

# Введение в текстовую аналитику



# Русаков Алексей

[vk.com/RusAlm](https://vk.com/RusAlm)   [RusAl@bk.ru](mailto:RusAl@bk.ru)

# Ссылка на материалы

- ▶ [github.com/RusAl84/text\\_analitic](https://github.com/RusAl84/text_analitic)
- ▶ [https://vk.com/text\\_analitic](https://vk.com/text_analitic)
- ▶ [http://gg.gg/text\\_analitic](http://gg.gg/text_analitic)

# Список литературы

- ▶ Jurafsky D., Martin J. H. Speech and language processing (draft) <https://web.stanford.edu/~jurafsky/slp3/>
- ▶ Митренина О. В., Николаев И. С., Ландо Т. М. Прикладная и компьютерная лингвистика. - 2016. - 320 с.
- ▶ Бенгфорт Б., Билбро Р., Охеда Т. Прикладной анализ текстовых данных на Python // Машинное обучение и создание приложений обработки естественного языка. СПб.: Питер. - 2019. - 368 с.
- ▶ **NLPub** — каталог ресурсов для обработки естественного языка. <https://nlpub.ru/>
- ▶ <https://www.kaggle.com/>
- ▶ <https://openai.com/>



**Системы  
мгновенного обмена  
сообщениями**



**Отчеты**



**Социальные сети**



**Документы**

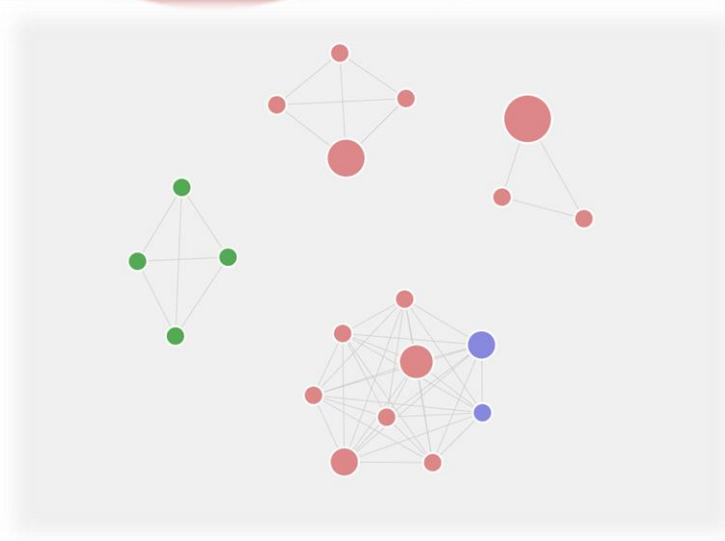
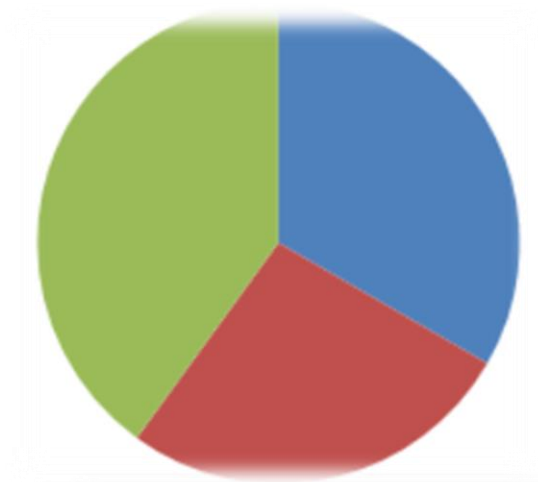


Выявление: таблица									
Номер	Код	Дата	Имя	Пол	Возраст	Рост	Вес	Описание	Статус
1	1	06.11.2008	Савченко	муж	до 45 лет	15 000 кг	50	официальное удостоверение, служебный транспорт	открыто
2	2	20.11.2008	Савченко	муж	от 20 до 50 лет	12 000 кг	50	официальное удостоверение, служебный транспорт	открыто
3	2	20.11.2008	Савченко	муж	до 40 лет	16 000 кг	50	официальное удостоверение, служебный транспорт	открыто
4	4	25.11.2008	Литовский	муж	возможно без опыта, обучен	15 000 кг	50	официальное удостоверение, служебный транспорт	открыто
5	4	26.11.2008	Литовский	муж	от 20 до 45 лет, опыт	9 000 кг	50	официальное удостоверение, служебный транспорт	открыто
6	1	31.11.2008	Савченко	муж	до 45 лет	15 000 кг	50	официальное удостоверение, служебный транспорт	открыто
7	3	02.12.2008	Савченко	муж	до 55 лет	8 500 кг	50	официальное удостоверение, служебный транспорт	открыто
8	5	04.12.2008	Савченко	муж	до 40 лет, опыт от 2 лет	15 000 кг	202	официальное удостоверение, служебный транспорт	открыто
9	6	04.12.2008	Савченко	муж	до 30 лет, высшее образование	25 000 кг	50	официальное удостоверение, служебный транспорт	открыто
10	7	05.12.2008	Савченко	муж	до 45 лет, опыт	15 000 кг	50	официальное удостоверение, служебный транспорт	открыто
11	7	04.12.2008	Савченко	муж	до 40 лет	14 000 кг	50	официальное удостоверение, служебный транспорт	открыто
12	8	05.12.2008	Савченко	муж	до 40 лет, опыт от 2 лет	12 000 кг	50	официальное удостоверение, служебный транспорт	открыто
13	10	06.12.2008	Савченко	муж	до 55 лет	8 000 кг	173	официальное удостоверение, служебный транспорт	открыто
14	1	06.12.2008	Савченко	муж	до 45 лет, опыт, высшее образование	12 000 кг	50	официальное удостоверение, служебный транспорт	открыто
15	5	06.12.2008	Савченко	муж	до 45 лет	3 500 кг	50	официальное удостоверение, служебный транспорт	открыто

10	Егоров Сергей Геннадьевич	РФ
11	Ежов Станислав Станиславович	Украина
12	Еремич Николай Николаевич	Украина
13	Жидких Аркадий Юрьевич	РФ
14	Кимаковский Игорь Владимирович	РФ
15	Ковалис Ольга Валерьевна	РФ
16	Кореньковский Дмитрий Евстафьевич	Украина
17	Костенко Андрей Сергеевич	Украина
18	Лазарев Сергей Александрович	Украина
19	Ломачко Юрий Николаевич	Украина
20	Мельничук Петр Николаевич	Молдова
21	Мельников Евгений Игоревич	РФ
22	Одинцов Максим Евгеньевич	РФ
23	Пикалов Валерий Валерьевич	Украина
24	Прозорова Юлия Вадимовна	Украина
25	Ракушин Александр Сергеевич	Украина
26	Родионова Антонина Петровна	Украина
27	Саттаров Александр Валерьевич	Украина
28	Седиков Алексей Сергеевич	РФ
29	Сидоров Денис Владимирович	РФ
30	Тарасенко Александр Сергеевич	Украина
31	Федоров Виктор Андреевич	Украина
32	Хитров Денис Васильевич	Украина
33	Хоменко Олег Юрьевич	Украина
34	Цемах Владимир Борисович	Украина

