

Лекция 7: Введение в обработку текста на естественном языке

Автор: Сергей Вячеславович Макрушин, 2022 г.

e-mail: s-makrushin@yandex.ru (<mailto:s-makrushin@yandex.ru>)

V 0.4 23.10.2022

Разделы:

- [Метрики расстояния между строками](#)
 - [Расстояние Левенштейна](#)
 - [Динамическое программирование](#)
 - [Алгоритм Вагнера - Фишера](#)
- [Стемминг и лемматизация](#)
 - [Стемминг](#)
 - [Лемматизация](#)
- [Стоп-слова](#)
- [Мешок слов](#)
- [Векторное представление документа](#)

-

- [к оглавлению](#)

```
In [1]: # загружаем стиль для оформления презентации
from IPython.display import HTML
from urllib.request import urlopen
html = urlopen("file:./lec_v2.css")
HTML(html.read().decode('utf-8'))
```

Out[1]:

Метрики расстояния между строками

-

[к оглавлению](#)

- Расстояние Левенштейна
- Алгоритм поиска минимального редакционного расстояния Вагнера - Фишера
- Задача динамического программирования

Метрики расстояния для строк

Часто требуется понять, насколько близкими являются две несовпадающих строки (слова). Это может потребоваться для:

- для сравнения текстов, предложений
- поиска ошибок и опечаток в слове
- поиска словоформ слова
- в других областях (в биоинформатике для сравнения генов, хромосом и белков)

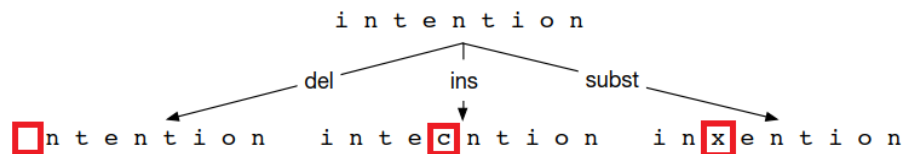
Расстояние Левенштейна

-

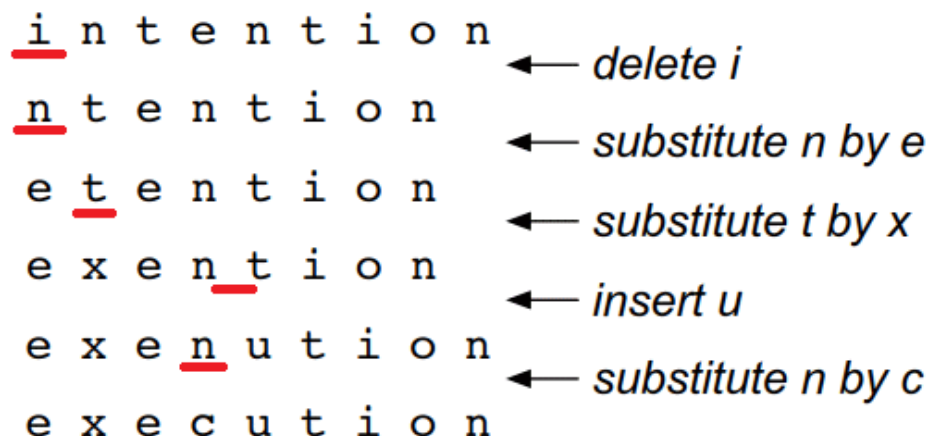
[к оглавлению](#)

Расстояние Левенштейна (редакционное расстояние, дистанция редактирования) - **минимальное** количество операций необходимых для превращения одной строки в другую. Рассматриваются следующие операции:

- вставка одного символа
- удаление одного символа
- замена одного символа на другим.



Пример выполнения операций вставки, удаления и замены для слова "intention"



Пример преобразования слова "intention" в "execution" с помощью операций вставки, удаления и замены

В общем случае **стоимость различных операций** может быть различной. Обычно цена отражает **разную вероятность** событий и может зависеть от:

- вида операции (вставка, удаление, замена)
- и/или от участвующих в ней символов

Если к списку разрешённых операций добавить **транспозицию** (два соседних символа меняются местами), получается **расстояние Дамерау - Левенштейна**.

- Дамерау показал, что 80 % ошибок при наборе текста человеком являются транспозициями.
- Кроме того, это расстояние используется и в биоинформатике.

Для поиска расстояния Левинштайна и расстояния Дамерау - Левинштайна **существуют эффективные алгоритм**, требующий $O(MN)$ операций (M и N это длины первой и второй строки соответственно).

Пусть S_1 и S_2 - две строки (длиной M и N соответственно, здесь и далее считается, что элементы строк нумеруются с первого, как принято в математике) над некоторым алфавитом, тогда расстояние Левенштейна $d(S_1, S_2)$ можно подсчитать используя вспомогательную функцию

$D(M, N)$, находящую редакционное расстояние для подстрок $S_1[0..M]$ и $S_2[0..N]$

по следующей рекуррентной формуле:

$$d(S_1, S_2) = D(M, N)$$

$$D(i, j) = \begin{cases} \max(i, j) & \text{if } \min(i, j) = 0, \\ \min \begin{cases} D(i-1, j) + 1 \\ D(i, j-1) + 1 \\ D(i-1, j-1) + m(S_1[i], S_2[j]) \end{cases} & \text{otherwise.} \end{cases}$$

$D(i-1, j) + 1$, операция удаления (цена: 1, на схеме обозначается как: \uparrow)

$D(i, j-1) + 1$, операция вставки (цена: 1, на схеме обозначается как: \leftarrow)

$D(i-1, j-1) + m(S_1[i], S_2[j])$, операция замены (цена m , на схеме обозначается как: \searrow)

Цена операции замены зависит от заменяемых символов:

$$m(s_1, s_2) = \begin{cases} 0, & \text{if } s_1 = s_2 \\ 2, & \text{if } s_1 \neq s_2 \end{cases}$$

Очевидно, что для расстояния Левинштейна справедливы следующие утверждения:

- $d(S_1, S_2) \geq ||S_1| - |S_2||$
- $d(S_1, S_2) \leq \max(|S_1|, |S_2|)$
- $d(S_1, S_2) = 0 \Leftrightarrow S_1 = S_2$

Редакционным предписанием называется последовательность действий, необходимых для получения второй строки из первой кратчайшим образом. Обычно действия обозначаются так:

- D (англ. delete) — удалить
- I (англ. insert) — вставить
- R (replace) — заменить
- M (match) — совпадение.

По сути редакционное предписание это кратчайшие пути на графе с весами, в котором существует 3 вида ориентированных ребер (D, I, M), а вершинами являются строки (слова). В общем случае для конкретной пары слов может существовать несколько редакционных предписаний (кратчайших путей на графе).

Динамическое программирование

-

- [к оглавлению](#)

Динамическое программирование - способ решения сложных задач путём разбиения их на более простые подзадачи. Он применим к задачам с *оптимальной подструктурой*, выглядящим как *набор перекрывающихся подзадач*, сложность которых чуть меньше исходной. В этом случае время вычислений, по сравнению с «наивными» методами, можно значительно сократить.

Идея динамического программирования:

Оптимальная подструктура в динамическом программировании означает, что оптимальное решение подзадач меньшего размера может быть использовано для решения исходной задачи.

В общем случае мы можем решить задачу, в которой присутствует оптимальная подструктура, проделывая следующие три шага.

1. Разбиение задачи на подзадачи меньшего размера.
2. Нахождение оптимального решения подзадач рекурсивно, проделывая такой же трехшаговый алгоритм.
3. Использование полученного решения подзадач для конструирования решения исходной задачи.

Часто многие из рассматриваемых подзадач одинаковы. Подход динамического программирования состоит в том, чтобы *решить каждую подзадачу только один раз*, сократив тем самым количество вычислений. Это особенно полезно в случаях, когда число повторяющихся подзадач экспоненциально велико.

- Метод динамического программирования **сверху-вниз** (top-down approach) - это простое *запоминание результатов решения тех подзадач*, которые могут повторно встретиться в дальнейшем.
- Динамическое программирование **снизу-вверх** (bottom-up approach) включает в себя переформулирование сложной задачи в виде рекурсивной последовательности более простых подзадач.

Пример использования динамического программирования

Пример: подсчет факториалов последовательности чисел от 0 до m.

Наивное решение задачи:

```
In [72]: def iter_factorial(n):
        factorial = 1
        if n == 0 or n == 1:
            return 1
        else:
            for i in range(1, n + 1):
                factorial = factorial * i
            return factorial

        def factorial_seq_1(m):
            return [iter_factorial(i) for i in range(m+1)]
```

```
In [74]: iter_factorial(4)
```

```
Out[74]: 24
```

```
In [76]: factorial_seq_1(10)
```

```
Out[76]: [1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800]
```

```
In [78]: %%timeit
        factorial_seq_1(100)
```

546 µs ± 37.6 µs per loop (mean ± std. dev. of 7 runs, 1,000 loops each)

Рекурсивное решение задачи:

(Динамическое программирование снизу-вверх)

In [79]: *# рекурсивная реализация:*

```
def rec_factorial(n):  
    if n == 0 or n == 1:  
        return 1  
    else:  
        return n * rec_factorial(n-1)
```

In [80]: rec_factorial(4)

Out[80]: 24

```
In [81]: def rec_factorial_for_seq(n, result):  
    if n == 0:  
        result.append(1)  
    else:  
        result.append(n * rec_factorial_for_seq(n-1, result))  
    return result[-1]  
  
def factorial_seq_2(m):  
    res = []  
    rec_factorial_for_seq(m, res)  
    return res
```

In [82]: factorial_seq_2(10)

Out[82]: [1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800]

```
In [83]: %%timeit  
factorial_seq_2(100)
```

37 μ s \pm 2 μ s per loop (mean \pm std. dev. of 7 runs, 10,000 loops each)

Рекурсивное решение задачи:

(Динамическое программирование сверху-вниз, на основе кеширования)

In [12]: from functools import lru_cache

```
In [84]: @lru_cache(maxsize=1024)  
def rec_factorial_cache(n):  
    if n == 0:  
        return 1  
    else:  
        return n * rec_factorial(n-1)
```

In [85]: rec_factorial_cache(4)

Out[85]: 24

```
In [70]: def factorial_seq_3(m):  
    return [rec_factorial_cache(i) for i in range(m+1)]
```

In [86]: factorial_seq_3(10)

Out[86]: [1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800]

```
In [87]: %%timeit
factorial_seq_3(100)
```

15.2 μ s \pm 1.53 μ s per loop (mean \pm std. dev. of 7 runs, 100,000 loops each)

Алгоритм Вагнера - Фишера

-
- [к оглавлению](#)

Используя рекурсивное определение расстояния Левинштейна $D(i, j)$ через расстояния для слов меньшей длины: $D(i - 1, j)$, $D(i, j - 1)$, $D(i - 1, j - 1)$ мы применим принцип динамического программирования снизу-вверх, комбинируя решения подзадач, для решения более сложной задачи.

