

# Введение в обработку естественного языка на Python



# Русаков Алексей

[vk.com/RusAlm](https://vk.com/RusAlm) [github.com/RusAl84/text\\_mining](https://github.com/RusAl84/text_mining)

[t.me/RusAl84](https://t.me/RusAl84)    [vk.com/text\\_mining](https://vk.com/text_mining)



**Системы  
мгновенного обмена  
сообщениями**



**Отчеты**



**Социальные сети**



**Документы**

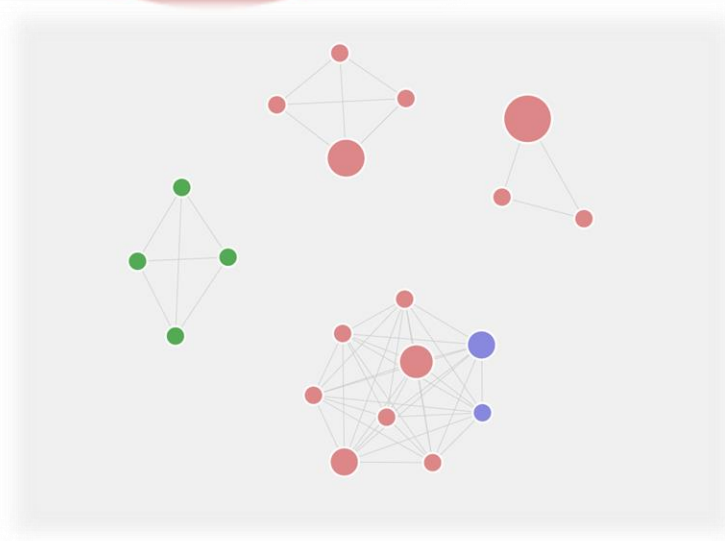
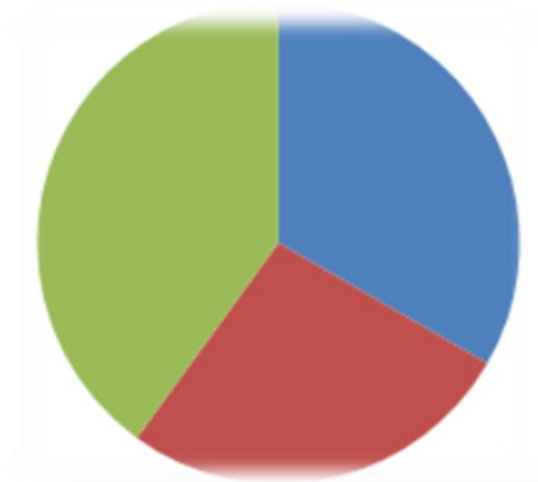


Номер	Код	Дата	Имя	Пол	Возраст	Рост	Вес	Цвет	Порода	Статус	Примечания
1	1	06.11.2008	Савицкий	муж	до 45 лет	15 000	100	белый	Савицкий	открыт	муж, до 45 лет, опыт
2	2	20.11.2008	Савицкий	муж	до 45 лет	15 000	100	белый	Савицкий	открыт	муж, до 45 лет, опыт
3	2	20.11.2008	Савицкий	муж	до 45 лет	15 000	100	белый	Савицкий	открыт	муж, до 45 лет, опыт
4	4	25.11.2008	Литвицкий	муж	до 45 лет	15 000	100	белый	Литвицкий	открыт	муж, до 45 лет, опыт
5	4	26.11.2008	Литвицкий	муж	до 45 лет	15 000	100	белый	Литвицкий	открыт	муж, до 45 лет, опыт
6	1	31.11.2008	Савицкий	муж	до 45 лет	15 000	100	белый	Савицкий	открыт	муж, до 45 лет, опыт
7	3	02.12.2008	Савицкий	муж	до 45 лет	15 000	100	белый	Савицкий	открыт	муж, до 45 лет, опыт
8	5	04.12.2008	Савицкий	муж	до 45 лет	15 000	100	белый	Савицкий	открыт	муж, до 45 лет, опыт
9	6	04.12.2008	Савицкий	муж	до 45 лет	15 000	100	белый	Савицкий	открыт	муж, до 45 лет, опыт
10	7	05.12.2008	Савицкий	муж	до 45 лет	15 000	100	белый	Савицкий	открыт	муж, до 45 лет, опыт
11	7	04.12.2008	Савицкий	муж	до 45 лет	15 000	100	белый	Савицкий	открыт	муж, до 45 лет, опыт
12	8	05.12.2008	Савицкий	муж	до 45 лет	15 000	100	белый	Савицкий	открыт	муж, до 45 лет, опыт
13	10	06.12.2008	Савицкий	муж	до 45 лет	15 000	100	белый	Савицкий	открыт	муж, до 45 лет, опыт
14	1	06.12.2008	Савицкий	муж	до 45 лет	15 000	100	белый	Савицкий	открыт	муж, до 45 лет, опыт
15	5	06.12.2008	Савицкий	муж	до 45 лет	15 000	100	белый	Савицкий	открыт	муж, до 45 лет, опыт