4	Bean Goose	Anser fabalis	Metsahanhi
5	Pink-footed Goose	Anser brachyrhynchus	Lyhytnokkahanhi
6	Greater White-fronted Goose	Anser albifrons	Tundrahanhi
7	Lesser White-fronted Goose	Anser erythropus	Kiljuhanhi
8	Greylag Goose	Anser anser	Merihanhi
9	Snow Goose	Anser caerulescens	Lumihanhi
10	Canada Goose	Branta canadensis	Kanadanhanhi (Cat. C)
11	Barnacle Goose	Branta leucopsis	Valkoposkikanhi
12	Brent Goose	Branta bernicla	Sepelhanhi
13	Red-breasted Goose	Branta ruficollis	Punakaulahanhi
14	Ruddy Shelduck	Tadoma ferruginea	Ruostesorsa
15	Common Shelduck	Tadoma tadoma	Ristisorsa
16	Mandarin Duck	Aix galericulata	Mandariinisorsa (Cat. C)
17	Eurasian Wigeon	Anas penelope	Haapana
18	American Wigeon	Anas americana	Amerikanhaapana
19	Gadwall	Anas strepera	Hamaasorsa
20	Common Teal	Anas crecca	Tavi
21	Green-winged Teal	Anas carolinensis	Amerikantavi
22	Mallard	Anas platyrhynchos	Sinisorsa
23	American Black Duck	Anas rubripes	Nokisorsa
24	Northern Pintail	Anas acuta	Jouhisorsa
25	Garganey	Anas querquedula	Heinatavi

Таблицы и списки  
(ненаглядно и  
неудобно)



**Визуализация**



# ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА

## ЗАДАЧИ:

- ▶ машинный перевод - это область вычислительной лингвистики, что исследует применение программного обеспечения для перевода текста с одного языка на другой. В простом приближении это простая замена слов одного языка на другой;
- ▶ вопрос-ответные системы - системы, которые могут отвечать на вопросы. Для поиска ответа, к примеру, используется Википедия. По входному вопросу одна нейронная сеть сужает список статей, в которых искать, а другая уже делает подробный поиск ответа;
- ▶ распознавание речи - это область вычислительной лингвистики, которая изучает применение технологий для распознавания разговорного языка и перевода его в текст;
- ▶ классификация текстов - определение тем документов. Важная задача в доменных областях;

# ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА

## ЗАДАЧИ:

- ▶ извлечение фактов - процесс анализа документов на основе множества теорий и технологий. Извлечение объектов, связи между этими объектами и в конечном итоге в извлечении необходимых фактов;
- ▶ диалоговые системы - это компьютерные системы, предназначенные для общения с человеком. Эти системы состоят из множества компонентов: распознавания речи, жестов, понимания языка (понимание языка уже сама по себе комплексная задача), выдача заготовленного или сгенерированного ответа, перевод в разговорный язык или на язык жестов;
- ▶ анализ тональности текста - определение эмоциональной окраски, настроения текста;



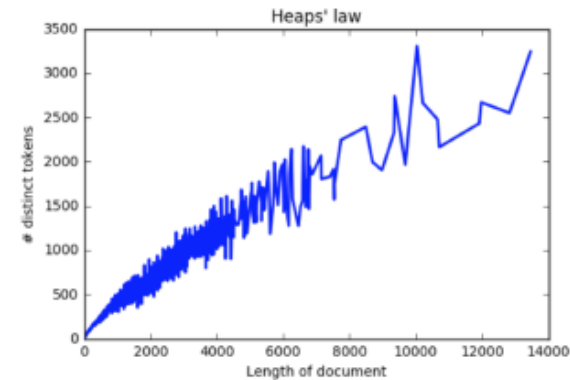
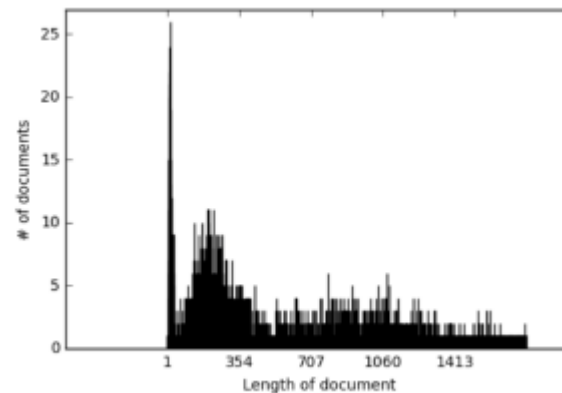
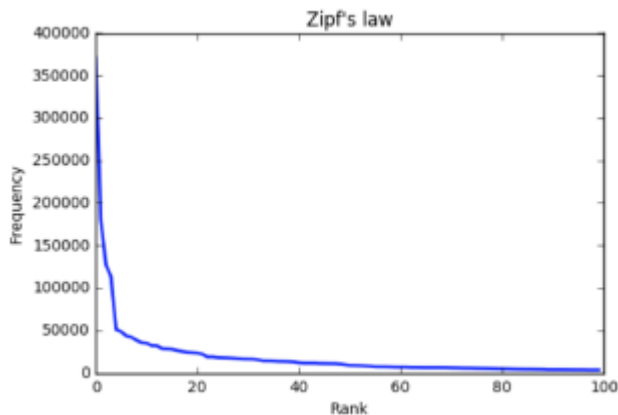
# ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА

