Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(УНИВЕРСИТЕТ ИТМО)

Факультет «Систем управления и робототехники»

ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ № 6

По дисциплине «Практическая линейная алгебра»
на тему: «Singular Value Decomposition»

Студенты:
Гизбрехт В.Д. группа 1
Ли Х.С. группа 1
Лаврик В.В. группа 4

Преподаватель:
Догадин Егор Витальевич

г. Санкт-Петербург
2024

Оглавление

Задание 1

Задание 1 «применение сингулярного разложения» выполнялось с помощью python и библиотек *os, numpy, matplotlib, skimage* (код отображен далее).

Для сингулярного разложения нами была выбрана картинка:

Преобразуем картинку к серым оттенкам с помощью команды color.rgb2gray из библиотеки skimage:

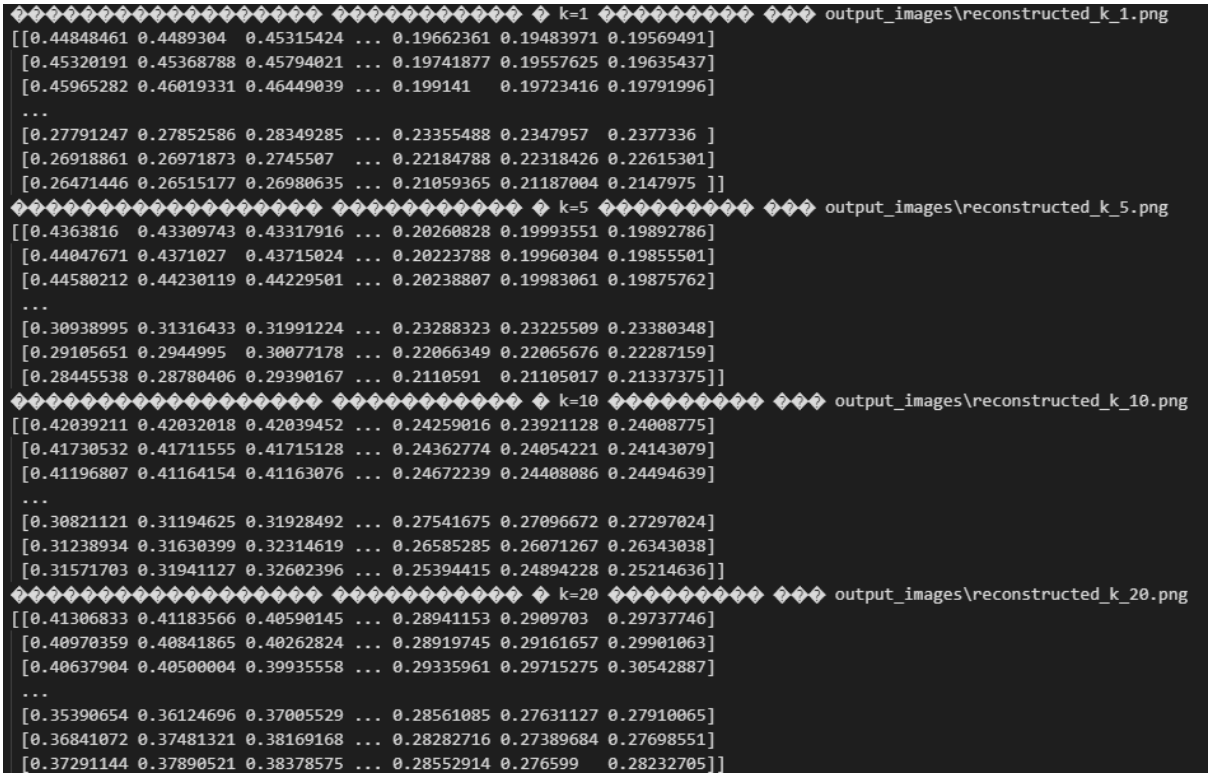
Картинка была выбрана 2650x1600 пикселей, выполнение SVD-разложения происходит с помощью кода на python. Разложение выполняется в функции “compute_svd” с помощью библиотеки *numpy*, в которой есть функция *linalg.svd,* после выполнения которой мы получаем 3 матрицы:

```
[[ 0.02736946  0.00855454  0.00219713 ...  0.09479291  0.0271263
  -0.04818915]
 [ 0.02748734  0.00853041  0.00243676 ... -0.16255331  0.00733634
   0.13850594]
 [ 0.02767408  0.00851548  0.00283466 ...  0.05945769 -0.05907983
  -0.10750206]
 ...
 [ 0.01973948  0.02214608  0.01045164 ...  0.00199352 -0.00465344
  -0.02485651]
 [ 0.01944643  0.02050097  0.01081904 ... -0.00462528  0.00304013
   0.00995952]
 [ 0.01927502  0.01896784  0.01017252 ... -0.00188438  0.00138372
  -0.00397887]] [9.49584689e+02 2.05359496e+02 1.33463831e+02 ... 2.87694934e-02
2.84874748e-02 2.75939586e-02] [[ 0.01061409  0.0105681   0.0106089  ...  0.0145272   0.01461606
   0.01478772]
 [ 0.02177281  0.0221473   0.0226     ...  0.02226493  0.0218891
   0.02154161]
 [ 0.01927217  0.01974515  0.02034579 ... -0.00373763 -0.00354724
  -0.0039576 ]
 ...
 [-0.01402135  0.02123284  0.02288092 ...  0.04358346 -0.02572014
  -0.02223231]
 [ 0.00431633 -0.00814451 -0.00680842 ... -0.01622627  0.01126848
  -0.00507913]
 [-0.00153466  0.01590169 -0.02100385 ... -0.01600994 -0.00122866
  -0.01217329]]
```