10	Егоров Сергей Геннадьевич	РФ
11	Ежов Станислав Станиславович	Украина
12	Еремин Николай Николаевич	Украина
13	Жидких Аркадий Юрьевич	РФ
14	Кимаковский Игорь Владимирович	РФ
15	Ковалис Ольга Валерьевна	РФ
16	Кореновский Дмитрий Евстафьевич	Украина
17	Костенко Андрей Сергеевич	Украина
18	Лазарев Сергей Александрович	Украина
19	Ломако Юрий Николаевич	Украина
20	Мельничук Петр Николаевич	Молдова
21	Мефедов Евгений Игоревич	РФ
22	Одинцов Максим Евгеньевич	РФ
23	Пикалов Валерий Валерьевич	Украина
24	Прозолова 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	Фёдоров Виктор Андреевич	Украина
32	Хитров Денис Васильевич	Украина
33	Хоменко Олег Юрьевич	Украина
34	Цемах Владимир Борисович	Украина

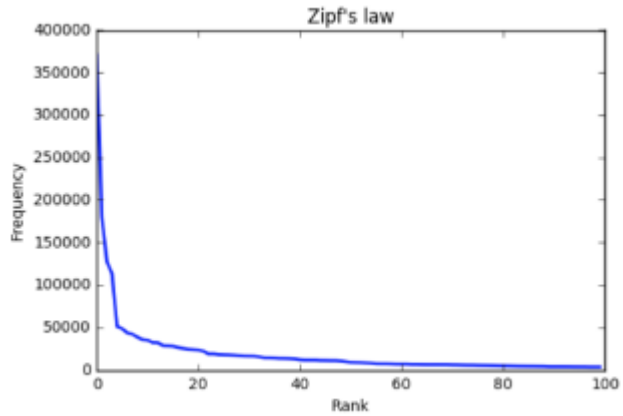
4	Bean Goose	Anser fabalis	Metsahanhi
5	Pink-footed Goose	Anser brachyrhynchus	Lyhytnokkahanhi
6	Greater White-fronted Goose	Anser albifrons	Tundrahanhi
7	Lesser White-fronted Goose	Anser erythropus	Kiljuhanhi
8	Greylag Goose	Anser anser	Merihanhi
9	Snow Goose	Anser caerulescens	Lumihanhi
10	Canada Goose	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)