- Для получения базового решения когда конечная строка длины 0 или исходная строка длинны 0:
 - $D(i, 0) = i$ - используется i операций удаления (на схеме операция удаления обозначается, как: " \uparrow ")
 - $D(0, j) = j$ - используется j операций вставки (на схеме операция вставки обозначается, как: " \leftarrow ")
- После расчета $D(i, j)$ для малых i и j мы рассчитываем значения расстояния для бОльших i и j на основе рекурсивной формулы:

$$D(i, j) = \min \begin{cases} D(i - 1, j) + 1, \text{ операция удаления, на схеме обозначается как: } \uparrow \\ D(i, j - 1) + 1, \text{ операция вставки, на схеме обозначается как: } \leftarrow \\ D(i - 1, j - 1) + m(S_1[i], S_2[j]), \text{ операция замены, на схеме обозначается} \end{cases}$$



	#	е	х	е	с	u	t	i	o	n
#	0	$\leftarrow 1$	$\leftarrow 2$	$\leftarrow 3$	$\leftarrow 4$	$\leftarrow 5$	$\leftarrow 6$	$\leftarrow 7$	$\leftarrow 8$	$\leftarrow 9$
i	$\uparrow 1$	$\nwarrow \leftarrow \uparrow 2$	$\nwarrow \leftarrow \uparrow 3$	$\nwarrow \leftarrow \uparrow 4$	$\nwarrow \leftarrow \uparrow 5$	$\nwarrow \leftarrow \uparrow 6$	$\nwarrow \leftarrow \uparrow 7$	$\nwarrow 6$	$\leftarrow 7$	$\leftarrow 8$
n	$\uparrow 2$	$\nwarrow \leftarrow \uparrow 3$	$\nwarrow \leftarrow \uparrow 4$	$\nwarrow \leftarrow \uparrow 5$	$\nwarrow \leftarrow \uparrow 6$	$\nwarrow \leftarrow \uparrow 7$	$\nwarrow \leftarrow \uparrow 8$	$\uparrow 7$	$\nwarrow \leftarrow \uparrow 8$	$\nwarrow 7$
t	$\uparrow 3$	$\nwarrow \leftarrow \uparrow 4$	$\nwarrow \leftarrow \uparrow 5$	$\nwarrow \leftarrow \uparrow 6$	$\nwarrow \leftarrow \uparrow 7$	$\nwarrow \leftarrow \uparrow 8$	$\nwarrow 7$	$\leftarrow \uparrow 8$	$\nwarrow \leftarrow \uparrow 9$	$\uparrow 8$
e	$\uparrow 4$	$\nwarrow 3$	$\leftarrow 4$	$\nwarrow \leftarrow 5$	$\leftarrow 6$	$\leftarrow 7$	$\leftarrow \uparrow 8$	$\nwarrow \leftarrow \uparrow 9$	$\nwarrow \leftarrow \uparrow 10$	$\uparrow 9$
n	$\uparrow 5$	$\uparrow 4$	$\nwarrow \leftarrow \uparrow 5$	$\nwarrow \leftarrow \uparrow 6$	$\nwarrow \leftarrow \uparrow 7$	$\nwarrow \leftarrow \uparrow 8$	$\nwarrow \leftarrow \uparrow 9$	$\nwarrow \leftarrow \uparrow 10$	$\nwarrow \leftarrow \uparrow 11$	$\nwarrow \uparrow 10$
t	$\uparrow 6$	$\uparrow 5$	$\nwarrow \leftarrow \uparrow 6$	$\nwarrow \leftarrow \uparrow 7$	$\nwarrow \leftarrow \uparrow 8$	$\nwarrow \leftarrow \uparrow 9$	$\nwarrow 8$	$\leftarrow 9$	$\leftarrow 10$	$\leftarrow \uparrow 11$
i	$\uparrow 7$	$\uparrow 6$	$\nwarrow \leftarrow \uparrow 7$	$\nwarrow \leftarrow \uparrow 8$	$\nwarrow \leftarrow \uparrow 9$	$\nwarrow \leftarrow \uparrow 10$	$\uparrow 9$	$\nwarrow 8$	$\leftarrow 9$	$\leftarrow 10$
o	$\uparrow 8$	$\uparrow 7$	$\nwarrow \leftarrow \uparrow 8$	$\nwarrow \leftarrow \uparrow 9$	$\nwarrow \leftarrow \uparrow 10$	$\nwarrow \leftarrow \uparrow 11$	$\uparrow 10$	$\uparrow 9$	$\nwarrow 8$	$\leftarrow 9$
n	$\uparrow 9$	$\uparrow 8$	$\nwarrow \leftarrow \uparrow 9$	$\nwarrow \leftarrow \uparrow 10$	$\nwarrow \leftarrow \uparrow 11$	$\nwarrow \leftarrow \uparrow 12$	$\uparrow 11$	$\uparrow 10$	$\uparrow 9$	$\nwarrow 8$

Пример поиска расстояния Левинштейна для слов "intention" и "execution" с помощью алгоритма Вагнера - Фишера

function MIN-EDIT-DISTANCE(*source*, *target*) **returns** *min-distance*

```
n ← LENGTH(source)
m ← LENGTH(target)
Create a distance matrix distance[n+1,m+1]

# Initialization: the zeroth row and column is the distance from the empty string
D[0,0] = 0
for each row i from 1 to n do
    D[i,0] ← D[i-1,0] + del-cost(source[i])
for each column j from 1 to m do
    D[0,j] ← D[0,j-1] + ins-cost(target[j])

# Recurrence relation:
for each row i from 1 to n do
    for each column j from 1 to m do
        D[i,j] ← MIN( D[i-1,j] + del-cost(source[i]),
                        D[i-1,j-1] + sub-cost(source[i],target[j]),
                        D[i,j-1] + ins-cost(target[j]))

# Termination
return D[n,m]
```

Алгоритм Вагнера - Фишера для поиска расстояния Левинштейна

```
In [12]: # from nltk.metrics import *
```

```
In [88]: from nltk.metrics.distance import (
        edit_distance,
        edit_distance_align,
        binary_distance,
        jaccard_distance,
        masi_distance,
        interval_distance,
        custom_distance,
        presence,
        fractional_presence,
    )
```

```
In [89]: edit_distance('intention', 'execution', substitution_cost=2)
```

Out[89]: 8

```
In [90]: # результат npu substitution_cost=1
edit_distance('intention', 'execution')
```

Out[90]: 5

```
In [91]: edit_distance('пирвет', 'привет', substitution_cost=2)
```

Out[91]: 2

```
In [93]: # расстояние Домрау-Левинштайна:
edit_distance('пирвет', 'привет', substitution_cost=2, transpositions=True)
```

Out[93]: 1

```
In [23]: s1 = 'intention'
        s2 = 'execution'
```

```
In [94]: ed = edit_distance_align(s1, s2, substitution_cost=2)
ed
```

```
Out[94]: [(0, 0),
          (1, 0),
          (2, 0),
          (3, 0),
          (4, 1),
          (4, 2),
          (4, 3),
          (4, 4),
          (5, 5),
          (6, 6),
          (7, 7),
          (8, 8),
          (9, 9)]
```

```
In [95]: s1
```

```
Out[95]: 'intention'
```

```
In [26]: l1 = list(s1)
l1
```

```
Out[26]: ['i', 'n', 't', 'e', 'n', 't', 'i', 'o', 'n']
```

```
In [27]: s = ''.join(l1)
s
```

```
Out[27]: 'intention'
```

```
In [28]: res = l1
i = 3
```

```
In [29]: sh_res = ''.join('_'+s+'_' if ind==i else s for ind, s in enumerate(res))
sh_res
```

```
Out[29]: 'int_e_ntion'
```



```
In [96]: def show_ed_path(as1, as2, ed):
s1 = '#' + as1 # shift index
s2 = '#' + as2 # shift index
ip, jp = ed[0]
res = list(s1)
cost = 0
print(f'i:{ip}, j:{jp}; init, cost: {cost}; res: {"".join(res)[1:]}')
def sh_res(res, i):
    return ''.join(s.upper() if ind==i else s for ind, s in enumerate(res))[1:]

for i, j in ed[1:]:
    if i == ip+1 and j == jp+1:
        if s1[i] == s2[j]:
            # res = res
            cost += 0
            print(f'i:{i}, j:{j}; save {s1[i]}, cost: {cost}; res: {sh_res(res, j)}')
        else:
            res[j] = s2[j]
            cost += 2
            print(f'i:{i}, j:{j}; change {s1[i]} -> {s2[j]}; cost: {cost}; res: {sh_res(res, j)}')
    elif i == ip+1 and j == jp:
        cost += 1
        print(f'i:{i}, j:{j}; remove {res[j+1]}, cost: {cost}; res: {sh_res(res, j+1)}')
        rs = res.pop(j+1)
    elif i == ip and j == jp+1:
        rs = res.insert(j, s2[j])
        cost += 1
        print(f'i:{i}, j:{j}; insert {s2[j]}, cost: {cost}; res: {sh_res(res, j)}')
    else:
        assert False, f'i: {i}, j: {j}; ip: {ip}, jp: {jp}'
    ip = i
    jp = j
```

```
In [97]: # s1 = 'abcd'
# s2 = 'acfg'

s1 = 'intention'
s2 = 'execution'
da = edit_distance_align(s1, s2, substitution_cost=2)
da
```

```
Out[97]: [(0, 0),
(1, 0),
(2, 0),
(3, 0),
(4, 1),
(4, 2),
(4, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9)]
```

In [99]: `show_ed_path(s1, s2, da)`

```
i:0, j:0; init, cost: 0; res: intention
i:1, j:0; remove i, cost: 1; res: Intention
i:2, j:0; remove n, cost: 2; res: Ntention
i:3, j:0; remove t, cost: 3; res: Tention
i:4, j:1; save e, cost: 3; res: Ention
i:4, j:2; insert x, cost: 4; res: eXntion
i:4, j:3; insert e, cost: 5; res: exEntion
i:4, j:4; insert c, cost: 6; res: exeCntion
i:5, j:5; change n -> u; cost: 8; res: execuTion
i:6, j:6; save t, cost: 8; res: execuTion
i:7, j:7; save i, cost: 8; res: execuTion
i:8, j:8; save o, cost: 8; res: execuTion
i:9, j:9; save n, cost: 8; res: execuTion
```

С точки зрения приложений определение расстояния Левенштейна между словами или строками обладает следующими недостатками:

- При перестановке местами слов или частей слов получаются сравнительно большие расстояния.
- Расстояния между совершенно разными короткими словами оказываются небольшими, в то время как расстояния между очень похожими длинными словами оказываются значительными.