Разложение выполнено по формуле: $A = U\Sigma V^T$

Далее для укороченного SVD-разложения были выполнены срезы внутри функции “reconstruct_image”. Изначально мы создаем матрицу S_k с первыми k сингулярными числами, а потом получаем новую матрицу изображения по формуле:

$$A_k = U_k \Sigma_k V_K^T$$



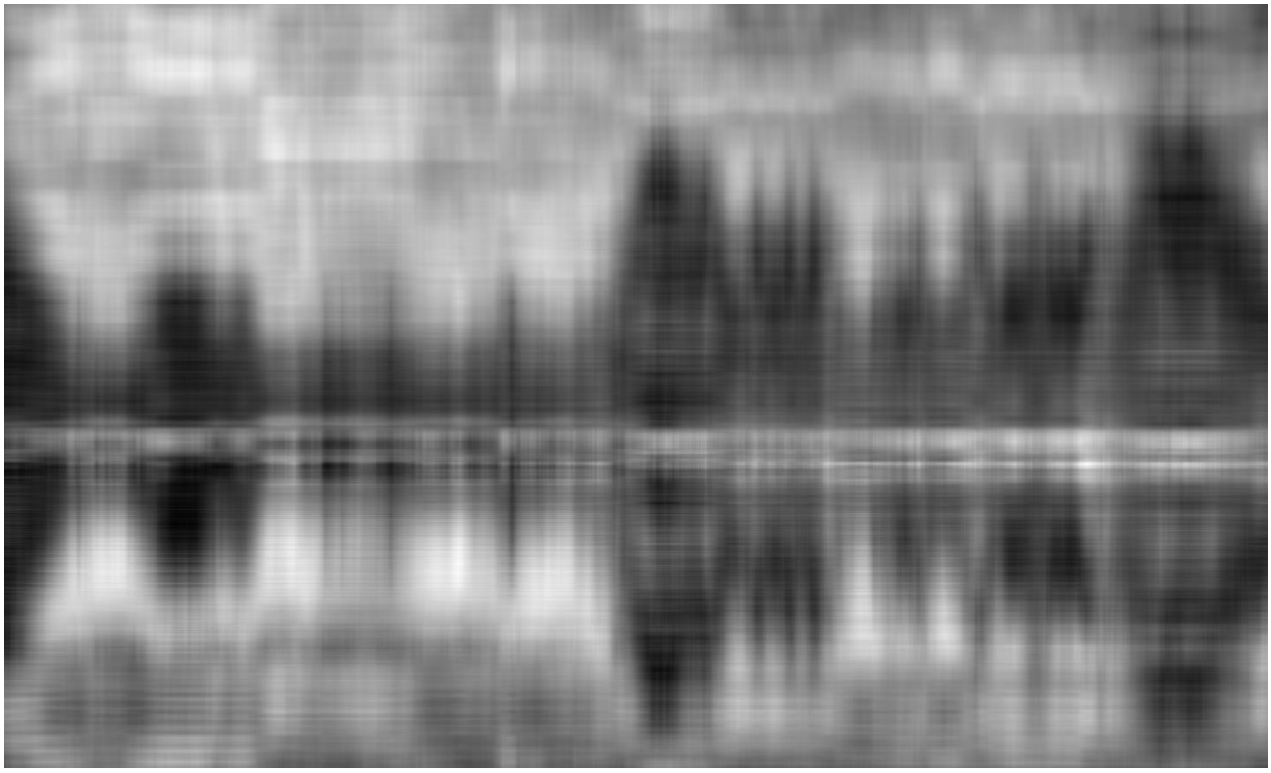
Где взятые значения K записаны в массив k_values и равны 9-ти различным значениям:

$$K = \{1\ ,5, 10, 20, 50, 100, 150, 200, 300\}$$

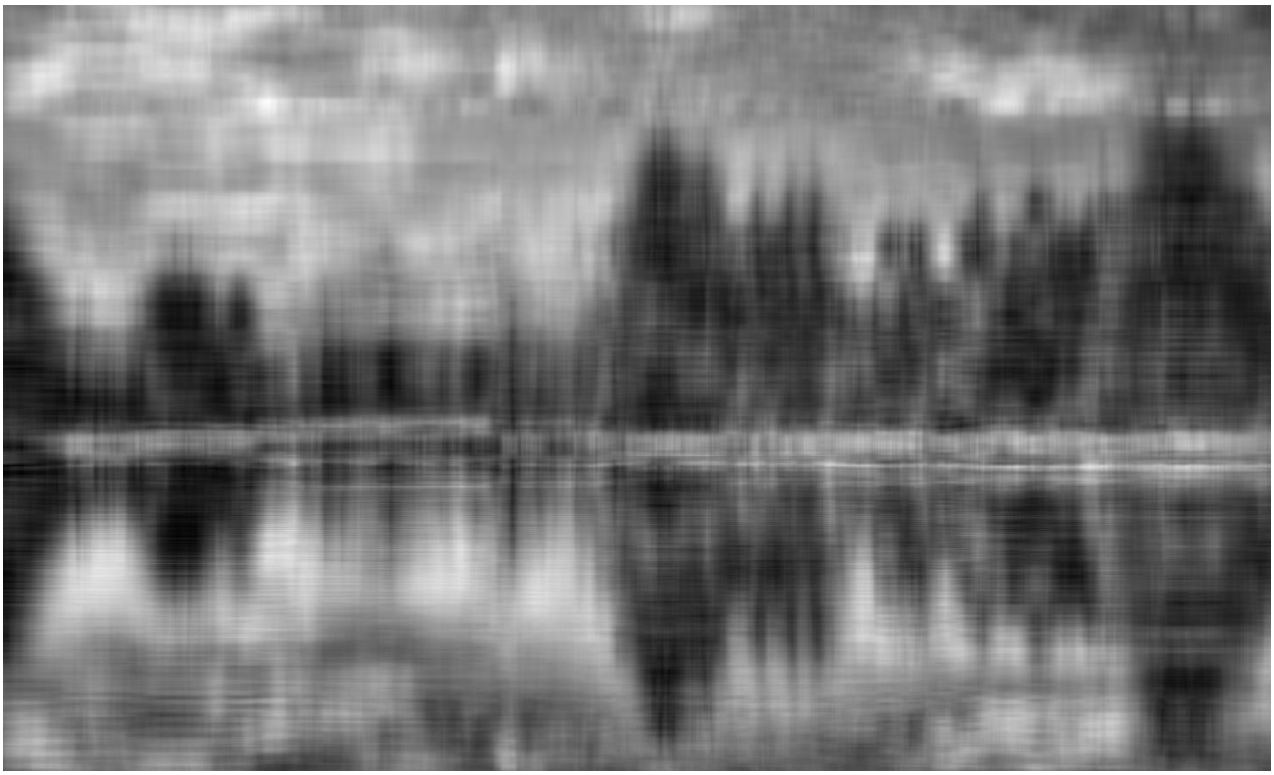
На предыдущем рисунке представлены матрицы для значений K = 1, 5, 10, 20.