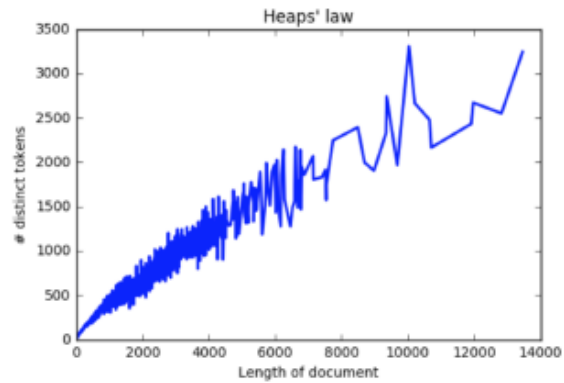


**Визуализация**

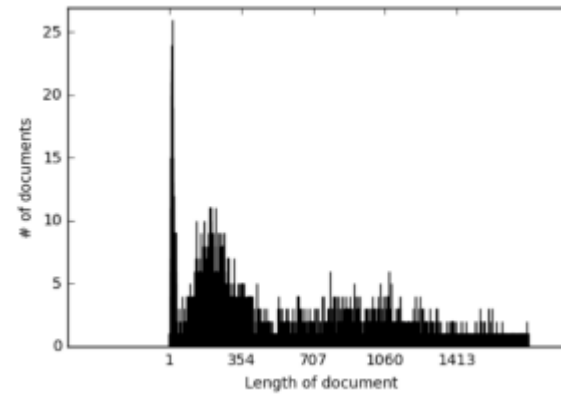
# Статистические свойства коллекций



Частоты слов в коллекции, закон Ципфа

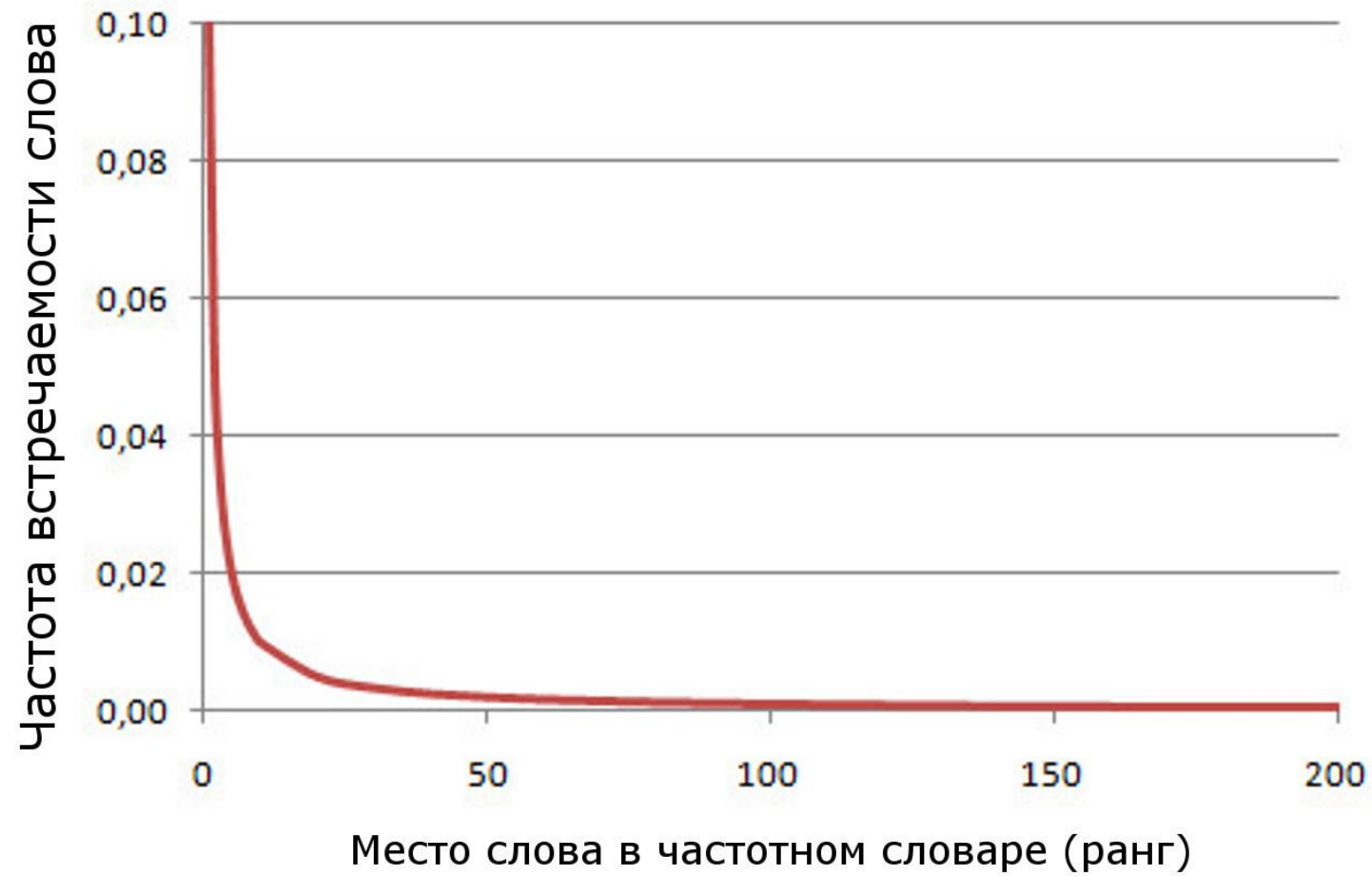


Закон Хипса для документов



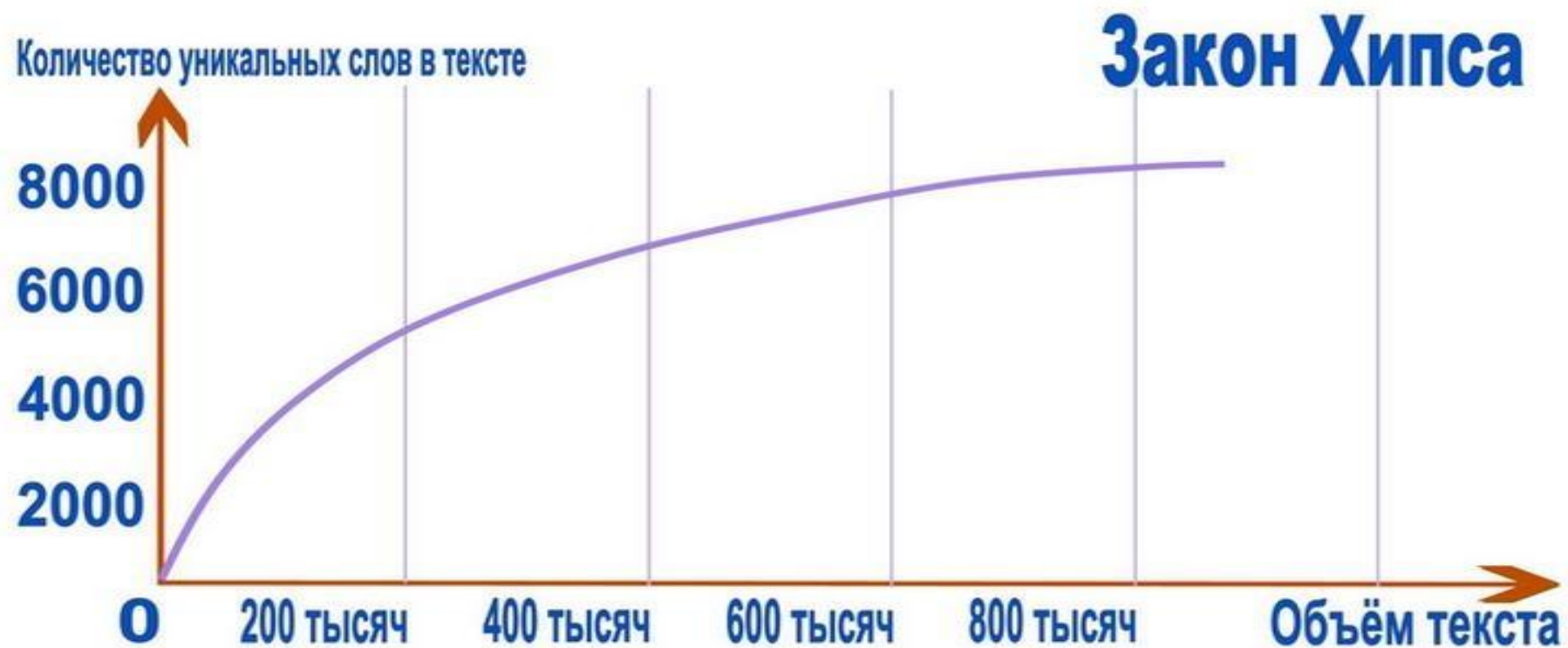
Гистограмма длин документов

# Закон Ципфа





# Закон Хипса



# 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 любого анализа данных - 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)
```

# 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.
  - `rumorphy2` (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

**ВСЕМ ПОНЯТНО???**

# 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)



# 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.

# RAKE

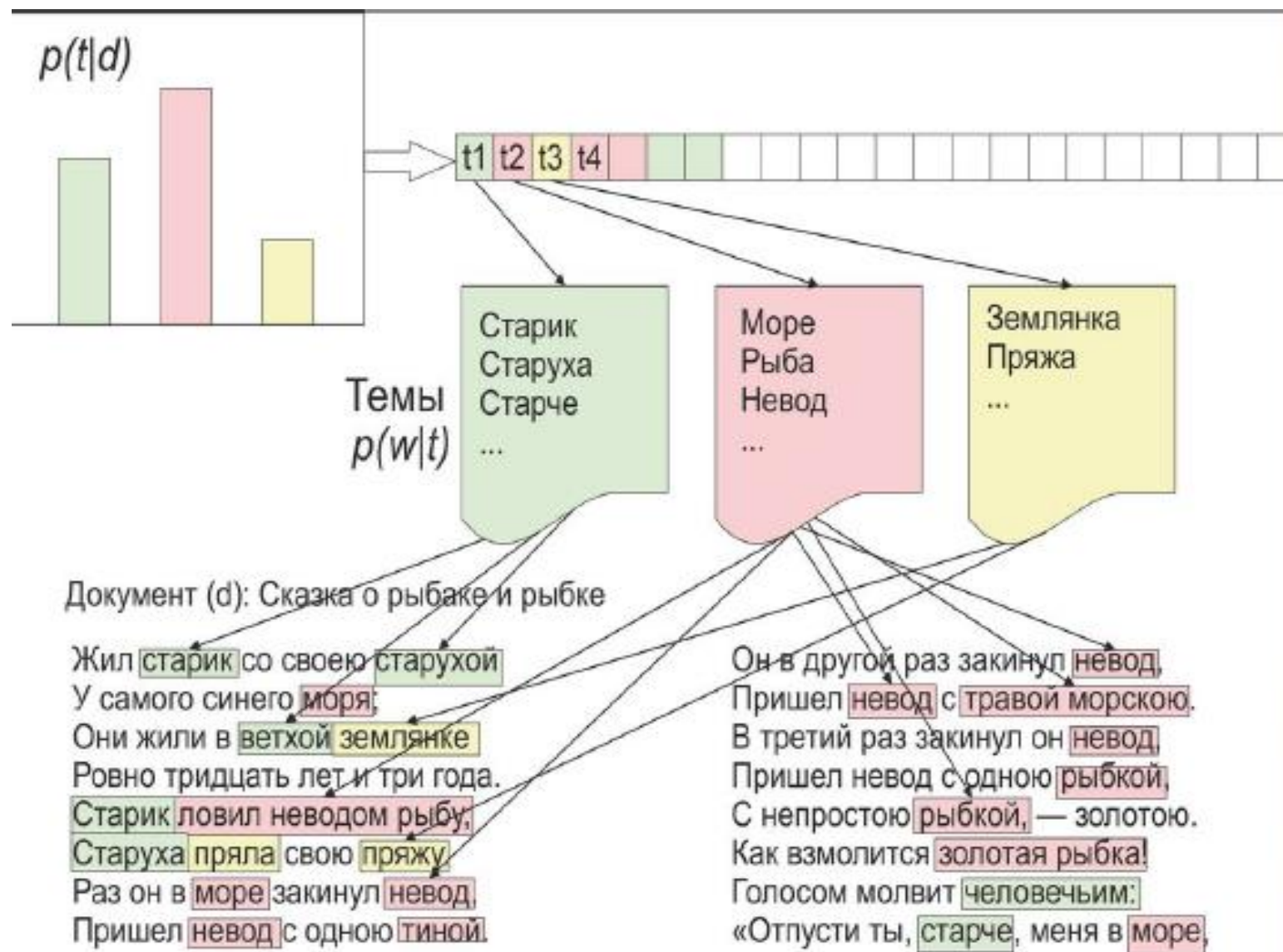
- ▶ *RAKE (Rapid Automatic Keyword Extraction):*
  - ▶ Фразы-кандидаты – все слова между разделителями.
  - ▶ Некоторым образом производится оценка фразы.
  - ▶ Фраза-кандидат ограничивается по частоте и количеству слов.
  - ▶ Модули: rake\_nltk, Rake, rake.

- ▶ 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)

# Тематическое моделирование



# Распознавание именованных сущностей

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

# demo.DeepPavlov.ai



Star

5,100

Ответы на  
вопросы  
по тексту

Ответы на  
вопросы  
по Википедии

Распознавание  
именованных  
сущностей

Анализ  
тональности

Ru

En

Zh

Multi-Lang

## Ответы на вопросы по тексту



deeppavlov at NGC



deeppavlov at Docker Hub

Ответы на вопросы по тексту (Text QA) — это задача поиска ответов на вопросы в известном контексте (например, в параграфе из Википедии). С помощью этого функционала можно искать ответы на вопросы по документации. Больше информации о модели и ее применении читайте в [документации](#).

[demo.deeppavlov.ai/#/ru/textqa](https://demo.deeppavlov.ai/#/ru/textqa)

