



Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ
по лабораторной работе № 7
по курсу «Основы искусственного интеллекта»
на тему: «Кластеризация»

Студент ИУ7-13М
(Группа)

(Подпись, дата)

Орду М. А.
(И. О. Фамилия)

Преподаватель

(Подпись, дата)

Строганов Ю. В.
(И. О. Фамилия)

2025 г.

СОДЕРЖАНИЕ

1 Теоретическая часть	3
1.1 Постановка задачи	3
1.2 Предобработка текстовых данных	3
1.2.1 Выделение словоформ	3
1.2.2 Выделение начальных форм слов	3
1.3 Векторизация документов	4
1.3.1 One-Hot Encoding	4
1.3.2 Term Frequency-Inverse Document Frequency	4
1.3.3 N-граммы	5
1.4 Методы кластеризации	5
1.4.1 Метод K-средних	5
1.4.2 Метод С-средних	6
1.4.3 Метод Гат-Гевы	7
1.5 Метрики оценки качества кластеризации	8
2 Практическая часть	9
2.1 Цель и задачи эксперимента	9
2.2 Описание набора данных	9
2.3 Предобработка текста и векторизация	9
2.4 Создание векторных представлений	10
2.5 Результаты кластеризации с лемматизацией	10
2.5.1 Метод K-средних	10
2.5.2 Метод С-средних	14
2.6 Результаты кластеризации без лемматизации	17
2.6.1 Метод K-средних	17
2.6.2 Метод С-средних	21
2.7 Метод Гат-Гевы	24
ПРИЛОЖЕНИЕ А Исходный код программы	31
ПРИЛОЖЕНИЕ Б Код программы для кластеризации . .	34
ПРИЛОЖЕНИЕ В Код программы для Гат-Гевы	43

1 Теоретическая часть

1.1 Постановка задачи

В данной лабораторной работе решается задача кластеризации текстовых документов. Для заданного набора текстов необходимо:

1. Преобразовать текстовые документы в векторное представление;
2. Применить методы кластеризации: k-средних, с-средних и Гат-Гевы;
3. Проанализировать качество кластеризации при различном количестве кластеров, определить среднее внутрикластерное расстояние и среднее межкластерное расстояние для каждого рассматриваемого случая.

1.2 Предобработка текстовых данных

1.2.1 Выделение словоформ

Словоформа — это конкретная грамматическая форма слова, встречающаяся в тексте. Процесс выделения словоформ из текста включает:

- Разбиение текста на отдельные слова;
- Приведение к нижнему регистру;
- Удаление служебных частей речи;
- Удаление пунктуации и специальных символов.

В данном подходе слова сохраняются в том виде, в котором они встречаются в тексте.

1.2.2 Выделение начальных форм слов

Лемматизация — процесс приведения слова к его начальной форме (лемме). В рамках этой работы применялась библиотека `pymorphy3`, которая:

- Определяет часть речи слова

- Учитывает морфологические характеристики (род, число, падеж и др.)
- Возвращает нормальную форму слова

Преимущество лемматизации можно назвать уменьшение размерности пространства признаков за счет объединения различных форм одного слова. В нашем случае, лемматизация приводит к уменьшению размерности вектора текстового документа.

1.3 Векторизация документов

После предобработки тексты преобразуются в числовые векторы. Преобразовать текст можно различными способами. В рамках работы рассмотрим статистические методы векторизации.

В основе статистических подходов лежит парадигма «Мешка слов», которая абстрагируется от порядка слов и рассматривает текст как неупорядоченную коллекцию терминов. Эти методы основаны на подсчёте частот слов, что приводит к созданию векторных представлений высокой размерности. Несмотря на свою простоту и интерпретируемость, они обладают общими недостатками, такими как семантическая слепота, пренебрежение порядком слов и проблема разреженности данных.

1.3.1 One-Hot Encoding

Самый просто и примитивный метод векторизации текста, результатом которого является матрица с единицами и нулями внутри. 1 говорит о том, что какой-то текстовый элемент встречается в предложении (или в нашем случае документе). 0 говорит о том, что элемент не встречается в предложении.

1.3.2 Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency или сокращенно TF-IDF — один из наиболее распространённых и эффективных статистических методов. Вес слова вычисляется как произведение двух компонент: количество раз, когда слово встретилось в документе и натурального логарифма от количества документов деленное на количество документов содержащее

этот символ. Формула для расчета представлена в (1.1).

$$w_{i,j} = tf_{i,j} \times \log \frac{N}{df_i} \quad (1.1)$$

где:

- $tf_{i,j}$ — количество раз, когда слово i встретилось в документе j ;
- df_i — количество документов содержащее символ i ;
- N — общее количество документов.

Логарифмическая мера снижает влияние служебных частей речи и повышает значимость слов, характерных для конкретного документа.

1.3.3 N-граммы

Для учёта локального контекста и фиксации устойчивых словосочетаний применяется расширение классического подхода — N-граммы. Элементами векторизации становятся не отдельные слова (униграммы), а последовательности из N соседних слов или символов. Это позволяет моделировать такие выражения, как «машинное обучение», в виде единого семантического токена. Однако использование N-грамм приводит к комбинаторному росту размерности пространства признаков, что создаёт серьёзные вычислительные сложности и требует больше данных для обучения моделей.

1.4 Методы кластеризации

1.4.1 Метод К-средних

К-средних — классический метод четкой кластеризации. Четкая кластеризация — кластеризация с заранее известным количеством кластеров. Алгоритм можно описать следующим образом:

1. Начальный выбор координат центроидов k кластеров/ Выбор, как правило, носит случайный характер, однако существуют модификации метода, где начальный выбор выполняется на основе поиска максимально удалённых друг от друга потенциальных центроидов кластеров;

2. Назначение каждого объекта ближайшему центроиду;
3. Пересчет центроидов кластеров;
4. Повторение шагов 2-3 до сходимости.

Целевая функция:

$$J = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|^2 \quad (1.2)$$

1.4.2 Метод С-средних

Метод с-средних – метод нечёткой кластеризации, где каждый объект принадлежит всем кластерам с определённой степенью принадлежности от 0 до 1.

Метод минимизирует взвешенную сумму квадратов расстояний:

$$J_m(U, V) = \sum_{i=1}^c \sum_{j=1}^n (u_{ij})^m d_{ij}^2 \quad (1.3)$$

где:

- $d_{ij} = \|\mathbf{x}_j - \mathbf{v}_i\|$ — евклидово расстояние
- $V = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_c\}$ — множество центроидов
- $m > 1$ — параметр нечёткости.

Алгоритм метода с-средних можно описать следующим образом.

1. Пересчёт центров кластеров. Центр каждого кластера вычисляется как взвешенное среднее всех точек данных. Весом для точки служит её степень принадлежности к этому кластеру, возведённая в степень m . Таким образом, точки с высокой степенью принадлежности сильнее влияют на положение центра;
2. Для каждой точки данных заново вычисляется её принадлежность ко всем кластерам. Степень принадлежности обратно пропорциональна расстоянию от точки до центра кластера: чем точка ближе к центру, тем выше её принадлежность к этому кластеру. Для одной точки сумма всех принадлежностей равна 1;

3. Вычисления прекращаются, когда изменения в матрице принадлежностей между двумя последовательными итерациями становятся меньше заданного порога ε ;

1.4.3 Метод Гат-Гевы

Метод Гат-Гевы — модификация алгоритма с-средних, который используется адаптивная метрика расстояния (*англ. fuzzy maximum likelihood estimation, FMLE*) вместо стандартной евклидовой.

Алгоритм можно описать следующим образом:

1. Случайным образом задаются начальные степени принадлежности точек к кластерам;
2. Центроиды вычисляются как взвешенные средние всех точек, где веса — степени принадлежности в степени m ;
3. Для каждого кластера вычисляется ковариационная матрица, которая описывает:
 - Направление наибольшего разброса точек;
 - Степень вытянутости кластера;
 - Ориентацию кластера в пространстве.
4. Расстояние от точки до центроиды кластера вычисляется с учётом его формы. Точка может быть ближе к центру вытянутого кластера по его длинной оси, даже если евклидово расстояние велико;
5. Степени принадлежности пересчитываются на основе новых расстояний;
6. Процесс повторяется, пока изменения в матрице принадлежностей не станут меньше заданного значения.

1.5 Метрики оценки качества кластеризации

Среднее внутрикластерное расстояние показывает насколько похожи объекты внутри одного кластера. Вычисляется по формуле (1.4):

$$W = \frac{1}{N} \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\| \quad (1.4)$$

где N — общее количество объектов, μ_i — центроид кластера C_i .

Среднее межкластерное расстояние показывает насколько хорошо кластеры отделены друг от друга. Вычисляется по формуле (1.5):

$$B = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^k \|\mu_i - \mu_j\| \quad (1.5)$$

где k — число кластеров.

2 Практическая часть

2.1 Цель и задачи эксперимента

Целью работы является исследование эффективности методов кластеризации Гат-Гевы, К-средних и С-средних на размеченном наборе текстовых данных. Для этого необходимо:

1. Предобработать тексты и создать векторные представления документов двумя методами;
2. Провести кластеризацию с использованием указанных методов, варьируя количество кластеров k ;
3. Рассчитать метрики внутрикластерного и межкластерного расстояний для оценки качества разбиения;
4. Сравнить полученные кластеризации с экспертной разметкой.

2.2 Описание набора данных

Работа проводились на предоставленном размеченном наборе текстовых данных. Набор содержит $N = 50$ текстовых документов, относящихся к $K = 7$ тематическим кластерам согласно экспертной разметке.

2.3 Предобработка текста и векторизация

Векторизация документов выполнялась в два этапа: предобработка текста и создание числового вектора.

В ходе первого этапа исходный текст очищался от знаков препинания, символов перевода строк и приводился к нижнему регистру. После этого каждый документ разбивался на отдельные слова (токены) с помощью простого разделителя по пробелам.