Другие метрики в NLTK: <http://www.nltk.org/howto/metrics.html> (<http://www.nltk.org/howto/metrics.html>)

Стемминг и лемматизация

-

- [к оглавлению](#)

Часто необходимо обрабатывать разные формы слова одинаково. В этом случае поможет переход от словоформ к их леммам (словарным формам лексем) или основам (ядерным частям слова, за вычетом словоизменятельных морфем)

Например, при поиске: по запросам “кошками” и “кошкам” ожидаются одинаковые ответы.

- **Стемминг** - это процесс нахождения основы слова, которая не обязательно совпадает с корнем слова.
- **Лемматизация** - приведение слова к словарной форме.

Морфология - это раздел лингвистики, который изучает структуру слов и их морфологические характеристики. Классическая морфология проанализирует слово *собака* примерно так: это существительное женского рода, оно состоит из *корня* собак и *окончания* а, окончание показывает, что слово употреблено в единственном числе и в именительном падеже.

Компьютерная морфология анализирует и синтезирует слова программными средствами. В наиболее привычной формулировке под морфологическим анализом слова подразумевается:

- определение леммы (базовой, канонической формы слова)
- определение грамматических характеристик слова.

В области автоматической обработки данных также используется термин **нормализация**, обозначающий постановку слова или словосочетания в **каноническую форму** (грамматические характеристики исходной формы при этом не выдаются). Обратная задача, т. е. постановка леммы

в нужную грамматическую форму, называется **порождением словоформы**.

Стемминг

-

[к оглавлению](#)

Стемминг отбрасывает суффиксы и окончания до неизменяемой формы слова

Примеры:

- кошка -> кошк
- кошками -> кошк
- пылесосы -> пылесос

В школьной грамматике **основой** считается **часть слова без окончания**.

- В большинстве случаев она не меняется при грамматических изменениях самого слова — так ведет себя, например, основа *слон* в словоформах: *слон, слону, слонами, слонов*.
- Но в некоторых словах основа может изменяться. Например, для словоформ *день, дню и дне* основами будут ден-, дн- и дн-, такое явление называется **чередованием**. Поэтому самый популярный на сегодня подход использует псевдоосновы (или машинные основы). Это неизменяемые начальные части слов. Для слова *день* такой неизменяемой частью будет *д-*. Формы некоторых слов могут образовываться от разных корней. Например, у слова *ходить* есть форма *шел*. Это называется **супплетивизмом**.

В русском языке супплетивизм и чередования очень распространены, поэтому **псевдоосновы часто получаются очень короткими**. Для русского языка **стемминг работает гораздо хуже, чем лемматизация**.

В стемминге есть только правила обрабатывания суффиксов и, возможно, небольшие словари исключений. Существует бесплатный инструмент для написания стеммеров — Snowball.

```
In [9]: # Snowball - Наиболее распространенный стеммер из проекта Apache Lucene
# Работает для нескольких языков, включая русский
from nltk.stem import SnowballStemmer
SnowballStemmer.languages
```

```
Out[9]: ('arabic',
'danish',
'dutch',
'english',
'finnish',
'french',
'german',
'hungarian',
'italian',
'norwegian',
'porter',
'portuguese',
'romanian',
'russian',
'spanish',
'swedish')
```

```
In [11]: import re
snb_stemmer_ru = SnowballStemmer('russian')
print(snb_stemmer_ru.stem('кошки'))
print(snb_stemmer_ru.stem('кошечки'))
```

КОШК
КОШЕЧК

```
In [12]: # загружаем текст:
with open('phm.txt ') as f:
    lines = [l for l in f]
print(len(lines))
print(lines[0])
```

3
Постгуманизм – рациональное мировоззрение, основанное на представлении, что эволюция человека не завершена и может быть продолжена в будущем. Эволюционное развитие должно привести к становлению постчеловека – гипотетической стадии эволюции человеческого вида, строение и возможности которого стали бы отличными от современных человеческих в результате активного использования передовых технологий преобразования человека. Постгуманизм признаёт неотъемлемыми правами совершенствование человеческих возможностей (физических, интеллектуальных и т. п.) и достижение физического бессмертия. В отличие от трансгуманизма, под определением постгуманизма также понимается критика классического гуманизма, подчёркивающая изменение отношения человека к себе, обществу, окружающей среде и бурно развивающимся технологиям, но окончательно разница между транс- и постгуманизмом не определена и остаётся предметом дискуссий.

```
In [13]: from razdel import sentenize
from razdel import tokenize
```

```
In [14]: snt = list(sentenize(lines[0]))
tok = list(tokenize(snt[0].text))
w = re.compile('^[а-яА-ЯёЁ]*$')
# предложение превращено в последовательность стем русских слов:
[snb_stemmer_ru.stem(t.text) for t in tok if w.search(t.text)]
```

```
Out[14]: ['постгуманизм',
'рациональн',
'мировоззрен',
'основа',
'на',
'представлен',
'что',
'эволюц',
'человек',
'не',
'заверш',
'и',
'может',
'быт',
'продолж',
'в',
'будущ']
```

Snowball использует **систему суффиксов и окончаний** для предсказания части речи и грамматических параметров. Так как одно и то же окончание может принадлежать разным частям речи или различным парадигмам, его оказывается недостаточно для точного предсказания. Применение суффиксов позволяет повысить точность.

Система реализуется на языке программирования в виде большого количества условных операторов, анализирующих самый длинный постфикс и его контекст. По окончании анализа слову приписывается часть речи и набор параметров, а найденное окончание (или псевдоокончание) отрезается. В итоге, помимо параметров, система возвращает стем.

Лемматизация

-

- [к оглавлению](#)

Лемматизация

У разных слов часто совпадает основа:

- пол : полу , пола , поле , полю , поля , пол , полем , полях , полям
- лев : левый, левая, лев

Из-за этого увеличивается многозначность и ухудшаются результаты работы приложений.

Лемматизация - приведение слова к словарной форме, например:

- кошки -> кошка
- кошками -> кошка

Морфологические анализаторы для русского языка:

Название	Open	Доб. словари	Объем слов.	Скорость	Python?
АОТ	Y	N	160 тыс.	60-90	N
MyStem	N	Y/N	>250 тыс.	100-120	Есть оболочка на Python
Pymorphy2	Y	N	250 тыс.	80-100	Y
TreeTagger	N	Y	210 тыс.	20-25	N

pymorphy2

- Код проекта: <https://github.com/kmike/pymorphy2> (<https://github.com/kmike/pymorphy2>)
- Документация проекта: <https://pymorphy2.readthedocs.io/en/stable/> (<https://pymorphy2.readthedocs.io/en/stable/>)

pip install pymorphy2

Словари распространяются отдельными пакетами. Для русского языка:

pip install -U pymorphy2-dicts-ru

Есть оптимизированная версия, потребуется настроенное окружение для сборки (компилятор C/C++ и т.д.).

Морфологический процессор с открытым исходным кодом, предоставляет все функции полного морфологического анализа и синтеза словоформ. 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. Словарь OpenCorpora содержит около 250 тыс. лемм, а также является полностью открытым и регулярно пополняемым.

Для анализа неизвестных слов в Pymorphy2 используются несколько методов, которые применяются последовательно. Изначально от слова отсекается префикс из набора известных префиксов и если остаток слова был найден в словаре, то отсеченный префикс приписывается к результатам разбора. Если этот метод не сработал, то аналогичные действия выполняются для префикса слова длиной от 1 до 5, даже если такой префикс является неизвестным. Затем, в случае неудачи, словоформа разбирается по окончанию. Для этого используется дополнительный автомат всех окончаний, встречающихся в словаре с имеющимися разборами.

```
In [1]: %pip install pymorphy2
```

```
In [15]: import pymorphy2

morph = pymorphy2.MorphAnalyzer()
```

```
In [162]: p = morph.parse('стали')
p
```

```
Out[162]: [Parse(word='стали', tag=OpencorporaTag('VERB,perf,intr plur,past,indc'), normal_form='стать', score=0.975342, methods_stack=((DictionaryAnalyzer(), 'стали', 945, 4),)),
Parse(word='стали', tag=OpencorporaTag('NOUN,inan,femn sing,gent'), normal_form='стал ь', score=0.010958, methods_stack=((DictionaryAnalyzer(), 'стали', 13, 1),)),
Parse(word='стали', tag=OpencorporaTag('NOUN,inan,femn plur,номн'), normal_form='стал ь', score=0.005479, methods_stack=((DictionaryAnalyzer(), 'стали', 13, 6),)),
Parse(word='стали', tag=OpencorporaTag('NOUN,inan,femn sing,dativ'), normal_form='стал ь', score=0.002739, methods_stack=((DictionaryAnalyzer(), 'стали', 13, 2),)),
Parse(word='стали', tag=OpencorporaTag('NOUN,inan,femn sing,loct'), normal_form='стал ь', score=0.002739, methods_stack=((DictionaryAnalyzer(), 'стали', 13, 5),)),
Parse(word='стали', tag=OpencorporaTag('NOUN,inan,femn plur,accs'), normal_form='стал ь', score=0.002739, methods_stack=((DictionaryAnalyzer(), 'стали', 13, 9),))]
```

```
In [163]: p[0].tag
```

```
Out[163]: OpencorporaTag('VERB,perf,intr plur,past,indc')
```

Метод `MorphAnalyzer.parse()` возвращает один или несколько объектов типа `Parse` с информацией о том, как слово может быть разобрано.

