**Лабораторная работа №1.  
Обработка текстовых данных**

Выполнил: Зарипзянов Данила Сергеевич, ПИ-20-1.

Ссылка на папку со всеми материалами:

<https://drive.google.com/drive/folders/1hzfVV5uj407q2Inc7zxQtOGFWe6EBvsl?usp=sharing>

В данной папке находится ноутбук с выполненными заданиями 9-12, а также вводные и результаты.

Задание 9: input.txt, output.txt.

Задание 10: input.txt, output\_stem.txt

Задание 11: input.txt

Задание 12: input12.txt, output12.txt

## Регулярные выражения

Для выполнения использовал синтаксис регулярных выражений из python.

**Задание 1 (0,5 балла).**

Найти в тексте запись об указании времени в формате чч:мм:сс. В заданном формате чч - целое число из диапазона от 00 до 23, мм и сс - целые числа из диапазона от 00 до 59 (если какая-то часть формата нарушена, то данная подстрока в качестве верного значения не рассматривается).

\b([0-1]{1}\d{1}|2{1}[0-3]{1}):([0-5]{1}\d{1}):([0-5]{1}\d{1})\b

**Задание 2 (0,5 балла).**

Найти в тексте знаки препинания (точка, запятая, точка с запятой, двоеточие, тире («минус», длинное и короткое), вопросительный и восклицательный знаки, многоточие (три идущие подряд точки и отдельный символ «…» ) .

[^.]\.{3}[^.]|[\.,;:\-—–?!…]

**Задание 3 (0,5 балла).**

В тексте могут содержаться IP-адреса компьютеров в формате d.d.d.d, где d – целое число из диапазона от 0 до 255. Найти все IP-адреса, содержащиеся в тексте.

\b(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\b

**Задание 4 (0,5 балла).**

Проверить надежно ли составлен пароль. Пароль считается надежным, если он состоит из 8 или более символов. Пароль должен содержать хотя бы одну заглавную букву, одну маленькую букву одну цифру, хотя бы один специальный символ («!», «@», «#», «$», «%», «^», «&», «\*») и не является последовательностью («123456Qwerty», «Ps123456789», «Qwerty1»).

Не знаю, как проверить строку на последовательность, поэтому сначала я проверил, что пароли имеют нужную длину и что в них содержатся все требуемые символы и ничего кроме них:

(?=.\*[!@#$%^&\*])(?=.\*[A-Z@#])(?=.\*[a-z])(?=.\*\d)[A-z\d!@#$%^&\*]{8,}

Далее проверка на последовательности. Как я понял из условия, последовательность - это 6 букв или 6 цифр (так как оставшиеся 2 символа – это спец. символ и цифра/буква соответственно). Таким образом, содержание скрипта будет примерно такое:

import re

def check\_passwords(unchecked\_passwords : str) -> list[str]:

sequences = [‘qwertyuiop’, ‘asdfghjkl’, ‘zxcvbnm’, ‘0123456789’, ‘abcdefghijklmnopqrstuvwxyz’]

passwords = re.findall(‘(?=.\*[!@#$%^&\*])(?=.\*[A-Z@#])(?=.\*[a-z])(?=.\*\d)[A-z\d!@#$%^&\*]{8,}’, unchecked\_passwords)

correct\_passwords = []

for password in passwords:

incorrect\_password = false

pas = password.lower()

for seq in sequences:

for i in range(0, len(seq) - 6):

if seq[i : i+6] in pas:

incorrect\_password = true

break

if incorrect\_password:

break

if not incorrect\_password:

correct\_passwords.append(password)

return correct\_passwords

**Задание 5 (1 балл).**

Выделить в тексте прямую речь с опорой на пунктуацию (см. <http://www.gramota.tv/biblio/textbooks/gramota/punct/343-45-192?ysclid=lne281u9yq246756449>).

Пустые строки “” / «» прямой речью не считал. Чтобы учитывать их, надо заменить “.+” на “.\*”. Кавычки частью прямой речи не считал.