Для получения начальных форм слов использовалась библиотека `rutagger3`. Каждый токен, полученный на предыдущем этапе, преобразовывался в свою нормальную форму (лемму).

2.4 Создание векторных представлений

После лемматизации для каждого из двух методов (мешок словоформ и мешок лемм) выполнялась векторизация. В результате каждый документ d_i был представлен как вектор \mathbf{x}_i , где каждая компонента соответствовала весу конкретного слова (словоформы или леммы) в документе. Служебные части речи при векторизации не учитывались.

2.5 Результаты кластеризации с лемматизацией

2.5.1 Метод К-средних

Ниже на рисунках 2.1-2.7 представлены результаты кластеризации методом К-средних с лемматизацией.

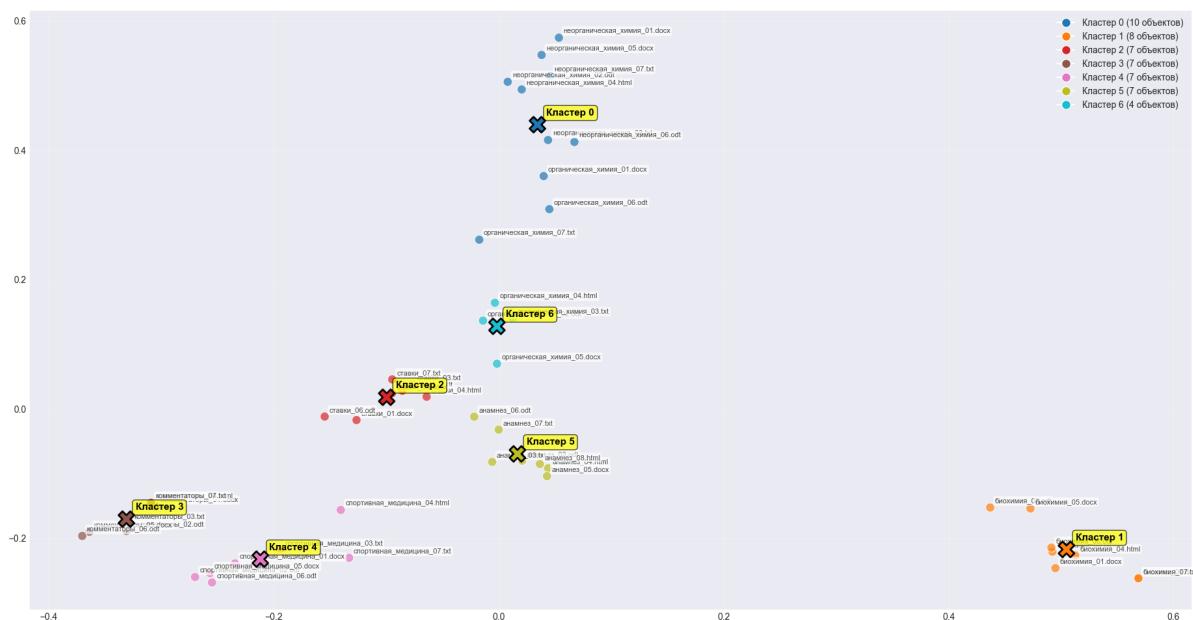


Рисунок 2.1 – Результат кластеризации методом К-средних, при $K=7$

