — это объект, к нему применяются действие, а "тряпка" — это инструмент. Где же здесь график? На самом деле, ребра обычно проводятся между главным словом и аргументами, а также между аргументами напрямую — это называется "семантические отношения".

## Дерево зависимостей

```
[[Token("Мама", init="мама", pos="сущ", synt_parent=1), Token("мыла", init="мыть", pos="гл, synt_parent=None), ...
```

Мама      мыла      оконную      раму      тряпкой.

- 5 Семантический анализ - граф, описывающий ситуацию на верхнем уровне.

Предикат - главное слово, определяющее структуру ситуации.

Рёбра связывают участников ситуации, метки рёбер характеризуют их роли и отношения.

## Семантический график



Ну и, наконец, результаты всех выполненных шагов агрегируются для того, чтобы решать конечную задачу — то, зачем мы взялись за обработку текста. Итак, мы рассмотрели — что такое [естественный язык](#) вообще, увидели, что правила бывают разных уровней, и что, даже когда мы не знаем ни одного слова в предложении, мы всё равно можем понять его смысл, потому что мы знаем язык, на котором написан этот текст. А ещё мы рассмотрели общий алгоритм обработки текста — снизу вверх, от графематического анализа до семантического, и дальше.

- 
- Что такое естественный язык?
  - Правила бывают разных уровней
  - Общий алгоритм обработки текста

## 1.3 Особенности обработки естественных языков

Давайте поговорим о некоторых особенностях [естественных языков](#). Первая особенность заключается в том, что эти языки никем специально не составлялись, а появились и эволюционировали в результате потребности людей в коммуникации. Никто полностью не знает правил естественных языков существует множество неписанных правил, которые меняются от региона к региону и со временем, особенно сейчас, в эпоху глобализации. Не обязательно соблюдать правила. Если вы скажете какую-то фразу неправильно, то, скорее всего, вас всё равно поймут (до определённого предела). А ещё, правила в естественном языке неоднозначны — позже мы рассмотрим парочку таких примеров. Естественным языкам часто противопоставляются [формальные языки](#). Например, языки программирования или языки математических формул. Такие языки подчиняются строгим правилам, которые задаются, например, порождающими формальными грамматиками. Для таких языков процесс разбора и поиска ошибок детерминирован и есть эффективные алгоритмы, реализующие этот процесс.

- никем специально не составлялся, а появился в результате потребности людей в коммуникации<sup>1</sup>
- его правила строго не регламентированы
- "неписанные" правила меняются от региона к региону и во времени
- от правил можно отклоняться и все всё равно будут понимать, о чём речь
- правила неоднозначные



- например:
  - языки программирования (Python, C++, и т.п.)
  - математические формулы
- строго определены (например, формальными грамматиками)
- процесс разбора и поиска ошибок детерминирован
- есть эффективные алгоритмы ( $LL^*$ -, LR-парсеры)



На Земле есть множество [естественных языков](#) и они отличаются множеством характеристик. За группировку и классификацию языков отвечает раздел науки "лингвистическая типология". Самые простые отличия (естественно, кроме алфавита) заключаются в способе образования и изменения словоформ в зависимости от ситуации, а также в том, как слова связываются друг с другом и как реализуется синтаксическое подчинение. Естественно, смысл тоже передаётся

разными способами. Давайте рассмотрим различие между английским и русским языком по самым наглядным характеристикам (естественно, не считая алфавита). Это флексивность, то есть возможность менять форму слова в зависимости от ситуации, неоднозначность по смыслу, по части речи, а также свобода, которая у нас есть в порядке расположения слов в предложении. В русском языке флексивность выражена гораздо сильнее, чем в английском. В английском слова "кошка" всегда — "cat". Только для множественного числа добавляется окончание "-s". В русском же, то же самое слово можно просклонять по падежам: "бежит кошка", "покормить кошку", "нет кошки", и так далее. Другой пример флексивности — это словообразование с помощью приставок. Приставки могут сильно менять смысл слова. Например, если взять слово "брать", а потом попробовать добавить к нему приставки и перевести каждый вариант на английский, получим три разных перевода. При автоматической обработке часто удобно получить начальную форму слова или какой-то суррогат начальной формы. Самый простой способ — это "стэмминг", или получение основы слова путём отбрасывания окончания и приставок. Для английского стэмминг часто неплохо работает. Например, достаточно, часто, всего лишь убрать окончание "-s". Для русского есть риск отбросить слишком много значимых частей от слова. Более сложный, но и более достоверный подход — лемматизация. В ней используется знание о точной части речи слова, поэтому можно с большей уверенностью предположить, как будет выглядеть его начальная форма.

- Лингвистическая типология

- отличия в образовании и изменении слов в зависимости от ситуации (аналитические, синтетические языки и т.п.)
- отличия в связывании слов друг с другом
- отличия в способах передачи смысла

- Самые яркие различия на практике:

	Английский	Русский
Флективность (словоизменение)	слабая	сильная
Смысловая омонимия	высокая	высокая
Частеречная омонимия <sup>2</sup>	сильная	умеренная
Порядок слов	фиксированный	свободный

<sup>2</sup>омонимы – слова, одинаковые по написанию, но разные по значению



- Склонение по падежам:

- Английский: a cat, to the cat, many cats ...
- Русский: кошка, кошке, кошки ...

- Словообразование с помощью приставок:

- брать - to take
- прибрать - to tidy up
- перебрать - to sort out

- Задачи:

- стемминг = cats -> cat, кошки -> кош
- лемматизация = cats -> cat, кошки -> кошка



Следующая характеристика — смысловая неоднозначность словоупотреблений. Одно и то же слово, употреблённое в разных контекстах, может иметь разный смысл. Такое явление называется "омонимей". Пример слова, имеющего множество смыслов в английском языке — "well". В первом случае оно соответствует русскому слову "хорошо". Во втором случае она ближе по смыслу к частицам "ладно", "так уж и быть"... А в третьем случае — это уже

существительное, обозначающее "углубление". Надо заметить, что неоднозначность смысла здесь идёт вместе с неоднозначностью части речи: глагол, наречие, частица или существительное. Смысловую омонимию в русском языке рассмотрим на примере слова "прибрать". В первом случае это — "сделать уборку", а во втором — ближе по смыслу к слову "присвоить". Или можно взять тот же самый глагол с другой приставкой. Глагол "перебрать" можно понимать как действие разбора на малейшие детали с последующей сборкой — ремонт, а ещё можно "перебрать" — "хватить лишка". Интересно, что в примерах на русском языке смысловая [омонимия](#) не обязательно сопровождается частеречной. На практике — лучше, когда неоднозначности нет вообще. Снять смысловую неоднозначность призваны соответствующие методы. На английском эта задача называется "word sense disambiguation". Частеречная омонимия — это как смысловая, только неоднозначность заключается не в выборе смысла, а в выборе части речи. Немножко мы уже коснулись этой темы. Например, в предложении "мама мыла раму", какую часть речи имеет слово "мыла"? А в предложении "в ванной не было мыла"? Где-то это глагол, как первом случае, а где-то это существительное, хотя выглядят они абсолютно одинаково. Есть и более экстремальные примеры: "косил косой косой косой" — попробуйте сами определить, какие части речи здесь употребляются. На практике всякая неоднозначность нам мешает. Есть соответствующие методы, которые относятся к задаче снятия частиречной неоднозначности — на английском это звучит как "[part of speech tagging](#)", или "POS-tagging".

- Смысл слова зависит от контекста практически во всех языках.
- Английский:
  - Morphology is a well studied area.
  - Well, I will do that.
  - Put the flour on a flat surface and make a well to hold the eggs
- Русский:
  - Прибрать комнату (сделать уборку)
  - Прибрать к рукам (присвоить, забрать)
  - Перебрать мотор (разобрать и собрать)
  - Перебрать спиртного ( выпить слишком много)
- Задача: word sense disambiguation - выбор смысла слова.



- Это глагол, существительное или прилагательное?
  - Мама мыла раму?
  - В ванной не было мыла.
  - Косил косой косой косой.
- Задача: Part of speech (POS) tagging - снятие частеречной неоднозначности



Ну, и наконец, последняя характеристика на сегодня — это вариативность порядка слов. В русском порядок может быть, практически, каким угодно: "дом, который построил, Джек, я разрушил" — можно поменять сказуемое и подлежащие местами, можно составные части предложения поменять местами, можно даже активный залог на пассивный поменять, можно даже подчинение между частями предложения поменять... На английском же есть только один

вариант: "I destroyed the house that Jack built", И всё. Подлежащее, сказуемое, дополнение, обстоятельства, и ни шага в сторону. И даже магистр Йода и Звёздных Войн — не исключение: слов порядок обратный, но фиксированный, всё равно. На практике мы абстрагируемся от конкретного порядка слов в предложении посредством [синтаксического анализа](#). Порядок слов убираем, но сохраняем связи подчинения между словами.

SAMSUNG  
Research  
Russia

- На русском:

- Дом, который построил Джек, я разрушил.
- Дом, который построил Джек, разрушил я.
- Я разрушил дом, который построил Джек.
- Джек построил дом, который я разрушил.

- На английском:

- I destroyed the house that Jack built.
- Подлежащее, сказуемое, дополнение, обстоятельство – и всё.



SAMSUNG  
Research  
Russia

- На русском:

- Дом, который построил Джек, я разрушил.
- Дом, который построил Джек, разрушил я.
- Я разрушил дом, который построил Джек.
- Джек построил дом, который я разрушил.

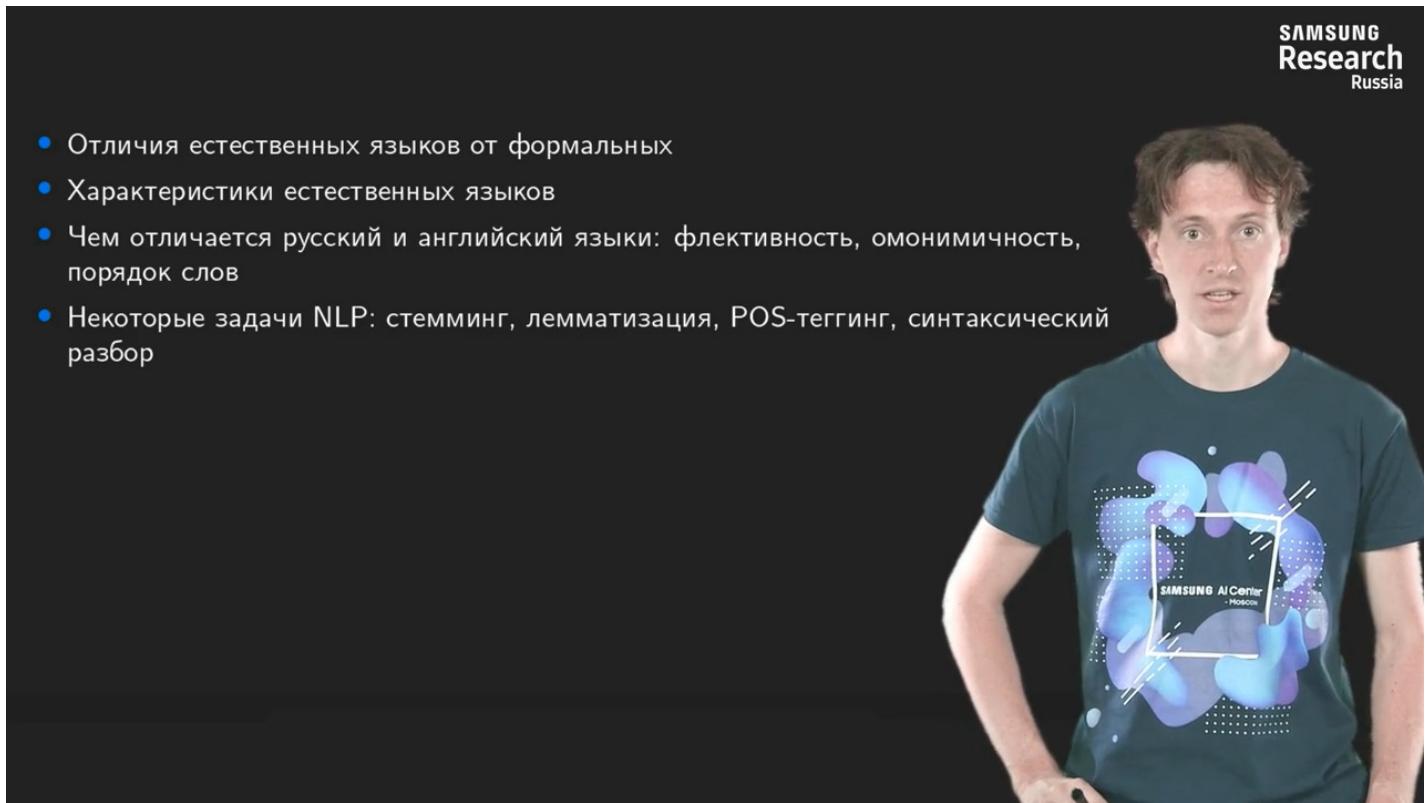
- На английском:

- I destroyed the house that Jack built.
- Подлежащее, сказуемое, дополнение, обстоятельство – и всё.
- Магистр Йода – не исключение – слов порядок обратный, но фиксированный всё равно.

- Задача – синтаксический разбор (построение дерева составляющих или дерева зависимостей)



В этом видео мы рассмотрели, чем отличаются [естественные языки](#) от формальных, а также, чем отличаются разные естественные языки друг от друга. А ещё, мы рассмотрели несколько примеров для русского и английского языка, в том числе различия по флексивности, омонимичности и по порядку слов. Кроме того, мы упомянули несколько типовых задач обработки естественного языка, в том числе стэмминг — получение основы слова, лемматизацию — получение нормальной формы, снятие частицерной и смысловой омонимии, а также синтаксический разбор.



- Отличия естественных языков от формальных
- Характеристики естественных языков
- Чем отличается русский и английский языки: флексивность, омонимичность, порядок слов
- Некоторые задачи NLP: стэмминг, лемматизация, POS-теггинг, синтаксический разбор

## 1.4 В общих чертах: Лингвистический анализ

В этой и паре следующих лекций мы рассмотрим разные задачи, которые в нашей предметной области есть. Можно выделить три высокоуровневые группы задач обработки текстов на [естественном языке](#). Первая группа — это лингвистический анализ. Методы из области лингвистического анализа направлены на разбор структуры текста на разных уровнях. Задачи из этой группы мы рассмотрим прямо в этой лекции. Вторая группа — методы извлечения признаков из текстов. Они частично пересекаются с первой группой, но, в то время как задачи лингвистического анализа — это самостоятельные задачи, задачи извлечения признаков всегда предшествуют применению методов машинного обучения. Третья группа — прикладные задачи, они ближе к бизнесу, к пользователю. Как правило, для их решения используются методы из первых групп плюс какие-то специальные надстройки. Рассмотрим задачи, входящие в лингвистический анализ текстов. Цель: извлечение структуры текста. Как мы уже

говорили ранее, решаются задачи в порядке от низкоуровневых к более высокоуровневым. Сначала мы разбиваем текст на предложение и токены. За это отвечает графематический анализ. Потом разбираем каждое предложение от морфологии до семантики. Затем разбираем связи предложений друг с другом, чтобы структурировать повествование, анализируем дискурс. Отдельно стоит задача генерации текста — это не про анализ, это про синтез. Как правило, для решения каждой задачи используются результаты всех предыдущих. Итак, графематический анализ принимает на вход сырой текст и возвращает разбиение на токены и предложения. Для его реализации, в простейшем случае, используются [регулярные выражения](#). Мы просто находим все разделительные символы — пробелы и знаки препинания, а потом из них отбираем только те, которые соответствуют окончанию предложения. Часто не очень просто определить, какая точка обозначает окончание предложения, а какая — сокращение. Например, в этом предложении есть фамилия, имя, отчество. Имя и отчество сокращены, а дальше следует "точка". Какая из этих трёх точек обозначает конец предложения? Чтобы убрать эту неоднозначность на практике, после регулярных выражений, используются вероятностные модели, описывающие [совместное распределение вероятностей](#) меток токенов. К таким моделям относятся [случайные условные поля](#) или [скрытые марковские модели](#). Мы не будем подробно их рассматривать в данном курсе, но важно, чтобы вы знали, что такие модели есть и знали, как они называются, чтобы могли самостоятельно почитать. Морфологический анализ работает с отдельными токенами. Для каждого токена анализатор предлагает набор сочетаний характеристик, к которым относится часть речи, падеж, число, начальная форма... При этом анализатор не может выбрать только одно сочетание характеристик, потому что для этого нужно привлекать более широкий контекст. Другими словами, частиречная [омонимия](#) пока не разрешается. Например, для слова "мыла" вот в такой форме анализатор предложит два варианта: это прошедшее время и женский род глагола "мыть", а также родительный падеж и единственное число "мыло". Для реализации используются разные методы, самый простой из которых — словарный. Для частотных словоформ все возможные варианты уже известны наперёд, надо их только достать. А для неизвестных или редких слов используются регулярные выражения, системы правил, которые по окончанию слова пытаются предположить, как должна выглядеть начальная форма этого слова, а также морфологические характеристики этой словоформы. Логичный следующий шаг — это для каждого токена, из всех вариантов, предложенных для него морфологическим анализатором, выбрать только один, то есть разрешить частиречную омонимию. Омонимия снимается в рамках одного предложения. Такого контекста на практике достаточно почти всегда. В результате из всего множества вариантов выбирается только один — например, "глагол". Для решения этой задачи существуют методы, основанные на правилах, в том числе полученных с помощью машинного обучения. Однако более современный и более распространённый вариант — это вероятностные модели и последовательности, то есть условные случайные поля и марковские модели. Когда размеченный [корпус](#) достаточно большой, [рекуррентные нейросети](#) позволяют улучшить качество ещё сильнее.

## Список литературы:

Графематика <https://en.wikipedia.org/wiki/Graphemics>

## Регулярные выражения:



Токенизация - SpaCy Tokenizer <https://spacy.io/usage/spacy-101/>

Морфологический анализатор для русского языка

PyMorphy2 <https://pymorphy2.readthedocs.io/en/latest/>

## Список работ с кодом по морфологическому

анализу <https://paperswithcode.com/task/morphological-analysis>

## Как эффективно хранить словари -

Trie <https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%84%D0%B8%D0%BA%D1%81%D0%BD%D0%BE%D0%B5%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE>

Разрешение частеречной омонимии - POS-теггинг (part of speech):

- TreeTagger - система, основанная на правилах <https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/>
  - POS-теггинг в SpaCy <https://spacy.io/usage/linguistic-features/#pos-tagging>
  - POS-теггинг в Stanford NLP <https://nlp.stanford.edu/software/tagger.html>

Вероятностные модели, в том числе вероятностные модели последовательностей:

- Burr Settles, Probabilistic Sequence Models, <http://pages.cs.wisc.edu/~bsettles/ibs08/lectures/03-sequencemodels.pdf>
  - Hidden Markov Models, HMM, [https://en.wikipedia.org/wiki/Hidden\\_Markov\\_model](https://en.wikipedia.org/wiki/Hidden_Markov_model)
  - Probabilistic Graphical Models, Coursera, <https://www.coursera.org/specializations/probabilistic-graphical-models>

- Лингвистический анализ
- Извлечение признаков - построение векторного представления, графового, сопоставление со словарями.
- Прикладные задачи - классификация, поиск по запросу, поиск похожих, извлечение именованных сущностей и фактов



Цель - извлечение структуры текста. Фундамент для решения других задач.

Задачи решаются в следующем порядке:

① Подготовка

- 1 графематический анализ

② Анализ отдельных предложений

- 1 морфологический анализ
- 2 POS-теггинг
- 3 извлечение именованных сущностей
- 4 синтаксический анализ
- 5 семантический анализ
- 6 извлечение отношений между сущностями внутри предложения

③ Анализ целых текстов

- 1 разрешение анафорических связей
- 2 дискурсивный анализ

④ Генерация текста



## Исходные данные

"Сырой" текст - последовательность символов.  
"Днём мама мыла раму. Вечером мы пошли гулять."

## Результат

Разбиение текста на предложения и лексемы (токены).  
[["Днём", "мама", "мыла", "раму", "."], ["Вечером", ...], ...]

## Механизмы

- Регулярные выражения для простого разбиения на токены.
- Вероятностные модели (Hidden Markov Model, Conditional Random Fields) для разрешения неоднозначностей.

"Я прочитал роман М.А. Булгакова. Великий писатель" - на какой из трёх точек заканчивается первое предложение?



## Исходные данные

Текст, разбитый на предложения и токены. Единица обработки - один токен.  
['Мама', 'мыла', 'раму', '.']

## Результат

Набор сочетаний морфологических характеристик и нормальных форм для каждого слова.

Частеречная омонимия - разные слова в разных формах могут выглядеть одинаково. Варианты для слова "мыла":

- глагол - нормальная форма "мыть", прошедшее время, женский род.
- существительное - нормальная форма "мыло", родительный падеж.

Token("мыла", variants=[Form("мыло", "сущ.", "ед.", "ср.", ...), Form("мыть", "гл.", "прош.", "ед.", "ж.", ...), ...])

## Механизмы

- Словари
- Системы правил, регулярные выражения



## Исходные данные

Результаты всех предыдущих шагов. Единица анализа - предложение.

## Результат

Для каждого словоупотребления выбирается только одна нормальная форма и одно значение для каждой морфологической характеристики. Частеречная омонимия снята.

Token("мыла", norm=Lexeme("мыть", "гл.", "прош.", "ед.", "ж.", ...))