# Визуализация семантических связей между словами (rusvectors.org)

Визуализация взаимного расположения слов при помощи t-SNE\*

новый Москва Лондон клавиатура транспорт машина Париж быстрый чистый мотоцикл компьютер мышь

## НКРЯ

Визуализировать в TensorFlow Projector



## НКРЯ и Wikipedia

Визуализировать в TensorFlow Projector



<http://rusvectors.org/ru/visual/>



# Спасибо за внимание!



Русаков Алексей

[vk.com/RusAlM](https://vk.com/RusAlM) [t.me/RusAl84](https://t.me/RusAl84)

# Современные библиотеки Python

**Gensim** — библиотека обработки естественного языка предназначена для «Тематического моделирования». С его помощью можно обрабатывать тексты, работать с векторными моделями слов (такими как Word2Vec, FastText и т. д.) и создавать тематические модели текстов.

Тематическое моделирование — это метод извлечения основных тем которым посвящен обрабатываемый текст. В пакете Gensim реализованы основные алгоритмы тематического моделирования: скрытое распределение Дирихле (LDA) и скрытое семантическое индексирование (LSI).

Какие типы текстов может обрабатывать Gensim? Входной текст может быть в одной из 3 разных форм:

- Как предложения, хранящиеся в собственном объекте списка Python.
- Как один текстовый файл.
- В нескольких текстовых файлах.

Документация на английском языке:

[https://radimrehurek.com/gensim/auto\\_examples/index.html#documentation](https://radimrehurek.com/gensim/auto_examples/index.html#documentation)

Руководство для начинающих на русском языке:

<https://webdevblog.ru/gensim-rukovodstvo-dlya-nachinajushhih/>

# Word2vec

**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.

Принцип работы:

Нахождение связей между контекстами слов согласно предположению, что слова, находящиеся в похожих контекстах, имеют тенденцию значить похожие вещи, т.е. быть семантически близкими. Более формально задача стоит так: максимизация косинусной близости между векторами слов (скалярное произведение векторов), которые появляются рядом друг с другом, и минимизация косинусной близости между векторами слов, которые не появляются друг рядом с другом. Рядом друг с другом в данном случае значит в близких контекстах.