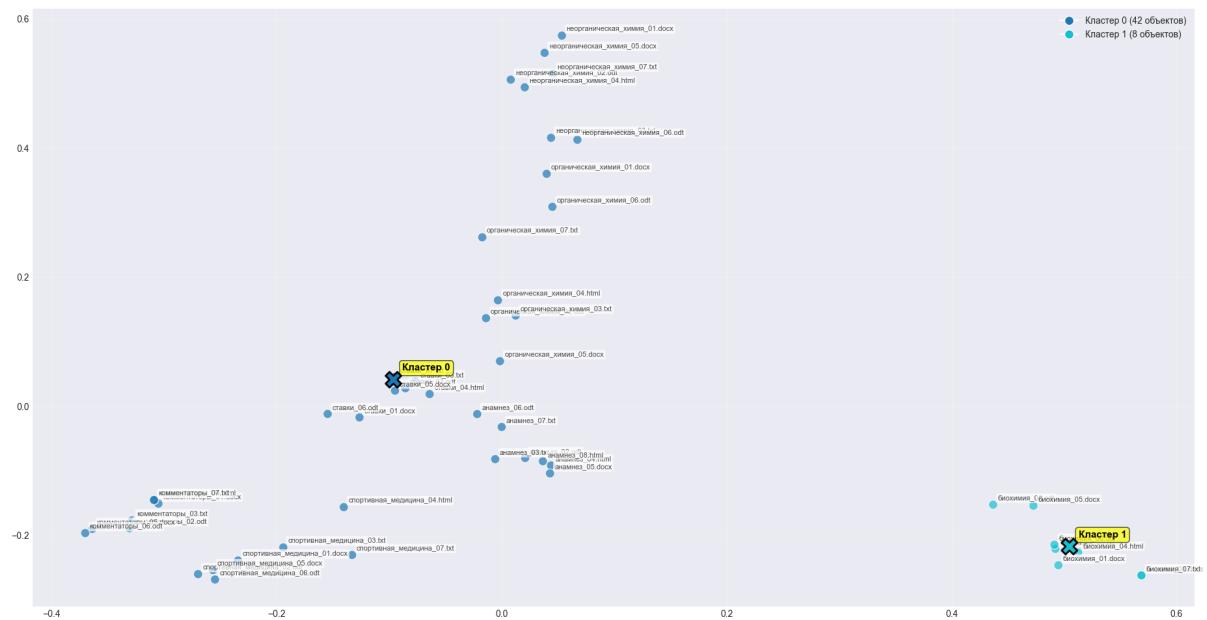


Рисунок 2.2 – Результат кластеризации методом K-средних, при K=2

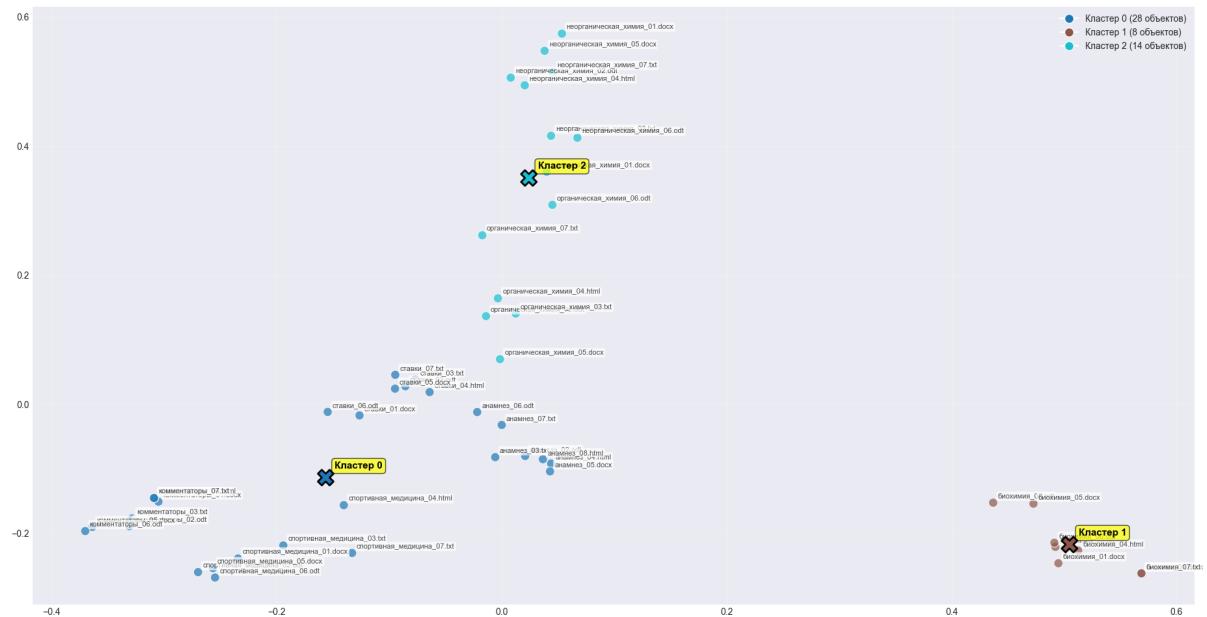


Рисунок 2.3 – Результат кластеризации методом K-средних, при K=3

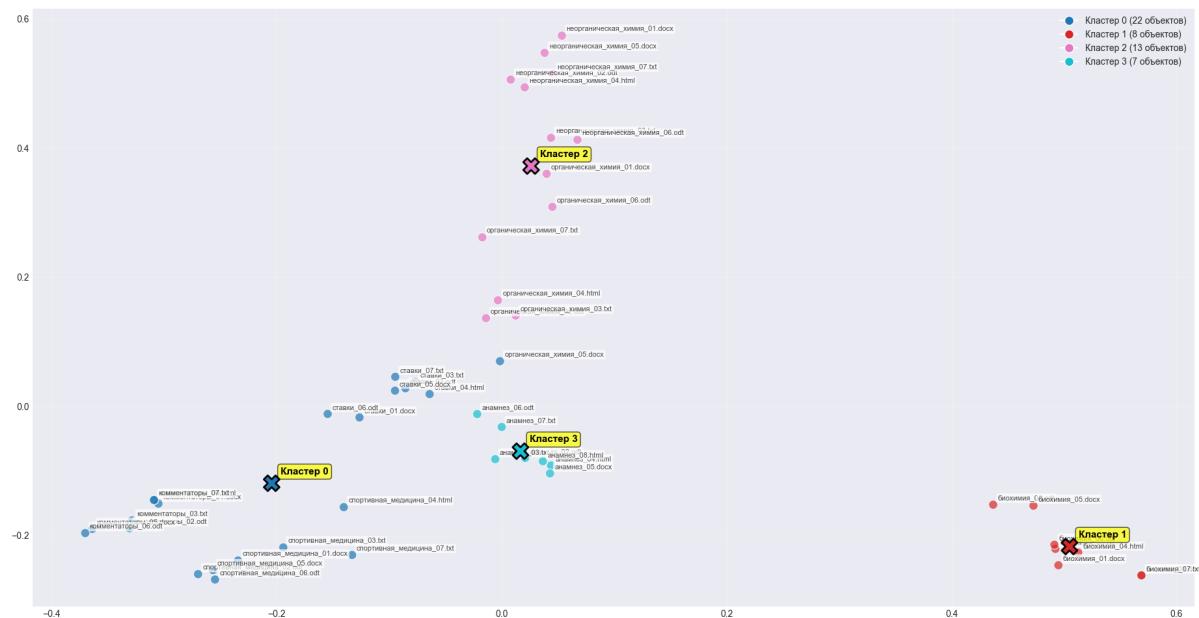


Рисунок 2.4 – Результат кластеризации методом K-средних, при K=4

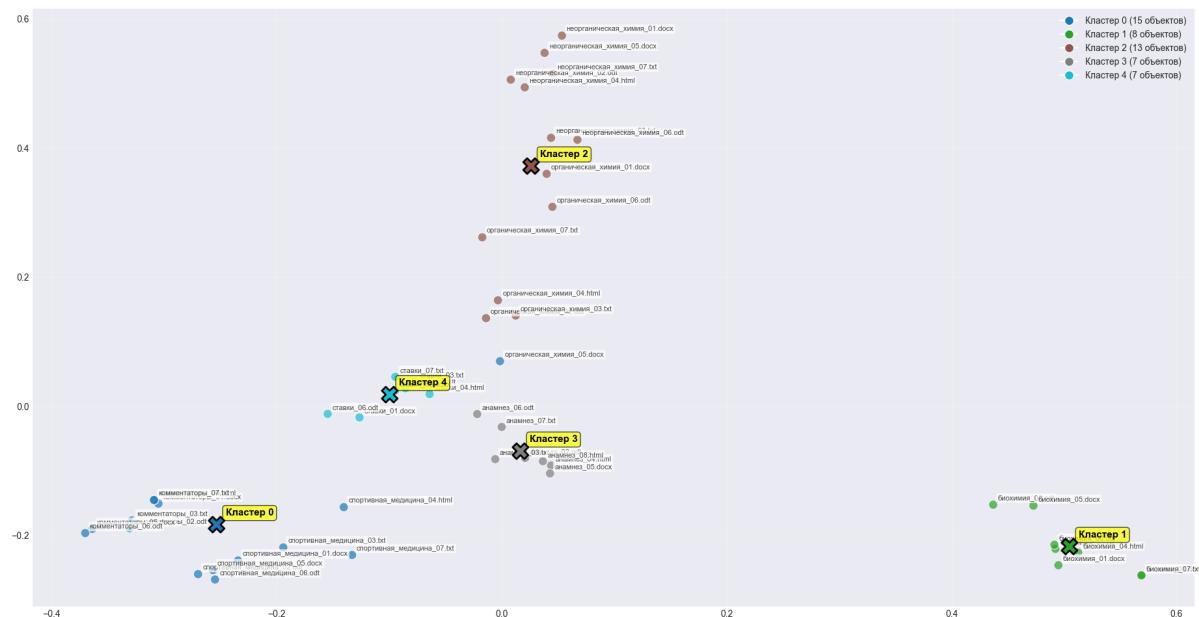


Рисунок 2.5 – Результат кластеризации методом K-средних, при K=5

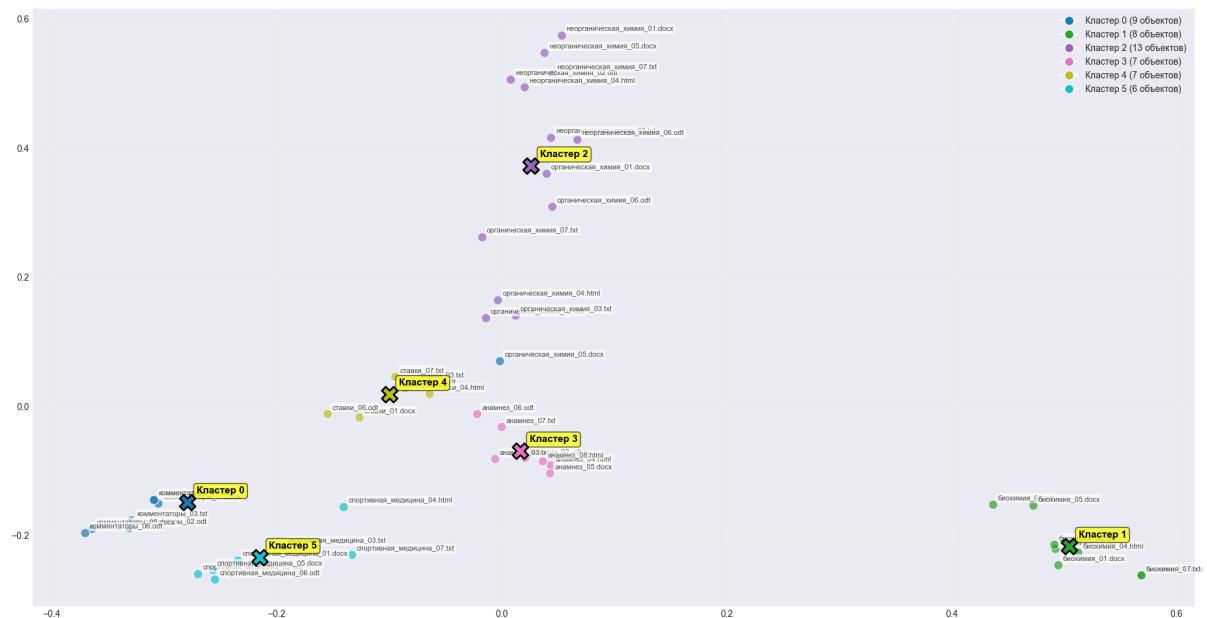


Рисунок 2.6 – Результат кластеризации методом K-средних, при K=6

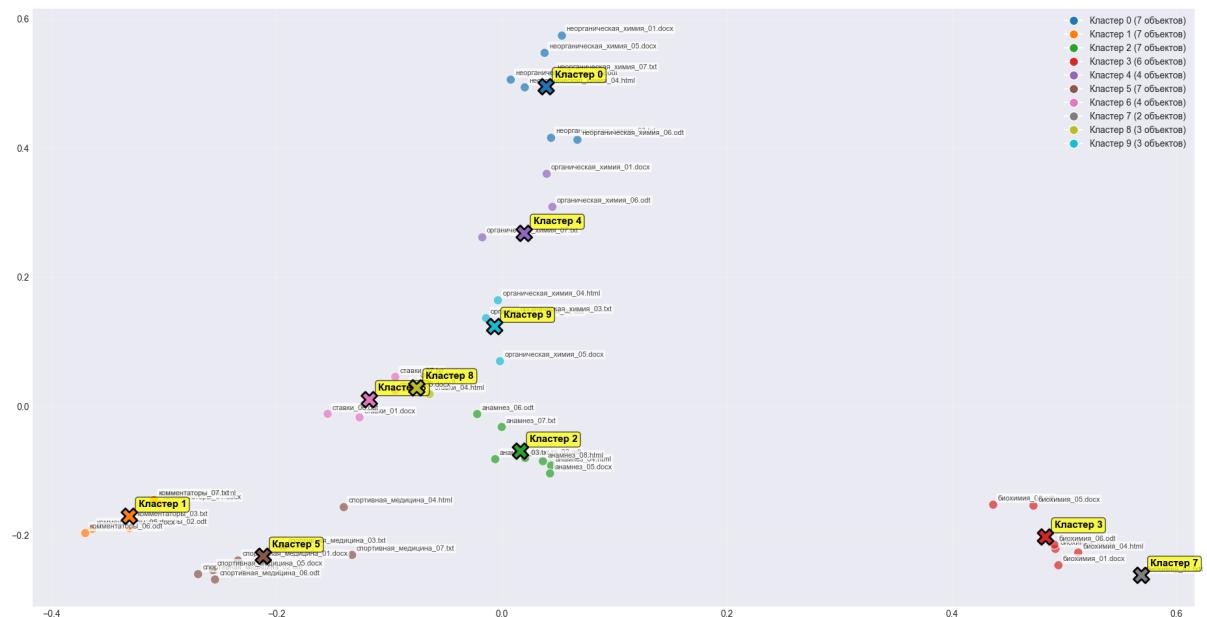


Рисунок 2.7 – Результат кластеризации методом K-средних, при K=10

2.5.2 Метод С-средних

Ниже на рисунках 2.8-2.14 представлены результаты кластеризации методом С-средних с лемматизацией.