## ЗАДАЧИ:

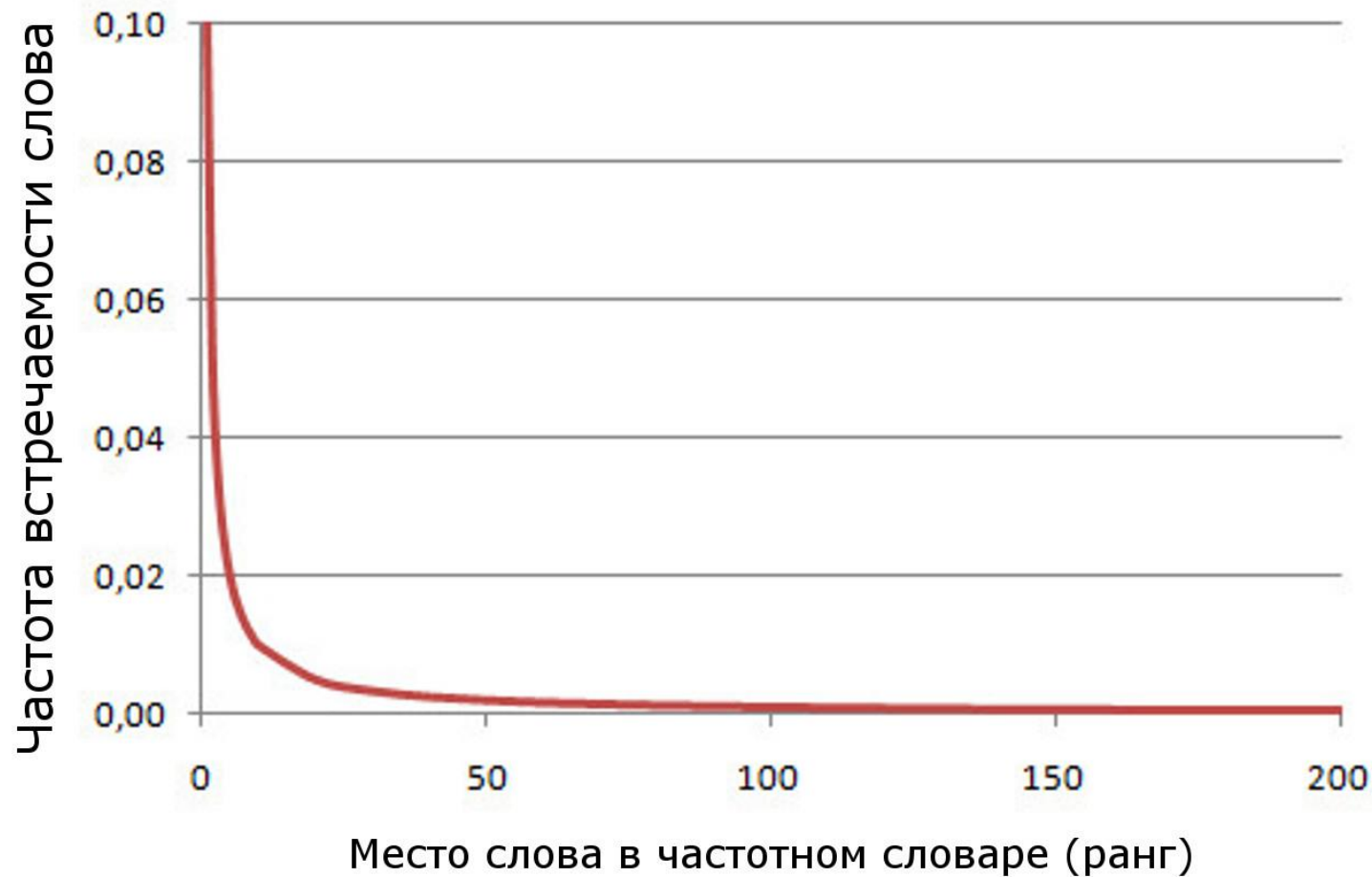
- ▶ суммаризация текстов - процесс сокращения текстов, извлечения только важной информации;
- ▶ topic modeling - определение списка тем из большой коллекции документов. Каждая тема представляется распределением слов в ней;
- ▶ разрешение кореферентности - определение связей между объектами и компонентами высказывания;
- ▶ разрешение лексической многозначности - определение смысла слова в зависимости от контекста;
- ▶ определение частей речи;
- ▶ синтаксический разбор;
- ▶ приведение слова в начальную форму;

# Статистические свойства коллекций

- ▶ Перед тем, как анализировать коллекцию текстов, необходимо узнать её статистические свойства
- ▶ Каждая задача требует подсчёта специфичных ей величин.
- ▶ Есть характеристики, на которые нужно смотреть всегда:
  - ▶ Частоты слов в коллекции, закон Ципфа
  - ▶ Гистограмма длин документов
  - ▶ Закон Хипса для документов

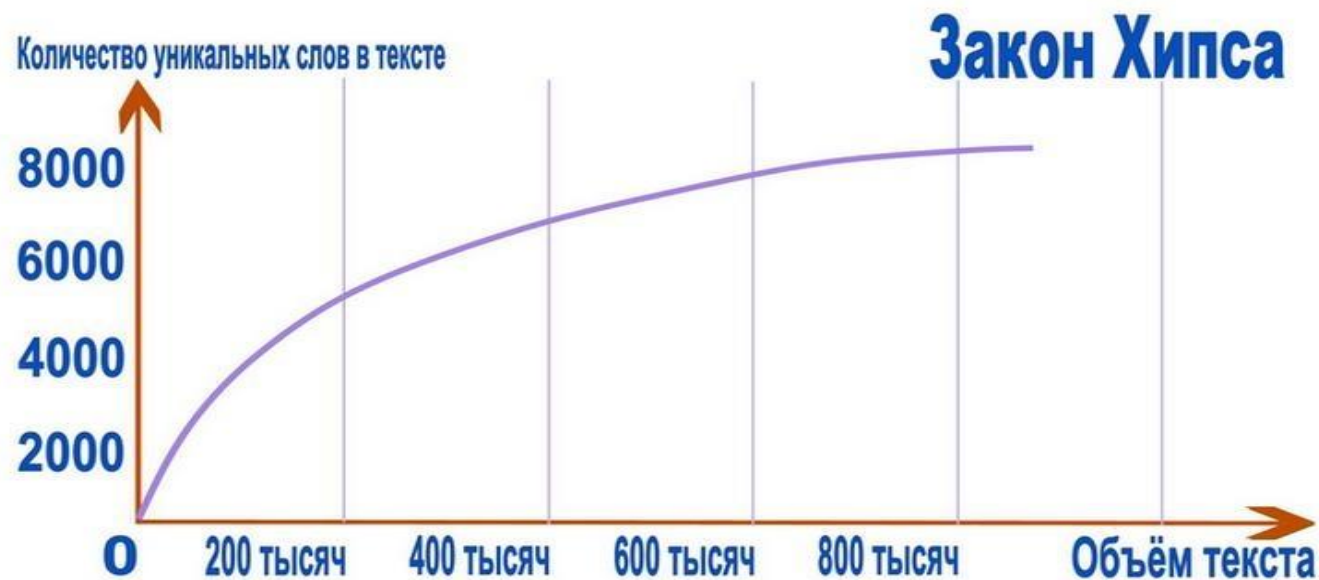


# Закон Ціпфа



**Закон Ціпфа** («ранг—частота») — емпірическая закономерность распределения частоты слов естественного языка: если все слова языка (или просто достаточно длинного текста) упорядочить по убыванию частоты их использования, то частота  $n$ -го слова в таком списке окажется приблизительно обратной пропорциональной его порядковому номеру  $n$  (так называемому рангу этого слова, см. шкала порядка).

# Закон Хипса для документов



$$v_R(n) = Kn^\beta$$

где  $v$  — это объем словаря уникальных слов, составленный из текста, который состоит из  $n$  уникальных слов,  
 $\alpha$  и  $\beta$  — определенные эмпирически параметры.  
Для европейских языков  $\alpha$  принимает значение от 10 до 100,  
а  $\beta$  — от 0.4 до 0.6.

# Подход TF\*IDF

Произведение  $TF*IDF$  определяет уровень соответствия документа запросу.

Множитель  $TF$  – прямая частота вхождения слова в документ (отвечает за встречаемость термина в содержании документа)

Множитель  $IDF$  – обратная частота термина в коллекции (отвечает за редкость употребления запроса во всех документах коллекции)

# Классический случай подхода TF\*IDF

$$TF = \frac{n_i}{\sum_k n_k}$$

где  $n_i$  - количество употреблений  $i$ -й N-граммы, знаменатель – общая длина документа в словах

$$IDF = \log \frac{D}{DF_i}$$

где  $D$  – общее количество документов в коллекции,  
знаменатель - число документов, содержащих  $i$ -й N-грамму

# Предобработка текста

- ▶ Первый шаг любой аналитики - получение данных.  
Предположим, что данные есть в некотором подходящем для работы формате.
- ▶ Следующая задача - предобработка
- ▶ Базовые шаги предобработки:
  1. токенизация
  2. приведение к нижнему регистру
  3. удаление стоп-слов
  4. удаление пунктуации
  5. фильтрация по частоте/длине/соответствию регулярному выражению
  6. лемматизация или стемминг
- ▶ Чаще всего применяются все эти шаги, но в разных задачах какие-то могут опускаться, поскольку приводят к потере информации.