((?<=[\"]{1}).\*(?=[\"]{1}))|((?<=[«]{1}).\*(?=[»]{1}))

Также в прямую речь скорее всего не надо было включать выделенный фрагмент: «Тише, **– говорит она мне, –** отец болен, при смерти, и желает с тобою проститься» (Пушкин). Но я не смог исключить него из прямой речи, только выделить отдельным выражением (по тире условий никаких не было, так что я использовал только то, что использовалось в примерах по ссылке).

([\–]{1}.\*[,]{1}[ ]{1}[\–]{1}[ ]{1})

**Задание 6 (1 балл).**

Выделить в тексте предложения с обобщающими словами. Рассмотреть все варианты пунктуации. (Например: «Всякий день стал приносить старый Мосеич **разную крупную рыбу**: щук, язей, голавлей, линей, окуней.») (см. <http://gramota.ru/class/coach/punct/45_190?ysclid=lne2kwryum489092268>).

(((?=.\*[:])(?=.\*[,]))|((?=.\*[,])(?=.\*[–])))\b(.+)|([\.])\b

**Задание 7 (1 балл).**

Дан текст с географическими названиями населенных пунктов (г. Пермь, п. Скобелевка, …). Найти вхождения населенных пунктов:

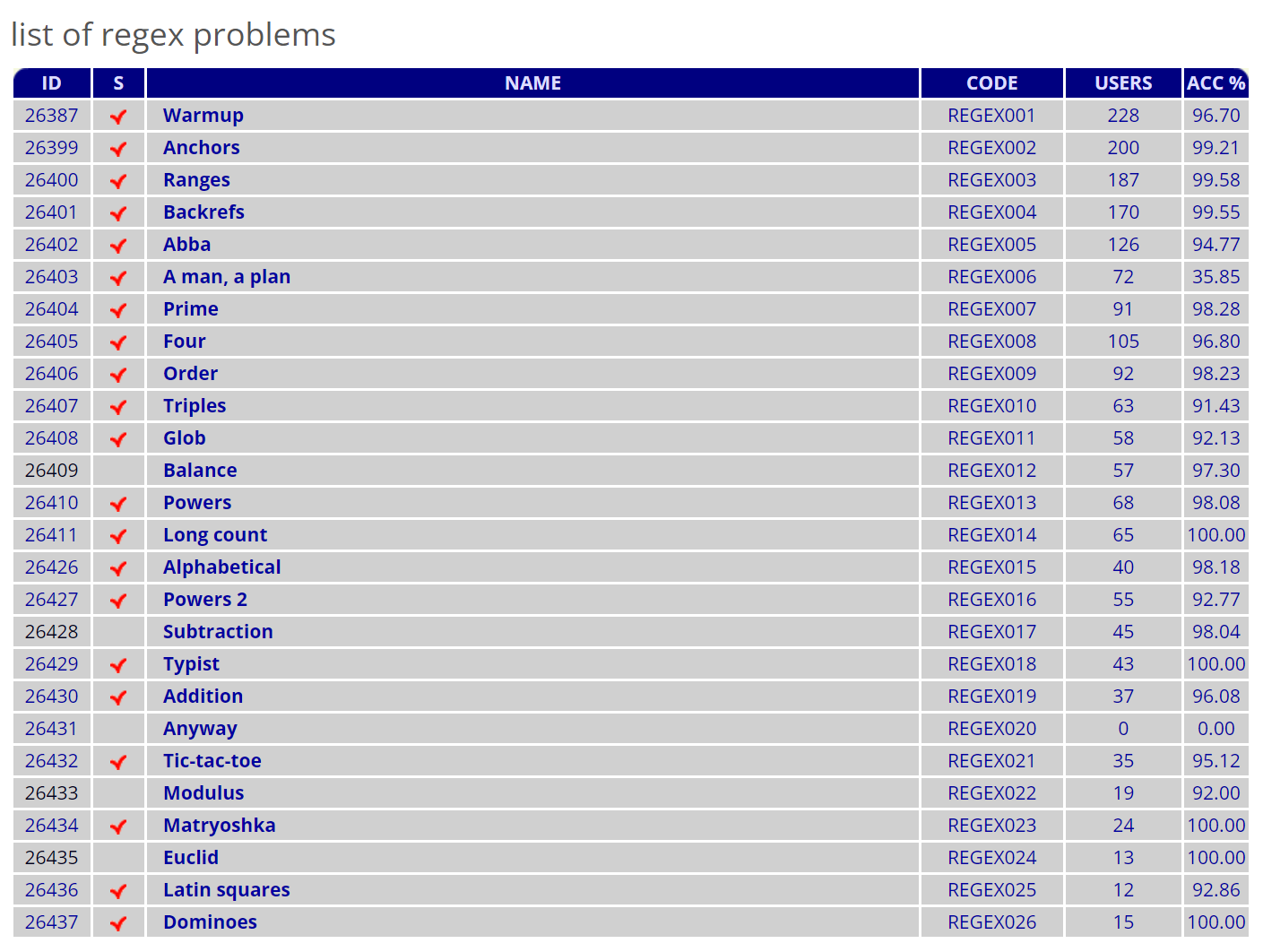
* г - город;
* пгт - поселок городского типа;
* рп - рабочий поселок;
* кп - курортный поселок;
* к - кишлак;
* дп - дачный поселок;
* п - поселок;
* п. ст - поселок при станции (поселок станции);
* с - село;
* д - деревня;
* сл - слобода;
* ст - станция;
* ст-ца - станица;
* х - хутор;
* у - улус;
* рзд - разъезд;
* зим - зимовье.