## Механизмы

- Системы правил
- Вероятностные модели последовательностей (Hidden Markov Model, Conditional Random Field)
- Нейросетевые модели



После выполнения всех предыдущих шагов, как правило, появляется возможность выполнить первую более-менее прикладную задачу: [извлечь именованные сущности](#). Рассмотрим такое вот предложение для примера. В рамках задачи извлечения именованных сущностей, необходимо для каждого словосочетания или короткой последовательности токенов, установить класс. Например, "организация" — у нас и здесь две организации: "Рога и копыта" и "Хвосты и лапы". Другой класс именованной сущности — это имя человека. У нас здесь есть "Василий Петров". Могут быть географические наименования и ещё множество разных вариантов. Самый простой метод — словарный. Он позволяет достаточно надёжно выделить широко известные и при этом уникальные имена, такие как "Samsung" или "Сбербанк". Эти слова не имеют других смыслов или функций, кроме обозначения данных конкретных организаций. Для некоторых типов сущностей (например, адрес электронной почты или номер телефона) хорошо подходят [регулярные выражения](#). Однако на практике для повышения точности, то есть уменьшения количества ложноположительных предсказаний, необходимо применять вероятностные модели. Сначала по словарю или с помощью регулярных выражений выделяем какой-то набор кандидатов, а затем мы применяем вероятностную модель, чтобы выбрать наиболее подходящее сочетание классов для каждого токена. Нейросети позволяют объединить все эти подходы в один алгоритм, все шаги которого настраиваются совместно с помощью методов оптимизации. Теперь мы можем выполнить [синтаксический анализ](#), результатом которого является, например, дерево зависимостей. В дереве зависимостей для каждого токена указана ссылка на его предка. Дерево зависимостей может выглядеть, например, так. Самый распространённый подход для синтаксического анализа сейчас — это хорошо известные классические алгоритмы класса "[сдвиг-свёртка](#)",

которые используются для синтаксического анализа не только [естественных языков](#), но и языков программирования. Однако существенное отличие в их вариантах для естественных языков заключается в том, что для принятия решений внутри анализатора используются обучаемые классификаторы, которые могут быть как линейными моделями, так и нейросетями. Именно так работают одни из лучших известных сейчас реализаций синтаксических анализаторов — [MaltParser](#) или [SyntaxNet](#).

Список литературы:

Извлечение именованных сущностей:

- [https://en.wikipedia.org/wiki/Named-entity\\_recognition](https://en.wikipedia.org/wiki/Named-entity_recognition)
- AllenNLP демо с извлечением именованных сущностей <https://demo.allennlp.org/named-entity-recognition>
- DeepPavlov демо с извлечением именованных сущностей <https://demo.deeppavlov.ai/#/en/ner>
- Томита-парсер - инструмент для извлечения сущностей и фактов с помощью правил <https://yandex.ru/dev/tomita/>
- Natasha - инструмент для извлечения сущностей и фактов с помощью правил <https://github.com/natasha/natasha>
- Список статей с кодом <https://paperswithcode.com/task/named-entity-recognition-ner>

Алгоритмы синтаксического разбора и анализаторы сдвиг-свёртка:

- [https://en.wikipedia.org/wiki/Canonical\\_LR\\_parser](https://en.wikipedia.org/wiki/Canonical_LR_parser)
- Economopoulos, Giorgios Robert. *Generalised LR parsing algorithms*. Diss. University of London, 2006. <https://pdfs.semanticscholar.org/2ff7/4ea9a0147318bc19e30c6d0f72e29a5f92c3.pdf>
- Breveglieri, Luca, Stefano Crespi Reghizzi, and Angelo Morzenti. "Parsing methods streamlined." *arXiv preprint arXiv:1309.7584* (2013). <https://arxiv.org/abs/1309.7584>
- SyntaxNet <https://ai.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html>
- SyntaxNet TensorFlow <https://github.com/tensorflow/models/tree/master/research/syntaxnet>
- Nivre, Joakim, Johan Hall, and Jens Nilsson. "Maltparser: A data-driven parser-generator for dependency parsing." *LREC*. Vol. 6. 2006. <http://lrec-conf.org/proceedings/lrec2006/pdf/162.pdf.pdf>
- Список статей с кодом <https://paperswithcode.com/task/dependency-parsing>

## Исходные данные

Результаты всех предыдущих этапов. Единица анализа - предложение.

Гендиректор ООО "Рога и копыта" Василий Петров из Москвы подписан соглашение об объединении с петербургской ООО "Хвосты и лапы".

## Результат

Для каждой короткой последовательности токенов установлен класс из числа:

- организация (ООО "Рога и копыта", ООО "Хвосты и лапы")
- имя (Василий Петров)
- географическое наименование (Москвы, петербургской)
- ...

## Механизмы

- Словари
- Системы правил, регулярные выражения
- Вероятностные модели последовательностей (Conditional Random Fields, Hidden Markov Models)
- Нейросетевые модели



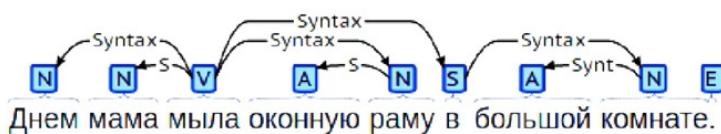
## Исходные данные

Результаты всех предыдущих этапов. Единица анализа - предложение.

## Результат

Построено дерево зависимостей<sup>12</sup>:

- для каждого токена  $T$  указан токен-родитель  $P$ , такой что выбор словоформы  $T$  подчиняется  $P$  (то есть согласуется род, число, время и т.п.).



## Механизмы

- Shift-reduce-анализаторы с вручную составленными или обучаемыми решающими правилами, в том числе нейросетевыми (Malt parser, SyntaxNet)



Отлично! Синтаксический анализ выполнен, следующий шаг — [семантический анализ](#). Мы работаем только с простыми предложениями. Сложные предложения сначала разбиваются на простые. Рассмотрим пример. Первый шаг — это выделение главного слова, предиката. Как правило, это глагол или отглагольное существительное. В данном примере у нас два предиката.

Первый — это "подписал" а второй — это "объединение". Когда предикат найден, мы можем понять структуру ситуации, то есть выделить набор [семантических ролей](#), которые мы ожидаем рядом с этим предикатом увидеть. Существуют две семантические роли, которые есть практически у всех глаголов — это субъект (тот, кто делает) и объект (к кому действия применяется). В нашем случае аргумента три: первый — это "генератор ООО "Рога и копыта", Василий Петров из Москвы" (такой большой один аргумент); второй аргумент — это "соглашение", третий аргумент — это "петербургское ООО "Хвосты и лапы". Напомню, что у нас здесь есть два простых предложения, вернее 2 клаусы (они разделяются вот так). На следующем шаге каждому аргументу назначается роль — например, "генератор" — это субъект, "соглашение" — это объект. Часто в качестве вспомогательного инструмента применяются словари, особенно для глаголов, потому что глаголов не очень много, и мы сразу можем понять по глаголу, какая ролевая структура ему должна соответствовать. Аргументы, как правило, выделяются с помощью правил, применяемых к синтаксическому дереву. Более современные подходы основаны на методах из области [синтаксического анализа](#). Они объединяют все эти шаги воедино. Следующий шаг — извлечь отношения между сущностями. К таким отношениям можно отнести, например, часть/целое или принадлежит, или предшествует/следует, и так далее. В нашем примере можно выделить отношения между, например, сущностями "Василий Петров" и "Рога и копыта". Отношение описывается словом "генератор". Таким образом мы знаем, что Василий Петров — это генератор ООО "Рога и копыта". Ещё мы можем выделить другое отношение — это то, что "Рога и копыта" находится в Москве. Общий подход к решению такого класса задач — выделить всевозможные сущности в предложении, а затем применять некоторый классификатор ко всем возможным их парам.

Список литературы:

Семантический анализ:

- [https://en.wikipedia.org/wiki/Semantic\\_role\\_labeling](https://en.wikipedia.org/wiki/Semantic_role_labeling)
- AllenNLP демо с семантическим анализом <https://demo.allennlp.org/semantic-role-labeling>
- Yih, Scott Wen-tau, and Kristina Toutanova. "Automatic semantic role labeling." *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Tutorial Abstracts*. Association for Computational Linguistics, 2006.
- Gildea, Daniel, and Daniel Jurafsky. "Automatic labeling of semantic roles." *Computational linguistics* 28.3 (2002): 245-288.
- Neural Methods for Semantic Role Labeling <https://diegma.github.io/slides/TutorialNNforSRL.pdf>
- He, Luheng, et al. "Deep semantic role labeling: What works and what's next." *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017.
- Shelmanov A. O., Devyatkin D. A. Semantic Role Labeling with Neural Networks for Texts in Russian <http://www.dialog-21.ru/media/3945/shelmanovaodevatkinda.pdf>

- Список работ с кодом <https://paperswithcode.com/task/semantic-role-labeling>

Извлечение отношений между сущностями:

- AllenNLP демо с извлечением сущностей в открытом домене и отношений между ними <https://demo.allennlp.org/open-information-extraction/>
- Fundel, Katrin, Robert Küffner, and Ralf Zimmer. "RelEx—Relation extraction using dependency parse trees." *Bioinformatics* 23.3 (2006): 365-371.
- Список ключевых статей и оценки качества [http://nlpprogress.com/english/relationship\\_extraction.html](http://nlpprogress.com/english/relationship_extraction.html)
- Список статей по извлечению отношений (в основном нейроети) <https://github.com/roomylee/awesome-relation-extraction>
- Список репозиториев с реализациями извлечения отношений (в основном нейроети) <https://paperswithcode.com/task/relation-extraction>

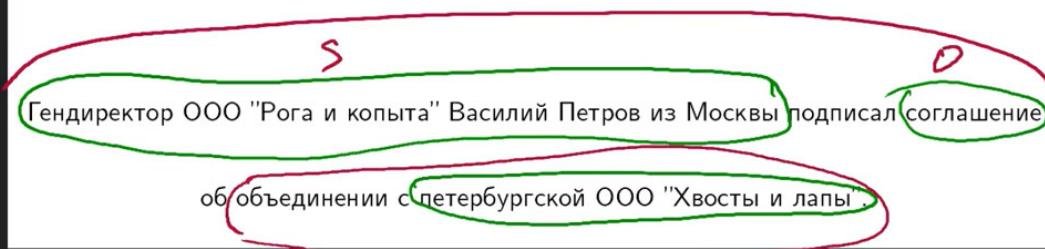
## Исходные данные

Результаты всех предыдущих этапов. Единица анализа - клауза - простое предложение.

SAMSUNG  
Research  
Russia

## Результат

- Определяется главное слово - предикат - он задаёт структуру ситуации
- Ситуация состоит из слотов (валентностей, ролей) и из заполняющих эти слоты аргументов.



## Исходные данные

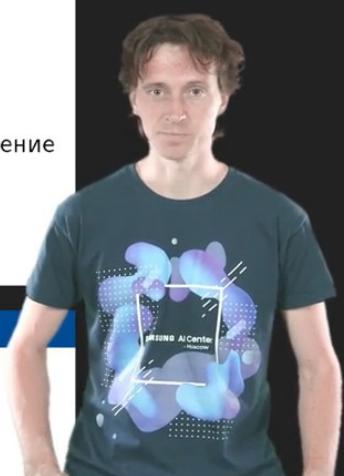
Результаты всех предыдущих этапов. Единица анализа - клауза - простое предложение.

SAMSUNG  
Research  
Russia

## Результат

- Определяется главное слово - предикат - он задаёт структуру ситуации
- Ситуация состоит из слотов (валентностей, ролей) и из заполняющих эти слоты аргументов.
- Выделяются аргументы и сопоставляются со слотами.

Гендиректор ООО "Рога и копыта" Василий Петров из Москвы подписал соглашение об объединении с петербургской ООО "Хвосты и лапы".



## Механизмы

- Словари
- Системы правил
- Все оставшиеся методы из синтаксического анализа

## Исходные данные

Результаты всех предыдущих этапов. Единица анализа - клауз - простое предложение.

## Результат

Для каждой пары сущностей устанавливается класс отношения между ними:

- является частью
- принадлежит
- следует за/предшествует
- ...

Гендиректор ООО "Рога и копыта" Василий Петров из Москвы подpisáл соглашение об объединении с петербургской ООО "Хвосты и лапы".