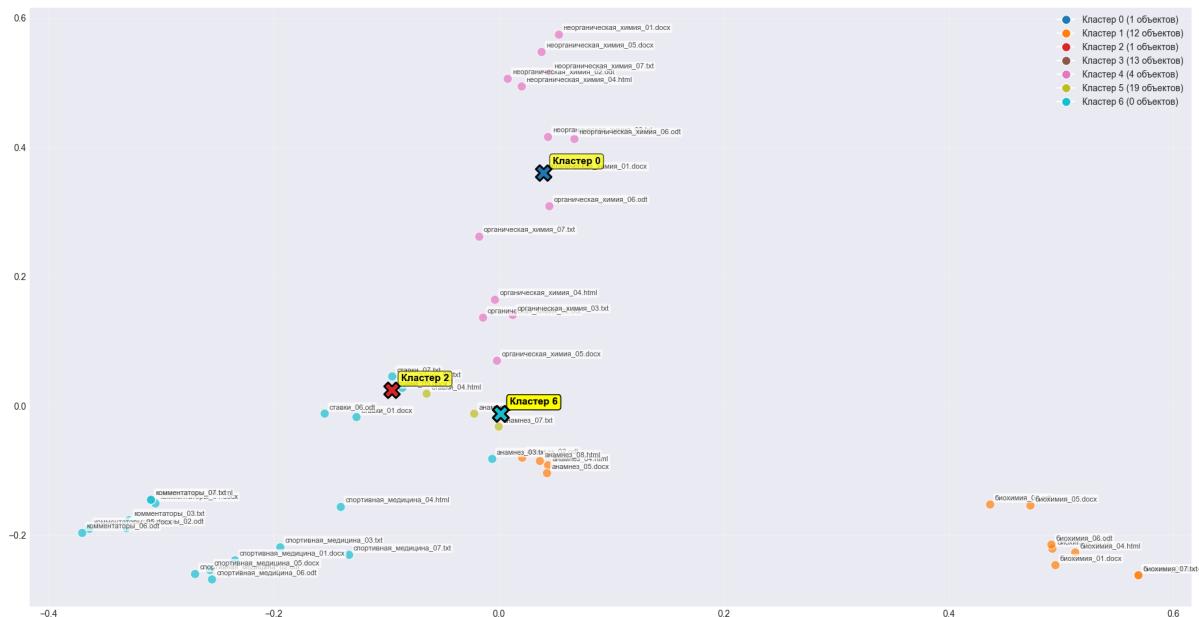


Рисунок 2.8 – Результат кластеризации методом С-средних, при $K=7$

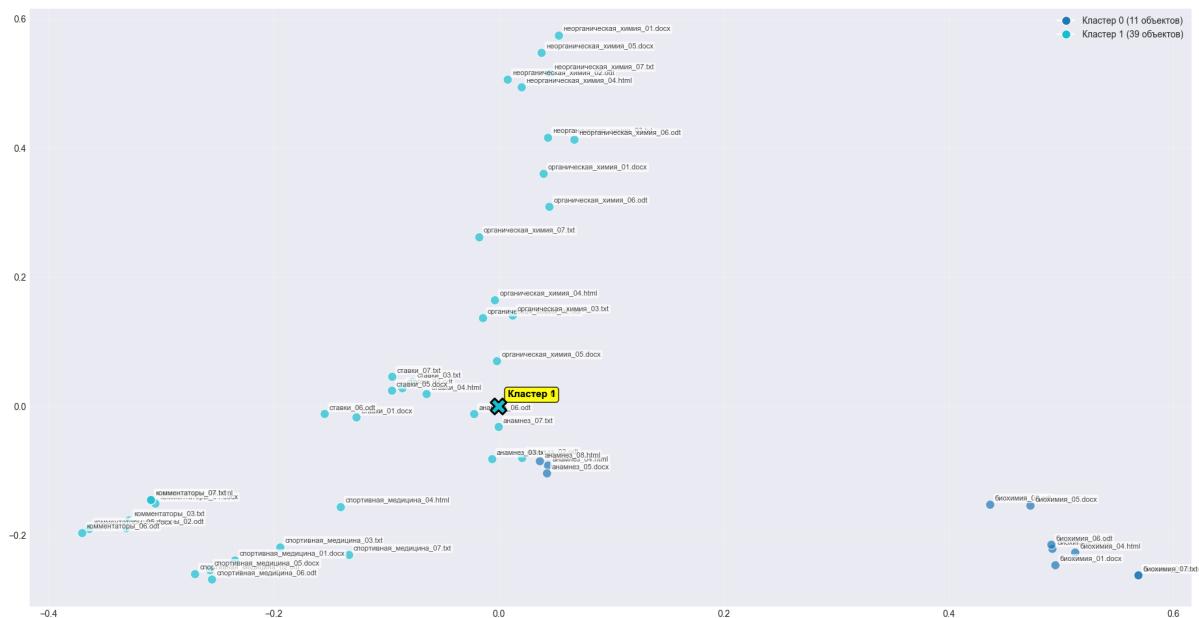


Рисунок 2.9 – Результат кластеризации методом С-средних, при $K=2$

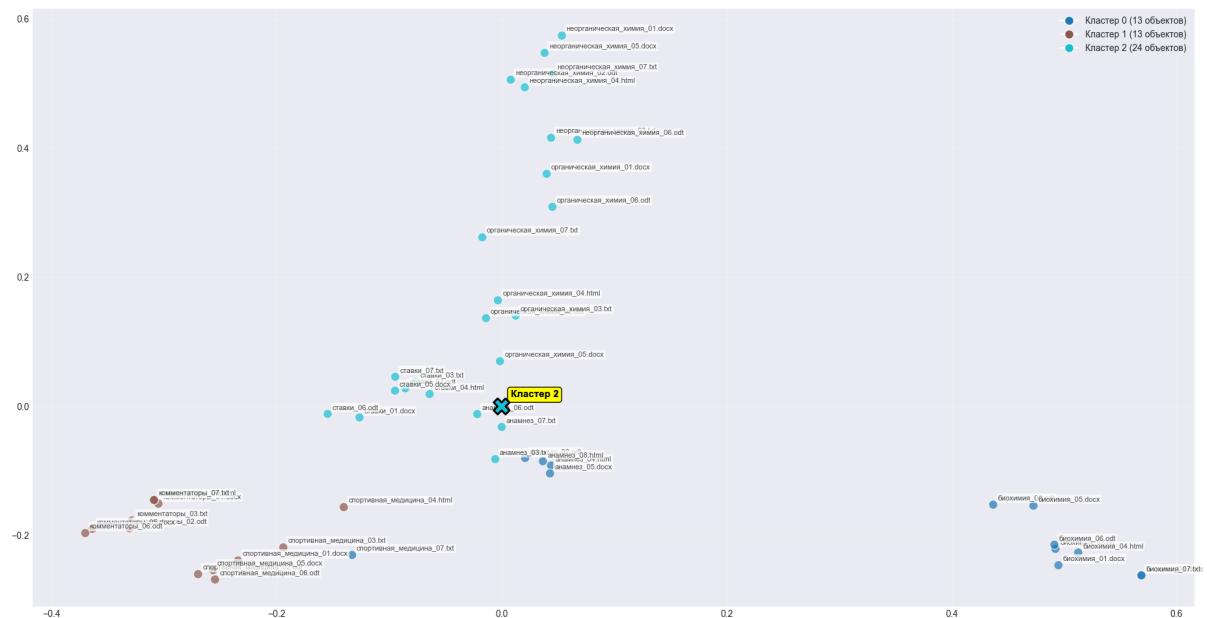


Рисунок 2.10 – Результат кластеризации методом С-средних, при K=3

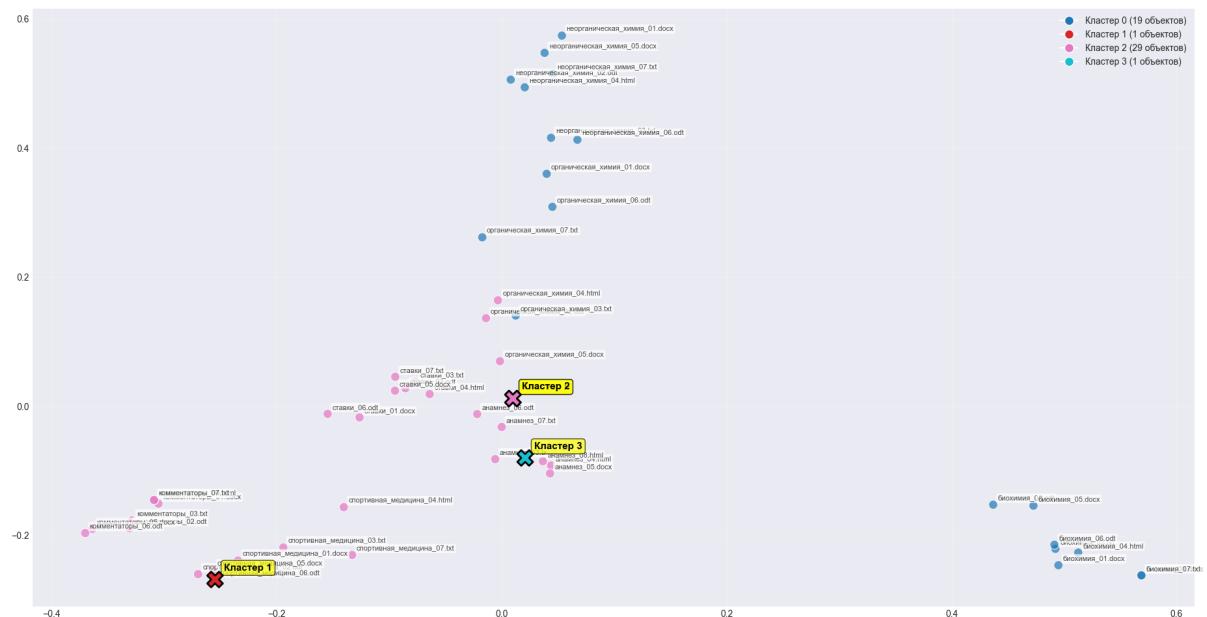


Рисунок 2.11 – Результат кластеризации методом С-средних, при K=4



Рисунок 2.12 – Результат кластеризации методом С-средних, при K=5

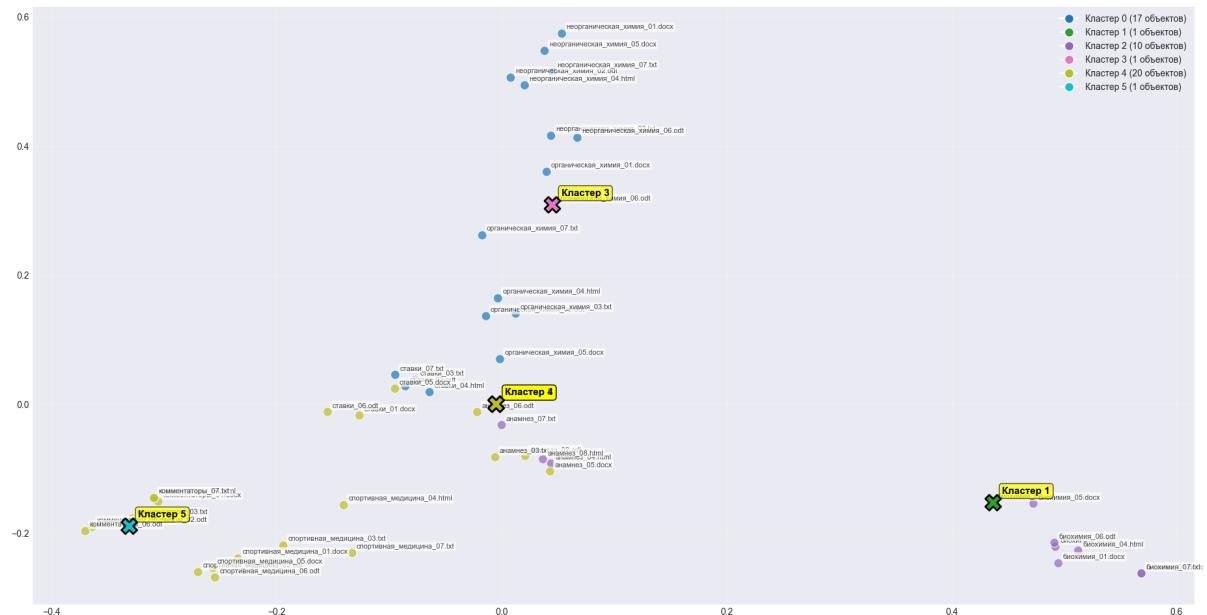


Рисунок 2.13 – Результат кластеризации методом С-средних, при K=6

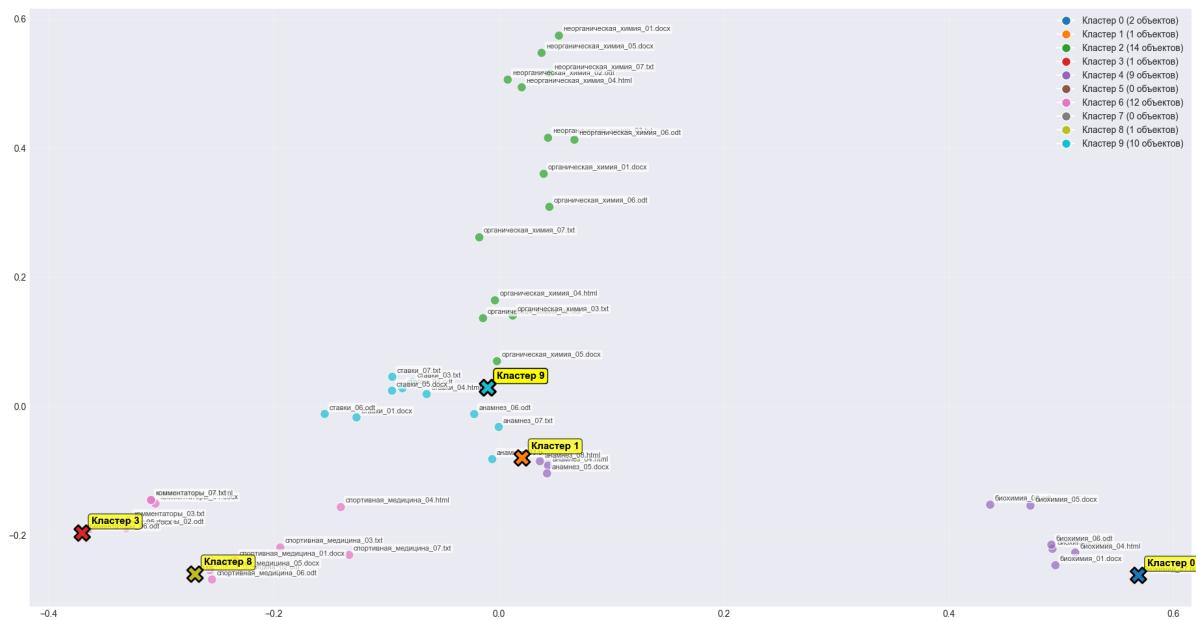


Рисунок 2.14 – Результат кластеризации методом С-средних, при $K=10$

2.6 Результаты кластеризации без лемматизации

2.6.1 Метод К-средних

Ниже на рисунках 2.15-2.21 представлены результаты кластеризации методом К-средних без лемматизации.