# FastText

**FastText** — это библиотека, содержащая предобученные готовые векторные представления слов и классификатор, то есть алгоритм машинного обучения разбивающий слова на классы. FastText разработала команда исследователей из Facebook AI Research, в составе которых был и создатель Word2vec Томаш Миколов

Так что же отличает FastText? Более быстрая работа по сравнению с другими пакетами и моделями. Для модели векторных представлений слов используется skip-gram с негативным сэмплированием. Негативное сэмплирование — это способ создать для обучения векторной модели отрицательные примеры, то есть показать ей пары слов, которые не являются соседями по контексту. Для каждого положительного примера (когда слова в тексте стоят рядом, например, «пушистый котик») мы подбираем несколько отрицательных («пушистый утюг», «пушистый радиосигнал», «пушистое бегство»). Всего подбирается от 3 до 20 случайных слов. Такой случайный подбор нескольких примеров не требует много компьютерного времени и позволяет ускорить работу FastText.

Документация на английском языке: <https://fasttext.cc/docs/en/supervised-tutorial.html>

# pymorphy2

**Pymorphy2** – морфологический анализатор. Он умеет:

- приводить слово к нормальной форме (например, “люди -> человек”, или “гулял -> гулять”).
- ставить слово в нужную форму. Например, ставить слово во множественное число, менять падеж слова и т.д.
- возвращать грамматическую информацию о слове (число, род, падеж, часть речи и т.д.)

При работе используется словарь OpenCorpora; для незнакомых слов строятся гипотезы. Библиотека достаточно быстрая: в настоящий момент скорость работы - от нескольких тыс слов/сек до > 100тыс слов/сек (в зависимости от выполняемой операции, интерпретатора и установленных пакетов); потребление памяти - 10...20Мб; полностью поддерживается буква ё. Лицензия - MIT.