## Исходные данные

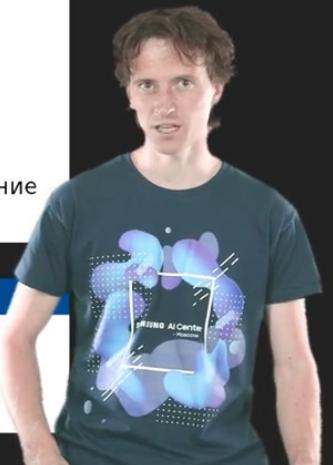
Результаты всех предыдущих этапов. Единица анализа - клауз - простое предложение.

## Результат

Для каждой пары сущностей устанавливается класс отношения между ними:

- является частью
- принадлежит
- следует за/предшествует
- ...

Гендиректор ООО "Рога и копыта" Василий Петров из Москвы подpisáл соглашение об объединении с петербургской ООО "Хвосты и лапы".



## Механизмы

- Попарные классификаторы, основанные на системах правил, классическом машинном обучении или на нейросетях.

Мы закончили с анализом отдельных предложений (то есть мы рассмотрели вот эту часть задачи). Давайте перейдём теперь к анализу текстов в целом. Анафорические связи — это когда местоимение ссылается на какой-то объект, ранее упоминавшейся, или тот, который будет упоминаться в дальнейшем. В результате применения метода разрешения анафорических связей мы ожидаем, что будут установлены связи между такими местоимениями и словами, на

которые они ссылаются — референтами. Рассмотрим пример из двух предложений. Здесь есть два местоимения и есть несколько вариантов референтов, например, для слова "он", можно выделить все слова мужского рода: "вечер", "дом" или "Марк". Соответственно задача заключается в том, чтобы из всех этих вариантов референтов выбрать только один. В данном случае, явно, он — это "Марк". Во втором случае всё попроще, потому что у нас только одно слово женского рода в предыдущем предложении: "осень". Подход в целом такой же как и в предыдущей задаче — задаче извлечения связей между сущностями. Мы сначала извлекаем все возможные пары "местоимение и референт", а потом применяем классификатор, который нам говорит, кто на кого ссылается, действительно. Дискурсивный анализ — это как [синтаксический анализ](#), только для документа в целом. Идея в том, чтобы получить структуру повествования или аргументации — что за чем следует, что является подтверждением, что — опровержением, что с чем сравнивается, и так далее. Дискурс начинает работать, когда у нас есть несколько простых предложений — Например, у нас есть следующее предложение — в нём есть следующие части — 'по сравнению с "x", "y" лучше подходит для "z", потому что "a" и "b"'. Между этими частями мы можем установить, например, следующие связи: "сравнение", "аргументация", а также "однородность". Так как задача выглядит как синтаксический разбор, логично её решать такими же методами. Действительно, так на практике и делают. На практике — это анализаторы класса "[сдвиг-свёртка](#)", в которых внутри работают классификаторы, основанные на машинном обучении.

Список литературы:

Разрешение анафорических связей:

- [https://en.wikipedia.org/wiki/Anaphora\\_\(linguistics\)](https://en.wikipedia.org/wiki/Anaphora_(linguistics)).
- AllenNLP демо с разрешением кореференции [https://demo.allennlp.org/coreference\\_resolution/](https://demo.allennlp.org/coreference_resolution/)
- Poesio, Massimo, Roland Stuckardt, and Yannick Versley. *Anaphora resolution*. Springer, 2016.
- Список работ с кодом <https://paperswithcode.com/sota/coreference-resolution-on-ontonotes>

Анализ дискурса:

- Shelmanov, Artem, et al. "Towards the Data-driven System for Rhetorical Parsing of Russian Texts." *Proceedings of the Workshop on Discourse Relation Parsing and Treebanking 2019*. 2019.
- Yangfeng Ji, Jacob Eisenstein, Representation Learning for Text-level Discourse Parsing, Proceedings of ACL, 2014 <https://github.com/jiyfeng/RSTParser>
- Morey, Mathieu, Philippe Muller, and Nicholas Asher. "How much progress have we made on RST discourse parsing? A replication study of recent results on the RST-DT." 2017. <https://www.aclweb.org/anthology/D17-1136.pdf>

## 1 Подготовка

- 1 графематический анализ

## 2 Анализ отдельных предложений

- 1 морфологический анализ
- 2 POS-теггинг
- 3 извлечение именованных сущностей
- 4 синтаксический анализ
- 5 семантический анализ
- 6 извлечение отношений между сущностями внутри предложения

## 3 Анализ целых текстов

- 1 разрешение анафорических связей
- 2 дискурсивный анализ

## 4 Генерация текста



### Исходные данные

Результаты всех предыдущих этапов. Единица анализа - текст - последовательность простых предложений.

### Результат

Установлены связи между словами-ссылками (например, анафорическими местоимениями) и словами-адресатами (референтами).

"Марк надел пальто и вышел из дома осенним вечером. Он не очень любил холод осени, но её красота согревала."

### Механизмы

- Попарные классификаторы, основанные на системах правил или на классическом машинном обучении.



## Исходные данные

Результаты всех предыдущих этапов. Единица анализа - текст - последовательность простых предложений.

## Результат

Построено дерево, описывающее структуру документа.

По сравнению с X, Y лучше подходит для Z, потому что A и B.



## Исходные данные

Результаты всех предыдущих этапов. Единица анализа - текст - последовательность простых предложений.

## Результат

Построено дерево, описывающее структуру документа.

По сравнению с X, Y лучше подходит для Z, потому что A и B.



## Механизмы

- Методы синтаксического анализа - анализатор "сдвиг-свёртка" (shift-reduce) с решающим правилом в виде линейного классификатора или нейросети.

Давайте ещё поверхностно коснёмся задачи генерации текста. Она достаточно сложная и, вообще, она относится скорее к прикладным задачам, чем к лингвистическому анализу. На вход генератору, как правило, подаётся структурированное описание того, что нужно сказать — что нужно преобразовать в текст на [естественном языке](#). Один из способов описать ситуацию — это [семантический фрейм](#). Это объект, у которого указано, какое действие кто делает, с чем

делают, и так далее. Другой способ задать смысл предложения — это [вектор признаков](#), полученный из нейросети или какими-то другими способами. Ожидаемо, на выходе получается текст в виде последовательности [лексем](#) или прямо в виде строки. Исторически, первый и один из самых практических подходов — это генерация через поиск. Мы не создаём новые тексты, мы просто стараемся найти достаточно подходящий текст из уже имеющихся в базе. Такой подход часто используется в чат-ботах. Альтернативный подход — это генерация по шаблону.

Шаблоны задаются программистами. Этот метод позволяет упоминать в тексте сущности, о которых речь шла ранее в диалоге (например, если речь идёт о чат-ботах). Самый современный, но пока что плохо изученный — это генерация текста с помощью нейросетей. Итак, мы начали рассматривать задачи обработки текстов на естественном языке, а именно — задачи лингвистического анализа. В следующих видео мы рассмотрим остальные группы задач.

SAMSUNG  
Research  
Russia

Исходные данные

Структурированное описание содержания:

- фрейм predicate=подписать, subject=гендиректор, object=соглашение, ...
- вектор признаков

Результат

Текст в виде цепочки лексем или символов.

Механизмы

- Генерация через поиск (retrieval based text generation)
- Генерация по шаблону (template based text generation)
- Генерация с помощью нейросетей

## 1.5 В общих чертах: Извлечение признаков

Всем привет! Сейчас мы поговорим о задачах извлечения признаков.<sup>[1]</sup> Центральное положение занимают методы, основанные на машинном обучении (так уж сейчас сложилось). Такие методы получают на вход специальные структуры данных — чаще всего это вектора или матрицы ([тензоры](#)). Иногда алгоритмы принимают на вход графы или деревья. Зачем рассматривать вообще задачу извлечения признаков отдельно от конечной задачи? На самом деле на это есть множество причин. Первая заключается в том, что значительная часть методов извлечения признаков не требует разметки. Следовательно, мы можем применить

методы к большим неразмеченным [корпусам](#), получить хорошие признаки, а потом уже работать с небольшой размеченной выборкой. С другой стороны, возможно, получится извлечь признаки один раз, а потом поверх них накручивать классификаторы для решения разных задач. А вообще бывает, что можно получить неплохое представление, не используя машинного обучения вообще. В этой лекции мы рассмотрим несколько самых популярных методов по мере их усложнения. Для каждого подхода мы выделим основные преимущества и недостатки. В первую очередь, рассмотрим [разреженное векторное представление](#), популярное в классических подходах, а затем перейдём к более сложным нейросетевым и [ядерным подходам](#). Это, в первую очередь — обзор, призванный расширить кругозор и дать правильную терминологию для дальнейшего самостоятельного изучения. Простейший метод — это двоичный вектор. Элементы вектора соответствуют отдельным словам. Элемент равен "1", например вот здесь, если слово в документе присутствует, и "0", если нет. Это очень простой метод. Он подходит, в том числе, тогда, когда тексты сильно отличаются по длине. Размерность векторного пространства, получаемого таким образом, достаточно большая, и поэтому почти любые классы линейно разделимы, и [линейные модели](#) хорошо работают на таких пространствах. Однако главная проблема заключается в том, что и частотные слова общепотребимые, и специальная лексика, имеют одинаковый вес, то есть мы не знаем, как много используется каждое слово в документе. По этой же причине метод чувствителен к опечаткам, случайным словам, и так далее. Общая проблема [разреженных векторных моделей](#) — в так называемом "предположении о независимости". Элементы вектора, соответствующие разным словам заполняются независимо, и таким образом мы теряем информацию о том, что какие-то два слова являются синонимами и если одно встретилось, то, возможно, может встретиться и другое. Из-за этого модель может не очень хорошо обобщаться на новые данные. И это происходит не из-за [переобучения](#), а из-за того, что просто признаки не очень хорошие, в них просто нет такой информации. В то время как высокая размерность вектора может рассматриваться как преимущество, она может также являться и недостатком, потому что если обучающая выборка не очень большая, а признаков при таком подходе получается очень много — сотни тысяч, мы можем переобучиться даже если работаем с простой линейной моделью.

[1] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. <https://nlp.stanford.edu/IR-book/>

Список литературы:

Извлечение признаков:

- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. <https://nlp.stanford.edu/IR-book/>
- Извлечение признаков в Scikit-Learn [https://scikit-learn.org/stable/modules/feature\\_extraction.html](https://scikit-learn.org/stable/modules/feature_extraction.html)
- Извлечение признаков без разметки [https://en.wikipedia.org/wiki/Feature\\_learning](https://en.wikipedia.org/wiki/Feature_learning)

- Линейные модели [https://en.wikipedia.org/wiki/Linear\\_model](https://en.wikipedia.org/wiki/Linear_model)
- Переобучение <https://en.wikipedia.org/wiki/Overfitting>

Многие задачи решаются классификаторами, основанными на машинном обучении.

Такие классификаторы принимают на вход объекты, представленные в определённом виде:

- вектора или матрицы (тензоры)
- графы (в том числе деревья)

Зачем отделять извлечение признаков от основной задачи?

- не нужна разметка – можно использовать гораздо больше данных
- одни и те же признаки могут быть хороши для нескольких задач
- можно выделять признаки без машинного обучения вообще



- Двоичный вектор, описывающий встречаемость слов в документе
- Вектор вещественных чисел, описывающий встречаемость слов в документе, с учетом их частотности
- N-граммы символов и токенов, словосочетания
- Плотные векторные представления слов, предложений и текстов (word embeddings, doc embeddings)
- Ядерные методы (kernel methods) и графовые ядра



"Днём мама мыла раму. Вечером мы пошли гулять."