# Полезные модули

- ▶ **nltk** — один из основных модулей Python для анализа текстов, содержит множество инструментов.
- ▶ **re/regex** — модули для работы с регулярными выражениями
- ▶ **pymorphy2/pymystem3** — лемматизаторы
- ▶ Специализированные модули для обучения моделей (например, CRF)
- ▶ **numpy/pandas/scipy** — модули общего назначения
- ▶ **Scikit-learn** - самый распространенный выбор для решения задач классического машинного обучения. Использовали для TF-IDF
- ▶ **GENSIM** - библиотека обработки естественного языка, предназначенная для «Тематического моделирования».
- ▶ **FastText** -это библиотека, содержащая предобученные готовые векторные представления слов (читать, что это такое) и классификатор, то есть алгоритм машинного обучения, разбивающий слова на классы.
- ▶ **Word2vec** -Библиотека word2vec - это инструмент (набор алгоритмов) для расчета векторных представлений слов, реализует две основные архитектуры - Continuous Bag of Words (CBOW) и Skip-gram. На вход подается корпус текста, а на выходе получается набор векторов слов.

# Токенизация, удаление стоп-слов и пунктуации

- ▶ В nltk есть разные токенизаторы:

- ▶ RegexpTokenizer
- ▶ BlanklineTokenizer
- ▶ И ещё около десятка штук

- ▶ Стоп-слова тоже можно удалять с помощью nltk (но лучше дополнительно фильтровать вручную):

```
from nltk.corpus import stopwords  
words = [word for word in word_list  
          if word not in stopwords.words('russian')]
```

- ▶ Пунктуацию можно удалять с помощью регулярных выражений, а можно просто:

```
from string import punctuation  
s = ''.join(c for c in s if c not in punctuation)
```

# Регулярные выражения

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

# Пример из жизни

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

`.*онлайн-|–трансляция.*`

`.*в эт(у?и?) минут.*`

`.*в эт((от)?и?) час.*`

`.*в этот день.*`

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

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

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

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

# СТЭММИНГ И ЛЕММАТИЗАЦИЯ

- ▶ **Стэмминг** — процесс приведения слова к основе (отрезание окончания и формообразующего суффикса), грубо, но быстро.
  - [Porter stemmer](#)
  - [Snowball stemmer](#)
  - [Lancaster stemmer](#)
- ▶ **Лемматизация** — процесс приведения слова к нормальной форме, качественно, но долго.
  - **py morphology2** (язык русский, украинский)
  - **mystem3** (язык русский, английский?)
  - **Wordnet Lemmatizer** (NLTK, язык английский, требует POS метку)
  - **Metaphraz** (язык русский)
  - **Coda/Cadenza** (языки русский и английский)

# Пример лемматизации

```
import pymorphy2

text_ru = u'Где твоя ложка, папа?'
pymorph = pymorphy2.MorphAnalyzer()

for word in text_ru.split(u' '):
    if re.match(u'([^\a-zA-яе\`]+)', word):
        word = pymorph.parse(word)[0].normal_form
    print(word)
```

## Вывод:

где твой ложка , папа ?

**ЕЩЁ пример???** [github.com/RusAl84/text\\_analitic/blob/master/lem\\_test.py](https://github.com/RusAl84/text_analitic/blob/master/lem_test.py)

- **ВАЗЕЛИН** - мазеобразная белая жидкость без запаха и вкуса. При неполной очистке цвет меняется от чёрного до жёлтого, при полной - до полупрозрачного.
- **вазелин** мазеобразный белый жидкость без запах и вкус при неполный очистка цвет меняться от черный до желтый при полный до полупрозрачный

**ТЕПЕРЬ ВСЕМ ПОНЯТНО???**

# TextRank

► TextRank – приложение алгоритма PageRank к задачам NLP:

- ❑ Строим граф на основе исходного текста  
     $V$  – вершины (слова)  
     $E$  – рёбра (связи)

- ❑ Вычисляем веса вершин по мерам центральности, например, по *PageRank*:

$$PR(V_i) = (1 - d) + d \sum_{V_j \in In(V_i)} \frac{PR(V_j)}{Out(V_j)}$$

- ❑ Извлекаем цепочки с наибольшими весами.



# Описание TextRank

- ▶ В качестве  $V$  можно взять все уникальные леммы текста (допустимо ограничиться прилагательными и существительными: термины в основном являются именными группами).
- ▶ Сканируем текст с окном из  $N \in [2, 10]$  слов.
- ▶ На каждой итерации считаем для пары слов величину связи

$$WC(w_1, w_2) = \begin{cases} 1 - \frac{d(w_1, w_2) - 1}{N - 1}, & \text{if } d(w_1, w_2) \in (0, N), \\ 0, & \text{if } d(w_1, w_2) \geq N, \end{cases}$$

где  $w_i$  - слова,  $d(w_1, w_2)$  - расстояние между ними (можно просто взять модуль разности позиций).

- ▶ Основание подобной связи - между двумя рядом стоящими словами часто существует семантическое отношение.

# Описание TextRank

- ▶ Ранжируем вершины графа на основании значения TextRank, получаемого случайным блужданием для каждой вершины  $t \in V$ :

$$WC(w_1, w_2) = \begin{cases} 1 - \frac{d(w_1, w_2) - 1}{N - 1}, & \text{if } d(w_1, w_2) \in (0, N), \\ 0, & \text{if } d(w_1, w_2) \geq N, \end{cases}$$

где  $d$  – фактор затухания,  $In(t)$  – вершины, входящие в  $t$ ,  $Out(t)$  – выходящие из  $t$ ,  $w_{ij}$  – вес соответствующего ребра.

- ▶ Упорядочиваем вершины по TR и отбираем  $T$  самых лучших (например,  $T = 1/3 |V|$ ). Это множество кандидатов  $C$ .
- ▶ Извлекаем из текста все последовательности слов, состоящие из элементов множества  $C$