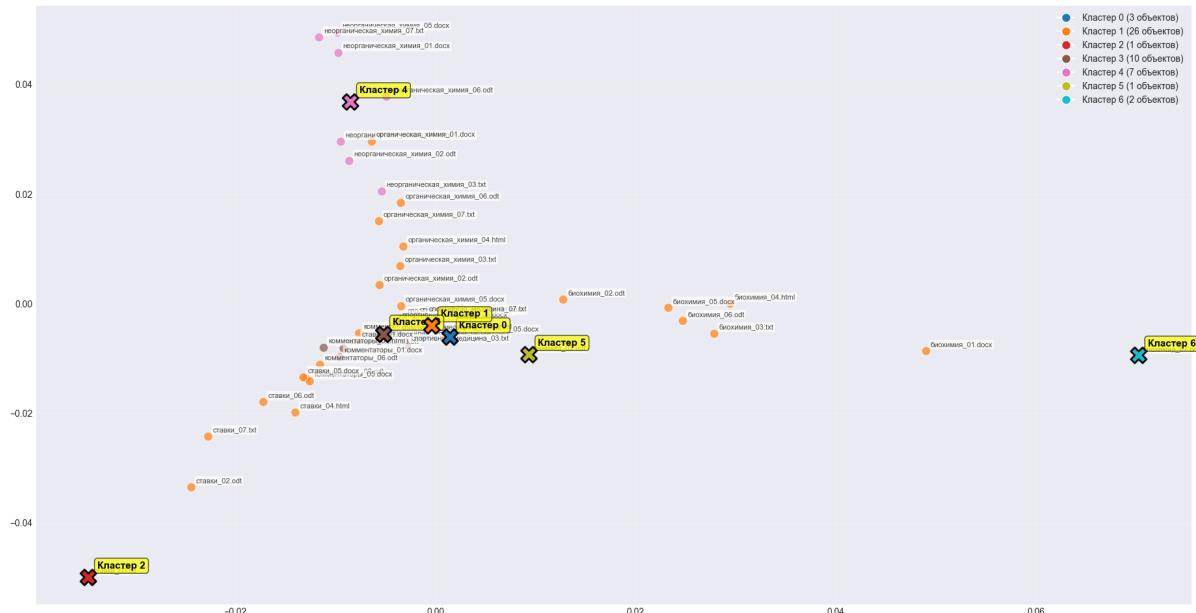


Рисунок 2.15 – Результат кластеризации методом К-средних, при $K=7$

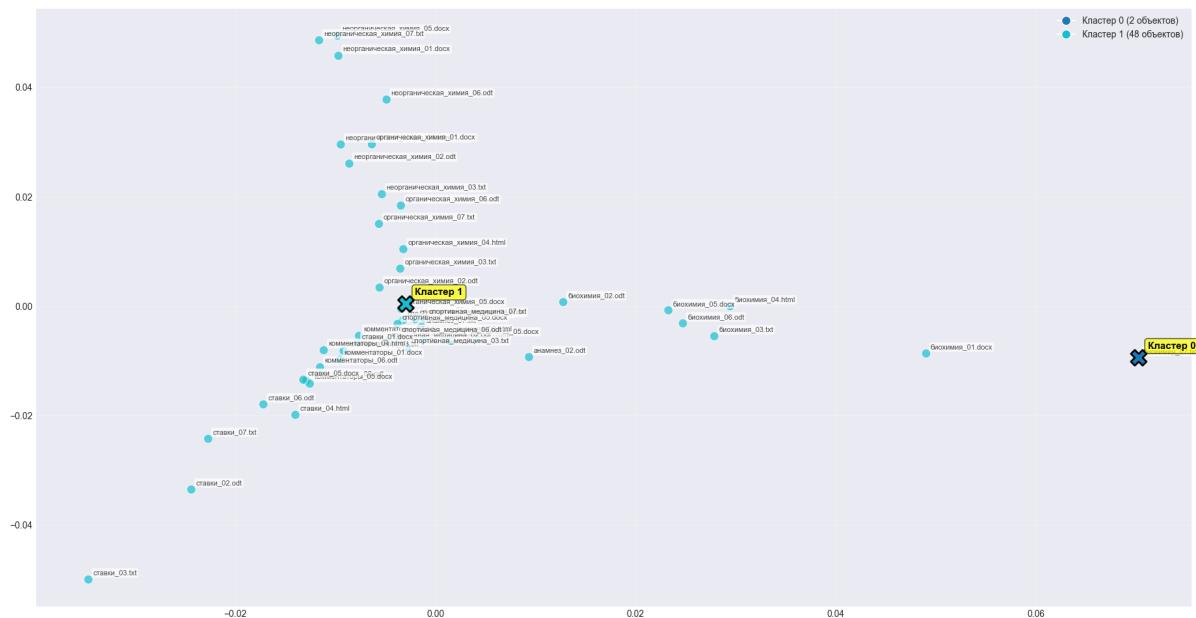


Рисунок 2.16 – Результат кластеризации методом К-средних, при $K=2$

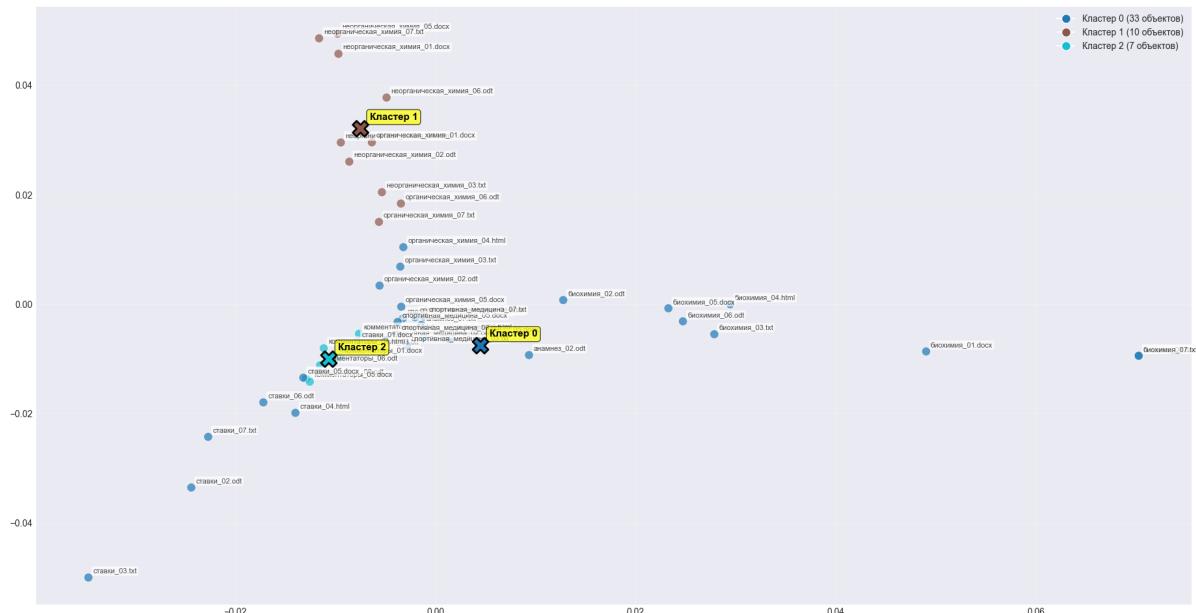


Рисунок 2.17 – Результат кластеризации методом К-средних, при $K=3$

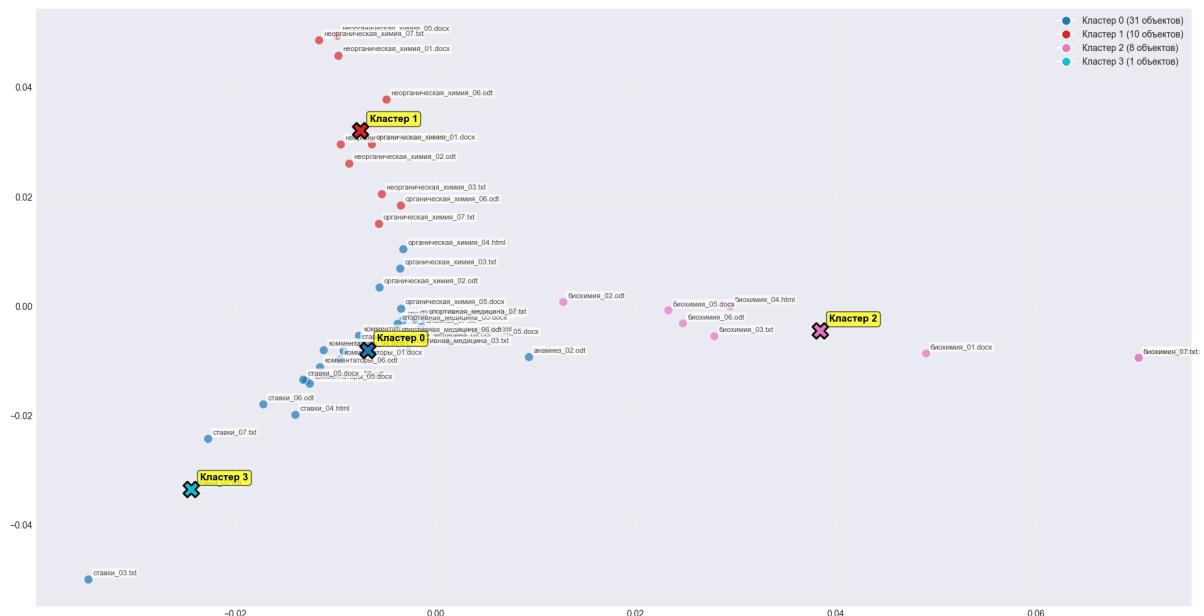


Рисунок 2.18 – Результат кластеризации методом К-средних, при $K=4$