и	мы	мама	велосипед	...	рама
0	1	1	0	...	1

Преимущества:

- простота
- подходит, когда тексты сильно отличаются по длине
- размерность вектора очень большая  $\Rightarrow$  простые линейные модели хорошо работают

Недостатки:

- значимость (вес) специальных слов такая же, как и частотных общеупотребимых
- чувствителен к шуму
- близкие по смыслу слова ("кружка" и "чашка") кодируются не зависящими друг от друга элементами вектора  $\Rightarrow$  может страдать обобщающая способность
- размерность вектора очень большая  $\Rightarrow$  на небольшой обучающей выборке высока вероятность переобучения



Ну хорошо. Логичный следующий шаг — это исправить самый главный недостаток бинарного вектора — кодировать не нулями и единичками, а как-то взвесить. Самый простой источник для взвешивания — это частотность слова, причём для каждого слова мы можем посчитать не только частоту его использования в одном документе, но и частоту его использования в [корпусе](#) в целом, а потом их как-то скомбинировать (например, с помощью формулы [TF-IDF](#) — мы поговорим о ней подробнее чуть позже). Итак, это по-прежнему достаточно простая модель. Если перед вами стоит задача тематической классификации длинных текстов, то этот подход очень хорошо будет работать и вполне возможно, что вы не сможете улучшить качество классификации, используя сложные методы (например, нейросети). По-прежнему мы сохраняем основные преимущества предыдущего подхода, то есть линейную разделимость данных, а также мы можем играться с весами, искусственно завышив веса для каких-то слов, или занизив — таким образом, закладывая в модель наши знания о том, что должно быть важно для принятия решения, а что — нет. Как всегда, у метода есть и свои недостатки. TF-IDF чувствителен к опечаткам в обучающей выборке и другим редким словам, которые, тем не менее, достаточно частотные, чтобы не быть отфильтрованными. Остальные недостатки предыдущего подхода, пока что, сохраняются — это предположение о независимости словоупотреблений и слишком большая размерность вектора.

Список литературы:

Извлечение признаков (метод - вектор вещественных чисел, описывающий встречаемость слов в документе, с учетом их частотности):

- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. <https://nlp.stanford.edu/IR-book/>
- TF-IDF <https://ru.wikipedia.org/wiki/TF-IDF>
- Закон Ципфа [https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%BE%D0%BD\\_%D0%A6%D0%B8%D0%BF%D1%84%D0%B0](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%BE%D0%BD_%D0%A6%D0%B8%D0%BF%D1%84%D0%B0)

Вопрос:

разве взвешенные по частоте в каких-то ситуациях проигрывают бинарным векторам?

Ответ (автора):

у меня такое было на паре датасетов, но в целом это зависит от особенностей задачи - иногда большая частота некоторого токена или н-грамм не говорит о том, что он более важный. Если датасет большой и на каждый класс достаточно примеров, это не будет играть роли, а если датасет недостаточный (но другого нет), то вполне себе играет.

"Днём мама мыла раму. Вечером мы пошли гулять."

и	мы	мама	велосипед	...	рама
0	0.01	0.1	0	...	2

Term Frequency - Inverse Document Frequency (TF-IDF)

Преимущества:

- простота
- очень хорошо описывает особенности тематики документов
- размерность вектора очень большая  $\Rightarrow$  простые линейные модели хорошо работают
- легко заложить в модель знания о предметной области (экспертное мнение)

Недостатки:

- чувствителен к шуму
- близкие по смыслу слова ("кружка" и "чашка") кодируются не зависящими друг от друга элементами вектора  $\Rightarrow$  может страдать обобщающая способность
- размерность вектора очень большая  $\Rightarrow$  на небольшой обучающей выборке высока вероятность переобучения



Отлично! Со взвешиванием разобрались. Но мы по-прежнему в векторе считаем просто слова. Иногда выгоднее считать не слова, а N-граммы, то есть последовательности подряд идущих символов или токенов. N-граммы бывают символьными, а бывают пословными. В символьных N-граммах мы просто берём 3 (или сколько-то) подряд идущих символа (как, например, здесь). А в пословных N-граммах, соответственно, то же самое, только со словами. N-граммы — это достаточно популярный трюк, который используется много где, в том числе с TF-IDF, а также в современных методах построения плотных векторных представлений и

дистрибутивной семантики — например, в модели [FastText](#) (речь о FastText пойдёт чуть позже в этом курсе). Итак, использование N-грамм не сильно усложняет модель. Кроме того, мы получаем какую-то устойчивость к опечаткам, а также словоизменению, то есть мы можем до какой-то степени обойтись без исправления опечаток, а также без сложных алгоритмов нормализации текста — например, лемматизации и морфологического анализа. Пословные N-граммы — более специфичны по сравнению с отдельными словами, то есть они встречаются реже, но при этом (если они встречаются) являются более сильным фактором. И поэтому они могут лучше описывать особенности тематики текстов. Как всегда, недостатки подхода следуют из его преимуществ. Размерность пространства растёт очень быстро, а вектора получаются очень разреженными. Чем больше N, тем реже соответствующая N-грамма встречается. Ну, и опять, мы никак не избавились от недостатков, присущих предыдущим векторным моделям. Хорошая новость том, что вектора можно сжать. Например, если у нас есть гигантская матрица, описывающая встречаемость слов в документах (например, вот эта), то мы её можем факторизовать, то есть представить в виде произведения двух матриц меньшей размерности, меньшего ранга. Например, первая матрица будет представлять документы в некотором латентном пространстве, а вторая матрица будет представлять слова в том же самом пространстве. Кроме [матричного разложения](#) используются предиктивные модели дистрибутивной семантики<sup>[1]</sup>, которые обучаются предсказывать соседние слова (в текстах) для данного слова.

[1] Дистрибутивно-семантические модели для русского языка: <https://rusvectores.org/ru/>

Список литературы:

Извлечение признаков (метод - N-граммы символов и токенов, словосочетания):

- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press. 2008. <https://nlp.stanford.edu/IR-book/>
- N-граммы <https://ru.wikipedia.org/wiki/N-%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0>
- Разложение матриц [https://en.wikipedia.org/wiki/Matrix\\_decomposition](https://en.wikipedia.org/wiki/Matrix_decomposition)

Вопрос:

латентные пространства - это о чём? очень нужен пример

Ответ (автора):

по поводу латентных пространств:

Есть переменные наблюдаемые, а есть скрытые (их ещё называют латентными). Наблюдаемые переменные - это те переменные, для вычисления которых не надо строить какую-то особую модель. В случае с текстами к таким переменным относятся, например, частоты слов, которые в тексте встречаются. К латентным переменным можно отнести, например, тематики (когда текст на 50% про котиков, на 30% про психологию и ещё на 20% про что-то).

Другими словами, латентные пространства (или пространство латентных переменных) – это когда мы пытаемся описать объекты (тексты) с помощью каких-то переменных (признаков), которые не вычисляются непосредственно из текста, а требуют построения специальной модели (тематической модели, нейросети и т.п.).

Тематические модели и факторизация матриц в общем выходят за рамки этого курса (мы только поговорим про Word2Vec и FastText позже), по этой теме есть [видео](#).

N-грамма – объект, состоящий из нескольких подряд идущих базовых элементов  
"Днём мама мыла раму. Вечером мы пошли гулять."

SAMSUNG  
Research  
Russia

- Символьные 3-граммы: днё, нём\_, \_м\_м, \_ма, мам
- Словные 2-граммы: день мама, мама мыть, мыть рама

Применяется:

- ① вместе с TF-IDF
- ② в дистрибутивной семантике, FastText word embeddings

Преимущества:

- простота
- работа с флексивными языками без полноценной морфологии и POS-теггинга
- более специфические признаки, чем отдельные слова

Недостатки:

- высокая размерность и разреженность
- близкие по смыслу слова ("кружка" и "чашка") кодируются не зависящими друг от друга элементами вектора  $\Rightarrow$  может страдать обобщающая способность
- размерность вектора очень большая  $\Rightarrow$  на небольшой обучающей выборке высока вероятность переобучения



## Латентные векторные представления (embeddings)

SAMSUNG  
Research  
Russia

Вектора можно сжать!



- Матричные разложения и тематическое моделирование (SVD, pLSA, LDA, ARTM)
- Предиктивные дистрибутивно-семантические модели (Word2Vec, FastText)
- Предиктивные модели текста (language model - BERT, ELMo, OpenAI Transformer)



Более сложный и мощный класс — языковые модели. Они предсказывают следующее слово по известному префиксу предложения, и, как правило, реализуются глубокими нейронными сетями. Хорошая новость в том, что, несмотря на то, что методы — обучаемые, разметка им не требуется. Такие методы называются методами с самонаблюдением (self-supervised), то есть целевая переменная берётся из самих анализируемых объектов — например, это просто следующее слово. То есть, руками размечать ничего не надо, и мы можем автоматически обработать гигантские объёмы текста. Методы из этой группы, особенно — основанные на языковых моделях, позволяют достичь наилучшего качества на некоторых сложных задачах. Наконец-то мы отвергли предположение о независимости словаупотреблений и наконец-то можем начать работать с синонимами. Так как в основе этих методов лежит оптимизация, мы можем настраивать веса моделей с учётом сразу нескольких критериев, таким образом делая признаки полезными сразу для нескольких задач. Это очень популярное сейчас направление — оно называется "multitask learning". Но за все надо платить — эти модели гораздо дороже с точки зрения объёмов вычислений, особенно нейросетевые модели. И другая сложность заключается в том, что им нужны гигантские объёмы текста. Если у вас меньше нескольких миллионов документов, обучить с нуля качественную языковую модель у вас, скорее всего, не получится, однако сейчас в интернете доступны веса моделей, уже обученных на некоторых больших [корпусах](#), их можно просто скачать и использовать в своём проекте (это здорово).

