Лекция 9: Введение в обработку текста на естественном языке

Автор: Сергей Вячеславович Макрушин e-mail: SVMakrushin@fa.ru (mailto:SVMakrushin@fa.ru)

Финансовый университет, 2020 г.

При подготовке лекции использованы материалы:

- ...

V 0.1 18.10.2020

Разделы:

-

- к оглавлению

- расстояние левенштейна
- стемминг / лемматизация
-

In [1]:

```
# загружаем стиль для оформления презентации
from IPython.display import HTML
from urllib.request import urlopen
html = urlopen("file:./lec_v1.css")
HTML(html.read().decode('utf-8'))
```

Out[1]:

Метрики расстояния между строками

-

- к оглавлению
- Расстояние Левенштейна
- Задача динамического программирования
- Алгоритм поиска расстояния Левинштайна (The minimum edit distance algorithm was named by Wagner and Fischer)

Метрики расстояния для строк

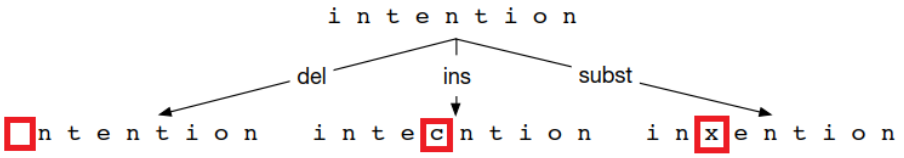
Часто требуется понять, насколько близкими являются две не совпадающих строки (слова). Это может потребоваться для:

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

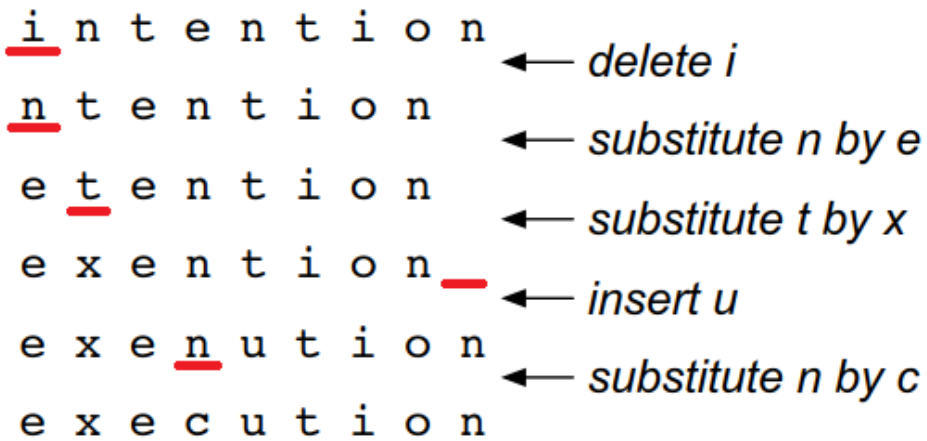
Расстояние Левенштейна

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 Дамрау - 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 $\mathrm{d}(S_1, S_2)$ 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:

$$\mathrm{d}(S_1, S_2) = \mathrm{D}(M, N)$$

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

$\mathrm{D}(i-1,j)+1$, операция удаления (цена: 1, на схеме обозначается как: $\uparrow$)
$\mathrm{D}(i,j-1)+1$, операция вставки (цена: 1, на схеме обозначается как: $\leftarrow$)
$\mathrm{D}(i-1,j-1)+\mathrm{m}(S_1[i], S_2[j])$, операция замены (цена m, на схеме обозначается как: $\nwarrow$)

Цена операции замены зависит от заменяемых символов:

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

Очевидно, что для расстояния Левинштайна справедливы следующие утверждения:

- $\mathrm{d}(S_1, S_2) \geqslant \big||S_1| - |S_2|\big|$
- $\mathrm{d}(S_1, S_2) \leqslant \max\big(|S_1|, |S_2|\big)$
- $\mathrm{d}(S_1, S_2) = 0 \Leftrightarrow S_1 = S_2$

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

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.

Алгоритм Вагнера - Фишера

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

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

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

	#	e	x	e	c	u	t	i	o	n
#	0	← 1	← 2	← 3	← 4	← 5	← 6	← 7	← 8	← 9
i	↑ **1**	↖← 2	↖← 3	↖← 4	↖← 5	↖← 6	↖← 7	↖ 6	← 7	← 8
n	↑ 2	↖← **3**	↖← 4	↖← 5	↖← 6	↖← 7	↖← 8	↑ 7	↖← 8	↖ 7
t	↑ 3	↖← 4	↖← **5**	↖← 6	↖← 7	↖← 8	↖ 7	←↑ 8	↖← 9	↑ 8
e	↑ 4	↖ 3	← 4	↖← **5**	← **6**	← 7	←↑ 8	↖← 9	↖← 10	↑ 9
n	↑ 5	↑ 4	↖← 5	↖← 6	↖← 7	↖← **8**	↖← 9	↖← 10	↖← 11	↖↑ 10
t	↑ 6	↑ 5	↖← 6	↖← 7	↖← 8	↖← 9	↖ **8**	← 9	← 10	←↑ 11
i	↑ 7	↑ 6	↖← 7	↖← 8	↖← 9	↖← 10	↑ 9	↖ **8**	← 9	← 10
o	↑ 8	↑ 7	↖← 8	↖← 9	↖← 10	↖← 11	↑ 10	↑ 9	↖ **8**	← 9
n	↑ 9	↑ 8	↖← 9	↖← 10	↖← 11	↖← 12	↑ 11	↑ 10	↑ 9	↖ **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 [2]:

```
from nltk.metrics.distance import (
    edit_distance,
    edit_distance_align,
    binary_distance,
    jaccard_distance,
    masi_distance,
    interval_distance,
    custom_distance,
    presence,
    fractional_presence,
)
```