Документация на русском языке:

<https://pymorphy2.readthedocs.io/en/stable/user/index.html>



# NLTK

NLTK (Natural Language Toolkit) — это важная библиотека, поддерживающая такие задачи, как классификация, стемминг, маркировка, синтаксический анализ и семантическое рассуждение в Python. Это бесплатный проект с открытым исходным кодом, поддерживаемый в данный момент.

Благодаря практическому руководству, представляющему основы программирования наряду с темами вычислительной лингвистики, а также исчерпывающей документации по API, NLTK подходит как для лингвистов, инженеров, студентов, преподавателей, исследователей, так и для обычных пользователей. NLTK доступен для Windows, Mac OS X и Linux.

Установка и документация на английском языке: <https://www.nltk.org/>

# spaCy

spaCy - это бесплатная библиотека с открытым исходным кодом для расширенной обработки естественного языка в Python.

spaCy разработан специально для производственного использования и помогает создавать приложения, обрабатывающие и «понимающие» большие объемы текста. Его можно использовать для построения систем извлечения информации или понимания естественного языка или для предварительной обработки текста для глубокого обучения.

spaCy - это не исследовательское программное обеспечение. Он основан на последних исследованиях, но предназначен для работы любых разработчиков. Это приводит к совершенно иным проектным решениям, чем NLTK или CoreNLP, которые были созданы в качестве платформ для обучения и исследований. Сохранение небольшого размера меню позволяет spaCy в целом обеспечивать лучшую производительность и удобство для разработчиков.

Документация на английском языке: <https://spacy.io/usage/spacy-101>