\b((г. )|(пгт. )|(рп. )|(кп. )|(к. )|(дп. )|(п. )|(п. ст. )|(с. )|(д. )|(сл. )|(ст. )|(ст-ца )|(х. )|(у. )|(рзд. )|(зим. ))[\w]\*\b

**Задание 8 (1 балл)**

Решить задачи (20 заданий) на ресурсе <https://www.spoj.com/SHORTEN/problems/regex/>. (Ресурс доступен через VPN. В качестве результата в отчете представить снимок экрана и пояснить логику решения.)

Ссылка на профиль: <https://www.spoj.com/SHORTEN/users/danil17/>



| **#** | **Название** | **Решение** | **Пояснение** |
| --- | --- | --- | --- |
| 1 | Warmup | foo | Во всех словах слева есть foo, а справа - нет |
| 2 | Anchor | ick$ | Слова слева оканчиваются на ick, а справа - нет |
| 3 | Ranges | ba|ead|beb|da|de|aff|ee | По очереди подбирал такие условия, которым удовлетворяли бы только слова слева |
| 4 | Backrefs | (.{3}).\*\1 | Только в словах слева первые 3 буквы повторялись дальше в слове |
| 5 | Abba | ^(([\w])(?!\2))+$|(^(ef|n)) | Все слова справа содержат двойные буквы (например, bb), слева двойные буквы содержат только слова effusive и noisefully, которые я добавил в исключение в правой части выражения: (^(ef|n)). Можно прочитать как: в слове нет двойных букв или оно начинается с ef/n |
| 6 | A man, a plan | ^(.)(.).\*(\2)(\1)$ | Слева палиндромы, справа - нет. Палиндромы произвольной длины с помощью регулярных выражений выбрать невозможно, также будут проблемы, если в одном списке очень длинные и очень короткие слова, но в данном случае достаточно было сопоставить первые две буквы с последними двумя соответственно |
| 7 | Prime | ^(.{33,}|.{2,3}|.{5}|.{7}|.{11}|.{13}|.{17}|.{19}|.{23}|.{29}|.{31})$ | Слева - строки, длины которых - обязательно простые числа, справа - наоборот. Не знаю, как проверять, является ли число простым, без кода, поэтому длины захардкодил, а также обнаружил, что справа нет строк больше 32 |
| 8 | Four | (?=[^T])^.\*([\w]).\*(\1).\*(\1).\*(\1).\*$ | В словах слева одна из букв повторяется 4+ раз. Справа есть 1 такое слово: Ternstroemiaceae. Чтобы его исключить, я запретил букву T |
| 9 | Order | ^([^o]).{2,5}$ | Длина слов слева не превышает 6, а справа - длина всех слов, кроме 1, больше 6. Я выбрал слова, которые начинались не на “o” и которые имели длину <=6 (в формуле от 2 до 5 - это кол-во букв без учета первой, это самое короткое решение, которое я нашел) |
| 10 | Triples | 000000000|000000003|000000006|000000009|000000012|000000015|066990060|140091876|173655750|312440187|321769005|368542278|390259104|402223947|443512431|714541758|747289572|819148602|878531775|905586303|953734824 | Ничего лучше не придумал, самое прямое решение |
| 11 | Glob | err|alle|log|tud|odi|bow|bron|chr|co|cr|dow|lea|mit|pal|pr|pup|resu|wr | Тоже решал напрямую, но при этом старался выбирать для каждого слова самую короткую комбинацию символов |