In [3]:

```
edit_distance('intention', 'execution', substitution_cost=2)
```

Out[3]:

```
8
```

In [4]:

```
# результат при substitution_cost=1
edit_distance('intention', 'execution')
```

Out[4]:

```
5
```

In [5]:

```
edit_distance('пирвет', 'привет', substitution_cost=2)
```

Out[5]:

```
2
```

In [6]:

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

Out[6]:

```
1
```

In [7]:

```
s1 = 'intention'
s2 = 'execution'
```

In [8]:

```
ed = edit_distance_align(s1, s2, substitution_cost=2)
ed
```

Out[8]:

```
[(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 [9]:

```
s1
```

Out[9]:

```
'intention'
```

In [16]:

```
l1 = list(s1)
l1
```

Out[16]:

```
['i', 'n', 't', 'e', 'n', 't', 'i', 'o', 'n']
```

In [17]:

```
s = ''.join(l1)
s
```

Out[17]:

```
'intention'
```

In [18]:

```
res = l1
i = 3
```

In [19]:

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

Out[19]:

```
'int_e_ntion'
```

In [20]:

```
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
        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 [21]:

```
# s1 = 'abcd'
# s2 = 'acfg'

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

Out[21]:

```
[(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 [22]:

```
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

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

словоизменительных 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

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 [23]:

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

Out[23]:

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

In [24]:

```
import re
snb_stemmer_ru = SnowballStemmer('russian')
print(snb_stemmer_ru.stem('кошку'))
print(snb_stemmer_ru.stem('кошечки'))
```

```
кошк
кошечк
```

In [25]:

```
# 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 э
волюция человека не завершена и может быть продолжена в будущем. Эволюционное
развитие должно привести к становлению постчеловека — гипотетической стадии э
волюции человеческого вида, строение и возможности которого стали бы отличным
и от современных человеческих в результате активного использования передовых
технологий преобразования человека. Постгуманизм признаёт неотъемлемыми права
ми совершенствование человеческих возможностей (физиологических, интеллектуал
ьных и т. п.) и достижение физического бессмертия. В отличие от трансгуманизм
а, под определением постгуманизма также понимается критика классического гума
низма, подчёркивающая изменение отношения человека к себе, обществу, окружающ
ей среде и бурно развивающимся технологиям, но окончательно разница между тра
нс- и постгуманизмом не определена и остаётся предметом дискуссий.
```

In [26]:

```
from razdel import sentenize
from razdel import tokenize
```

In [20]:

```
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[20]:

```
['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 реализовывается 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

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?
AOT	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 [21]:

```
%pip install pymorphy2
```

```
Requirement already satisfied: pymorphy2 in c:\programdata\anaconda3\envs\pyt
orch_1_6\lib\site-packages (0.8)
Requirement already satisfied: docopt>=0.6 in c:\programdata\anaconda3\envs\p
ytorch_1_6\lib\site-packages (from pymorphy2) (0.6.2)
Requirement already satisfied: pymorphy2-dicts<3.0,>=2.4 in c:\programdata\an
aconda3\envs\pytorch_1_6\lib\site-packages (from pymorphy2) (2.4.393442.37109
85)
Requirement already satisfied: dawg-python>=0.7 in c:\programdata\anaconda3\e
nvs\pytorch_1_6\lib\site-packages (from pymorphy2) (0.7.2)
Note: you may need to restart the kernel to use updated packages.
```

In [27]:

```
import pymorphy2

morph = pymorphy2.MorphAnalyzer()
```

In [29]:

```
p = morph.parse('стали')
p
```

Out[29]:

```
[Parse(word='стали', tag=OpencorporaTag('VERB,perf,intr plur,past,indc'), nor
mal_form='стать', score=0.984662, methods_stack=((<DictionaryAnalyzer>, 'стал
и', 904, 4),)),
 Parse(word='стали', tag=OpencorporaTag('NOUN,inan,femn sing,gent'), normal_f
orm='сталь', score=0.003067, methods_stack=((<DictionaryAnalyzer>, 'стали', 1
3, 1),)),
 Parse(word='стали', tag=OpencorporaTag('NOUN,inan,femn sing,datv'), normal_f
orm='сталь', score=0.003067, methods_stack=((<DictionaryAnalyzer>, 'стали', 1
3, 2),)),
 Parse(word='стали', tag=OpencorporaTag('NOUN,inan,femn sing,loct'), normal_f
orm='сталь', score=0.003067, methods_stack=((<DictionaryAnalyzer>, 'стали', 1
3, 5),)),
 Parse(word='стали', tag=OpencorporaTag('NOUN,inan,femn plur,nomn'), normal_f
orm='сталь', score=0.003067, methods_stack=((<DictionaryAnalyzer>, 'стали', 1
3, 6),)),
 Parse(word='стали', tag=OpencorporaTag('NOUN,inan,femn plur,accs'), normal_f
orm='сталь', score=0.003067, methods_stack=((<DictionaryAnalyzer>, 'стали', 1
3, 9),))]
```

In [30]:

```
p[0].tag
```

Out[30]:

```
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(tag|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(tag|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; pymorphy2 работает только на уровне отдельных слов; условная вероятность P(tag|word) оценена на основе сбалансированного набора текстов; в специализированных текстах вероятности могут быть другими - например, возможно, что в металлургических текстах P(NOUN|стали) > P(VERB|стали);

In [31]:

```
#у каждого разбора есть нормальная форма, которую можно получить, обратившись к атрибутам п
p[0].normalized
```

Out[31]:

```
Parse(word='стать', tag=OpencorporaTag('INFN,perf,intr'), normal_form='стат
ь', score=1.0, methods_stack=((<DictionaryAnalyzer>, 'стать', 904, 0),))
```

In [98]:

```
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[98]:

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

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 появлвения) имеют вид близкий к распределению Ципфа.

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

Пусть:

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

тогда распрпеделение 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

$$P_k = P_1/k$$

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 ...)

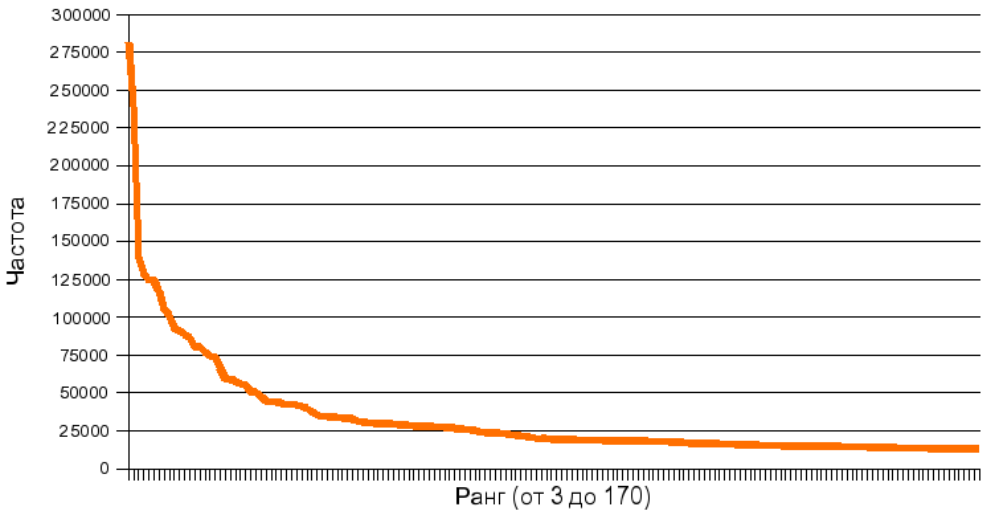
В естественных языках частоты слов имеют очень тяжелые ховосты и могут описываться распределением Ципфа с $s \to 1$ при $N \to \infty$ в случае если $s > 1$:

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

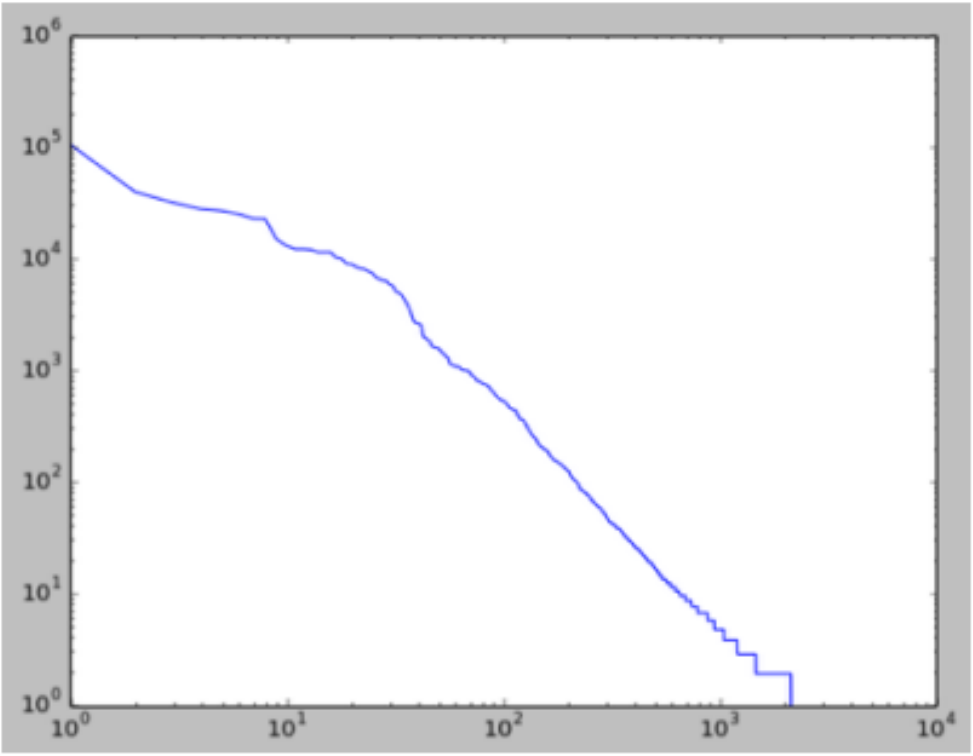
где ζ это Дзета-функция Римана.

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 художесвтенном произведении)

Стоп-слова

- Для крупных текстов большинство слов из головы распределения обычно характеризуют язык, а не текст
- Обычно это служебные слова, определяющие стрктуру 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 теримн **стоп-слова**(stop words).

Пример стоп-слов русского языка:

(конкретный состав стоп-слов зависит от рассматриваемого корпуса текстов, длинны 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 [32]:

```
# pip install -U nltk
import nltk
nltk.__version__
# # nltk.download()
```

Out[32]:

```
'3.4.5'
```

In [33]:

```
nltk.__version__
```

Out[33]:

```
'3.4.5'
```

In [34]:

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

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

Out[34]:

```
True
```

In [35]:

```
from nltk.corpus import stopwords
```

In [36]:

```
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're
e", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves',
'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'i
t', "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', 'ha
d', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but',
'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'wit
h', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'af
ter', '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', 'th
an', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'shoul
d', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren',
"aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',
"hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'might
n', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'sh
ouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'w
ouldn', "wouldn't"]
```

In [37]:

```
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', 'б
ы', 'по', 'только', 'ее', 'мне', 'было', 'вот', 'от', 'меня', 'еще', 'нет',
'о', 'из', 'ему', 'теперь', 'когда', 'даже', 'ну', 'вдруг', 'ли', 'если', 'уж
е', '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', 'са
м', 'чтоб', 'без', 'будто', 'чего', 'раз', 'тоже', 'себе', 'под', 'будет',
'ж', 'тогда', 'кто', 'этот', 'того', 'потому', 'этого', 'какой', 'совсем', 'н
им', 'здесь', 'этом', 'один', 'почти', 'мой', 'тем', 'чтобы', 'нее', 'сейча
с', 'были', 'куда', 'зачем', 'всех', 'никогда', 'можно', 'при', 'наконец', 'д
ва', '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', 'то
м', 'нельзя', 'такой', 'им', 'более', 'всегда', 'конечно', 'всю', 'между']
```

In [105]:

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

Мешок слов

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 [38]:

```
import re
import itertools as it
from razdel import sentenize
from razdel import tokenize
import pymorphy2
morph = pymorphy2.MorphAnalyzer()
```

In [39]:

```
# 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 [40]:

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

```
266954 ['лев', 'николаевич', 'толстой', 'анна', 'каренина', 'мне', 'отмщени
е', 'и', 'аз', 'воздам', 'часть', 'первая', 'все', 'счастливые', 'семьи', 'по
хожи', 'друг', 'на', 'друга', 'каждая', 'несчастливая', 'семья', 'несчастлив
а', 'все', 'смешалось', 'в', 'доме', 'облонских', 'жена', 'узнала', 'что', 'м
уж', 'был', 'в', 'связи', 'с', 'бывшею', 'в', 'их', 'доме', 'и', 'объявила',
'мужу', 'что', 'не', 'может', 'жить', 'с', 'ним', 'в', 'одном', 'доме', 'поло
жение', 'это', 'продолжалось', 'уже', 'третий', 'день', 'и', 'мучительно', 'ч
увствовалось', 'и', 'самими', 'супругами', 'и', 'всеми', 'членами', 'семьи',
'и', 'домочадцами', 'все', 'члены', 'семьи', 'и', 'домочадцы', 'чувствовали',
'что', 'нет', 'смысла', 'в', 'их', 'сожительстве', 'и', 'что', 'на', 'каждо
м', 'постоялом', 'дворе', 'случайно', 'сошедшиеся', 'люди', 'более', 'связан
ы', 'между', 'собой', 'чем', 'они', 'члены', 'семьи', 'и', 'домочадцы', 'обло
нских', 'жена', 'не', 'выходила', 'из', 'своих', 'комнат', 'мужа', 'третий',
'день', 'не', 'было', 'дома', 'дети', 'бегали', 'по', 'всему', 'дому', 'как',
'потерянные', 'англичанка', 'поссорилась', 'с', 'экономкой', 'и', 'написала',
'записку', 'приятельнице', 'прося', 'приискать', 'ей', 'новое', 'место', 'пов
ар', 'ушел', 'еще', 'вчера', 'со', 'двора', 'во', 'время', 'обеда', 'черная',
'кухарка', 'и', 'кучер', 'просили', 'расчета', 'на']
None
```

In [41]:

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

In [42]:

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

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

In [43]:

```
print('Содержимое:', list(it.islice(fdist.items(), 10)))
```

```
Содержимое: [('лев', 1), ('николаевич', 2), ('толстой', 2), ('анна', 499),
('каренина', 45), ('мне', 682), ('отмщение', 1), ('и', 12916), ('аз', 1), ('в
оздам', 1)]
```

In [44]:

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

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

In [46]:

```
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), ('был
о', 1656), ('так', 1415), ('сказал', 1412), ('а', 1391), ('то', 1388), ('же',
1325), ('ему', 1252), ('о', 1243), ('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), ('тепе
рь', 810), ('вы', 756), ('из', 735), ('была', 728), ('еще', 699), ('ей', 68
9), ('мне', 682), ('кити', 661), ('они', 646), ('него', 622), ('уже', 601),
('нет', 592), ('очень', 573)]
```

Видим, что в мешке слов большинство самых частотных слов - стоп-слова.

In [47]:

```
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', 'возда
м', 'часть', 'первая', 'счастливые', 'семьи', 'похожи', 'друг', 'друга', 'каж
дая', 'несчастливая', 'семья', 'несчастлива', 'смешалось', 'доме', 'облонски
х', 'жена', 'узнала', 'муж', 'связи', 'бывшею', 'доме', 'объявила', 'мужу',
'жить', 'одном', 'доме', 'положение', 'это', 'продолжалось', 'третий', 'ден
ь', 'мучительно', 'чувствовалось', 'самими', 'супругами', 'всеми', 'членами',
'семьи', 'домочадцами', 'члены', 'семьи', 'домочадцы', 'чувствовали', 'смысл
а', 'сожительстве', 'каждом', 'постоялом', 'дворе', 'случайно', 'сошедшиеся',
'люди', 'связаны', 'собой', 'члены', 'семьи', 'домочадцы', 'облонских', 'жен
а', 'выходила', 'своих', 'комнат', 'мужа', 'третий', 'день', 'дома', 'дети',
'бегали', 'всему', 'дому', 'потерянные', 'англичанка', 'поссорилась', 'эконом
кой', 'написала', 'записку', 'приятельнице', 'прося', 'приискать', 'новое',
'место', 'повар', 'ушел', 'вчера', 'двора', 'время', 'обеда', 'черная', 'куха
рка', 'кучер', 'просили', 'расчета']
```

In [48]:

```
# 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, 'сказал
а', 827), (44, 'кити', 661), (49, 'очень', 573), (53, 'вронский', 509), (56,
'анна', 499), (66, 'алексей', 429), (68, 'степан', 423), (69, 'аркадьич', 42
2), (72, 'александрович', 395), (81, 'время', 366), (82, 'мог', 357), (83, 'г
оворил', 357), (89, 'руку', 309), (90, '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', 2
31), (124, 'пред', 224), (125, 'хотел', 219), (127, 'сергей', 219), (129, 'ну
жно', 217), (130, 'человек', 215), (131, 'прежде', 215), (132, 'глаза', 214),
(134, 'могу', 214), (135, 'видел', 214), (137, 'тебе', 213), (139, 'тотчас',
211), (141, 'чувствовал', 210), (143, 'вронского', 205), (145, 'одно', 202),
(146, 'своего', 199), (147, 'могла', 199), (148, 'свое', 198), (149, 'иванови
ч', 191), (153, 'думал', 189), (154, 'глядя', 189), (156, 'говорила', 184)]
```

Векторное представление документа

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 подобия 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 терма 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 терма) — вес определяется как функция от количества вхождений терма в документе;
- tf-idf (term frequency — inverse document frequency, частота терма — обратная частота документа) — вес определяется как произведение функции от количества вхождений терма 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(θ), 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 терма (например, веса tf-idf) не может быть отрицательной. Угол между двумя векторами частоты терма не может быть больше, чем 90°.

Одна из причин популярности косинуснуго 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 [49]:

```
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 [50]:

```
# 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 [51]:

```
# рассмотренные токены:
cv.get_feature_names()
```

Out[51]:

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

In [52]:

```
cv_ar = corpus_cv.toarray()
cv_ar
```

Out[52]:

```
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 [53]:

```
norm(cv_ar, axis=1)
```

Out[53]:

```
array([2.23606798, 2.82842712, 2.44948974, 2.23606798])
```

In [54]:

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

In [55]:

```
ca_arn
```

Out[55]:

```
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 [56]:

```
ca_arn @ ca_arn.T
```

Out[56]:

```
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 [57]:

```
# создание векторизатора:
tv = TfidfVectorizer()

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

In [58]:

```
# рассмотренные токены:
tv.get_feature_names()
```

Out[58]:

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

In [59]:

```
corpus_tv.toarray()
```

Out[59]:

```
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 [60]:

```
pairwise_similarity = corpus_tv * corpus_tv.T
pairwise_similarity.toarray()
```

Out[60]:

```
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/RossiyaSegodnya/ria_news_dataset (https://github.com/RossiyaSegodnya/ria_news_dataset)

Этап 1: загрузка данных

In [61]:

```
import json
from os import listdir
from os.path import isfile, join
```