## Важно:

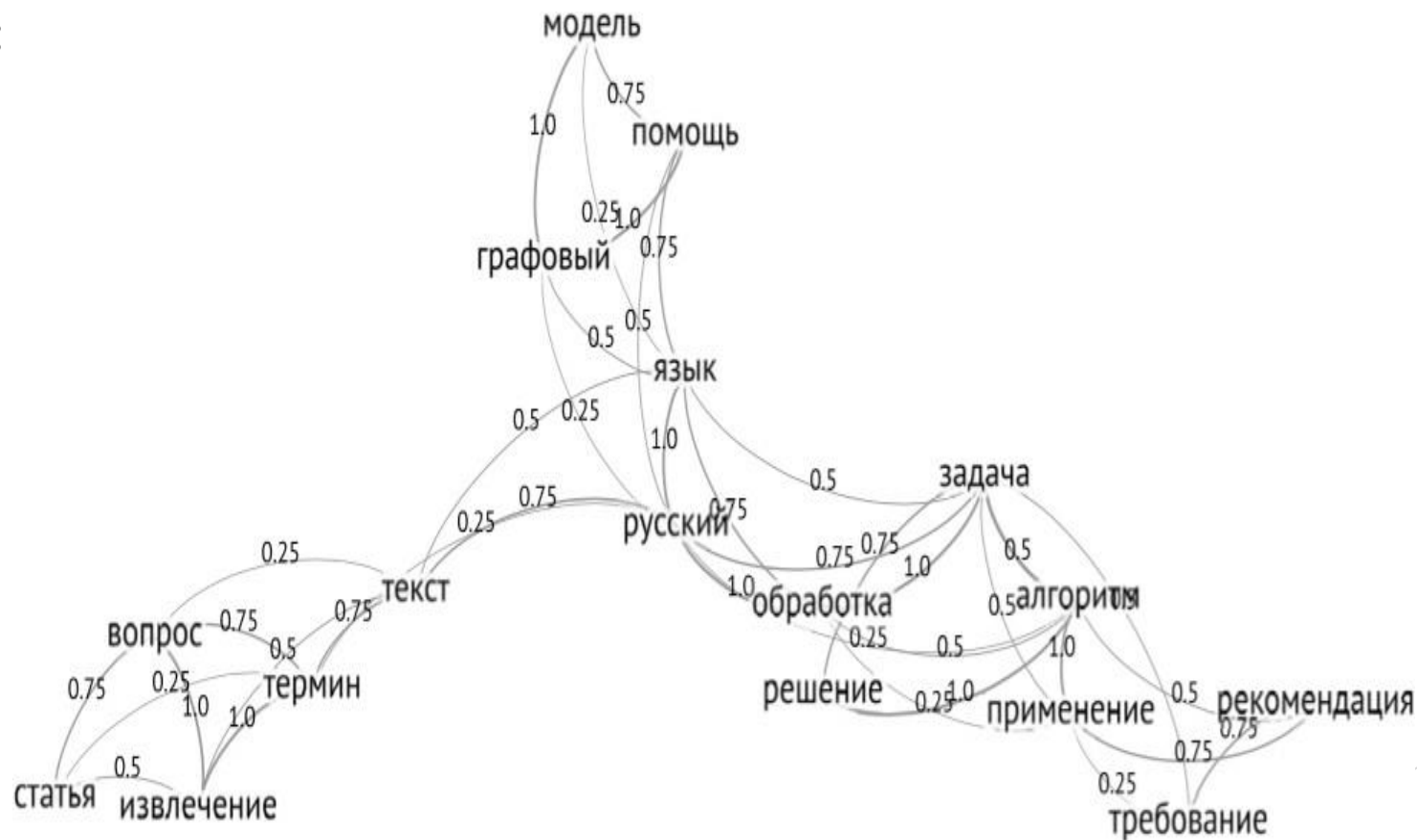
1. в последовательности должно быть хоть одно существительное;
2. в случае вложенности надо рассматривать только последовательность с большим весом

# TextRank: пример

► Текст:

Статья посвящена вопросу извлечения терминов из текстов на русском языке при помощи графовых моделей экспериментально исследован алгоритм решения данной задачи. Сформулированы требования и рекомендации к применению алгоритма в задачах обработки русского языка.

► **Граф:**



# TextRank: пример

## Множество кандидатов:

- задача – 0,094
- русский – 0,084
- алгоритм – 0,083
- язык – 0,076
- извлечение – 0,064
- обработка – 0,063
- термин – 0,062
- вопрос – 0,060

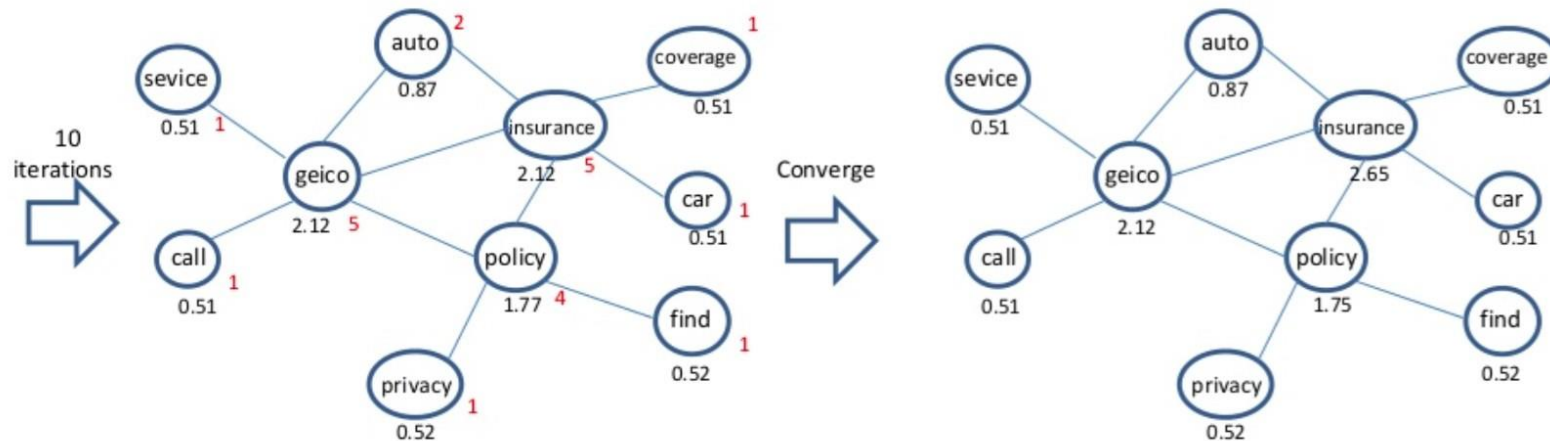
## Выделенные словосочетания:

Статья посвящена *вопросу извлечения терминов* из текстов на русском языке при помощи графовых моделей экспериментально исследован **алгоритм** решения данной **задачи**. Сформулированы требования и рекомендации к применению *алгоритма* в *задачах обработки русского языка*.

# TextRank: ещё пример

$$S(V_i) = (1 - d) + d * \sum_{j \in \text{nbr}(V_i)} \frac{1}{|\text{degree}(V_j)|} S(V_j)$$

$d$  is the damping factor  
that usually set to 0.85



**Converge Really Quick!**  
( $\leq 20$  iterations)

$$\text{Score}(\text{geico}) = 0.15 + 0.85 * \left( \underbrace{\frac{1}{1} * 0.51}_{\text{service}} + \underbrace{\frac{1}{1} * 0.51}_{\text{call}} + \underbrace{\frac{1}{2} * 0.87}_{\text{auto}} + \underbrace{\frac{1}{5} * 2.12}_{\text{insurance}} + \underbrace{\frac{1}{4} * 1.77}_{\text{policy}} \right) = 2.12$$

$$\text{Score}(\text{policy}) = 0.15 + 0.85 * \left( \underbrace{\frac{1}{1} * 0.52}_{\text{find}} + \underbrace{\frac{1}{1} * 0.52}_{\text{privacy}} + \underbrace{\frac{1}{5} * 2.12}_{\text{insurance}} + \underbrace{\frac{1}{5} * 2.12}_{\text{geico}} \right) = 1.75$$

$$\text{Score}(\text{service}) = 0.15 + 0.85 * \left( \underbrace{\frac{1}{5} * 2.12}_{\text{geico}} \right) = 0.51$$