Тег - это набор граммем, характеризующих данное слово. Например, тег `'VERB,perf,intr plur,past,indc'` означает, что слово - глагол (VERB) совершенного вида (perf), непереходный (intr), множественного числа (plur), прошедшего времени (past), изъявительного наклонения (indc). Доступные граммемы описаны тут: <https://pymorphy2.readthedocs.io/en/latest/user/grammemes.html#grammeme-docs> (<https://pymorphy2.readthedocs.io/en/latest/user/grammemes.html#grammeme-docs>).

Далее: <https://pymorphy2.readthedocs.io/en/latest/user/guide.html> (<https://pymorphy2.readthedocs.io/en/latest/user/guide.html>).

`score` - это оценка $P(\text{tag}|\text{word})$, оценка вероятности того, что данный разбор правильный.

Разборы сортируются по убыванию `score`, поэтому везде в примерах берется первый вариант разбора из возможных. Оценки $P(\text{tag}|\text{word})$ помогают улучшить разбор, но их недостаточно для надежного снятия неоднозначности, как минимум по следующим причинам:

то, как нужно разбирать слово, зависит от соседних слов; rymorphy2 работает только на уровне отдельных слов; условная вероятность $P(\text{tag}|\text{word})$ оценена на основе сбалансированного набора текстов; в специализированных текстах вероятности могут быть другими - например, возможно, что в металлургических текстах $P(\text{NOUN}|\text{стали}) > P(\text{VERB}|\text{стали})$;

```
In [164]: #у каждого разбора есть нормальная форма, которую можно получить, обратившись к атрибуту
p[0].normalized
```

```
Out[164]: Parse(word='стать', tag=OpencorporaTag('INFN,perf,intr'), normal_form='стать', score=
1.0, methods_stack=((DictionaryAnalyzer(), 'стать', 945, 0),))
```

```
In [165]: snt = list(sentenize(lines[0]))
tok = list(tokenize(snt[0].text))
w = re.compile('^[а-яА-ЯёЁ]*$')
# предложение превращено в последовательность нормальных форм русских слов:
pt = [morph.parse(t.text) for t in tok if w.search(t.text)]
[w[0].normalized.word for w in pt]
```

```
Out[165]: ['постгуманизм',
'рациональный',
'мировоззрение',
'основать',
'на',
'представление',
'что',
'эволюция',
'человек',
'не',
'завершить',
'и',
'мочь',
'быть',
'продолжить',
'в',
'будущее']
```

Стоп-слова

-

- [к оглавлению](#)

Закон Ципфа

Закон Ципфа (Zipf's law, «ранг-частота») - эмперический закон, наблюдаемый для различных объектов в области физики, социологии, лингвистики и т.д., указывающий на то, что характеристики объектов (в частности, частота появления) имеют вид близкий к распределению Ципфа.

Распределение Ципфа - это дискретный закон распределения, имеющий степенную природу и близкий (но не идентичный) Дзета-распределению.

Пусть:

- N - количеств различных объектов (например, различных слов в тексте);
- k - ранг, т.е. порядковый номер объекта (например, слова), в отсортированной по частоте последовательности объектов;
- s - параметр распределения, отражающий степень убывания частоты.

тогда распределение имеет вид:

$$f(k; s, N) = \frac{1/k^s}{\sum_{n=1}^N (1/n^s)}$$

Свойство объектов распределенных по этому закону:

- P_k - частота встречаемости объекта с рангом k

$$P_k = P_1/k^s$$

- при $s = 1$:

$$P_k = P_1/k$$

Закон Ципфа в лингвистике - эмпирическая 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 обратно пропорциональной его порядковому номеру n (так называемому рангу этого слова).

Например:

- второе по используемости слово встречается примерно в два раза реже, чем первое
- третье - в три раза реже, чем первое (и так далее ...)

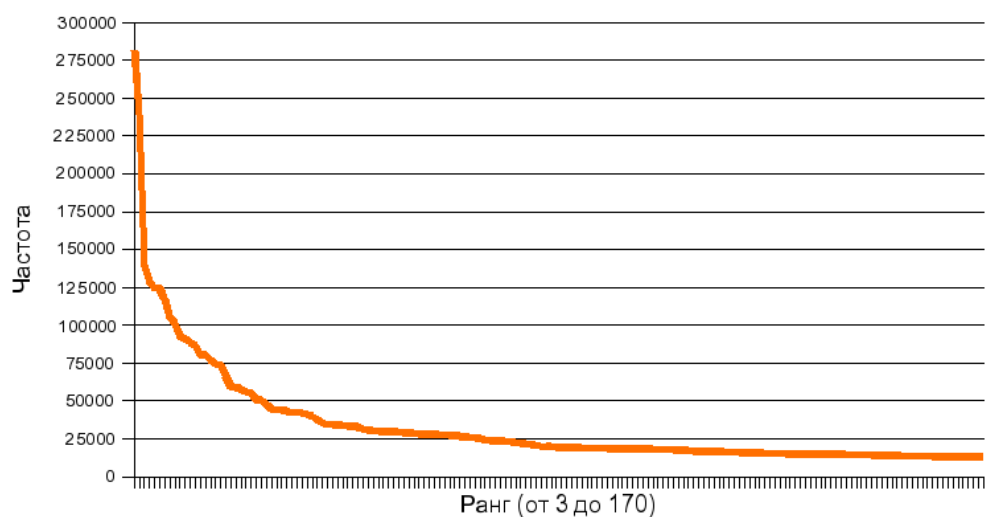
В естественных языках частоты слов имеют очень тяжелые хвосты и могут описываться распределением Ципфа с $s \rightarrow 1$ при $N \rightarrow \infty$ в случае если $s > 1$:

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} < \infty$$

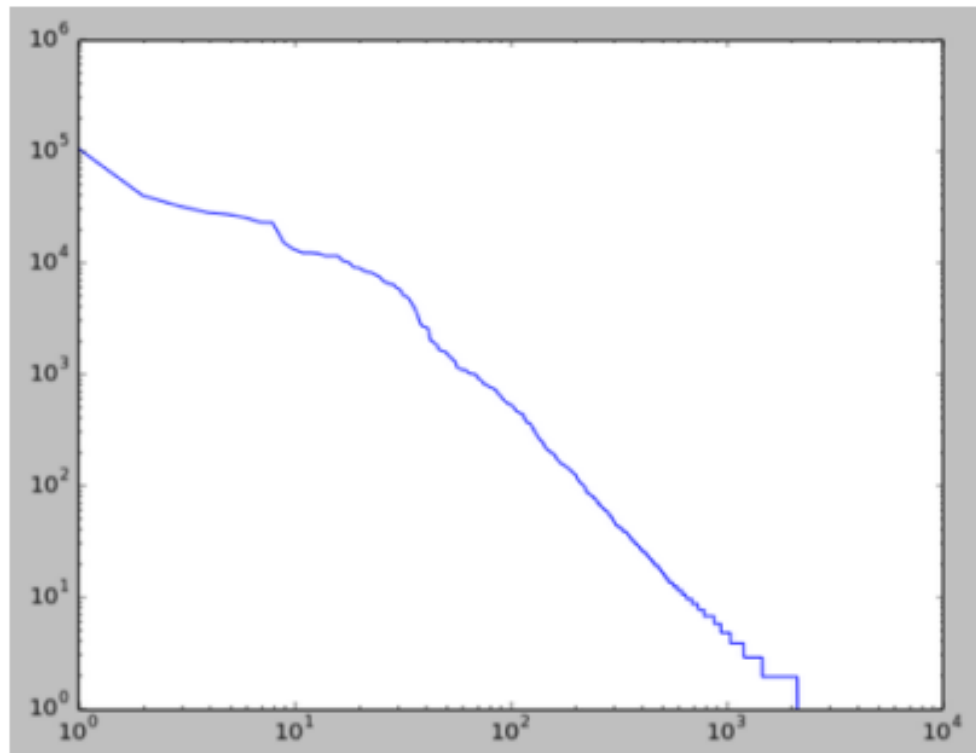
где ζ это Дзета-функция Римана.

В этом случае распределение Ципфа можно заменить Дзета распределением (дискретным распределением, в котором $k \in [1, \infty]$):

$$P(x = k; s) = \frac{k^{-s}}{\zeta(s)}$$



Пример: (распределение частот слов в статьях русской Википедии)



Пример (распределение частот слов в крупном художественном произведении)

Стоп-слова

- Для крупных текстов большинство слов из головы распределения обычно характеризуют язык, а не текст
- Обычно это служебные слова, определяющие структуру предложения (например: предлоги, артикли, частицы), местоимения (фактически, универсальные указатели) и самые общие понятия используемые в письменной речи
- Во многих задачах использование наиболее частотных слов создает шум и их выгодно исключать из рассмотрения. За такими словами закрепился термин **стоп-слова**(stop words).

Пример стоп-слов русского языка:

(конкретный состав стоп-слов зависит от рассматриваемого корпуса текстов, длины списка и т.д.)

-	еще	него	сказать
а	ж	нее	со
без	же	ней	совсем
более	жизнь	нельзя	так
больше	за	нет	такой
будет	зачем	ни	там
будто	здесь	нибудь	тебя
бы	и	никогда	тем
был	из	ним	теперь
была	из-за	них	то
были	или	ничего	тогда
было	им	но	того
быть	иногда	ну	тоже
в	их	о	только
вам	к	об	том
вас	кажется	один	тот

вдруг	как	он	три
ведь	какая	она	тут
во	какой	они	ты
вот	когда	опять	у
впрочем	конечно	от	уж
все	которого	перед	уже
всегда	которые	по	хорошо
всего	кто	под	хоть
всех	куда	после	чего
всю	ли	потом	человек
вы	лучше	потому	чем
г	между	почти	через
где	меня	при	что
говорил	мне	про	чтоб
да	много	раз	чтобы
даже	может	разве	чуть
два	можно	с	эти
для	мой	сам	этого
до	моя	свое	этой
другой	мы	свою	этом
его	на	себе	этот
ее	над	себя	эту
ей	надо	сегодня	я
ему	наконец	сейчас	
если	нас	сказал	
есть	не	сказала	

```
In [21]: # pip install -U nltk
import nltk
nltk.__version__
# # nltk.download()
```

Out[21]: '3.7'

```
In [22]: # Загрузка списков стоп-слов в NLTK:
nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\alpha\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

Out[22]: True