Список литературы:

Извлечение признаков (метод - Плотные векторные представления слов, предложений и текстов (word embeddings, doc embeddings):

- Дистрибутивно-семантические модели для русского языка <https://rusvectores.org/ru/>
- Предобученные языковые модели для нескольких языков <http://docs.deeppavlov.ai/en/master/features/models/bert.html>
- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. <https://nlp.stanford.edu/IR-book>



- Матричные разложения и тематическое моделирование (SVD, pLSA, LDA, ARTM)
- Предиктивные дистрибутивно-семантические модели (Word2Vec, FastText)
- Предиктивные модели текста (language model - BERT, ELMo, OpenAI Transformer)

Преимущества:

- размеченная выборка не требуется
- state-of-the-art качество для некоторых задач
- позволяет учитывать совместную встречаемость слов и отношения их смыслов
- multitask learning - настройка представлений с учетом сразу нескольких прикладных задач

Недостатки:

- намного дороже с вычислительной точки зрения
- требуются гигантские массивы текстов для обучения



Надо помнить, что правило 80/20 никто не отменял:<sup>[1]</sup> если вам надо быстро получить хоть какое-то решение, возможно, нет смысла заморачиваться с языковыми моделями или нейросетями вообще и лучше начать с векторной модели и [линейных классификаторов](#), в части случаев этого будет вполне достаточно. Другой недостаток в том, что процесс обучения не всегда стабилен. Может потребоваться перебор гиперпараметров или несколько запусков. Когда мы работаем с векторами, у нас есть миллион способов оценить их сходство. Один из таких способов — это [скалярное произведение](#). Физический смысл скалярного произведения, как вы все, конечно, знаете — это косинус угла между двумя векторами. Однако, когда мы работаем с текстами, в этом есть небольшая проблема. Текст — это сложная нелинейная структура, и если мы её представляем вектором, мы неизбежно теряем часть информации. OK, вполне естественная идея — представить текст как граф или семантическую сеть. Отлично, теперь дело за малым — нужно определить хорошую операцию сходства на графах. Такую операцию принято называть "ядром". Например, мы можем запускать некоторое количество случайных обходов в этих графах, а потом смотреть, насколько обходы, полученные по разным графикам, пересекаются друг с другом: чем больше пересекаются, тем больше графы похожи друг на друга. Или ядро можно выучить (с помощью нейросети, например). Когда мы нашли подходящее ядро, мы можем использовать относительно простые модели — например, метод опорных векторов<sup>[2]</sup> или поиск ближайших соседей<sup>[3]</sup>. Хорошая новость в том, что, даже не имея разметки, мы можем получить достаточно выразительный способ представления текстов. Для некоторых сложных задач — например тех, в которых размечать большие [корпуса](#) достаточно дорого, [ядерные методы](#) позволяют получить наилучшее на сегодняшний день

качество. К таким областям, например, относится извлечение информации из медицинских текстов. Посредством ядра мы можем заложить в модель наши экспертные знания о предметной области, тем самым ещё сильнее сокращая требования к обучающей выборке. К сожалению, ядерные методы, с вычислительной точки зрения, иногда даже медленнее нейросетевых, так как они хуже распараллеливаются с помощью графических ускорителей. Ну и, вполне логично, модель будет ровно настолько хорошей, насколько хорошее ядро вы придумали.<sup>[4]</sup>

[1] Закон Парето или правило 80/20:

[https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%BE%D0%BD\\_%D0%9F%D0%B0%D1%80%D0%B5%D1%82%D0%BE](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%BE%D0%BD_%D0%9F%D0%B0%D1%80%D0%B5%D1%82%D0%BE)

[2] [SVM с нелинейным ядром](#)

[3] [Задача поиска ближайшего соседа](#)

[4] Kriege, Nils M., Fredrik D. Johansson, and Christopher Morris. "A Survey on Graph Kernels." arXiv preprint arXiv:1903.11835 (2019). <https://arxiv.org/abs/1903.11835>

Список литературы:

Извлечение признаков (метод - Ядерные методы (kernel methods) и графовые ядра):

- Закон Парето или правило 80/20 [https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%BE%D0%BD\\_%D0%9F%D0%B0%D1%80%D0%B5%D1%82%D0%BE](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%BE%D0%BD_%D0%9F%D0%B0%D1%80%D0%B5%D1%82%D0%BE)
- Ядерные методы [https://en.wikipedia.org/wiki/Kernel\\_method](https://en.wikipedia.org/wiki/Kernel_method)
- SVM с нелинейным ядром <https://towardsdatascience.com/understanding-support-vector-machine-part-2-kernel-trick-mercers-theorem-e1e6848c6c4d>
- Kriege, Nils M., Fredrik D. Johansson, and Christopher Morris. "A Survey on Graph Kernels." arXiv preprint arXiv:1903.11835 (2019). <https://arxiv.org/abs/1903.11835>

- Матричные разложения и тематическое моделирование (SVD, pLSA, LDA, ARTM)
- Предиктивные дистрибутивно-семантические модели (Word2Vec, FastText)
- Предиктивные модели текста (language model - BERT, ELMo, OpenAI Transformer)

Преимущества:

- размеченная выборка не требуется
- state-of-the-art качество для некоторых задач
- позволяет учитывать совместную встречаемость слов и отношения их смыслов
- multitask learning - настройка представлений с учетом сразу нескольких прикладных задач

Недостатки:

- намного дороже с вычислительной точки зрения
- требуются гигантские массивы текстов для обучения
- для некоторых задач нет существенного преимущества перед более простым частотным векторным представлением
- сложнее управлять процессом обучения



Скалярное произведение - оценка сходства двух векторов -  $\langle a, b \rangle = \frac{\sum_i^N a_i b_i}{\sqrt{\sum_i^N a_i^2} \sqrt{\sum_i^N b_i^2}}$

Однако текст - нелинейная многосвязная структура, его крайне сложно закодировать одним вектором, ничего не потеряв!



Можно переопределить операцию скалярного произведения так, чтобы она работала на графах.

- случайные обходы различной длины
- обучаемые ядра (tree recursive neural network)

Когда ядро определено, можно применять простые классификаторы - Support Vector Machine, Kernel Nearest Neighbor и т.п.

Преимущества:

- размеченная выборка не требуется
- state-of-the-art качество для некоторых задач
- позволяет учитывать очень сложные отношения слов
- позволяет абстрагироваться от порядка слов
- есть возможность заложить в модель знания о предметной области

Недостатки:

- дорого с вычислительной точки зрения



## 1.6 Прикладные задачи обработки текста и итоги

Всем привет! В этот раз мы поговорим о прикладных задачах. К прикладным задачам мы отнесли те, которые можно так или иначе положить в основу продукта. Самая популярная

задача — это классификация.<sup>[2]</sup> Следующая крупная группа методов — поиск по разным видам запросов. Извлечение структурированной информации<sup>[1]</sup> — важная задача, связанная с наполнением баз данных, автоматизацией, переносом компетенций, процессом принятия решений... Ещё две задачи связаны с переводом текста в текст, а именно — диалоговые системы и машинный перевод. Немного в стороне стоит эксплоративный анализ — задача без чётких критериев качества, когда надо понять, что вообще происходит. Итак, самая базовая задача — это тематическая классификация длинных текстов, для которых простираются золотые метки. Длина текстов должна позволять набрать хоть какую-то статистику. Природа меток такова — они зависят от состава текста в целом, а не от отдельных фраз или формулировок. Самый подходящий алгоритм для этой задачи — это [линейный классификатор](#) с разреженными признаками, взвешенными по частоте<sup>[3]</sup> (например, [TF-IDF](#)). Нейросети тоже можно применять, но они часто не дают большого прироста в качестве. Когда мы имеем дело с короткими текстами, ситуация несколько меняется: частоты слов уже не подсчитать, да и метки чаще связаны со структурой фраз, а не с тематикой. Например, по составу слов в этом предложении достаточно сложно определить его эмоциональную окраску, однако если мы рассмотрим связи тональных слов с теми словами, которые они определяют, станет понятно, что предложение, в целом, положительное. И вот тут нейросети работают уже существенно лучше — особенно, когда есть большой размеченный [корпус](#). Если меток не очень много, [ядерные методы](#) могут сработать лучше. А если же разметки нет вообще, то, возможно, проще запрограммировать правила, анализирующие структуру предложений<sup>[4]</sup>, обходящие синтаксические деревья или семантические сети<sup>[5]</sup> и принимающее решения.

[1] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. <https://nlp.stanford.edu/IR-book/>

[2] Список статей с кодом по тематике классификации текстов

<https://paperswithcode.com/task/text-classification>

[3] [https://en.wikipedia.org/wiki/Bag-of-words\\_model](https://en.wikipedia.org/wiki/Bag-of-words_model)

[4] Лексико-синтаксические шаблоны в системе GATE

<https://gate.ac.uk/sale/tao/splitch8.html#chap:jape>

[5] Лексико-синтаксические шаблоны Томита-парсера <https://github.com/yandex/tomita-parser/blob/master/docs/ru/tutorial/basic-rules.md>

Список литературы:

Прикладные задачи обработки текста (классификация):

- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. <https://nlp.stanford.edu/IR-book/>
- [https://en.wikipedia.org/wiki/Bag-of-words\\_model](https://en.wikipedia.org/wiki/Bag-of-words_model)
- <https://ru.wikipedia.org/wiki/TF-IDF>
- Линейные модели, логистическая регрессия [https://en.wikipedia.org/wiki/Logistic\\_regression](https://en.wikipedia.org/wiki/Logistic_regression)

- Линейные модели, SVM [https://en.wikipedia.org/wiki/Support-vector\\_machine](https://en.wikipedia.org/wiki/Support-vector_machine)
- Обобщённые линейные модели [https://en.wikipedia.org/wiki/Generalized\\_linear\\_model](https://en.wikipedia.org/wiki/Generalized_linear_model)
- Список статей с кодом по тематике классификации текстов <https://paperswithcode.com/task/text-classification>
- Лексико-синтаксические шаблоны в системе GATE <https://gate.ac.uk/sale/tao/splitch8.html#chap:jape>
- Лексико-синтаксические шаблоны Томита-парсера <https://github.com/yandex/tomita-parser/blob/master/docs/ru/tutorial>

- Классификация
  - Тематическая классификация длинных текстов
  - Классификация коротких текстов (по тональности, интенции)
- 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
- Эксплоративный анализ



#### Дано

- Коллекция текстов с проставленными метками.
- Каждый текст состоит хотя бы из нескольких предложений.
- Метки определяются составом текста в целом, а не отдельными фразами и их структурой.

#### Необходимо получить

- Алгоритм, предсказывающий эти метки.

#### Механизмы

- Линейные модели классификации, векторное представление, TF-IDF



- Каждый текст - одно или два предложения.
- Для текстов простираются метки.
- Метки определяются не столько составом текста в целом, сколько структурой фраз.

"Готовился к плохому, в итоге получилось хорошо."

### Необходимо получить

- Алгоритм, предсказывающий эти метки.

### Механизмы

- Нейросети
- Ядерные методы
- Системы правил (лексико-синтаксические шаблоны)



Перейдём к поисковым задачам. Самый популярный вид поиска — по запросу. При этом, запрос не обязательно является связным текстом на каком-то языке, чаще это просто набор ключевых слов. На входе имеем коллекцию текстов. Будет большим плюсом, если уже накоплена какая-то статистика посещения пользователей, по которой мы можем обучить оценку [релевантности](#). По сути, алгоритм должен сравнивать два текста по содержанию: короткий запрос и длинный ответ. Система должна работать даже тогда, когда статистика ещё не накоплена. Базовый подход — это [векторная модель](#) текста вместе с классическими формулами оценки релевантности — как правило, это всё основано на частотах слов. Следующее логичное развитие — это расширить частотную модель с помощью дистрибутивной семантики, то есть "[эмбеддингами](#)" слов (наподобие [word2vec](#)). Можно также использовать глубокий лингвистический анализ и графовые ядра, чтобы точнее сопоставлять тексты. Когда накапливается статистика посещения пользователей, появляется возможность применять обучаемые методы ранжирования (на английском эта задача называлася "learning to rank"). Самые популярные алгоритмы для ранжирования — это градиентный бустинг и нейросети. Более сложная постановка задачи поиска — поиск между модальностями, например, когда мы ищем картинку по тексту, или наоборот<sup>[1]</sup>. В этом случае исходные данные — это коллекция мультимедийных документов, состоящая из текста и иллюстраций, например — веб-страницы. Исторически первый подход работал через текстовый поиск, ключевой момент здесь — это определять, какой именно текст на странице описывает изображение. Тогда, если мы ищем по тексту, то мы сначала находим страницу, а потом, по положению текста и картинок, понимаем, какая картинка найденному тексту соответствует.<sup>[2]</sup> Если же

ищем по картинке, то — наоборот, сначала ищем ближайшую картинку, а потом со страницы берём ближайший к ней текст. Однако сейчас гораздо чаще используются нейросетевые архитектуры, получающие общее [векторное представление](#) — и для картинок, и для текста<sup>[3]</sup> — в этом случае у нас есть возможность искать похожие объекты в этом векторном пространстве<sup>[4]</sup>. Иногда мы не хотим руками составлять поисковый запрос, но у нас есть какой-то документ, который нам понравился — например, статья по психологии. Так как я не специалист по психологии, я не могу составить грамотный поисковый запрос, чтобы поискать похожие статьи на эту же тему. Почему бы тогда не использовать прямо этот документ, как запрос? На помощь приходит "поиск похожих документов".<sup>[5]</sup> В этой области достаточно хорошо работают классические подходы с [TF-IDF](#) и [N-граммами](#). Однако, как мы уже обсуждали, такие модели предполагают, что частоты словоупотреблений независимы, поэтому мы можем терять полноту из-за того, что не учитываем какие-то синонимы. Альтернативный подход — через тематическое моделирование.<sup>[6]</sup> Гигантская матрица размерности ["количество документов" на "количество слов"] факторизуется на две матрицы поменьше, и мы используем только первую из них для поиска документов. Этот механизм обладает большей полнотой, но уступает по точности первому подходу, так как при факторизации мы теряем информацию о специфических, достаточно редких словах.