# TextRank в Gensim

## ► Результат (примеры):

- дуров
- telegram пока
- роскомнадзор
- дать компания реестр
- заявление
- канал мессенджер
- возражать против
- блокировка
- россия создатель

[https://github.com/RusAl84/text\\_analitic/tree/master/DZ\\_3%20CompareKWE](https://github.com/RusAl84/text_analitic/tree/master/DZ_3%20CompareKWE)

# RAKE

- ▶ *RAKE (Rapid Automatic Keyword Extraction):*
  - ▶ Фразы-кандидаты – все слова между разделителями.
  - ▶ Некоторым образом производится оценка фразы.
  - ▶ Фраза-кандидат ограничивается по частоте и количеству слов.
  - ▶ Модули: rake\_nltk, Rake, rake.
- ▶ Пример использования:

```
from rake_nltk import Rake
r = Rake(stopwords.words('russian') + ['это', 'вне'])
r.extract_keywords_from_text(tokenized_text)
r.get_ranked_phrases()
```

[https://github.com/RusAl84/text\\_analitic/tree/master/DZ\\_3%20CompareKWE](https://github.com/RusAl84/text_analitic/tree/master/DZ_3%20CompareKWE)



# Выделение ключевых слов по tf-idf

- Идея: хотим выделить слова, которые часто встречаются в данном тексте, и редко - в других текстах.

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

где  $tf_{wd}$  – число раз, которое слово  $w$  встретилось в документе  $d$ ,  $df_w$  – число документов, содержащих  $w$ ,  $N$  – общее число документов.

- Такие слова, как правило, информативны, и значение tf-idf является хорошим признаком.
- Значения tf-idf для слов текста можно получить с помощью `sklearn.feature_extraction.text.TfidfVectorizer`

[https://github.com/RusAl84/text\\_analitic/tree/master/DZ\\_3%20CompareKWE](https://github.com/RusAl84/text_analitic/tree/master/DZ_3%20CompareKWE)

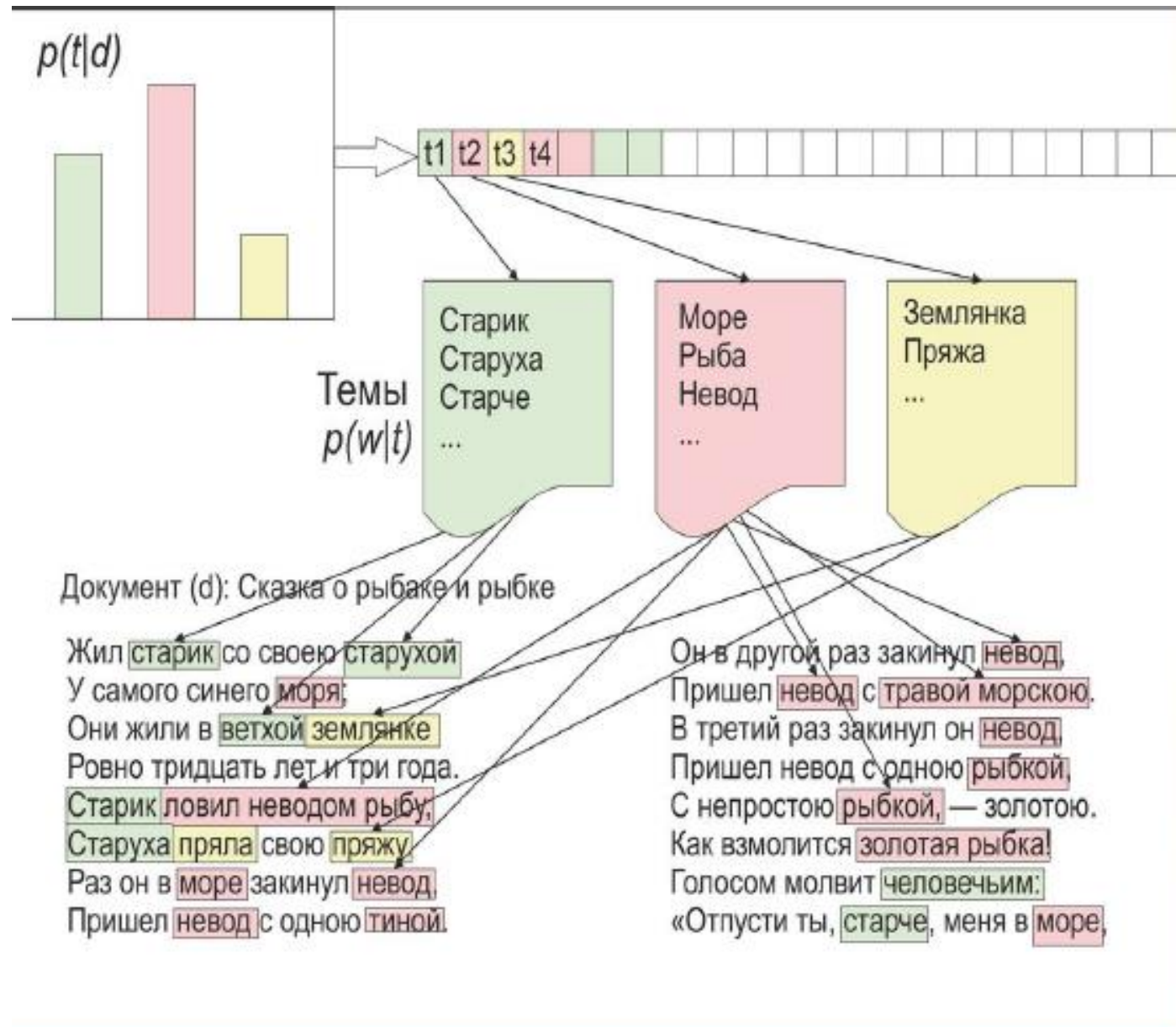
# Тематическое моделирование

- ▶ Тематическое моделирование - это способ построения модели коллекции текстовых документов, которая определяет, к каким темам относится каждый из документов.
- ▶ Тематическая модель (topic model) коллекции текстовых документов определяет, к каким темам относится каждый документ и какие слова (термины) образуют каждую тему.

Отвечает на вопросы:

1. Как выявлять смысл или тематику документов по их содержанию?
2. Как осуществлять классификацию документов на основе этих скрытых тематических закономерностей?