# Pattern

**Pattern** - это web-mining модуль, имеющий инструменты для интеллектуального анализа данных (Google, Twitter и Wikipedia API, веб-сканер, анализатор HTML DOM), обработки естественного языка (тегеры части речи, поиск n-граммов, анализ тональности, WordNet), машинного обучения (векторная модель, кластеризация, SVM), сетевого анализа и визуализации <canvas>.

Документация на английском языке: <https://github.com/clips/pattern/wiki>

# Natasha

**Natasha** — качественное компактное решение для извлечения именованных сущностей из новостных статей на русском языке

Библиотека Natasha решает базовые задачи обработки естественного русского языка: сегментация на токены и предложения, морфологический и синтаксический анализ, лемматизация, извлечение именованных сущностей. Для новостных статей качество на всех задачах сравнимо или превосходит существующие решения. Библиотека поддерживает Python 3.5+ и PyPy3, не требует GPU, зависит только от NumPy.

Документация на английском языке:

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

# Googletrans

**Googletrans** - неофициальная и бесплатная библиотека для перевода текста Google Translate API на Python.

Документация на английском языке:

<https://py-googletrans.readthedocs.io/en/latest/#>

## Yandex.Translate

Официальная библиотека для перевода текста от Yandex. Действовала до 2020 года, затем Yandex перестали выдавать бесплатные API ключи.

---

❗ **Note.** Starting May 27, 2020, free API keys aren't issued. [We recommend using Yandex.Translate](#) [↗](#) included in Yandex.Cloud, which supports the latest neural machine translation (NMT) technology.

Документация на английском языке:

<https://yandex.com/dev/translate/doc/dg/concepts/about.html>

# Scikit-learn

**Scikit-learn** - один из наиболее широко используемых пакетов Python для Data Science и машинного обучения с открытым исходным кодом, основанный на NumPy и SciPy. Он позволяет выполнять множество операций и предоставляет множество алгоритмов. Scikit-Learn поддерживает предварительную обработку данных, уменьшение размерности, выбор модели, регрессии, классификации, кластерный анализ.

Документация на английском языке: <https://scikit-learn.org/stable/>