- [1] Frome, Andrea, et al. "Devise: A deep visual-semantic embedding model." *Advances in neural information processing systems*. 2013.
- [2] Feng, Yansong, and Mirella Lapata. "Automatic image annotation using auxiliary text information." *Proceedings of ACL-08: HLT*. 2008.
- [3] Wang, Liwei, Yin Li, and Svetlana Lazebnik. "Learning deep structure-preserving image-text embeddings." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- [4] Gomaa, Wael H., and Aly A. Fahmy. "A survey of text similarity approaches." *International Journal of Computer Applications* 68.13 (2013): 13-18.
- [5] Zubarev, Denis, and Ilya Sochenkov. "Using Sentence Similarity Measure for Plagiarism Source Retrieval." *CLEF (Working Notes)*. 2014.
- [6] Ianina, Anastasia, Lev Golitsyn, and Konstantin Vorontsov. "Multi-objective topic modeling for exploratory search in tech news." *Conference on Artificial Intelligence and Natural Language*. Springer, Cham, 2017.

#### Список литературы:

Прикладные задачи обработки текста (поиск: по запросу, по изображению и изображений по текстам, похожих текстов):

- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press. 2008. <https://nlp.stanford.edu/IR-book/>
- Feng, Yansong, and Mirella Lapata. "Automatic image annotation using auxiliary text information." *Proceedings of ACL-08: HLT*. 2008.

- Wang, Liwei, Yin Li, and Svetlana Lazebnik. "Learning deep structure-preserving image-text embeddings." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- Frome, Andrea, et al. "Devise: A deep visual-semantic embedding model." *Advances in neural information processing systems*. 2013.
- Gomaa, Wael H., and Aly A. Fahmy. "A survey of text similarity approaches." *International Journal of Computer Applications* 68.13 (2013): 13-18.
- Ianina, Anastasia, Lev Golitsyn, and Konstantin Vorontsov. "Multi-objective topic modeling for exploratory search in tech news." *Conference on Artificial Intelligence and Natural Language*. Springer, Cham, 2017.
- Zubarev, Denis, and Ilya Sochenkov. "Using Sentence Similarity Measure for Plagiarism Source Retrieval." *CLEF (Working Notes)*. 2014.

Доп. информация из практического задания:

Процесс решения задачи поиска часто состоит из двух крупных шагов, каждый из которых разбивается на более мелкие:

1. Настройка поиска
  1. преобразование объектов (например, текстов) в вещественные вектора
  2. построение поискового индекса
  3. настройка функции ранжирования
2. Выполнение поиска
  1. преобразование запроса в вещественный вектор
  2. грубая выборка кандидатов
  3. сортировка кандидатов с помощью функции ранжирования

Поисковый индекс - набор специальных структур данных, ускоряющих процесс поиска. Так как процесс индексации тоже требует времени, поисковые индексы не всегда имеет смысл строить - например, когда данных мало или данные часто меняются и индекс устаревает быстрее, чем может быть перестроен.

Настройка функции ранжирования выполняется с помощью набора примеров вида "запрос - документ - оценка релевантности по мнению человека". Релевантность - численная величина, характеризующая соответствие найденного документа запросу (чем больше, тем лучше документ подходит под запрос).

Дано

- Коллекция текстов
- Статистика поиска - запросы и ссылки, по которым пользователи в итоге перешли

SAMSUNG  
Research  
Russia

Необходимо получить

- Алгоритм, который запросу в виде короткого текста (не длиннее одного предложения) будет находить релевантные документы
- Холодный старт

Механизмы

- TF-IDF + формулы вычисления релевантности BM25 и др.
- Дистрибутивно-семантические модели
- Лингвистический анализ и специальные алгоритмы для сопоставления структуры текстов
- Обучаемое ранжирование (learning to rank) на основе градиентного бустинга или нейросетей



SAMSUNG  
Research  
Russia

Дано

- Коллекция текстов со вставленными в них изображениями (HTML-страниц)

Необходимо получить

- Алгоритм, находящий изображения по тексту или наоборот

Механизмы

- Поиск через окружающий текст (не по содержимому изображения)
- Нейросети



## Дано

- Коллекция текстов без разметки
- Статистика поиска

## Необходимо получить

- Алгоритм, по запросу в виде текста (длиннее одного предложения) находящий близкие по содержанию или тематике документы

## Механизмы

- TF-IDF + векторная модель текста, N-граммы
- плотные векторные представления, тематическое моделирование
- сиамские нейросети (siamese neural networks)



Когда есть размеченная выборка или статистика посещений пользователя, есть возможность что-то обучить (например, градиентный бустинг или нейросеть). Сейчас постепенно набирают обороты попытки создать более человеческий интерфейс — например, человек может не задавать поисковый запрос как набор ключевых слов, вместо этого — писать текст на ему известном языке, задавать вопрос на [естественном языке](#)<sup>[1]</sup>. Например, "кто победил на прошлом чемпионате мира по футболу"? И система должна выдать непосредственно ответ на этот вопрос: например, "Франция". Для решения этой задачи нам нужна коллекция текстов, в которых мы будем искать ответы, а также желательно иметь обучающую выборку, то есть набор пар "вопрос" и "текст, вместе с отметками — где в этом тексте находится ответ".<sup>[2]</sup> Современные алгоритмы предназначены для поиска дословных ответов, то есть они не делают логический вывод. Если ответ встречается в тексте — они его выделяют, если нужно немного подумать — то лучше ничего не выделять. Майнстрим в этой области — нейросети.<sup>[4,5]</sup> Для них сейчас есть достаточно большие размеченные [корпуса](#). Однако не стоит забывать и про классические методы<sup>[3]</sup> — глубокий лингвистический анализ и алгоритмы сопоставления графов. В ряде случаев они позволяют получить сопоставимое качество без обучающей выборки.

[1] Chen, Danqi, et al. "Reading Wikipedia to Answer Open-Domain Questions." Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017.

[2] Rajpurkar, Pranav, et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text." Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016.

- [3] Gliozzo, Alfio, et al. "Semantic technologies in IBM watson." Proceedings of the fourth workshop on teaching NLP and CL. 2013.
- [4] Shelmanov, A. O., et al. "Semantic-syntactic analysis for question answering and definition extraction." Scientific and Technical Information Processing 44.6 (2017): 412-423.
- [5] Ferrucci, David, et al. "Building Watson: An overview of the DeepQA project." AI magazine 31.3 (2010): 59-79.

Список литературы:

Прикладные задачи обработки текста (поиск: вопросно-ответный поиск):

- Chen, Danqi, et al. "Reading Wikipedia to Answer Open-Domain Questions." *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017.
- Rajpurkar, Pranav, et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text." *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016.
- Ferrucci, David, et al. "Building Watson: An overview of the DeepQA project." *AI magazine* 31.3 (2010): 59-79.
- Gliozzo, Alfio, et al. "Semantic technologies in IBM watson." *Proceedings of the fourth workshop on teaching NLP and CL*. 2013.
- Shelmanov, A. O., et al. "Semantic-syntactic analysis for question answering and definition extraction." *Scientific and Technical Information Processing* 44.6 (2017): 412-423.

The diagram illustrates the Question Answering (QA) process through three main components:

- Дано** (Given):
  - Коллекция текстов
  - Коллекция вопросов, на которые есть ответы в текстах
- Необходимо получить** (Need to get):
  - Алгоритм, находящий ответ на вопрос в тексте.
  - Ответ должен быть дословным, то есть алгоритм не обязан делать логический вывод.
- Механизмы** (Mechanisms):
  - Нейросети
  - Лингвистический анализ и специальные алгоритмы сопоставления графов
  - Ядерные методы

**SAMSUNG Research Russia**

A man in a dark t-shirt with a graphic design stands to the right of the diagram.

Для того, чтобы компьютерная поддержка принятия решений работала и вообще имела смысл, необходимо составлять достаточно полные и актуальные базы фактов. Так уж сложилось, что в некоторых предметных областях значительная часть информации по-прежнему не публикуется в виде, пригодном для удобной обработки компьютерами, а публикуется, например, просто в текстах — например, новости. На помощь приходят методы извлечения структурированной информации. На вход методам даётся, в первую очередь, структура той информации, которую надо извлечь, а именно — набор полей и набор связей между ними. Это как схема базы данных. А также дан набор текстов, в которых факты точно содержатся, но представлены в неструктурированном виде. Пример такой области — это новости, или медицинские карты, в которых врач простым текстом описывает, что происходит, какое лечение, и так далее.<sup>[1]</sup> Соответственно, методы извлечения информации заполняют поля значениями, найденными в тексте.<sup>[2]</sup> Нам здесь подходят методы [извлечения именованных сущностей](#) и связей между ними — о них мы немножко говорили в разделе про лингвистический анализ. В основе таких методов часто лежат системы правил, сопоставление со словарями, лингвистический анализ, а также нейросети, но только когда есть достаточно большие размеченные коллекции текстов — от тысячи примеров на каждую сущность.

[3,4,5,6,7,8,9,10]

- [1] Peng, Fuchun, and Andrew McCallum. "Information extraction from research papers using conditional random fields." *Information processing & management* 42.4 (2006): 963-979.
- [2] Surdeanu, Mihai, et al. "Stanford's distantly-supervised slot-filling system." (2011).
- [3] Kelly, Liadh, et al. "Overview of the share/clef ehealth evaluation lab 2014." *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, Cham, 2014.
- [4] Mann, Gideon S., and David Yarowsky. "Multi-field information extraction and cross-document fusion." *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2005.
- [5] Ratner, Alexander J., et al. "Snorkel: Fast training set generation for information extraction." *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 2017.
- [6] Dernoncourt, Franck, Ji Young Lee, and Peter Szolovits. "NeuroNER: an easy-to-use program for named-entity recognition based on neural networks." *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2017.
- [7] Dalvi, Bhavana, et al. "IKE—an interactive tool for knowledge extraction." *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*. 2016.
- [8] Banko, Michele, et al. "Open information extraction from the web." *Ijcai*. Vol. 7. 2007
- [9] Huang, Zhiheng, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging." *arXiv preprint arXiv:1508.01991* (2015).
- [10] Arkhipov, Mikhail Y., and Mikhail S. Burtsev. "Application of a hybrid Bi-LSTM-CRF Model to the task of Russian named entity recognition." *Conference on Artificial Intelligence and Natural Language*. Springer, Cham, 2017.