```
In [23]: from nltk.corpus import stopwords
```

```
In [24]: print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'ca', 'n', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'does', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

```
In [25]: ru_stop_words = stopwords.words('russian')
print(ru_stop_words)
```

```
['и', 'в', 'во', 'не', 'что', 'он', 'на', 'я', 'с', 'со', 'как', 'а', 'то', 'все', 'она', 'так', 'его', 'но', 'да', 'ты', 'к', 'у', 'же', 'вы', 'за', 'бы', 'по', 'только', 'ее', 'мне', 'было', 'вот', 'от', 'меня', 'еще', 'нет', 'о', 'из', 'ему', 'теперь', 'когда', 'даже', 'ну', 'вдруг', 'ли', 'если', 'уже', 'или', 'ни', 'быть', 'был', 'него', 'до', 'вас', 'нибудь', 'опять', 'уж', 'вам', 'ведь', 'там', 'потом', 'себя', 'ничего', 'ей', 'может', 'они', 'тут', 'где', 'есть', 'надо', 'ней', 'для', 'мы', 'тебя', 'их', 'чем', 'была', 'сам', 'чтоб', 'без', 'будто', 'чего', 'раз', 'тоже', 'себе', 'под', 'будет', 'ж', 'тогда', 'кто', 'этот', 'того', 'потому', 'этого', 'какой', 'совсем', 'ним', 'здесь', 'этом', 'один', 'почти', 'мой', 'тем', 'чтобы', 'нее', 'сейчас', 'были', 'куда', 'зачем', 'всех', 'никогда', 'можно', 'при', 'наконец', 'два', 'об', 'другой', 'хоть', 'после', 'над', 'больше', 'тот', 'через', 'эти', 'нас', 'про', 'всего', 'них', 'какая', 'много', 'разве', 'три', 'эту', 'моя', 'впрочем', 'хорошо', 'свою', 'этой', 'перед', 'иногда', 'лучше', 'чуть', 'том', 'нельзя', 'такой', 'им', 'более', 'всегда', 'конечно', 'всю', 'между']
```

Мешок слов

-

- [к оглавлению](#)

Мешок слов (bag-of-words, BoW) – модель, которая используется при обработке естественного языка для представления текста. Для представления текста ведется подсчет того, сколько раз каждое отдельное слово появляется в тексте, таким образом текст преобразуется в вектор, координатами которого являются рассматриваемые слова, а значениями - частоты слов.

- Любая информация о порядке или структуре слов в документе отбрасывается. Модель касается только того, встречаются ли в документе известные слова, а не где в документе.
 - Интуиция заключается в том, что документы похожи, если они имеют похожее содержание.
- Модели мешка слов могут отличаться способами в определении словарного запаса известных слов (или токенов) и в том, как оценивать наличие известных слов.
- Перед подсчетом можно применить методы предварительной обработки, описанные в выше.

```
In [40]: import re
import itertools as it
from razdel import sentenize
from razdel import tokenize
import pymorphy2
morph = pymorphy2.MorphAnalyzer()
```

```
In [41]: # получаем все интересные нам токены:
w_regex = re.compile('[а-яА-ЯёЁ]*$') # re.compile('[а-яА-ЯёЁ\.]*$')
with open("AnnaKarenina_.txt", encoding="cp1251") as f:
    book_tokens = [t.text.lower() for t in tokenize(f.read()) if w_regex.search(t.text)]
```

```
In [42]: print(print(len(book_tokens), book_tokens[:150]))
```

```
266954 ['лев', 'николаевич', 'толстой', 'анна', 'каренина', 'мне', 'отмщение', 'и', 'а
з', 'воздам', 'часть', 'первая', 'все', 'счастливые', 'семьи', 'похожи', 'друг', 'на',
'друга', 'каждая', 'несчастливая', 'семья', 'несчастлива', 'все', 'смешалось', 'в', 'д
оме', 'облонских', 'жена', 'узнала', 'что', 'муж', 'был', 'в', 'связи', 'с', 'бывшею',
'в', 'их', 'доме', 'и', 'объявила', 'мужу', 'что', 'не', 'может', 'жить', 'с', 'ним',
'в', 'одном', 'доме', 'положение', 'это', 'продолжалось', 'уже', 'третий', 'день',
'и', 'мучительно', 'чувствовалось', 'и', 'самими', 'супругами', 'и', 'всеми', 'членам
и', 'семьи', 'и', 'домочадцами', 'все', 'члены', 'семьи', 'и', 'домочадцы', 'чувствова
ли', 'что', 'нет', 'смысла', 'в', 'их', 'сожительстве', 'и', 'что', 'на', 'каждом', 'п
остоялом', 'дворе', 'случайно', 'сошедшиеся', 'люди', 'более', 'связаны', 'между', 'со
бой', 'чем', 'они', 'члены', 'семьи', 'и', 'домочадцы', 'облонских', 'жена', 'не', 'вы
ходила', 'из', 'своих', 'комнат', 'мужа', 'третий', 'день', 'не', 'было', 'дома', 'дет
и', 'бегали', 'по', 'всему', 'дому', 'как', 'потерянные', 'англичанка', 'поссорилась',
'с', 'экономкой', 'и', 'написала', 'записку', 'приятельнице', 'прося', 'приискать', 'е
й', 'новое', 'место', 'повар', 'ушел', 'еще', 'вчера', 'со', 'двора', 'во', 'время',
'обеда', 'черная', 'кухарка', 'и', 'кучер', 'просили', 'расчета', 'на']
None
```

```
In [43]: # http://www.nltk.org/api/nltk.html#nltk.probability.FreqDist
from nltk.probability import FreqDist
fdist = FreqDist(book_tokens)
```

```
In [44]: print(f'Обработано токенов:{fdist.N()}; найдено различных токенов:{fdist.B()}')
```

```
Обработано токенов:266954; найдено различных токенов:32569
```

```
In [45]: print('Содержимое:', list(it.islice(fdist.items(), 10)))
```

```
Содержимое: [('лев', 1), ('николаевич', 2), ('толстой', 2), ('анна', 499), ('каренин
а', 45), ('мне', 682), ('отмщение', 1), ('и', 12916), ('аз', 1), ('воздам', 1)]
```

```
In [32]: print(f'Самое частое слово:{fdist.max()}, частота слова "анна":{fdist.get("анна")}')
```

```
Самое частое слово:и, частота слова "анна":499
```

```
In [33]: print(fdist.most_common(50))
```

```
[('и', 12916), ('не', 6537), ('что', 5765), ('в', 5720), ('он', 5551), ('на', 3594),
('она', 3434), ('с', 3327), ('я', 3212), ('как', 2660), ('но', 2581), ('его', 2578),
('это', 2223), ('к', 1983), ('ее', 1805), ('все', 1671), ('было', 1656), ('так', 141
5), ('сказал', 1412), ('а', 1391), ('то', 1388), ('же', 1325), ('ему', 1252), ('о', 12
43), ('за', 1139), ('левин', 1135), ('только', 1017), ('ты', 993), ('у', 913), ('был',
901), ('по', 834), ('когда', 831), ('для', 827), ('сказала', 827), ('бы', 822), ('от',
813), ('да', 812), ('теперь', 810), ('вы', 756), ('из', 735), ('была', 728), ('еще', 6
99), ('ей', 689), ('мне', 682), ('кити', 661), ('они', 646), ('него', 622), ('уже', 60
1), ('нет', 592), ('очень', 573)]
```

```
In [46]: ru_stop_words_s = set(ru_stop_words)
# фильтруем стоп-слова:
wtokens_wostw = [w for w in book_tokens[:150] if w not in ru_stop_words_s]
print(wtokens_wostw)
```

```
['лев', 'николаевич', 'толстой', 'анна', 'каренина', 'отмщение', 'аз', 'воздам', 'част
ь', 'первая', 'счастливые', 'семьи', 'похожи', 'друг', 'друга', 'каждая', 'несчастлива
я', 'семья', 'несчастлива', 'смешалось', 'доме', 'облонских', 'жена', 'узнала', 'муж',
'связи', 'бывшею', 'доме', 'объявила', 'мужу', 'жить', 'одном', 'доме', 'положение',
'это', 'продолжалось', 'третий', 'день', 'мучительно', 'чувствовалось', 'самими', 'суп
ругами', 'всеми', 'членами', 'семьи', 'домочадцами', 'члены', 'семьи', 'домочадцы', 'ч
увствовали', 'смысла', 'сожительстве', 'каждом', 'постоялом', 'дворе', 'случайно', 'со
шедшиеся', 'люди', 'связаны', 'собой', 'члены', 'семьи', 'домочадцы', 'облонских', 'же
на', 'выходила', 'своих', 'комнат', 'мужа', 'третий', 'день', 'дома', 'дети', 'бегал
и', 'всему', 'дому', 'потерянные', 'англичанка', 'поссорилась', 'экономкой', 'написал
а', 'записку', 'приятельнице', 'прося', 'приискать', 'новое', 'место', 'повар', 'уше
л', 'вчера', 'двора', 'время', 'обеда', 'черная', 'кухарка', 'кучер', 'просили', 'расч
ета']
```

```
In [35]: # самые частотные слова "Анны Карениной" после очистки от стоп-слов:
print(list(it.islice(((i, w, f) for i, (w, f) in enumerate(fdist.most_common(500)) \
                    if w not in ru_stop_words_s), 50)))
```

```
[(12, 'это', 2223), (18, 'сказал', 1412), (25, 'левин', 1135), (33, 'сказала', 827),
(44, 'кити', 661), (49, 'очень', 573), (53, 'вронский', 509), (56, 'анна', 499), (66,
'алексей', 429), (68, 'степан', 423), (69, 'аркадьич', 422), (72, 'александрович', 39
5), (81, 'время', 366), (82, 'мог', 357), (83, 'говорил', 357), (89, 'руку', 309), (9
0, 'долли', 302), (92, 'которые', 295), (97, 'лицо', 277), (98, 'сказать', 276), (102,
'дело', 272), (103, 'левина', 272), (108, 'который', 263), (111, 'своей', 251), (113,
'знал', 249), (116, 'жизни', 235), (117, 'говорить', 234), (118, 'знаю', 233), (121,
'которое', 231), (124, 'пред', 224), (125, 'хотел', 219), (127, 'сергей', 219), (129,
'нужно', 217), (130, 'человек', 215), (131, 'прежде', 215), (132, 'глаза', 214), (134,
'могу', 214), (135, 'видел', 214), (137, 'тебе', 213), (139, 'тотчас', 211), (141, 'чу
вствовал', 210), (143, 'вронского', 205), (145, 'одно', 202), (146, 'своего', 199), (1
47, 'могла', 199), (148, 'свое', 198), (149, 'иванович', 191), (153, 'думал', 189), (1
54, 'глядя', 189), (156, 'говорила', 184)]
```

Векторное представление документа

-

- [к оглавлению](#)

Все слова (в более общем случае - термы: слова и другие значимые элементы текста) которые встречаются в документах обрабатываемой коллекции, можно упорядочить. Если теперь для некоторого документа выписать по порядку веса всех термов, включая те, которых нет в этом документе, получится вектор, который и будет представлением данного документа в векторном пространстве.