In [62]:

```
# 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[62]:

```
(['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 [64]:

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

news_js
```

Out[64]:


```
{'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 Фетхуллаха Гюлена - ег
о брата',
  'performance_score': 0,
  'site': 'newsru.com',
  'participants_count': 0,
  'title_full': 'В Турции задержали очередного родственника Фетхуллаха 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 Фетхуллаха Гюлена - его
брата',
 'locations': [],
 'entities': {'persons': [], 'locations': [], 'organizations': []},
 'highlightText': '',
 'language': 'russian',
 'persons': [],
 'text': 'В Турции задержали очередного родственника Фетхуллаха Гюлена - его
брата   16:53   16:53 \nТурецкая полиция задержала в городе Измир на западе с
траны брата оппозиционного исламского проповедника Фетхуллаха Гюлена Ктубетти
на. Живущего в США проповедника Анкара считает вдохновителем попытки провалив
шегося 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 под
озреваемый скрывается в доме своего родственника в районе Газиемир, передает
РИА "Новости" . ТАСС напоминает, что 23 сентября власти Турции задержали плем
янницу Гюлена Эмине. Задержание прошло в уезде Эрдемит западной провинции Бал
ыкесир. Выяснилось, что она значительную часть телефонных разговоров вела с о
дним абонентом в США. Кроме того, у нее изъято большое количество фотографий
и книг Гюлена. В августе полиция задержала племянника Гюлена Кемаля Гюлена, т
елеведущего и адвоката. Он был задержан в одной из деревень в провинции Каста
```

```
мону, где скрывался с середины июля после провала заговора. Еще раньше был за
держан другой племянник проповедника Адбуллах Коруджук. В ночь на 16 июля в Т
урции группа мятежников совершила попытку военного переворота. Основное проти
востояние развернулось в Анкаре и Стамбуле. Погибли более 240 турецких гражда
н, 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 обвин
ению в причастности к организации Гюлена в Турции после мятежа были арестован
ы 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 [58]:

```
news_js['text']
```

Out[58]:

```
'В Турции задержали очередного родственника Фетхуллаха Гюлена - его брата  1
6:53  16:53 \nТурецкая полиция задержала в городе Измир на западе страны бра
та оппозиционного исламского проповедника Фетхуллаха Гюлена Ктубеттина. Живущ
его в США проповедника Анкара считает вдохновителем попытки провалившегося пе
реворота. Кутбеттин Гюлен разыскивался по обвинению в причастности к деятельн
ости организации, возглавляемой его братом. Его доставили на допрос в Управле
ние безопасности и, вероятно, вскоре предъявят обвинение. Операцию по задержа
нию провела полиция Измира на основе оперативных данных о том, что подозревае
мый скрывается в доме своего родственника в районе Газиемир, передает РИА "Но
вости" . 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 або
нентом в США. Кроме того, у нее изъято большое количество фотографий и книг Г
юлена. В августе полиция задержала племянника Гюлена Кемаля Гюлена, телеведущ
его и адвоката. Он был задержан в одной из деревень в провинции Кастамону, гд
е скрывался с середины июля после провала заговора. Еще раньше был задержан д
ругой племянник проповедника Адбуллах Коруджук. В ночь на 16 июля в Турции гр
уппа мятежников совершила попытку военного переворота. Основное противостояни
е 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 экстрадиц
ии. Сам Гюлен осудил мятеж и заявил о своей непричастности. По обвинению в пр
ичастности к организации Гюлена в Турции после мятежа были арестованы около 3
2 тысяч человек.'
```

In [65]:

```
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[65]:

```
('Уланова: чтобы охарактеризовать Гамову, достаточно одного слова – великая
22:02. Волейбол Либеро «Динамо Казань» Екатерина Уланова после прощального ма
тча Екатерины Гамовой поделилась эмоциями, связанными с уходом Гамовой из спо
рта. «Слёзы на глаза накатываются, и мурашки по коже. Грустно, хотя понимаеш
ь, конечно, что все мы рано или поздно будем уходить из спорта. Я очень счаст
ливый человек, потому что мне удалось поиграть с Катей и в сборной, и в клуб
е. Какими словами я охарактеризовала бы Гамову? Мне достаточно 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 женс
ком волейболе, не получается. Может быть, женский характер не позволяет раскр
епоститься и сыграть в своё удовольствие. Да, сегодня была борьба, игра. Прос
то мы ещё не умеем делать шоу, не готовы к этому», – приводит слова Улановой
«Спорт Бизнес Online».',
 999)
```

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

In [66]:

```
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 [67]:

```
w_regex = re.compile('^[А-а-яЯёЁ]*$')
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 [69]:

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

Out[69]:

```
['в',
 'турция',
 'задержать',
 'очередной',
 'родственник',
 'фетхуллах',
 'гюлена',
 'он',
 'брат',
 'турецкий',
 'полиция',
 'задержать',
 'в',
 'город',
 'измир',
 'на',
 'запад',
 'страна',
```

In [70]:

```
n_stop_words = stopwords.words('russian')

n_stop_words
```

Out[70]:

```
['и',
 'в',
 'во',
 'не',
 'что',
 'он',
 'на',
 'я',
 'с',
 'со',
 'как',
 'а',
 'то',
 'все',
 'она',
 'так',
 'его',
 'но',
```

In [71]:

```
news_texts_corpus[:3]
```

Out[71]:

```
['В Турции задержали очередного родственника Фетхуллаха Гюлена - его брата
16:53   16:53 \nТурецкая полиция задержала в городе Измир на западе страны бр
ата оппозиционного исламского проповедника Фетхуллаха Гюлена Ктубеттина. Живу
щего в США проповедника Анкара считает вдохновителем попытки провалившегося п
ереворота. Кутбеттин Гюлен разыскивался по обвинению в причастности к деятель
ности организации, возглавляемой его братом. Его доставили на допрос в Управл
ение безопасности и, вероятно, вскоре предъявят обвинение. Операцию по задерж
анию провела полиция Измира на основе оперативных данных о том, что подозрева
емый скрывается в доме своего родственника в районе Газиемир, передает РИА "Н
овости" . 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 або
нентом в США. Кроме того, у нее изъято большое количество фотографий и книг Г
юлена. В августе полиция задержала племянника Гюлена Кемаля Гюлена, телеведущ
его и адвоката. Он был задержан в одной из деревень в провинции Кастамону, гд
е скрывался с середины июля после провала заговора. Еще раньше был задержан д
ругой племянник проповедника Адбуллах Коруджук. В ночь на 16 июля в Турции гр
уппа мятежников совершила попытку военного переворота. Основное противостояни
е 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 экстрадиц
ии. Сам Гюлен осудил мятеж и заявил о своей непричастности. По обвинению в пр
ичастности к организации Гюлена в Турции после мятежа были арестованы около 3
2 тысяч человек.',
 'Aizvērt karti Высокие потолки, нужен ремонт-всё подготовлено для капитально
го ремонта, окна на одну сторону но шума нет, газовое отопление-колонка Viess
mann-выравненный платёж 68 euro в месяц, городские коммуникации, вода по счет
чикам только за холодную, пластиковые окна, камины действующие, два сарая во
дворе с занесением в земельную книгу (22, 77m2), земля в собственности. Pilsē
ta:',
 'Теги Локомотив Руслан Пименов Юрий Семин Премьер-лига Россия Арсенал Тула Б
ывший нападающий «Локомотива» Руслан Пименов после матча 9-го тура премьер-ли
ги с «Арсеналом» (1:1) выразил мнение, что многие футболисты «железнодорожник
ов» не отвечают требованиям главного тренера Юрия Семина. Футболист не стал к
онкретизировать. – Главный тренер дал указание футболистам подумать, как им и
грать лучше. Ожидаете какие-то меры со стороны Семина? — вопрос Пименову. – К
оманду нужно встряхнуть. Многие футболисты не отвечают требованиями Юрия Павл
овича. – Кто именно?']
```

In [72]:

```
%%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[:100])
```

```
C:\ProgramData\Anaconda3\envs\pyTorch_1_6\lib\site-packages\sklearn\feature_e
xtraction\text.py:300: UserWarning: Your stop_words may be inconsistent with
your preprocessing. Tokenizing the stop words generated tokens ['большой', 'в
есь', 'ещё', 'имя', 'мочь', 'нея', 'нибыть', 'ничто', 'свой', 'хороший', 'эт
о'] not in stop_words.
  'stop_words.' % sorted(inconsistent))

Wall time: 6.99 s
```

In [73]:

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

Out[73]:

```
(['абашидзе',
  'абзац',
  'абонент',
  'аборт',
  'абсолютно',
  'абсолютный',
  'абсурд',
  'абукаров',
  'абэ',
  'авария',
  'аватар',
  'август',
  'аветисов',
  'авиационный',
  'австриец',
  'австрия',
  'автобиографичный',
  'автобус',
  'автобусный',
  'автодору'],
 ['юрий',
  'юсупов',
  'явка',
  'явление',
  'являться',
  'явно',
  'явный',
  'яд',
  'язык',
  'якобы',
  'ям',
  'январь',
  'яндекс',
  'япония',
  'японский',
  'яр',
  'ярость',
  'ярый',
  'ясно',
  'яценюк'],
 3895)
```

In [74]:

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

Out[74]:

```
(array([0, 0, 1, 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, 1, 1, 0, 0, 0, 0, 0, 0, 0], dtype=int64),
 3895)
```

In [75]:

```
dict(zip(news_fn, news_ar[0]))
```

Out[75]:

```
{'абашидзе': 0,
 'абзац': 0,
 'абонент': 1,
 'аборт': 0,
 'абсолютно': 0,
 'абсолютный': 0,
 'абсурд': 0,
 'абукаров': 0,
 'абэ': 0,
 'авария': 0,
 'аватар': 0,
 'август': 1,
 'аветисов': 0,
 'авиационный': 0,
 'австриец': 0,
 'австрия': 0,
 'автобиографичный': 0,
 'автобус': 0,
```

In [76]:

```
cv_ar = corpus_cv.toarray()
cv_ar
```

Out[76]:

```
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 [77]:

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

```
C:\ProgramData\Anaconda3\envs\pyTorch_1_6\lib\site-packages\ipykernel_launche
r.py:2: RuntimeWarning: invalid value encountered in true_divide
```

In [78]:

```
news_sim_mx = news_arn @ news_arn.T
news_sim_mx
```

Out[78]:

```
array([[1.        , 0.0091863 , 0.00508263, ..., 0.02201779, 0.02030463,
        0.05418659],
       [0.0091863 , 1.        , 0.01844278, ..., 0.        , 0.        ,
        0.        ],
       [0.00508263, 0.01844278, 1.        , ..., 0.05157106, 0.01019109,
        0.01450495],
       ...,
       [0.02201779, 0.        , 0.05157106, ..., 1.        , 0.06254228,
        0.02094499],
       [0.02030463, 0.        , 0.01019109, ..., 0.06254228, 1.        ,
        0.0543243 ],
       [0.05418659, 0.        , 0.01450495, ..., 0.02094499, 0.0543243 ,
        1.        ]])
```

In [79]:

```
n_idx = 41
news_texts_corpus[n_idx]
```

Out[79]:

```
'2 октября 2016 02:45 SpaceX подозревает конкурентов во взрыве своей ракеты
\nАмериканская компания SpaceX подозревает, что ее конкурент – консорциум Uni
ted Launch Alliance – причастен к аварии ракеты Falcon 9. Информация об этом
появилась в газете The Washington Post . \nКак уточняется, сотрудник SpaceX п
осетил 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 особенно
сть, вызвавшую подозрение. На видеозаписи взрыва специалисты компании обнаруж
или странную тень, а позже – белое пятно на здании ULA, расположенном неподал
еку. \nКак представитель SpaceX объяснил конкурентам, его компания прорабатыв
ает все возможные версии аварии. Но в ULA ему не разрешили попасть на крышу т
ого самого здания. Сотрудники консорциума вызвали специалиста из Военно-воору
женных сил США. Он осмотрел крышу и не нашел ничего подозрительного, что могл
о быть связано со взрывом ракеты Falcon 9. \nПо данным газеты, сам Илон Маск
, глава SpaceX, не отрицает версию о саботаже. ULA – совместное предприятие а
виационного гиганта Boeing и Lockheed Martin. \n1 сентября на космодроме, рас
положенном на мысе Канаверал, на площадке SpaceX взорвалась ракета Falcon 9 с
израильским спутником связи Amos-6. В результате инцидента никто не пострада
л, напоминает ТАСС .'
```

In [80]:

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

Out[80]:

```
59
```

In [81]:

```
sim_news_idx = news_sim_mx[n_idx, :].argsort()
sim_news_idx
```

Out[81]:

```
array([49,  1,  2, 88, 85,  5, 82, 77, 45, 36, 22, 99, 53, 62, 61, 37, 39,
       38, 97, 11, 21, 12, 94, 63, 10,  7, 15,  6, 83, 84, 51, 90, 93,  9,
       64,  4, 42, 28, 96, 56, 17, 95, 58, 18, 46, 76, 91, 72, 29,  8, 89,
       81, 98, 24, 55, 65, 14,  0, 32, 86, 92, 71, 52, 70, 50, 74, 27, 31,
       73, 48, 57, 75, 87, 60, 40,  3, 23, 47, 13, 79, 33, 26, 66, 30, 35,
       69, 25, 67, 54, 68, 44, 34, 16, 78, 80, 20, 19, 43, 41, 59],
      dtype=int64)
```

In [87]:

```
news_texts_corpus[sim_news_idx[-4]]
```

Out[87]:

```
'× E-mail: В Испании прогремел взрыв: пострадали более 70 человек http://www.
segodnya.ua/img/article/7566/94_main.jpg (http://www.segodnya.ua/img/article/
7566/94_main.jpg) http://www.segodnya.ua/img/article/7566/94_tn.jpg (http://w
ww.segodnya.ua/img/article/7566/94_tn.jpg) 2016-10-02T02:18:34+03:00 Мир Взры
в прогремел в помещении кафе В кафе в Испании от взрыва пострадало 77 челове
к. Фото: pbs.twimg.com В Испании прогремел взрыв: пострадали более 70 человек
Взрыв прогремел в помещении кафе В результате взрыва в городе Велес-Малага на
юге Испании пострадали по меньшей мере 77 человек, сообщает канал " 112 Украи
на " со ссылкой на пресс-секретаря скорой помощи региона Андалусия. "Взрыв пр
огремел в помещении кафе около 19:00 по Киеву. Сообщается, что это был взрыв
 газового баллона. На фотографиях с места события видны последствия взрыва –
 разбросанные стулья, мусор и другие повреждения", – говорится в сообщении. Ф
ото: twitter.com/opiniondemalaga По данным испанского издания El Mundo, всего
были госпитализированы более 70 человек, двадцать из них после оказания перво
й медицинской помощи были отпущены из больницы. "Остальные остаются в медицин
ских учреждениях, у четверых пострадавших тяжелые травмы, но не опасные для ж
изни", подчеркивает издание. Читайте также:'
```

In [ ]:

```

```