## Список литературы:

Прикладные задачи обработки текста (извлечение структурированной информации):

- Kelly, Liadh, et al. "Overview of the share/clef ehealth evaluation lab 2014." *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, Cham, 2014.
- Surdeanu, Mihai, et al. "Stanford's distantly-supervised slot-filling system." (2011).
- Peng, Fuchun, and Andrew McCallum. "Information extraction from research papers using conditional random fields." *Information processing & management* 42.4 (2006): 963-979.
- Mann, Gideon S., and David Yarowsky. "Multi-field information extraction and cross-document fusion." *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2005.
- Ratner, Alexander J., et al. "Snorkel: Fast training set generation for information extraction." *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 2017.
- Dernoncourt, Franck, Ji Young Lee, and Peter Szolovits. "NeuroNER: an easy-to-use program for named-entity recognition based on neural networks." *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2017.
- Dalvi, Bhavana, et al. "IKE-an interactive tool for knowledge extraction." *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*. 2016.
- Banko, Michele, et al. "Open information extraction from the web." *Ijcai*. Vol. 7. 2007
- Huang, Zhiheng, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging." *arXiv preprint arXiv:1508.01991* (2015).
- Arkhipov, Mikhail Y., and Mikhail S. Burtsev. "Application of a hybrid Bi-LSTM-CRF Model to the task of Russian named entity recognition." *Conference on Artificial Intelligence and Natural Language*. Springer, Cham, 2017.

- Структура информации - набор полей и связей между ними
- Тексты, в которых есть нужная информация, но она находится в неструктурированном виде.
- Электронные медицинские карты пациентов - извлечение симптомов, их частоты, предыдущих заболеваний.

## Необходимо получить

- Заполнить поля значениями из текста в соответствии с заданной структурой.

## Механизмы

- Системы правил, сопоставление со словарями
- Лингвистический анализ и ядерные методы
- Нейросети в меньшей степени (из-за недостатка больших размеченных коллекций текстов)



Как же не поговорить о самой горячей теме последних лет: чат-ботах.<sup>[1,2,3,4,5,6]</sup> Чат-бот — это не какой-то один алгоритм, это набор из различных алгоритмов, выполняющих разные функции, но работающих вместе. Как правило, известен желаемый сценарий взаимодействия с пользователем, он может быть описан как явно, так и через примеры диалогов.<sup>[7,8]</sup> Цель общения может быть как определена, так и быть свободной, равно как и предметная область может быть ограниченной, а может быть открытой. Для построения чат-бота требуется сразу несколько алгоритмов для разных функций. Одна из таких функций — это определение интенции, то есть намерения пользователя: что он хочет от системы, с которой взаимодействует. Здесь подходят методы классификации коротких текстов. Также мы можем выдавать ответы посредством поиска похожих текстов по базе, ну, и конечно же нейросети здесь применяются — как для отдельных подзадач, так и для реализации чат-бота в рамках единого обучаемого алгоритма. Но это пока что на уровне эксперимента, скорее. Другая задача, связанная с генерацией текстов — это машинный перевод. Чаще всего, перевод выполняется на уровне фраз или отдельных предложений. Для настройки алгоритма используются как параллельные корпуса (тогда для каждого предложения на исходном языке известен хотя бы один вариант его перевода на целевом языке), так и непараллельные корпуса. Центральное место в машинном переводе сейчас занимают нейросети. Ранее использовались (а кое-где и сейчас используются) алгоритмы статистического перевода на уровне отдельных фраз.

- [1] Chen, Hongshen, et al. "A survey on dialogue systems: Recent advances and new frontiers." *Acm Sigkdd Explorations Newsletter* 19.2 (2017): 25-35.
- [2] Serban, Iulian V., et al. "Building end-to-end dialogue systems using generative hierarchical neural network models." *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.
- [3] Chuklin, Aleksandr, et al. "Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI." *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*. 2018.
- [4] Dinan, Emily, et al. "The second conversational intelligence challenge (convai2)." *arXiv preprint arXiv:1902.00098* (2019).
- [5] Burtsev, Mikhail, et al. "DeepPavlov: open-source library for dialogue systems." *Proceedings of ACL 2018, System Demonstrations*. 2018.
- [6] Gupta, Arshit, John Hewitt, and Katrin Kirchhoff. "Simple, Fast, Accurate Intent Classification and Slot Labeling." *arXiv preprint arXiv:1903.08268* (2019).
- [7] Serban, Iulian Vlad, et al. "A survey of available corpora for building data-driven dialogue systems: The journal version." *Dialogue & Discourse* 9.1 (2018): 1-49.
- [8] Lowe, Ryan Thomas, et al. "Training end-to-end dialogue systems with the ubuntu dialogue corpus." *Dialogue & Discourse* 8.1 (2017): 31-65.

Список литературы:

Прикладные задачи обработки текста (диалоговые системы/чат-боты, машинный перевод):

- Chen, Hongshen, et al. "A survey on dialogue systems: Recent advances and new frontiers." *Acm Sigkdd Explorations Newsletter* 19.2 (2017): 25-35.
- Serban, Iulian Vlad, et al. "A survey of available corpora for building data-driven dialogue systems: The journal version." *Dialogue & Discourse* 9.1 (2018): 1-49.
- Serban, Iulian V., et al. "Building end-to-end dialogue systems using generative hierarchical neural network models." *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.
- Chuklin, Aleksandr, et al. "Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI." *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*. 2018.
- Dinan, Emily, et al. "The second conversational intelligence challenge (convai2)." *arXiv preprint arXiv:1902.00098* (2019).
- Lowe, Ryan Thomas, et al. "Training end-to-end dialogue systems with the ubuntu dialogue corpus." *Dialogue & Discourse* 8.1 (2017): 31-65.
- Burtsev, Mikhail, et al. "DeepPavlov: open-source library for dialogue systems." *Proceedings of ACL 2018, System Demonstrations*. 2018.
- Gupta, Arshit, John Hewitt, and Katrin Kirchhoff. "Simple, Fast, Accurate Intent Classification and Slot Labeling." *arXiv preprint arXiv:1903.08268* (2019).

- Сценарий взаимодействия с пользователем
- Примеры диалогов
- Цель общения может быть задана (goal oriented) или отсутствовать (chit-chat)
- Домен (тематика) может быть открытым или закрытым

## Необходимо получить

- Алгоритм, взаимодействующий с пользователем посредством диалога на ЕЯ.

## Механизмы

- Алгоритмы классификации коротких текстов для, например, определения интенции.
- Алгоритмы поиска похожих текстов для выбора ответа.
- Нейросети - end-to-end диалоговые системы.



- Параллельный корпус - пары предложений на разных языках, являющиеся переводом друг друга.
- Непараллельные корпуса

## Необходимо получить

- Алгоритм, генерирующий предложение на целевом языке, по предложению на исходном языке.



## Механизмы

- Нейросети
- Алгоритмы статистического машинного перевода

Особняком стоит задача эксплоративного анализа.<sup>[1,2,3]</sup> Она заключается в том, что нам дают коллекцию текстов и просят ответить на вопрос — "а что здесь вообще происходит?" То есть нужно понять, какие тематики есть, как они пересекаются и как меняются во времени. Рабочая лошадка для таких задач — это методы тематического моделирования. Самые популярные методы — это латентное размещение Дирихле или тематические модели с

аддитивной регуляризацией. Мы пробежались по основным прикладным задачам обработки текстов на [естественном языке](#), а именно: классификации, поиску, извлечению информации, генерации текстов, эксплоративному анализу. На этом мы завершаем наш обзор задач обработки естественного языка.

[1] Вопросно-ответный поиск – описание соревнования и обзор решений, 2016

<https://www.aclweb.org/anthology/S16-1083.pdf>

[2] Поиск похожих предложений, 2017 <https://www.aclweb.org/anthology/S17-2028.pdf>

[3] Извлечение отношений, 2018 <https://www.aclweb.org/anthology/S18-1130.pdf>

Список литературы:

Прикладные задачи обработки текста (эксплоративный анализ):

- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press. 2008. <https://nlp.stanford.edu/IR-book/>
- O'Connor, Brendan, Michel Krieger, and David Ahn. "Tweetmotif: Exploratory search and topic summarization for twitter." *Fourth International AAAI Conference on Weblogs and Social Media*. 2010.
- Соченков, Илья Владимирович, Денис Владимирович Зубарев, and Илья Александрович Тихомиров. "Эксплоративный патентный поиск." *Информатика и её применения* 12.1 (2018): 89-94.
- Medlar, Alan, et al. "Pulp: A system for exploratory search of scientific literature." *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2016.
- Ianina, Anastasia, Lev Golitsyn, and Konstantin Vorontsov. "Multi-objective topic modeling for exploratory search in tech news." *Conference on Artificial Intelligence and Natural Language*. Springer, Cham, 2017.

Дано

- Коллекция текстов

Необходимо получить

- Знание, о чём идёт речь в этой коллекции
- Структура тематик, как они пересекаются и как развиваются со временем

Механизмы

- Тематический анализ (LDA, ARTM)



Вы посмотрели первый модуль курса по введению в обработку естественного языка с помощью нейросетей. Основной фокус в этом модуле был сделан на процесс и различные задачи обработки естественного языка, а также на место нейросетей в текущем состоянии нашей предметной области. Мы рассмотрели, чем разные языки отличаются друг от друга и как это влияет на методы их анализа, а именно — флективность (или изменчивость) словоформ, омонимия (или неоднозначность) — когда одна и та же словоформа может соответствовать разным словам с разным смыслом, вариативность в порядке слов — то, насколько перестановка слов предложений влияет на его смысл и корректность. А ещё мы поговорили о том, что делают с текстами. Лингвистический анализ направлен на выявление структуры текстов на разных уровнях, от разбиения на отдельные слова и предложения до семантики и дискурса. Методы извлечения признаков отвечают за преобразование текста в форму, пригодную для применения методов машинного обучения. А также мы рассмотрели несколько прикладных задач. Давайте попробуем вспомнить, какие подходы, классы алгоритмов, упоминались в настоящем модуле. Первая группа — это почти ручные методы, — это правила, регулярные выражения, словари. Вторая группа — это так называемые классические методы машинного обучения, то есть линейные модели, ядерные методы, факторизация матриц. Ну, и наконец, конечно же, — то, зачем мы здесь собрались — это нейросети. Важно помнить, что для каждой задачи всегда есть альтернативные варианты решения со своими преимуществами и недостатками. Не нейросетями единими живёт наша предметная область, хотя, конечно, они представляют наибольший интерес. Уверен, что вам, как и мне, не терпится скорее перейти к практике. Тогда — вперёд!

- Какие особенности есть у разных языков и как это влияет на методы их анализа?
- Флективность - изменение слов в зависимости от ситуации
- Омонимия - разные слова могут выглядеть одинаково в разных ситуациях
- Порядок слов - можно или нельзя переставлять слова местами и как это влияет на смысл предложения



- Системы правил, регулярные выражения, словари
- "Классическое" машинное обучение - линейные модели, ядерные методы, матричные факторизации
- Нейросети

Для каждой задачи всегда есть разные подходы,  
у каждого есть свои плюсы и минусы.