| 12 | Powers | ^(.{1}|.{2}|.{4}|.{8}|.{16}|.{32}|.{64}|.{128}|.{256}|.{512}|.{1024})$ | То же, что и в 7: выбрал строки, у которых длина - степень двойки |
| 13 | Long count | 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111 | Странное задание, надо было выбрать только 1 строку, я просто вбил ее, предварительно максимально сократив |
| 14 | Alphabetical | aerate (ass|ae|e)|(arrest) \2|^assent|assert ([er]|astern r)|^asses.\*(testes|snarer|testes)$|^astern (r|enseat e)|^en.\*(ts|tes)|^rer | Решил так же, как 11 |
| 15 | Powers 2 | ^(.{1}|.{3}|.{9}|.{27}|.{81}|.{243}|.{729}|.{2187}|.{6561,})$ | Логика та же, что и в 12 |
| 16 | Typist | address|afterwards|average|better|bread|card$|create|decrease|effect|exaggerate|face|freeze|garbage|grass|reverse|stress|sweater|target|tree|wave | Решал так же, как 10: вбил все слова в выражение |
| 17 | Addition | ^x \+ xxx = xx |^x \+ xxxx |^x \+ xxxxxxxxxxx = xxxxx|^xxxx \+ xx |^xxxx \+ xxx |xxxx \+ xxxx = xxxx \+ xxxx$|xxxxx \+ xx = xxxxxx \+ x|xxxxx \+ xxxxx = xx |xxxxx \+ xxxxxx |^xxxxx \+ xxxxxxx|xxxxxx \+ x = xxx \+ xxxx$|xxxxxx \+ xx = x \+ xx|xxxxxxx \+ xx = xxx \+ xxxx|^xxxxxxx \+ xxxx = xxx \+ xxxxxxxx|xxxxxxx \+ xxxxxxxxx = xxxx|xxxxxxxx \+ xxx |xxxxxxxxx \+ xxxxxxx |xxxxxxxxxxx \+ x = xxxxxx \+ xxx | Так же, как 11 |
| 18 | Tic-tac-toe | .OO XO. XXX|.OO XOX .OX|.XO .OO XOO|.XO X.O O.O|OOO .OO O.X|OOO XXX ..X|OX. OXO OX.|OX. X.. OOO|OXX OX. XO.|X.. XOO XOX|X.. XOX X.O|X.. XXX OOO|X.O .XO XOO|X.X O.X OOX|XO. .XX .XX|XOO .OO OXO|XOO XX. .OX|XX. .XO .OX|XXX X.O OX. | Так же, как 10 |
| 19 | Matryoshka | ^a |ab|all|an |be bet|cat|eat ea|fun |he |ice r|in pin |it sit|fundamental|promotion|flavour|spot sport support|pro|unde|stream$|to | Как 11 |
| 20 | Latin squares | bcow cbwo owbc wocb|bytz ztby tzyb ybzt|cuoh uhco ochu houc|erad dare ader reda|fbtp ptfb tpbf bfpt|feqp efpq qpef pqfe|gezw zwge wzeg egwz|gkih hgki ihgk kihg|hzdi dhiz idzh zihd|ivpc vpci pciv civp|jwok okjw kowj wjko|kxrh rhkx hrxk xkhr|ncib cnbi ibnc bicn|rick kcir ckri irkc|tqec qtce ecqt cetq|vpth thpv htvp pvht|whrl lwhr rlwh hrlw|xiey ixye eyxi yeix|zdvm vmzd dzmv mvdz|znqp qpzn pznq nqpz | Как 10 |
| 21 | Dominoes | 02$|04 43|05 50|13 33 53|15 14 46|16 15 35 33|21 01 01 14|25 26 63 33|25 54 14 51 52|30 02 25 53 33|34 45 56 65 55|43 41 31 30 03|45 45 40 01 51|50 30 13 31 31 16|50 55 65 06 00 50|53 03 03 13 61 36|60 20 02 20 20 10 11|61 66 61 12 02 03 23|64 14 11 31 23 23 36 63|65 51 15 51 51 15 10 10 13 03 | Как 10 |
| 22 | Anyway | anyway|bone|cinema|cover|define|economy|eleven|eye|hesitate|key|majority|music|palace|potato|relax|royal|surely|typical|vocabulary|water | Там проблемы с проверкой, но решено правильно. Логика: как в 10 |