- Размерность этого вектора, как и размерность пространства, равна количеству различных термов во всей коллекции, и является одинаковой для всех документов.

Записывая формально, документ j описывается вектором:

$$d_j = (w_{1j}, w_{2j}, \dots, w_{nj})$$

где d_j — векторное представление j -го документа, где w_{ij} — вес i -го слова в j -м документе, n — общее количество различных термов во всех документах коллекции.

Располагая таким представлением для всех документов, можно, например, находить расстояние между точками пространства и тем самым решать задачу подобию документов — чем ближе расположены точки, тем больше похожи соответствующие документы.

Методы взвешивания термов

Для полного определения векторной модели необходимо указать, каким именно образом будет отыскиваться вес термина в документе. Существует несколько стандартных способов задания функции взвешивания:

- **булевский вес** — равен 1, если терм встречается в документе и 0 в противном случае;
- **tf** (term frequency, частота термина) — вес определяется как функция от количества вхождений термина в документе;
- **tf-idf** (term frequency — inverse document frequency, частота термина — обратная частота документа) — вес определяется как произведение функции от количества вхождений термина в документ и функции от величины, обратной количеству документов коллекции, в которых встречается этот терм.

Косинусное сходство

Косинусное сходство — это мера сходства между двумя векторами предгильбертового пространства, которая используется для измерения косинуса угла между ними.

Если даны два вектора признаков, A и B, то косинусное сходство, $\cos(\theta)$, может быть представлено используя скалярное произведение и норму:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

косинусное сходство двух документов изменяется в диапазоне от 0 до 1, поскольку частота термина (например, веса tf-idf) не может быть отрицательной. Угол между двумя векторами частоты термина не может быть больше, чем 90° .

Одна из причин популярности косинусного сходства состоит в том, что **оно эффективно в качестве оценочной меры, особенно для разреженных векторов**, так как необходимо учитывать только ненулевые измерения.

Пример 1: Тривиальный пример с векторизацией на основе подсчета слов:

```
In [47]: import re
import itertools as it
from razdel import sentenize
from razdel import tokenize
import pymorphy2
from nltk.corpus import stopwords

import numpy as np
from numpy.linalg import norm

import nltk, string
from sklearn.feature_extraction.text import (CountVectorizer, TfidfVectorizer)
```

```
In [49]: # корпус текстов:
corpus = ['This is the first document.',
          'This document is the second document.',
          'And this is the third one.',
          'Is this the first document?']

# создание векторизатора:
cv = CountVectorizer()

# векторизуем корпус:
corpus_cv = cv.fit_transform(corpus)
```

```
In [38]: # рассмотренные токены:
cv.get_feature_names()
```

C:\Users\alpha\.conda\envs\teach_e2\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
warnings.warn(msg, category=FutureWarning)

```
Out[38]: ['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
```

```
In [51]: cv_ar = corpus_cv.toarray()
cv_ar
```

```
Out[51]: array([[0, 1, 1, 1, 0, 0, 1, 0, 1],
                [0, 2, 0, 1, 0, 1, 1, 0, 1],
                [1, 0, 0, 1, 1, 0, 1, 1, 1],
                [0, 1, 1, 1, 0, 0, 1, 0, 1]], dtype=int64)
```

```
In [52]: norm(cv_ar, axis=1)
```

```
Out[52]: array([2.23606798, 2.82842712, 2.44948974, 2.23606798])
```

```
In [53]: # нормализация:
ca_arn = cv_ar / norm(cv_ar, axis=1)[:, np.newaxis]
```

```
In [54]: ca_arn
```

```
Out[54]: array([[0.         , 0.4472136 , 0.4472136 , 0.4472136 , 0.         ,
                 0.         , 0.4472136 , 0.         , 0.4472136 ],
                [0.         , 0.70710678, 0.         , 0.35355339, 0.         ,
                 0.35355339, 0.35355339, 0.         , 0.35355339],
                [0.40824829, 0.         , 0.         , 0.40824829, 0.40824829,
                 0.         , 0.40824829, 0.40824829, 0.40824829],
                [0.         , 0.4472136 , 0.4472136 , 0.4472136 , 0.         ,
                 0.         , 0.4472136 , 0.         , 0.4472136 ]])
```

```
In [55]: ca_arn @ ca_arn.T
```

```
Out[55]: array([[1.         , 0.79056942, 0.54772256, 1.         ],
                [0.79056942, 1.         , 0.4330127 , 0.79056942],
                [0.54772256, 0.4330127 , 1.         , 0.54772256],
                [1.         , 0.79056942, 0.54772256, 1.         ]])
```

Использование TfidfVectorizer

```
In [56]: # создание векторизатора:
tv = TfidfVectorizer()

# векторизуем корпус:
corpus_tv = tv.fit_transform(corpus)
```

```
In [57]: # рассмотренные токены:
tv.get_feature_names()
```

```
Out[57]: ['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
```

```
In [58]: corpus_tv.toarray()
```

```
Out[58]: array([[0.          , 0.46979139, 0.58028582, 0.38408524, 0.          ,
                0.          , 0.38408524, 0.          , 0.38408524],
               [0.          , 0.6876236 , 0.          , 0.28108867, 0.          ,
                0.53864762, 0.28108867, 0.          , 0.28108867],
               [0.51184851, 0.          , 0.          , 0.26710379, 0.51184851,
                0.          , 0.26710379, 0.51184851, 0.26710379],
               [0.          , 0.46979139, 0.58028582, 0.38408524, 0.          ,
                0.          , 0.38408524, 0.          , 0.38408524]])
```

```
In [59]: pairwise_similarity = corpus_tv * corpus_tv.T
pairwise_similarity.toarray()
```

```
Out[59]: array([[1.          , 0.64692568, 0.30777187, 1.          ],
               [0.64692568, 1.          , 0.22523955, 0.64692568],
               [0.30777187, 0.22523955, 1.          , 0.30777187],
               [1.          , 0.64692568, 0.30777187, 1.          ]])
```

```
In [ ]:
```

Пример 2: Векторизация данных реального новостного потока

- Источник данных: <https://webhose.io/free-datasets/russian-news-articles/> (<https://webhose.io/free-datasets/russian-news-articles/>)
- альтернатива: https://github.com/RossiiaSegodnya/ria_news_dataset (https://github.com/RossiiaSegodnya/ria_news_dataset)