Далее с помощью укороченного SVD-разложения мы получаем наши новые картинки с различным качество, которое зависит от K. Прикрепляем все получившиеся картинки:

$K = 5$



$K = 10$



$K = 20$

$K = 50$

$K = 100$

$K = 150$

$K = 200$

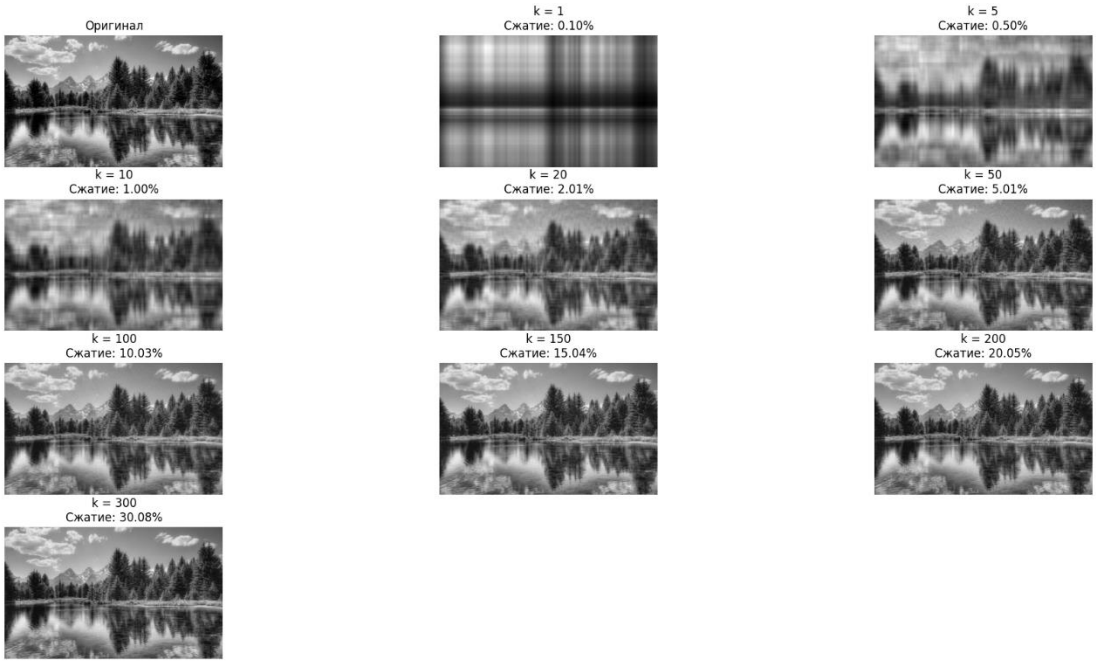
$K = 300$

Можно заметить, что при значении $K = 50$ мы уже хорошо видим картинку, а при значении $K = 100$ мы можем отчетливо различать элементы нашей картинки.

Также все полученные изображения можно посмотреть на диске

Далее был выполнен подсчет сжатия внутри функции “compression_ratio”, в которой мы находим «процент видимости» полученной картинки в отличие от исходной по формуле: $\frac{k*(1+n+m)}{n*m}$, где $m, n -$ размеры изображения

Также не была прикреплена картинка с $K = 1$, так как при сохранении она имеет слишком маленький размер, что не позволяет даже ее открыть. Поэтому прикрепляем изображение из matplotlib, где видно все картинки с их «сжатием».