# Отработка текстов

Задания 9 - 12 делал в colab’е (они там идут по порядку + подписаны), но самое важное все равно буду прикладывать здесь. Ссылка на colab: <https://colab.research.google.com/drive/1wMSzARIaaJeagBplnJJm1vFTyvpIPVMu?usp=sharing>

**Задание 9 (1 балл)**

Написать программу (Python, С# или любой другой язык программирования) для построения частотного словаря текста на русском языке без учета стоп-слов. Входной текст хранится в текстовом файле, построенный словарь также должен быть сохранен в текстовый файл в формате (словоформа, количество\_вхождений\_словоформы) с сортировкой по количеству вхождений. Для деления текста на токены и удаления стоп-слов использовать библиотеку nltk, а для нормализации использовать библиотеку Pymorphy2.

# построение словаря одной функцией

def make\_freq\_dict\_ru(input\_path : str, output\_path : str):

# функция удаляет заданные символы из текста

def replace\_chars\_in\_text(text, chars, sub\_char):

return sub\_char.join([ch for ch in text if ch not in chars])

# открываю файл и считываю текст

spec\_chars = string.punctuation + '\xa0«»\t—…\*'

f = open(input\_path, 'r', encoding='utf-8')

text = f.read()

# привожу к нижнему регистру, удаляю пунктуацию и цифры

text = text.lower()

text = replace\_chars\_in\_text(text, spec\_chars, '')

text = replace\_chars\_in\_text(text, string.digits, '')

# токенизирую и удаляю из текста стоп-слова

text\_tokens = word\_tokenize(text)

russian\_stopwords = stopwords.words("russian")

text\_tokens = [word for word in text\_tokens if not word in russian\_stopwords]

# нормализую слова

morph = pymorphy2.MorphAnalyzer()

for i in range(len(text\_tokens)):

text\_tokens[i] = morph.parse(text\_tokens[i])[0].normal\_form

# формирую словарь

nltk\_text = nltk.Text(text\_tokens)

fdist = FreqDist(nltk\_text)

# формирую файл со словарем

with open(output\_path, 'w', encoding='utf-8') as f:

for word in fdist:

f.write(f'{word}: {fdist[word]}\n')

make\_freq\_dict\_ru('input.txt', 'output.txt')

**Задание 10 (1 балл)**

Реализовать вторую версию программы (см. задание 1) построения частотного словаря, в которой для нормализации используется стеммер (найти необходимую библиотеку самостоятельно). Сравнить построенные частотные словари.

from nltk.stem.snowball import SnowballStemmer

# построение словаря одной функцией

def make\_freq\_dict\_ru\_stem(input\_path : str, output\_path : str):

# функция удаляет заданные символы из текста

def replace\_chars\_in\_text(text, chars, sub\_char):

return sub\_char.join([ch for ch in text if ch not in chars])

# открываю файл и считываю текст

spec\_chars = string.punctuation + '\xa0«»\t—…\*'

f = open(input\_path, 'r', encoding='utf-8')

text = f.read()

# привожу к нижнему регистру, удаляю пунктуацию и цифры

text = text.lower()

text = replace\_chars\_in\_text(text, spec\_chars, '')

text = replace\_chars\_in\_text(text, string.digits, '')