Этап 1: загрузка данных

```
In [60]: import json
from os import listdir
from os.path import isfile, join
```



```
In [61]: # получение имен всех файлов, находящихся по определенному пути:
news_path = './news'
news_files = [f for f in listdir(news_path) if isfile(join(news_path, f))]
news_files[:5], news_files[-5:], len(news_files)
```

```
Out[61]: (['news_0000001.json',
           'news_0000002.json',
           'news_0000003.json',
           'news_0000004.json',
           'news_0000005.json'],
          ['news_0000995.json',
           'news_0000996.json',
           'news_0000997.json',
           'news_0000998.json',
           'news_0000999.json'],
          999)
```

```
In [62]: with open(join(news_path, news_files[0]), 'r', encoding='utf-8') as f:
         news_js = json.load(f)
```

```
news_js
```

```
Out[62]: {'organizations': [],
'uuid': '99bbd8fc99f9458417204a7107d21a0e03272d60',
'thread': {'social': {'gplus': {'shares': 0},
'pinterest': {'shares': 0},
'vk': {'shares': 0},
'linkedin': {'shares': 0},
'facebook': {'likes': 1, 'shares': 1, 'comments': 0},
'stumbledupon': {'shares': 0}},
'site_full': 'www.newsru.com',
'main_image': 'http://image.newsru.com/v2/02/2016/10//.jpg',
'site_section': 'http://feeds.newsru.com/com/www/news/main',
'section_title': 'NEWSru.com :: Важные новости',
'url': 'http://www.newsru.com/world/02oct2016/gulens.html',
'country': 'US',
'domain_rank': 3073,
'title': 'В Турции задержали очередного родственника Фетхуллага Гюлена - его брата',
'performance_score': 0,
'site': 'newsru.com',
'participants_count': 0,
'title_full': 'В Турции задержали очередного родственника Фетхуллага Гюлена - его брата',
'spam_score': 0.0,
'site_type': 'news',
'published': '2016-10-02T21:53:00.000+03:00',
'replies_count': 0,
'uuid': '99bbd8fc99f9458417204a7107d21a0e03272d60'},
'author': '',
'url': 'http://www.newsru.com/world/02oct2016/gulens.html',
'ord_in_thread': 0,
'title': 'В Турции задержали очередного родственника Фетхуллага Гюлена - его брата',
'locations': [],
'entities': {'persons': [], 'locations': [], 'organizations': []},
'highlightText': '',
'language': 'russian',
'persons': [],
'text': 'В Турции задержали очередного родственника Фетхуллага Гюлена - его брата 1
6:53 16:53 \nТурецкая полиция задержала в городе Измир на западе страны брата оппози
ционного исламского проповедника Фетхуллага Гюлена Кутбеттина. Живущего в США проповед
ника Анкара считает вдохновителем попытки провалившегося переворота. Кутбеттин Гюлен р
азыскивался по обвинению в причастности к деятельности организации, возглавляемой его
братом. Его доставили на допрос в Управление безопасности и, вероятно, вскоре предьявя
т обвинение. Операцию по задержанию провела полиция Измира на основе оперативных данны
х о том, что подозреваемый скрывается в доме своего родственника в районе Газиемир, пе
редает РИА "Новости" . ТАСС напоминает, что 23 сентября власти Турции задержали племян
ницу Гюлена Эмине. Задержание прошло в уезде Эрдемит западной провинции Балыкесир. Выя
снилось, что она значительную часть телефонных разговоров вела с одним абонентом в СШ
А. Кроме того, у нее изъято большое количество фотографий и книг Гюлена. В августе пол
иция задержала племянника Гюлена Кемаля Гюлена, телеведущего и адвоката. Он был задерж
ан в одной из деревень в провинции Кастамону, где скрывался с середины июля после пров
ала заговора. Еще раньше был задержан другой племянник проповедника Адбуллах Коруджук.
В ночь на 16 июля в Турции группа мятежников совершила попытку военного переворота. Ос
новное противостояние развернулось в Анкаре и Стамбуле. Погибли более 240 турецких гра
ждан, более 2 тысяч человек получили ранения, мятеж был подавлен. Власти Турции обвини
ли Гюлена в причастности к попытке переворота и потребовали от США его экстрадиции. Са
м Гюлен осудил мятеж и заявил о своей непричастности. По обвинению в причастности к ор
ганизации Гюлена в Турции после мятежа были арестованы около 32 тысяч человек.',
'external_links': [],
'published': '2016-10-02T21:53:00.000+03:00',
'crawled': '2016-10-02T17:00:29.521+03:00',
'highlightTitle': ''}
```

```
In [63]: news_js['text']
```

```
Out[63]: 'В Турции задержали очередного родственника Фетхуллага Гюлена - его брата 16:53 1
6:53 \nТурецкая полиция задержала в городе Измир на западе страны брата оппозиционного
исламского проповедника Фетхуллага Гюлена Кутбеттина. Живущего в США проповедника Анка
ра считает вдохновителем попытки провалившегося переворота. Кутбеттин Гюлен разыскивал
ся по обвинению в причастности к деятельности организации, возглавляемой его братом. Е
го доставили на допрос в Управление безопасности и, вероятно, вскоре предъявят обвинен
ие. Операцию по задержанию провела полиция Измира на основе оперативных данных о том,
что подозреваемый скрывается в доме своего родственника в районе Газиимир, передает РИ
А "Новости" . ТАСС напоминает, что 23 сентября власти Турции задержали племянницу Гюле
на Эмине. Задержание прошло в уезде Эрдемит западной провинции Балыкесир. Выяснилось,
что она значительную часть телефонных разговоров вела с одним абонентом в США. Кроме т
ого, у нее изъято большое количество фотографий и книг Гюлена. В августе полиция задер
жала племянника Гюлена Кемаля Гюлена, телеведущего и адвоката. Он был задержан в одной
из деревень в провинции Кастамону, где скрывался с середины июля после провала заговор
а. Еще раньше был задержан другой племянник проповедника Амбуллах Коруджук. В ночь на
16 июля в Турции группа мятежников совершила попытку военного переворота. Основное про
тивостояние развернулось в Анкаре и Стамбуле. Погибли более 240 турецких граждан, боле
е 2 тысяч человек получили ранения, мятеж был подавлен. Власти Турции обвинили Гюлена
в причастности к попытке переворота и потребовали от США его экстрадиции. Сам Гюлен ос
удил мятеж и заявил о своей непричастности. По обвинению в причастности к организации
Гюлена в Турции после мятежа были арестованы около 32 тысяч человек.'
```

```
In [64]: news_texts_corpus = []
for nf in news_files:
    with open(join(news_path, nf), 'r', encoding='utf-8') as f:
        news_texts_corpus.append(json.load(f)['text'])

news_texts_corpus[42], len(news_texts_corpus)
```

```
Out[64]: ('Уланова: чтобы охарактеризовать Гамову, достаточно одного слова – великая 22:02. В
oleyбол Либери «Динамо Казань» Екатерина Уланова после прощального матча Екатерины Гам
овой поделилась эмоциями, связанными с уходом Гамовой из спорта. «Слёзы на глаза накат
ываются, и мурашки по коже. Грустно, хотя понимаешь, конечно, что все мы рано или позд
но будем уходить из спорта. Я очень счастливый человек, потому что мне удалось поиграт
ь с Катей и в сборной, и в клубе. Какими словами я охарактеризовала бы Гамову? Мне дос
таточно одного слова – великая. И в жизни, и в спорте. Почему у нас не получилось шоу?
Не знаю, что ответить на этот вопрос. Не мы решали. Сколько ни пытались сделать шоу в
женском волейболе, не получается. Может быть, женский характер не позволяет раскрепост
иться и сыграть в своё удовольствие. Да, сегодня была борьба, игра. Просто мы ещё не у
меем делать шоу, не готовы к этому», – приводит слова Улановой «Спорт Бизнес Onlin
e」.',
999)
```

```
In [65]: len(news_texts_corpus)
```

```
Out[65]: 999
```

Этап 2: подготовка: токенизация, очистка от стоп слов, лемматизация

```
In [158]: import re
import itertools as it
from razdel import sentenize
from razdel import tokenize
import pymorphy2
from nltk.corpus import stopwords

import nltk, string
from sklearn.feature_extraction.text import TfidfVectorizer
```

Готовим свой токенизатор (с нормализацией) и список стоп-слов:

```
In [159]: w_regex = re.compile('[а-яА-ЯёЁ]*$')
morph = pymorphy2.MorphAnalyzer()

def n_tokenizer(news_str):
    return [morph.parse(t.text.lower())[0].normalized.word for t in tokenize(news_str)
            if w_regex.search(t.text)]
```

```
In [160]: news_texts_corpus[0]
```

```
Out[160]: 'В Турции задержали очередного родственника Фетхуллага Гюлена - его брата 16:53 1
6:53 \nТурецкая полиция задержала в городе Измир на западе страны брата оппозиционного
исламского проповедника Фетхуллага Гюлена Кутбеттина. Живущего в США проповедника Анка
ра считает вдохновителем попытки провалившегося переворота. Кутбеттин Гюлен разыскивал
ся по обвинению в причастности к деятельности организации, возглавляемой его братом. Е
го доставили на допрос в Управление безопасности и, вероятно, вскоре предъявят обвинен
ие. Операцию по задержанию провела полиция Измира на основе оперативных данных о том,
что подозреваемый скрывается в доме своего родственника в районе Газиемир, передает РИ
А "Новости" . ТАСС напоминает, что 23 сентября власти Турции задержали племянницу Гюле
на Эмине. Задержание прошло в уезде Эрдемит западной провинции Балыкесир. Выяснилось,
что она значительную часть телефонных разговоров вела с одним абонентом в США. Кроме т
ого, у нее изъято большое количество фотографий и книг Гюлена. В августе полиция задер
жала племянника Гюлена Кемаля Гюлена, телеведущего и адвоката. Он был задержан в одной
из деревень в провинции Кастамону, где скрывался с середины июля после провала заговор
а. Еще раньше был задержан другой племянник проповедника Абуллах Коруджук. В ночь на
16 июля в Турции группа мятежников совершила попытку военного переворота. Основное про
тивостояние развернулось в Анкаре и Стамбуле. Погибли более 240 турецких граждан, боле
е 2 тысяч человек получили ранения, мятеж был подавлен. Власти Турции обвинили Гюлена
в причастности к попытке переворота и потребовали от США его экстрадиции. Сам Гюлен ос
удил мятеж и заявил о своей непричастности. По обвинению в причастности к организации
Гюлена в Турции после мятежа были арестованы около 32 тысяч человек.'
```

```
In [161]: # test:
n_tokenizer(news_texts_corpus[0])
```

```
Out[161]: ['в',
'турция',
'задержать',
'очередной',
'родственник',
'фетхуллаг',
'гюлен',
'он',
'брат',
'турецкий',
'полиция',
'задержать',
'в',
'город',
'измир',
'на',
'запад',
'страна',
'брат',
'-----']
```

n_stop_words

—

```
news_texts_corpus[:3]
```

Out[71]: ['В Турции задержали очередного родственника Фетхуллага Гюлена - его брата 16:53 16:53 \nТурецкая полиция задержала в городе Измир на западе страны брата оппозиционного исламского проповедника Фетхуллага Гюлена Кутбеттина. Живущего в США проповедника Анкара считает вдохновителем попытки провалившегося переворота. Кутбеттин Гюлен разыскивался по обвинению в причастности к деятельности организации, возглавляемой его братом. Его доставили на допрос в Управление безопасности и, вероятно, вскоре предъявят обвинение. Операцию по задержанию провела полиция Измира на основе оперативных данных о том, что подозреваемый скрывается в доме своего родственника в районе Газиимир, передает РИА "Новости" . ТАСС напоминает, что 23 сентября власти Турции задержали племянницу Гюлена Эмине. Задержание прошло в уезде Эрдемит западной провинции Балыкесир. Выяснилось, что она значительную часть телефонных разговоров вела с одним абонентом в США. Кроме того, у нее изъято большое количество фотографий и книг Гюлена. В августе полиция задержала племянника Гюлена Кемаля Гюлена, телеведущего и адвоката. Он был задержан в одной из деревень в провинции Кастамону, где скрывался с середины июля после провала заговора. Еще раньше был задержан другой племянник проповедника Амбуллах Коруджук. В ночь на 16 июля в Турции группа мятежников совершила попытку военного переворота. Основное противостояние развернулось в Анкаре и Стамбуле. Погибли более 240 турецких граждан, более 2 тысяч человек получили ранения, мятеж был подавлен. Власти Турции обвинили Гюлена в причастности к попытке переворота и потребовали от США его экстрадиции. Сам Гюлен осудил мятеж и заявил о своей непричастности. По обвинению в причастности к организации Гюлена в Турции после мятежа были арестованы около 32 тысяч человек.',
'Aizvērt karti Высокие потолки, нужен ремонт-всё подготовлено для капитального ремонта, окна на одну сторону но шума нет, газовое отопление-колонка Viessmann-выравненный паркет 68 евро в месяц, городские коммуникации, вода по счетчикам только за холодную, пластиковые окна, камин действующий, два сарая во дворе с занесением в земельную книгу (22, 77м2), земля в собственности. Pilsēta:',
'Теги Локомотив Руслан Пименов Юрий Семин Премьер-лига Россия Арсенал Тула Бывший нападающий «Локомотива» Руслан Пименов после матча 9-го тура премьер-лиги с «Арсеналом» (1:1) выразил мнение, что многие футболисты «железнодорожников» не отвечают требованиям главного тренера Юрия Семина. Футболист не стал конкретизировать. – Главный тренер дал указание футболистам подумать, как им играть лучше. Ожидаете какие-то меры со стороны Семина? – вопрос Пименову. – Команду нужно встряхнуть. Многие футболисты не отвечают требованиям Юрия Павловича. – Кто именно?']

```
In [109]: %%time
# создание векторизатора:
# vectorizer = TfidfVectorizer(tokenizer=n_tokenizer, stop_words=n_stop_words)
cv_news = CountVectorizer(tokenizer=n_tokenizer, stop_words=n_stop_words)

# векторизуем корпус:
news_corpus_cv = cv_news.fit_transform(news_texts_corpus[:])
```