Данный фрагмент можно также посмотреть на диске

Построим таблицу сжатия изображений в зависимости от K

K	1	5	10	20	50	100	150	200	300
Сжатие	0.1%	0.5%	1.0%	2.01%	5.01%	10.03%	15.04%	20.05%	30.06%

Таб.1 Таблица сжатия изображений в зависимости от K

После выполнения задания можно сделать вывод, что значение K напрямую влияет на качество нашего исходного изображения. Как было сказано выше, при анализе картинок можно заметить, что при значении $K = 100$ мы можем отчетливо различать элементы нашей картинки, при значении $K = 50$ мы уже хорошо видим картинку, понимая что примерно на ней изображено, при значении $K = 10$ становится сложно разобрать элементы нашего изображения. В последнем значении $K = 300$ мы получили максимально приближенное, из взятых значений, изображение относительно исходного. По полученному в процентах сжатию можно сделать вывод насколько меньше требуется данных для хранения изображения в сжатом виде, т.е. если для исходного изображения требуется примерно 3мБ памяти, то для изображения с $K = 10$, имеющего сжатие 1%, требуется в 100 раз меньше памяти, что равно 0.03мБ. Мы можем делать данный вывод, так как ранее было сказано, что формула сжатия равна $\frac{k*(1+n+m)}{n*m}$. В данной формуле мы обозначаем m, n как размеры нашего исходного изображения, где m – количество строк матрицы исходного изображения, n – столбцов. Далее можно сделать вывод, что укороченное SVD-разложение с формулой $A_k = U_k \Sigma_k V_k^T$ хранит в себе только первые K сингулярные числа, следовательно матрица U с размером $m \times m$ становится матрицей U_k с размером $m \times k$, Σ_k с размером $k \times k$, а матрица V_k^T с размером $k \times n$, итого для хранения сжатого изображения нам требуется $m \times k + k + k \times n$ элементов, откуда мы получаем формулу $\frac{k*(1+n+m)}{n*m}$.

Кроме того, на диске хранится код, хранящийся в файле “zad1.py”, с помощью которого происходит полное выполнение данного задания для любой картинки. Наша картинка также хранится на диске с именем “image.jpg”. Для того, чтобы произвести такие изменение с другой картинкой нужно выполнить код с другой картинкой, названной “image.jpg”, либо внутри кода поменять значение переменной *image_path = "image.jpg"*

Задание 2

Для работы с заданием 2 был выбран файл *data2.txt*.

Удаление знаков препинания происходит с помощью строчки кода processed_documents = [re.sub(r'[^\w\s]', '', doc.lower()) for doc in documents].

Далее производим лемматизацию текста с помощью библиотеки *pymorphy3*.

С помощью библиотеки *nltk* проведем удаление «шумных» слов. Для этого нужно скачать файл шумных слов с помощью команды “nltk.download('stopwords')”.

```
    def preprocess_text(text):  # Удаление знаков препинания
        text = re.sub(r"[^\w\s]", "", text)  # Токенизация
        words = word_tokenize(text.lower())  # Лемматизация и удаление стоп-слов
        res_words = [morph.parse(word)[0].normal_form for word in words if word not in
stop_words]
        return res_words
```

Создаем терм-документную матрицу из оставшихся слов, отображая частоту употребления слов. После выполнения TDM-матрицы получаем результаты вида:

Координаты	Значение
(0, 14)	2

Таб.2 Пример полученных данных TDM матрицы.

Также мы имеем список наших слов. Приводим пример списка до 15 слова.

['автономный' 'аспект' 'больший' 'важный' 'вариационный' 'вблизи' 'весь'
'взаимный' 'включать' 'влияние' 'внешний' 'возмущение' 'возникновение'
'волна' 'время']

Полный словарь прикреплен в приложении.