# токенизирую и удаляю из текста стоп-слова

text\_tokens = word\_tokenize(text)

russian\_stopwords = stopwords.words("russian")

text\_tokens = [word for word in text\_tokens if not word in russian\_stopwords]

# нормализую слова

snowball = SnowballStemmer("russian")

for i in range(len(text\_tokens)):

text\_tokens[i] = snowball.stem(text\_tokens[i])

# формирую словарь

nltk\_text = nltk.Text(text\_tokens)

fdist = FreqDist(nltk\_text)

# формирую файл со словарем

with open(output\_path, 'w', encoding='utf-8') as f:

for word in fdist:

f.write(f'{word}: {fdist[word]}\n')

make\_freq\_dict\_ru\_stem('input.txt', 'output\_stem.txt')

**Задание 11 (1 балл)**

Собрать корпус документов по выбранной теме и по основы программ, разработанных в заданиях 1 и 2, подсчитать метрику TF-IDF для ключевых слов документа.

from sklearn.feature\_extraction.text import TfidfVectorizer

# Текстовые данные, взяты из статьи: https://habr.com/ru/news/770514/

documents = [

"В конце октября 2022 года Илон Маск купил соцсеть Twitter, ныне X.",

"Ряд СМИ вместе с аналитиками попробовали подвести итоги прошедшего года.",

"27 октября 2022 года Маск официально взял на себя управление компанией.",

"Он немедленно уволил весь топ-менеджмент Twitter, а в течение следующих недель ещё около 5000 человек, или примерно две трети штата компании, оставшись с относительно небольшой командой в примерно 1500 сотрудников.",

"Сразу после покупки компании Маск заявил о планах жёсткой борьбы с ботами, а чуть позже — о намерении превратить Twitter в «приложение для всего», включающее в себя платформу цифровых платежей, а также аудио- и видеозвонки.",

"Он заявил сотрудникам, что, несмотря на массовые сокращения, он видит «ясный, но трудный путь» к будущей оценке компании в более чем $250 млрд.",

"Тем не менее, как считает большинство аналитиков, за прошедший год платформа пострадала от ряда необдуманных решений и сейчас сталкивается с огромной конкуренцией.",

"Как отмечает ArsTechnica, после того как Маск сделал компанию частной, она перестала предоставлять подробные отчёты о своём финансовом состоянии и числе пользователей, однако сторонние аналитики, ориентируясь на открытые и слитые данные, считают, что платформа находится в упадке.",

"Так, в отчёте The Information, опубликованном в начале октября, говорится, что с момента покупки Маском компании «количество ежедневных активных пользователей X упало на 3,7%до 245 миллионов».",

"Многих пользователей отпугнули спорные шаги, такие как сокращение команд модераторов, взимание платы за верификацию аккаунтов и неожиданный ребрендинг. ",

"Немного другие данные приводят аналитики из Sensor Tower.",

"Они утверждают, что число ежедневных активных пользователей X в августе 2023 года снизилось на 9% процентов по сравнению с августом 2022 года.",

"Падение числа пользователей сопровождается также сокращением времени, проведённого в соцсети (падение составило 13%), что, по мнению Sensor Tower, гораздо более важный показатель. ",

"Ещё одна исследовательская фирма Similarweb сообщила на прошлой неделе, что за год использование X «снизилось по всем параметрам».",

"«В сентябре глобальный трафик на twitter.com снизился на 14% по сравнению с аналогичным периодом прошлого года, а трафик на Ads.twitter.com для рекламодателей снизился на 16,5%.",

"В США, откуда приходится около четверти трафика twitter.com, сентябрьский трафик снизился на 19%.",

"Тенденция была аналогичной, хотя и не столь выраженной, в других странах: минус 11,6% в Великобритании, минус 13,4% во Франции, минус 17,9% в Германии и минус 17,5% в Австралии»."