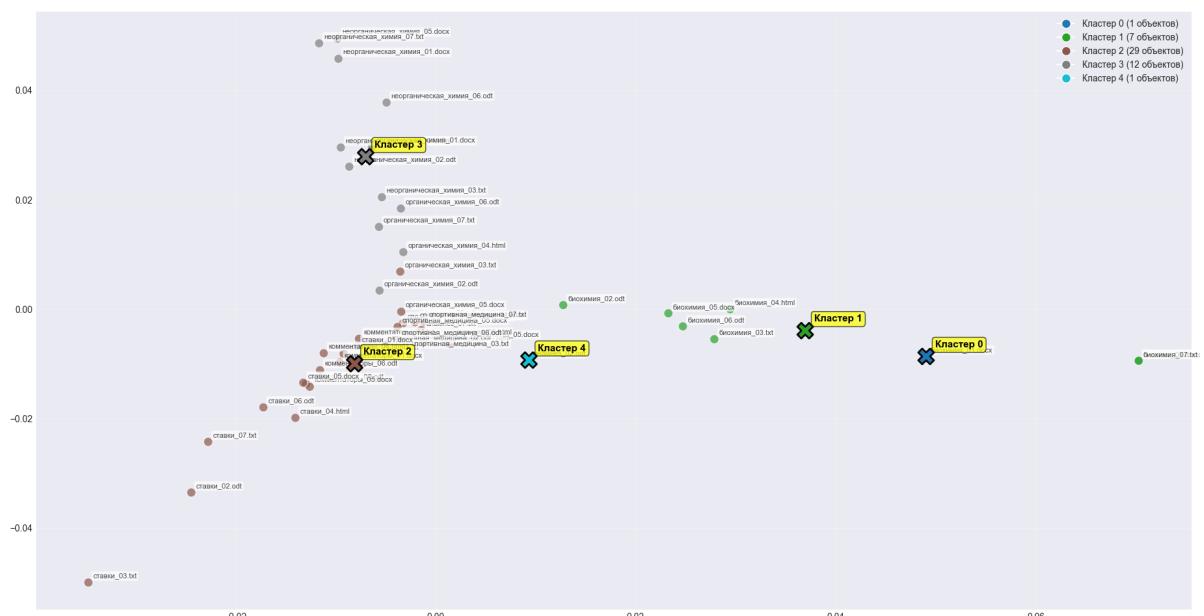


Рисунок 2.19 – Результат кластеризации методом К-средних, при $K=5$

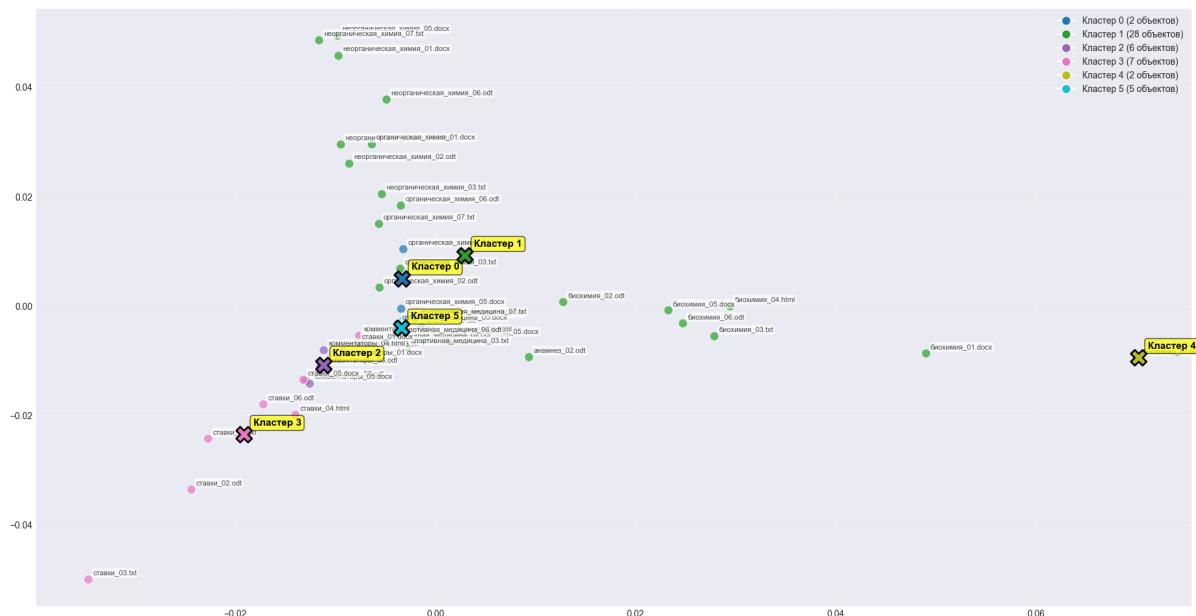


Рисунок 2.20 – Результат кластеризации методом К-средних, при K=6

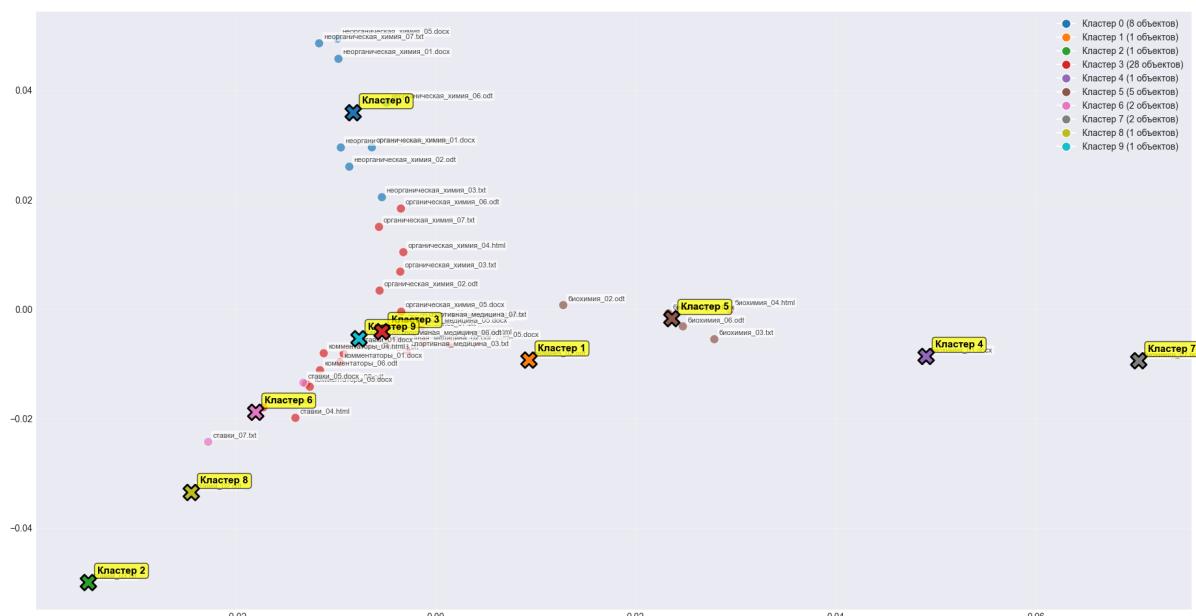


Рисунок 2.21 – Результат кластеризации методом К-средних, при K=10

2.6.2 Метод С-средних

Ниже на рисунках 2.22-2.28 представлены результаты кластеризации методом С-средних без лемматизации.