C:\Users\alpha\.conda\envs\teach_e2\lib\site-packages\sklearn\feature_extraction\text.py:396: UserWarning: Your stop_words may be inconsistent with your preprocessing. Tokenizing the stop words generated tokens ['большой', 'весь', 'всё', 'ещё', 'мочь', 'нибыть', 'свой', 'хороший', 'это'] not in stop_words.
warnings.warn(

CPU times: total: 1min 54s
Wall time: 1min 54s

```
In [163]: news_fn = cv_news.get_feature_names()
news_fn[:20], news_fn[-20:], len(news_fn)
```

```
Out[163]: (['аба',
'абав',
'абай',
'абашидзе',
'аббас',
'аббревиатура',
'абделаизиз',
'абдувахоб',
'абелла',
'абель',
'абер',
'абзац',
'абзелиловский',
'аблязов',
'абметко',
'абонемент',
'абонент',
'аборт',
'абрам',
'абрамс'],
['ясир',
'ясли',
'ясно',
'ясность',
'ясный',
'ясса',
'ястреб',
'яуза',
'яундубулт',
'яффа',
'яхрома',
'яхта',
'яхтсменка',
'яценко',
'яценюк',
'ячейка',
'яшин',
'ёвмар',
'ёжик',
'ёмкость'],
17761)
```

```
In [164]: news_ar = news_corpus_cv.toarray()
news_ar[0][:40], len(news_ar[0]), news_ar.shape
```

```
Out[164]: (array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64),
          17761,
          (999, 17761))
```

```
In [165]: print(news_ar[0,:], len(news_ar[0,:]), sum(news_ar[0,:]), max(news_ar[0,:]))

[0 0 0 ... 0 0 0] 17761 170 10
```

```
In [166]: dict(zip(news_fn, news_ar[0]))
```

```
Out[166]: {'аба': 0,
          'абаев': 0,
          'абай': 0,
          'абашидзе': 0,
          'аббас': 0,
          'аббревиатура': 0,
          'абделаизиз': 0,
          'абдувахоб': 0,
          'абелла': 0,
          'абель': 0,
          'абер': 0,
          'абзац': 0,
          'абзелиловский': 0,
          'аблязов': 0,
          'абметко': 0,
          'абонемент': 0,
          'абонент': 1,
          'аборт': 0,
          'абрам': 0,
          'абрамов': 0}
```

```
In [168]: # news_ar = news_corpus_cv.toarray()
news_ar
```

```
Out[168]: array([[0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                ...,
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

```
In [167]: news_ar.shape
```

```
Out[167]: (999, 17761)
```

```
In [169]: cv_ar = corpus_cv.toarray()
cv_ar
```

```
Out[169]: array([[0, 1, 1, 1, 0, 0, 1, 0, 1],
                [0, 2, 0, 1, 0, 1, 1, 0, 1],
                [1, 0, 0, 1, 1, 0, 1, 1, 1],
                [0, 1, 1, 1, 0, 0, 1, 0, 1]], dtype=int64)
```

```
In [170]: # нормализация:
news_arn = news_ar / norm(news_ar, axis=1)[:, np.newaxis]

C:\Users\alpha\AppData\Local\Temp\ipykernel_2700\3015992622.py:2: RuntimeWarning: invalid value encountered in true_divide
  news_arn = news_ar / norm(news_ar, axis=1)[:, np.newaxis]
```

```
In [171]: news_sim_mx = news_arn @ news_arn.T
news_sim_mx
```

```
Out[171]: array([[1.          , 0.00919795, 0.          , ..., 0.06234117, 0.07357274,
        0.02700276],
       [0.00919795, 1.          , 0.01873172, ..., 0.          , 0.00888757,
        0.          ],
       [0.          , 0.01873172, 1.          , ..., 0.          , 0.04994384,
        0.03142363],
       ...,
       [0.06234117, 0.          , 0.          , ..., 1.          , 0.08605355,
        0.08967453],
       [0.07357274, 0.00888757, 0.04994384, ..., 0.08605355, 1.          ,
        0.1574812 ],
       [0.02700276, 0.          , 0.03142363, ..., 0.08967453, 0.1574812 ,
        1.          ]])
```

```
In [172]: n_idx = 41
news_texts_corpus[n_idx]
```

```
Out[172]: '2 октября 2016 02:45 SpaceX подозревает конкурентов во взрыве своей ракеты \nАмериканская компания SpaceX подозревает, что ее конкурент – консорциум United Launch Alliance – причастен к аварии ракеты Falcon 9. Информация об этом появилась в газете The Washington Post . \nКак уточняется, сотрудник SpaceX посетил объект ULA, расположенный на мысе Канаверал (штат Флорида) и попросил предоставить ему доступ на крышу одного из зданий, принадлежащих консорциуму. Здание располагается недалеко от пусковой площадки, где и произошла авария. В рамках расследования инцидента компания SpaceX хотела проверить одну особенность, вызвавшую подозрение. На видеозаписи взрыва специалисты компании обнаружили странную тень, а позже – белое пятно на здании ULA, расположенном неподалеку. \nКак представитель SpaceX объяснил конкурентам, его компания прорабатывает все возможные версии аварии. Но в ULA ему не разрешили попасть на крышу того самого здания. Сотрудники консорциума вызвали специалиста из Военно-вооруженных сил США. Он осмотрел крышу и не нашел ничего подозрительного, что могло быть связано со взрывом ракеты Falcon 9. \nПо данным газеты, сам Илон Маск , глава SpaceX, не отрицает версию о саботаже. ULA – совместное предприятие авиационного гиганта Boeing и Lockheed Martin. \n1 сентября на космодроме, расположенном на мысе Канаверал, на площадке SpaceX взорвалась ракета Falcon 9 с израильским спутником связи Amos-6. В результате инцидента никто не пострадал, напоминает ТАСС .'
```



```
In [173]: news_sim_mx[n_idx, :]
```

```
Out[173]: array([0.03815077, 0.          , 0.          , 0.04725417, 0.02177002,
                0.          , 0.00817014, 0.01593402, 0.03005714, 0.0194717 ,
                0.01467734, 0.01147381, 0.01314109, 0.03856908, 0.03442142,
                0.01600563, 0.09652011, 0.02781671, 0.02829423, 0.1485761 ,
                0.11553844, 0.01278379, 0.          , 0.0506523 , 0.02590476,
                0.06227069, 0.0467695 , 0.02904462, 0.01529841, 0.02421883,
                0.05632596, 0.03751621, 0.03389587, 0.05422932, 0.08814422,
                0.05757427, 0.          , 0.00685012, 0.00746705, 0.00678329,
                0.03387535, 1.          , 0.01703419, 0.54029124, 0.07769439,
                0.          , 0.02819913, 0.04462646, 0.03569335, 0.          ,
                0.04226541, 0.01807902, 0.04093638, 0.00628446, 0.05635924,
                0.03382274, 0.0250073 , 0.03609808, 0.02808508,          , nan,
                0.04695043, 0.          , 0.00647619, 0.01435472, 0.02091267,
                0.03391643, 0.05519707, 0.05812518, 0.07618379, 0.06582634,
                0.04151379, 0.0413057 , 0.02980982, 0.04306417, 0.0372441 ,
                0.04579358, 0.02909141, 0.          , 0.10539364, 0.04707222,
                0.11473807, 0.01539373, 0.          , 0.01632751, 0.00874305,
                0.          , 0.03941929, 0.04196363, 0.          , 0.03066255,
                0.01839902, 0.0294082 , 0.04050357, 0.01996587, 0.01376857,
                0.02726502, 0.02446004, 0.01004475, 0.02770130, 0.00405510])
```

```
In [174]: news_sim_mx[n_idx, :].argmax()
```

```
Out[174]: 59
```

```
In [175]: news_sim_mx[n_idx, 59]
```

```
Out[175]: nan
```

```
In [177]: news_sim_mx[n_idx, :].argsort()
```

```
Out[177]: array([ 61,  77, 868, 493, 911, 968, 647, 326, 510, 843,  49, 192,  45,
                753, 528, 315, 351, 529, 353,  85, 695, 702, 703, 893, 685, 450,
                676, 707, 675, 931, 890, 463, 379, 373,  88,  82, 826, 791, 604,
                 22, 208, 274, 264, 798, 306, 617, 783, 292, 994, 290, 227, 218,
                 5, 537, 983,  2,  36,  1, 239, 231, 241, 167, 582, 967, 838,
                285, 151, 185, 624,  99, 102, 706, 322, 355, 741, 735, 511, 747,
                177,  53, 699,  62, 664, 312,  39,  37, 305, 371, 469, 896, 920,
                913, 183, 435,  38, 451, 389, 919, 581, 474, 385, 328, 701, 686,
                 6, 627, 849, 127, 825,  84, 201, 646, 503, 311, 655, 578, 347,
                672, 944, 594, 663, 393, 518,  97, 773, 288, 600, 548, 497, 960,
                635, 810, 461, 423, 210, 182,  11, 284, 708, 731, 666, 585, 813,
                336, 832, 730, 508, 391, 346, 927, 286, 325, 884,  21, 955, 786,
                787, 785, 270, 110,  12, 811, 989, 100, 746, 609, 263,  94, 115,
                948, 643, 533, 781, 569, 698, 665, 691,  63, 400,  10, 797, 213,
                992, 396, 424, 170, 296, 603, 340,  28, 725, 184, 767,  81, 420,
                742, 417, 795, 411, 176, 904, 238, 230, 850, 487,  7,  15, 142,
                640, 471, 768, 689, 504, 662, 828, 993,  83, 413, 615, 268, 752,
                788, 720, 645, 220, 116, 709, 527, 852,  42, 104, 380, 237, 334,
                857, 766, 500, 961, 729,  51, 272, 490, 153, 984, 657,  90, 465,
                100, 101, 711, 715, 670, 173, 501, 160, 611, 375, 500, 300, 301])
```

```
In [149]: news_sim_mx[n_idx, 43]
```

```
Out[149]: 0.5402912369036817
```

```
In [180]: news_texts_corpus[885]
```

```
Out[180]: 'По предварительным данным, причиной взрыва могла стать утечка бытового газа. Инцидент
произошел в городе Харбин. Также сообщается, что в результате взрыва пострадали нескол
ько человек, их число уточняется. Фото: x'
```