]

def get\_tf\_idf(documents):

# Создание объекта TfidfVectorizer

tfidf\_vectorizer = TfidfVectorizer()

# Применение TF-IDF к текстовым данным

tfidf\_matrix = tfidf\_vectorizer.fit\_transform(documents)

# Получение списка ключевых слов и их значения TF-IDF для первого документа

feature\_names = tfidf\_vectorizer.get\_feature\_names\_out()

tfidf\_scores = tfidf\_matrix.toarray()[0]

# Сортировка слов по значениям TF-IDF

sorted\_keywords = sorted(zip(tfidf\_scores, feature\_names), reverse=True)

return sorted\_keywords

sorted\_keywords = get\_tf\_idf(documents)

print(sorted\_keywords)

**Задание 12 (1 балл)**

Реализовать алгоритм автоматического реферирования (квазиреферирование) на основе статистического подхода, алгоритм приведен ниже. Входные данные: исходный текст и коэффициент сжатия. Единицей реферирования (фрагментом) должно являться предложение. Выходные данные: список ключевых слов с весами, список предложений с весами, текст реферата.

**Алгоритм**:

1. Разбить текст на предложения.
2. Разбить текст на слова произвести их нормализацию.
3. Удалить стоп-слова.
4. Подсчитать веса слов (tf или tf-idf).
5. Определить веса предложений, рассчитанный как сумма весов, входящих в предложение слов.
6. Отсортировать предложения по убыванию веса.
7. В отсортированном списке оставить те предложений, которые входят в задаваемый процент сжатия. Например, если процент сжатия 10%, а в исходном тексте 50 предложений, то в реферате будет 5 предложений.
8. Сформировать текст реферата из отобранных на предыдущем шаге предложений в порядке, котором они встретились в исходном тексте.

import re

def get\_summary(input : str, compr\_rate : int = 0.1) -> str:

# Получаю текст из файла, он там не структурирован

f = open(input, 'r', encoding='utf-8')

text = f.read()

# Заменяю переносы строки на пробелы и привожу к нижнему регистру

text = text.replace('\n', ' ')

# Разбиваю текст на предложения, получилось почти идеально

# Только последнее предложение неправильно определилось из-за иницалов

origin\_sentences = re.split("\. |! |\? |… ", text) # оригинальные предложения нужны для вывода

sentences = re.split("\. |! |\? |… ", text.lower()[:-1])

# Удаляю лишние пробелы в начале и в конце строки

sentences = [s.strip() for s in sentences]

prepared\_sentences = []

for sentence in sentences:

# удаляю пунктуацию

sentence = replace\_chars\_in\_text(sentence, spec\_chars, '')

# токенизирую

text\_tokens = word\_tokenize(sentence)

# нормализую слова

morph = pymorphy2.MorphAnalyzer()

for i in range(len(text\_tokens)):

text\_tokens[i] = morph.parse(text\_tokens[i])[0].normal\_form

# удаляю из текста стоп-слова

russian\_stopwords = stopwords.words("russian")

text\_tokens = [word for word in text\_tokens if not word in russian\_stopwords]

prepared\_sentences.append(' '.join(text\_tokens))

tf\_idf\_metrics = get\_tf\_idf(prepared\_sentences)

d\_metrics = {}

for pair in tf\_idf\_metrics:

d\_metrics[pair[1]] = pair[0]

# вычисляю весы предложений, предложения идут по порядку

# первому предложению в исходном тексте соотв первая запись в массиве

sentences\_tf\_idf = {}

i = -1

for sentence in prepared\_sentences:

val = 0

i += 1

words = sentence.split()

for word in words:

if word in d\_metrics:

val += d\_metrics[word]

sentences\_tf\_idf[i] = val

sentences\_tf\_idf\_sorted = sorted(sentences\_tf\_idf.items(), key=lambda x:x[1], reverse=True)

summary\_sentence\_count = round(len(sentences\_tf\_idf) \* compr\_rate)

sentence\_ids = sorted([x[0] for x in sentences\_tf\_idf\_sorted[:summary\_sentence\_count]])

summary = ""

for i in sentence\_ids:

summary += origin\_sentences[i] + '. '

return summary

def write\_to\_file(output, result):

# формирую файл со результатом

with open(output, 'w', encoding='utf-8') as f:

f.write(result)

Про вывод ничего не сказано, поэтому вывел в файл и на консоль.

print(get\_summary('input12.txt', 0.1))

write\_to\_file('output12.txt', get\_summary('input12.txt', 0.1))