По данным TDM-матрицы можно сделать вывод, что слово *'время'* употребляется в тексте 2 раза

Выполнение SVD-разложения происходит по формуле:

$$TDM = U * \Sigma * V^T$$

Выполняем с помощью python

```
vectorizer = TfidfVectorizer()
term_doc_matrix = vectorizer.fit_transform(processed_texts)

# Выполнение SVD
svd = TruncatedSVD(n_components=2)  # Выбираем 2 темы
U = svd.fit_transform(term_doc_matrix)
Sigma = svd.singular_values_
VT = svd.components_
```

После выполнения мы получили:

Σ: Это сингулярное число, которое показывает "важность" компоненты. Чем больше это число, тем более значимой является компонента для модели.

U: Это матрица левых сингулярных векторов, столбцы которой показывают направление векторов в документе, каждая строка

V: Матрица правых сингулярных векторов содержит информацию о том, как термины распределяются по компонентам. В матрице V каждая строка отвечает за одно слово и показывает насколько слово связано с темой.

После того, как мы получили SVD-разложение, нам нужно проанализировать два первых сингулярных числа и соответствующие им правые и левые векторы, а также восстановить две основные темы из текста по топ-5 словам. С помощью python находим эти слова:

```
feature_names = vectorizer.get_feature_names_out()

def display_top_words(model, feature_names, n_top_words):
    for topic_idx, topic in enumerate(model):
        print(f"Тема {topic_idx + 1}:")
        top_features = [feature_names[i] for i in topic.argsort()[:-n_top_words - 1:-
1]]
        print(", ".join(top_features))

print("Ключевые слова для каждой темы:")
display_top_words(VT, feature_names, 5)
```

Получаем:

Тема 1:

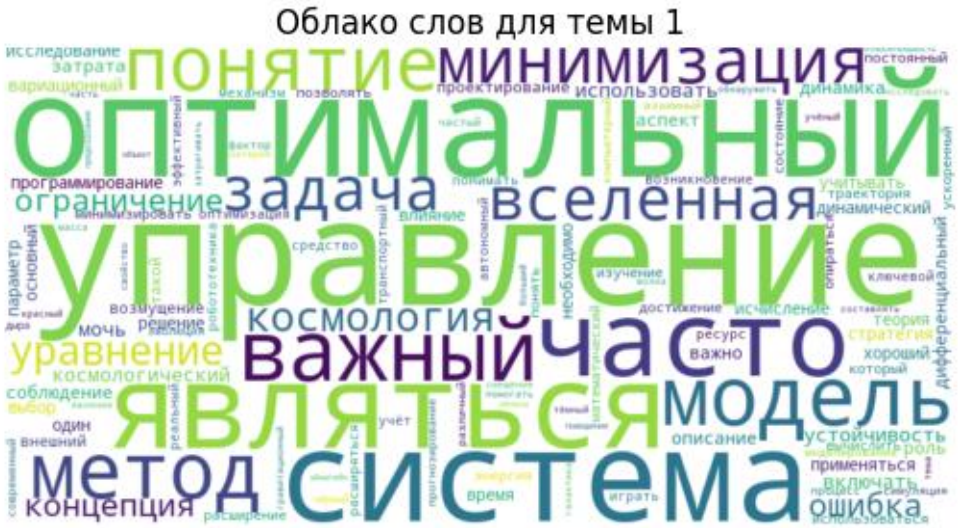
управление, оптимальный, система, являться, часто.