# Тематическое моделирование



# Практика



# Дорогу осилит идущий...

## Что будет – инструментарий



**Язык программирования Python**

<https://www.python.org/>



**Библиотека для матричных  
вычислений и линейной алгебры**

<http://www.numpy.org/>



**Библиотека для научных вычислений**

<https://www.scipy.org/>



**Библиотека для визуализации**

<https://matplotlib.org/>



**Библиотека для машинного обучения**

<http://scikit-learn.org/>



## Совет по инструментарию



**научный дистрибутив  
Anaconda Python  
от Continuum**

**<https://www.anaconda.com/download/>**


**Python 3.6 version \***

 **Download**

[64-Bit Graphical Installer \(631 MB\)](#) ⓘ

[32-Bit Graphical Installer \(506 MB\)](#)

**Python 2.7 version \***

 **Download**

[64-Bit Graphical Installer \(564 MB\)](#) ⓘ

[32-Bit Graphical Installer \(443 MB\)](#)



**Python, R, Julia, Scala, F#**

<http://jupyter.org/>



<https://www.jetbrains.com/pycharm/>

## **эволюция IPython Notebook**

**для создания и обмена  
«ноутбуками»:**

- код
- полнотекстовые комментарии
- уравнения
- визуализация

**интегрированная среда  
разработки для языка  
программирования Python**

[colab.research.google.com](https://colab.research.google.com)

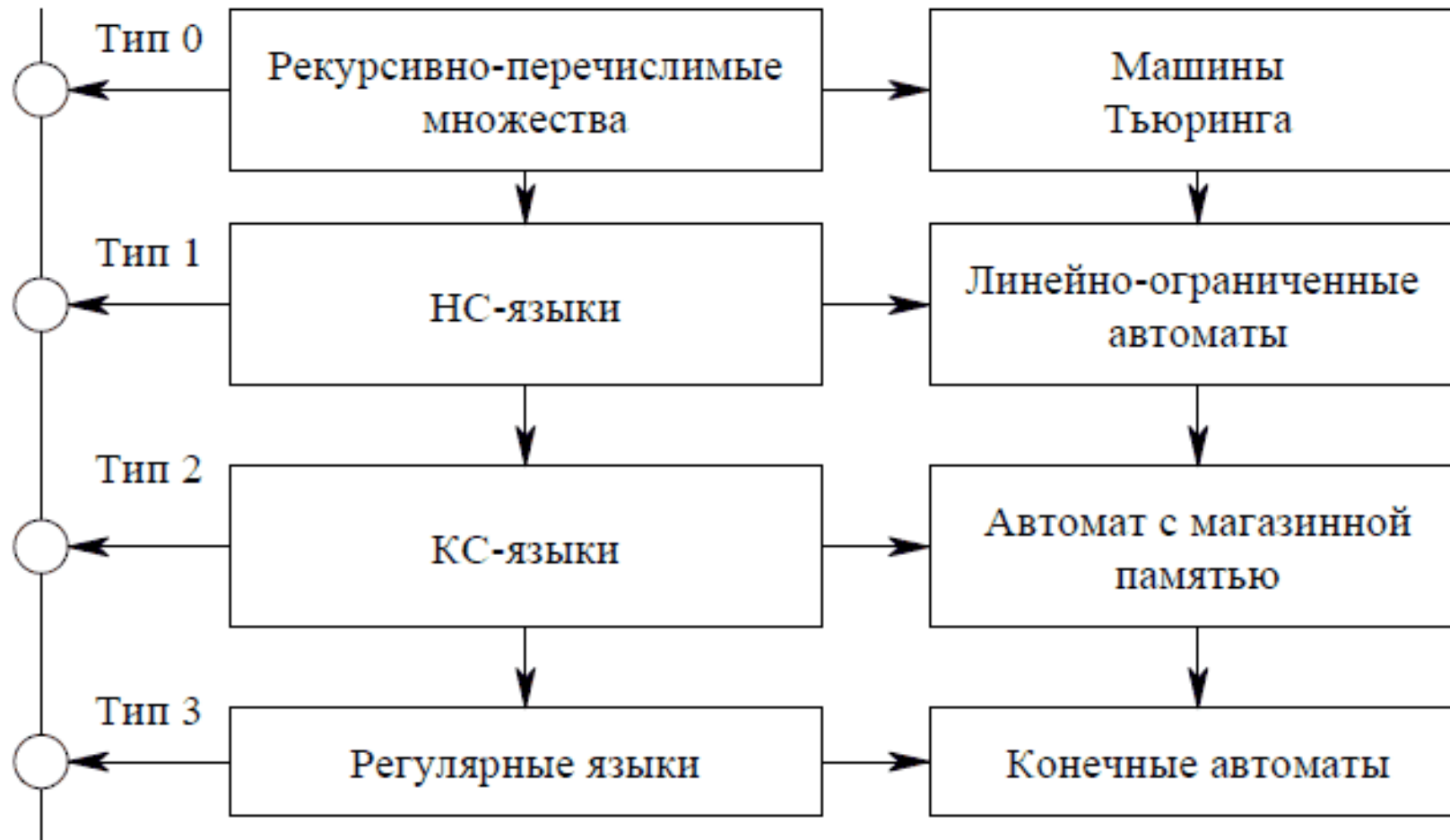


# Классификация формальных языков по Хомскому

Грамматика

Языки

Автоматы



**Конечный автомат** — это устройство для распознавания строк какого-либо языка. Конечный автомат — это пятерка  $M = (K, \Sigma, \delta, S, F)$ ,

где  $K$  — конечное множество состояний;

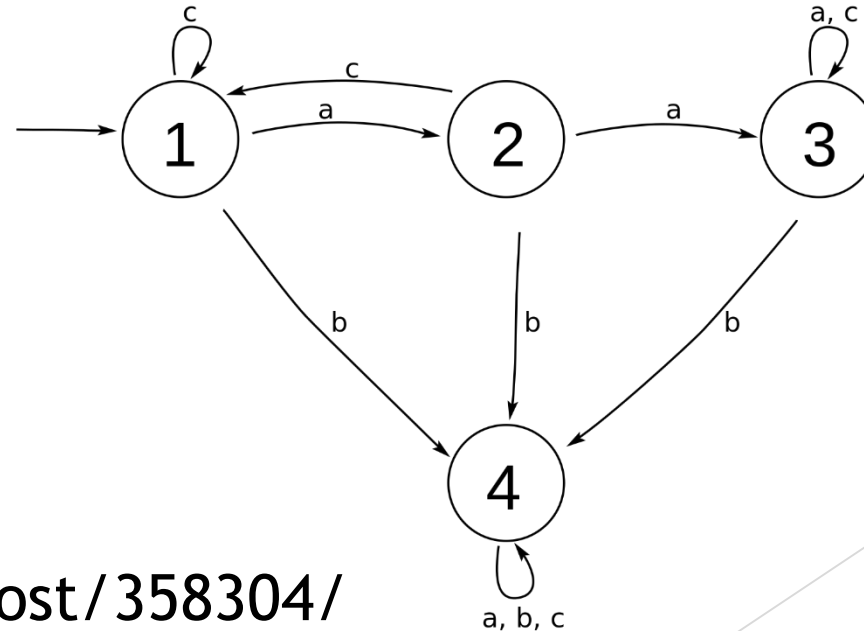
$\Sigma$  — конечный входной алфавит;

$\delta$  — множество переходов;

$S$  — начальное состояние ( $S \in K$ );

$F$  — множество последних состояний ( $F \subseteq K$ ).

По мере считывания каждой литеры строки автоматом контроль передается от состояния к состоянию в соответствии с заданным множеством переходов.