# Transformers

**Transformers** — это библиотека на Python для обучения state-of-the-art моделей в обработке естественного языка. Разработкой библиотеки занимается компания HuggingFace. Transformers предоставляет API для использования таких архитектур трансформеров, как BERT, RoBERTa, GPT-2 или DistilBERT.

Архитектуры трансформера показывают наиболее точно решают задачи текстовой классификации, вычленения информации из текста, вопросно-ответных систем и текстовой генерации. Трансформеры в библиотеке предобучены и доступны в виде набора весов.

Transformers базируется на идее предобучения трансформер моделей. Трансформер модели доступны в разных формах, размерах и архитектурах. На вход трансформеры принимают данные в токенизированном формате.

Документация на английском языке: <https://huggingface.co/transformers/>

# DeepPavlov

**DeepPavlov** — открытая программная библиотека разговорного AI для создания виртуальных диалоговых ассистентов и анализа текста.

На основе этой библиотеки вы можете собрать чат бота, ассистента или решение для анализа текста.

Документация на английском языке: <http://docs.deeppavlov.ai/en/master/>

# Beautiful Soup

**Beautiful Soup** — это библиотека Python для извлечения данных из файлов HTML и XML, для парсинга. Она создаёт деревья считывания данных, позволяющие с лёгкостью эти данные получать. Но проблема с Beautiful Soup в том, что он не может выполнить всю работу самостоятельно. Эта библиотека требует определенных модулей для работы.

Документация на русском языке:

<https://www.crummy.com/software/BeautifulSoup/bs4/doc.ru/bs4ru.html#id11>

# Спасибо за внимание!



# Русаков Алексей

[vk.com/RusAlM](https://vk.com/RusAlM) [t.me/RusAl84](https://t.me/RusAl84)