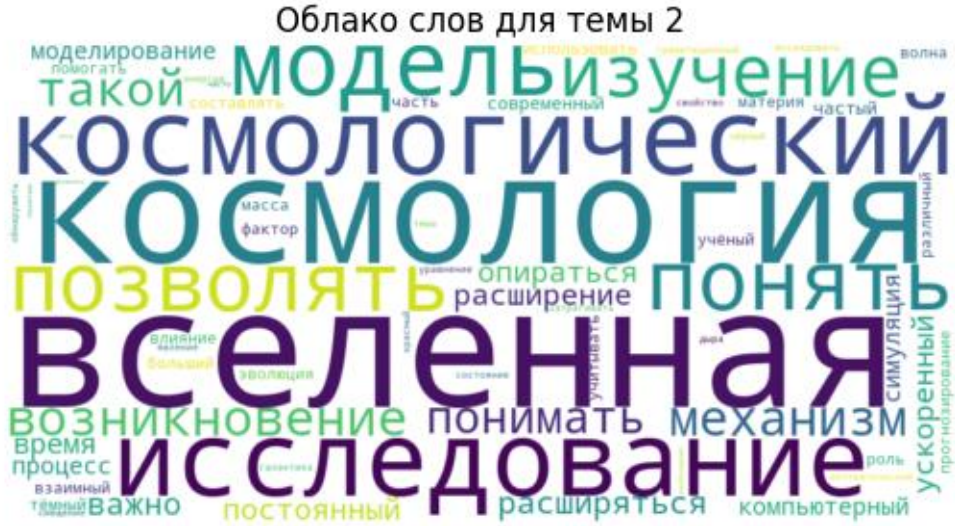
Тема 2:

вселенная, космология, космологический, исследование, модель.

На основе полученных тем можно сделать вывод, что первая тема связана с оптимальным управлением, а вторая тема связана с космосом и его исследованием.

После полученных тем создадим облака слов с помощью библиотеки *wordcloud* на python. Получим:





В полученных облаках мы можем наблюдать значимость слов по их размеру. Так мы получаем значимость слов по темам:

Тема 1:

управление, оптимальный, система, являться, часто.

Тема 2:

вселенная, космология, космологический, исследование, модель.

Они также были получены ранее с помощью python.

Вывод

В проделанной работе мы научились использовать SVD-разложение картинки, где с помощью SVD-разложения мы научились сжимать картинку. Кроме того, мы научились работать с текстом, с помощью SVD-разложения мы смогли выделить основную тему текста и часто используемые слова.

Мы научились использовать мощный инструмент линейной алгебры — SVD-разложение — для решения задач, связанных с обработкой изображений и текстов, что является важным шагом в изучении анализа данных и машинного обучения и поможет нам в дальнейших исследованиях и обучении.

Приложение 1

Словарь слов

['автономный' 'аспект' 'больший' 'важно' 'важный' 'вариационный' 'вблизи'
'взаимный' 'включать' 'влияние' 'внешний' 'возмущение' 'возникновение'
'волна' 'время' 'вселенная' 'выбор' 'вычислить' 'галактика'
'гравитационный' 'динамика' 'динамический' 'дифференциальный'
'достижение' 'дыра' 'задача' 'затрагивать' 'затрата' 'играть' 'изучение'
'использовать' 'использоваться' 'исследование' 'исследовать' 'исчисление'
'ключевой' 'компьютерный' 'концепция' 'космологический' 'космология'
'который' 'красный' 'масса' 'математический' 'материя' 'метод' 'механизм'
'минимизация' 'минимизировать' 'моделирование' 'модель' 'мочь'
'необходимо' 'обнаружить' 'объект' 'ограничение' 'один' 'опираться'
'описание' 'оптимальный' 'оптимизация' 'основный' 'относительность'
'ошибка' 'параметр' 'поведение' 'позволять' 'помогать' 'понимать'
'понятие' 'понять' 'постоянный' 'предсказание' 'применяться'
'прогнозирование' 'программирование' 'проектирование' 'процесс'
'различный' 'расширение' 'расширяться' 'реальный' 'ресурс' 'решение'
'робототехника' 'роль' 'свойство' 'симуляция' 'система' 'смещение'
'соблюдение' 'современный' 'составлять' 'состояние' 'средство'
'стратегия' 'такой' 'тема' 'теория' 'траектория' 'транспортный' 'тёмный'
'управление' 'уравнение' 'ускоренный' 'устойчивость' 'учитывать' 'учёный'
'учёт' 'фактор' 'хороший' 'часто' 'частый' 'часть' 'чёрный' 'эволюция'
'эйнштейн' 'энергия' 'эффективный' 'явление' 'являться']