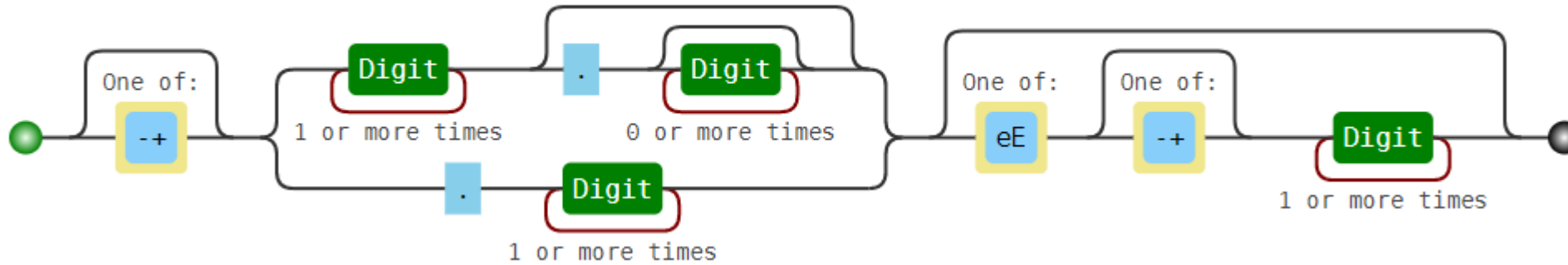


Регулярка	Её смысл
simple text	В точности текст «simple text»
\d{5}	Последовательности из 5 цифр \d означает любую цифру {5} — ровно 5 раз
\d\d/\d\d/\d{4}	Даты в формате ДД/ММ/ГГГГ (и прочие куски, на них похожие, например, 98/76/5432)
\b\w{3}\b	Слова в точности из трёх букв \b означает границу слова (с одной стороны буква, а с другой — нет) \w — любая буква, {3} — ровно три раза
[-+]? \d+	Целое число, например, 7, +17, -42, 0013 (возможны ведущие нули) [-+]? — либо -, либо +, либо пусто \d+ — последовательность из 1 или более цифр
[-+]?(?:\d+(?:\.\d*)? \.\d+)(?:[eE][-+]? \d+)?	Действительное число, возможно в экспоненциальной записи Например, 0.2, +5.45, -.4, 6e23, -3.17E-14. См. ниже картинку.

# Примеры «Регулярок»

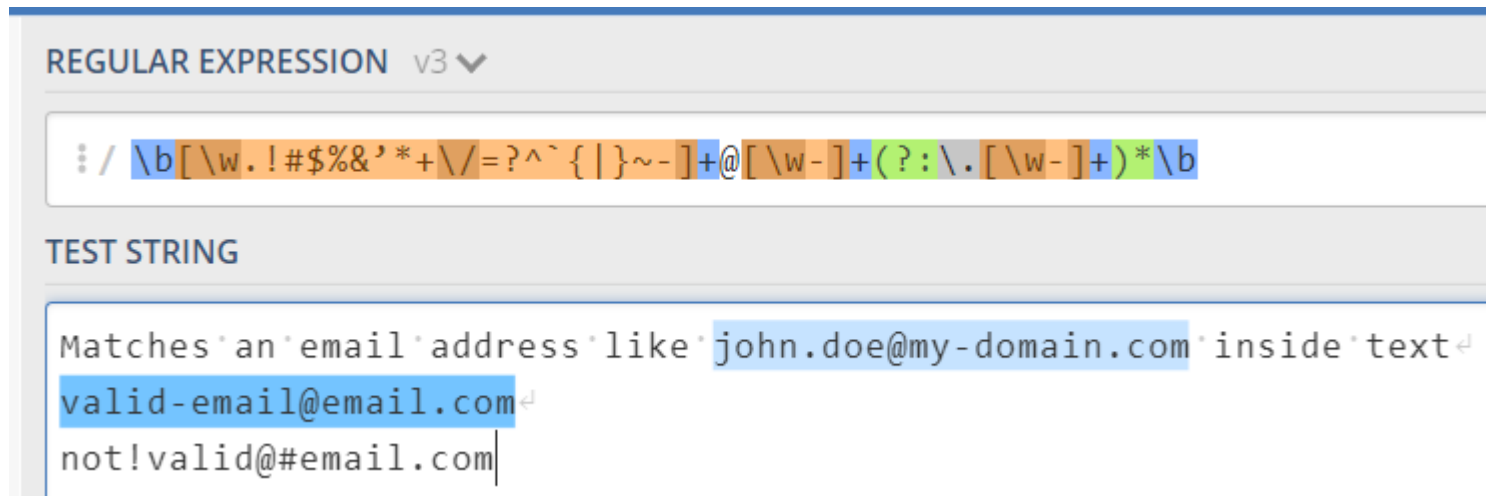
<https://regex101.com/>

RegExp: `/[-+]?(?:\d+(?:\.\d*)?|\.\d+)(?:[eE][-+]?\d+)?/`



Top 10 Most wanted regex

<https://medium.com/factory-mind/regex-cookbook-most-wanted-regex-aa721558c3c1>



# ИСТОРИЯ И СУТЬ ЗАДАЧИ РАСПОЗНАВАНИЯ ИМЕНОВАННЫХ СУЩНОСТЕЙ

На шестой конференции Message Understanding (MUC-6) в 1996 году задачи фокусировались в области извлечения информации.

В процессе постановки задач выявилась отдельная задача в извлечении объектов из документов.

Для определения объекта ввели термин «именованная сущность», а задачу назвали **распознавание именованных сущностей (named entity recognition, NER)**, так её в области обработки естественного языка и называют до сих пор.

# Распознавание именованных сущностей

На международной конференции по компьютерной лингвистике «Диалог 2016» проводилось соревнование по извлечению информации из текстов на русском языке

Необходимо было распознать следующие классические именованные сущности:

- ▶ 1. Персона (person, PER).
- ▶ 2. Местоположение (location, LOC).
- ▶ 3. Организация (organization, ORG).
- ▶ 4. Остальное (O) - тег для остальных слов в тексте, которые не являются обозначенными выше именованными сущностями.

# Пример использования

Задача NER - понять, что участок текста “1 января 1997 года” является датой, “Кофи Аннан” - персоной, а “ООН” - организацией.

## News NER example

Kofi Atta Annan is a Ghanaian diplomat who served as the seventh Secretary General of the United Nations from January 1, 1997, to January 1, 2007, serving two five-year terms. Annan was the co-recipient of the Nobel Peace Prize in October 2001.

Kofi Annan was born on April 8, 1938, to Victoria and Henry Reginald Annan in Kumasi, Ghana. He is a twin, an occurrence that is regarded as special in Ghanaian culture. Efua Atta, his twin sister, shares the same middle name, which means 'twin'. As with most Akan names, his first name indicates the day of the week he was born: 'Kofi' denotes a boy born on a Friday. The name Annan can indicate that a child was the fourth in the family, but in his family it was simply a name which Annan inherited from his parents.

In 1962, Annan started working as a Budget Officer for the World Health Organization, an agency of the United Nations. From 1974 to 1976, he was the Director of Tourism in Ghana. Annan then returned to work for the United Nations as an Assistant Secretary General in three consecutive positions.

Person
Location
Organization
Date
Nationality
Title



# Спасибо за внимание!



Русаков Алексей

[vk.com/RusAlm](https://vk.com/RusAlm) [RusAl@bk.ru](mailto:RusAl@bk.ru)