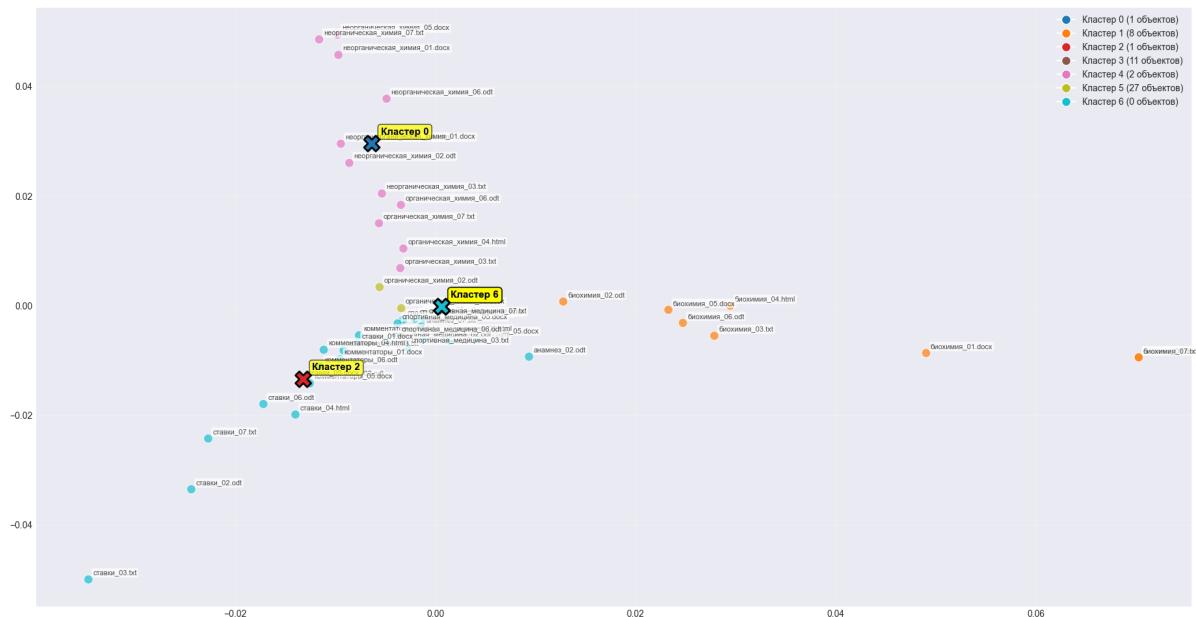


Рисунок 2.22 – Результат кластеризации методом С-средних, при $K=7$

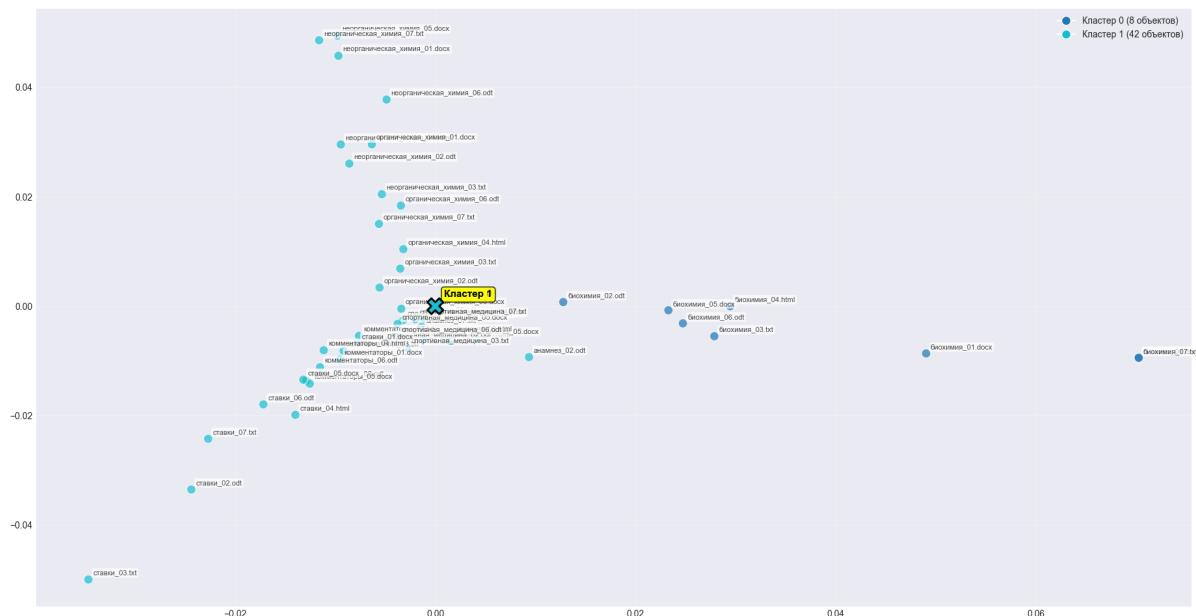


Рисунок 2.23 – Результат кластеризации методом С-средних, при $K=2$

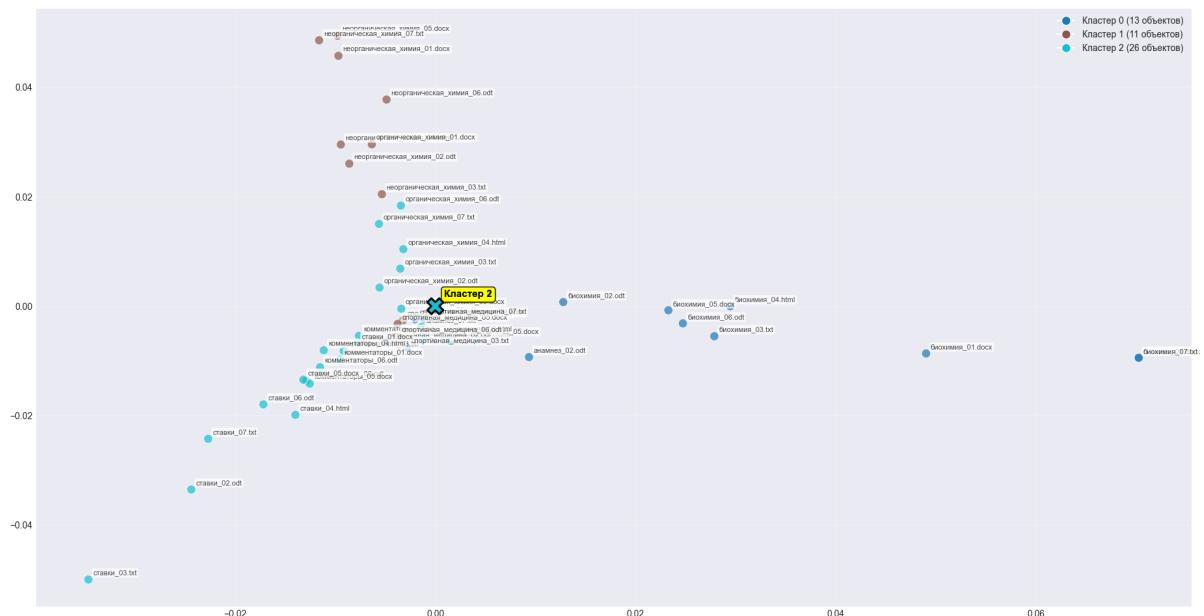


Рисунок 2.24 – Результат кластеризации методом С-средних, при $K=3$

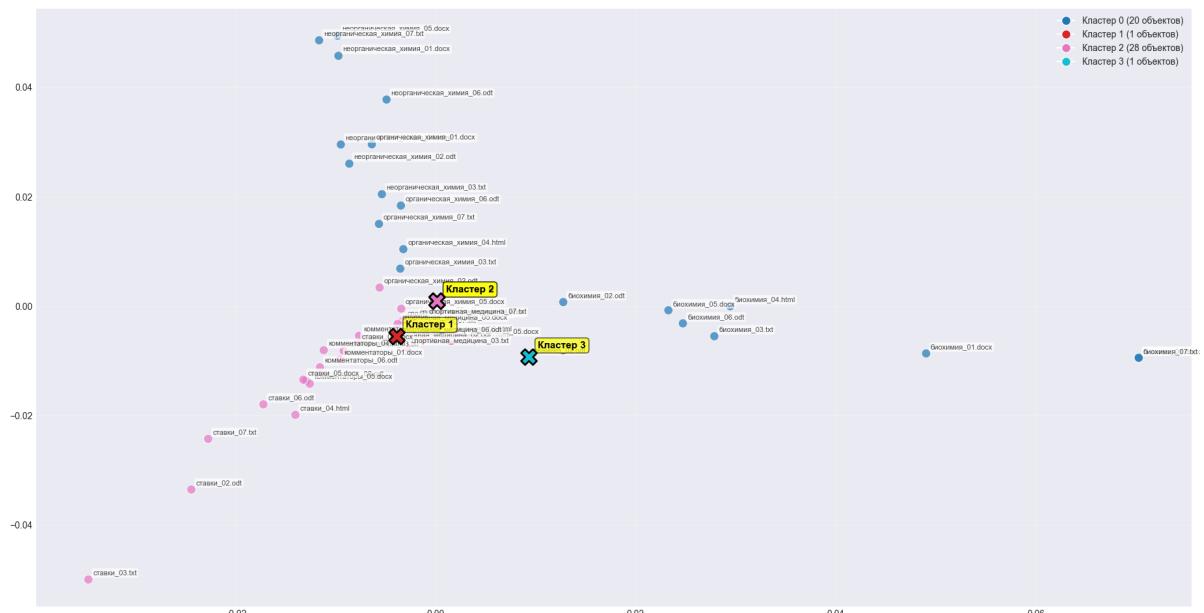


Рисунок 2.25 – Результат кластеризации методом С-средних, при $K=4$

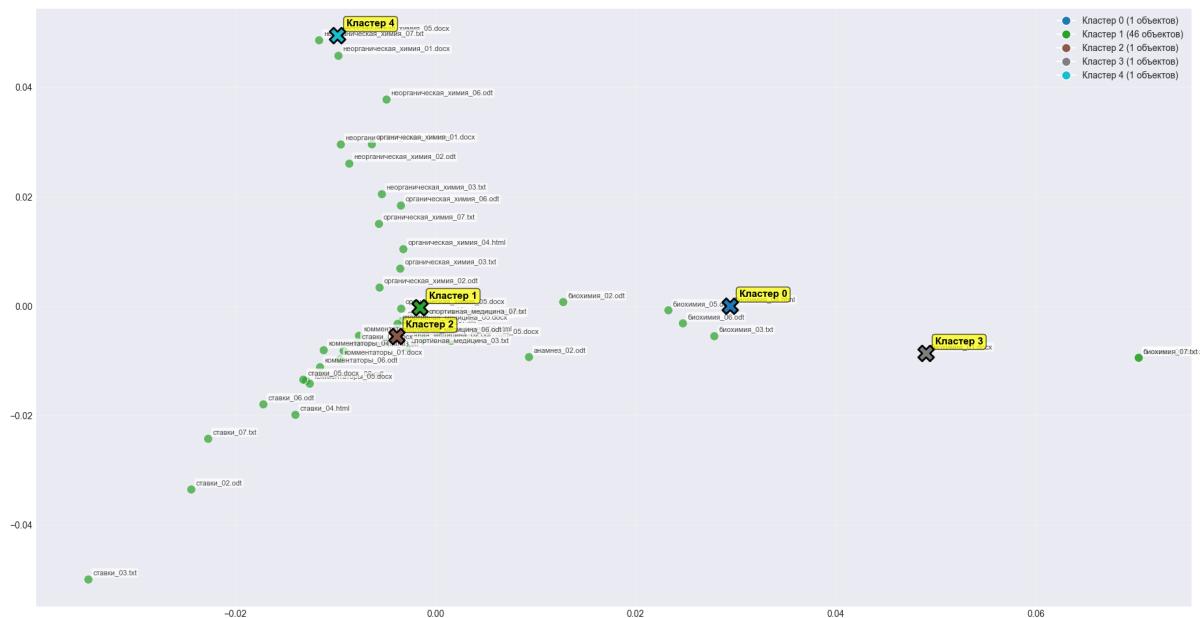


Рисунок 2.26 – Результат кластеризации методом С-средних, при $K=5$

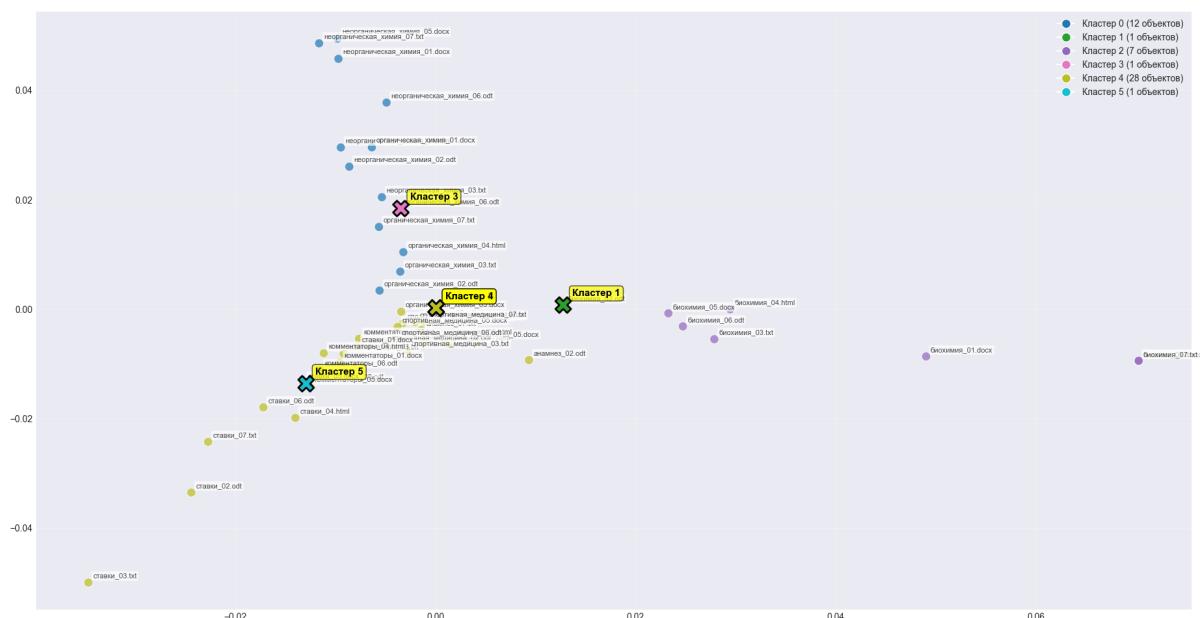


Рисунок 2.27 – Результат кластеризации методом С-средних, при $K=6$

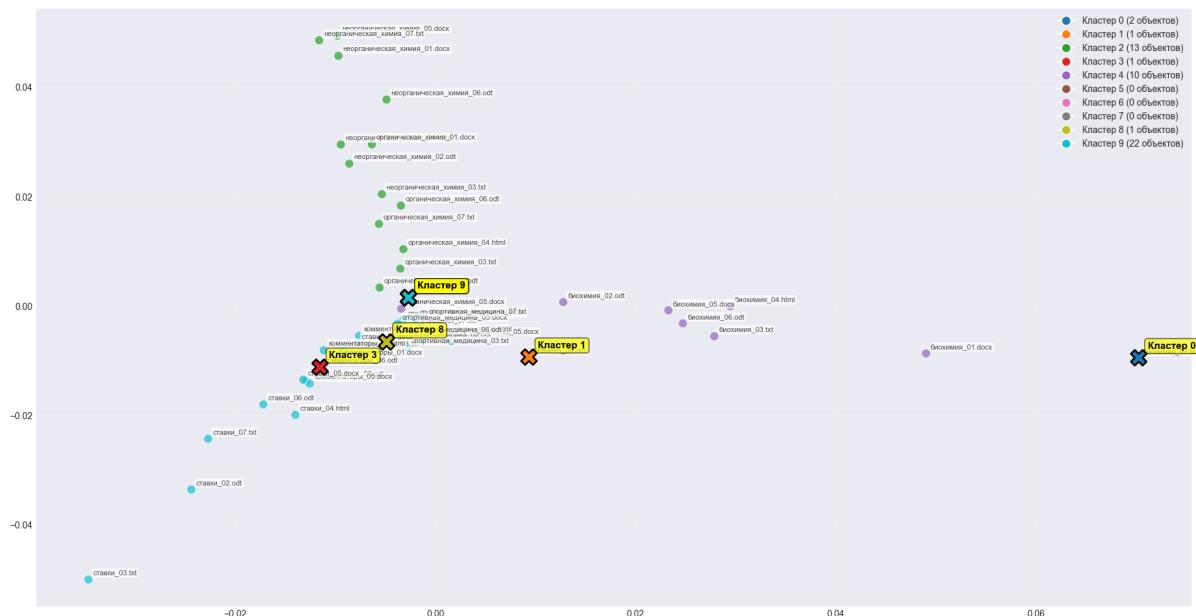


Рисунок 2.28 – Результат кластеризации методом С-средних, при $K=10$

2.7 Метод Гат-Гевы

Решение задачи кластеризации методом Гат-Гевы выполнялась на языке `matlab` ввиду отсутствия готовых реализаций на языке `python` (справедливости ради, мною были найдены готовые реализации этого метода кластеризации, однако ввиду того что они были написаны достаточно давно, текущие версии библиотек их не поддерживали, а заниматься их реанимацией мне не хотелось, посему было принято решение реализовать этот метод в `matlab`, где Гат-Гева входит в состав одного из *toolbox*).

В `matlab` метод Гат-Гевы реализован в рамках Fuzzy Logic Toolbox. Результат задачи кластеризации методом Гат-Гевы при стандартных размерностях векторов документов, представленный на рисунке 2.29 не дают ожидаемого результата.

Однако, предварительно перед кластеризацией уменьшив размерность векторов методом главных компонент удалось получить результат представленный на рисунке 2.30.

Ниже на рисунках 2.31 и 2.34 представлены примеры решения задачи кластеризации методом Гат-Гевы при различных параметрах.

Ниже в таблицах 2.1 и 2.2 представлены результаты решения задачи кластеризации разными методами при различном количестве кластеров K .

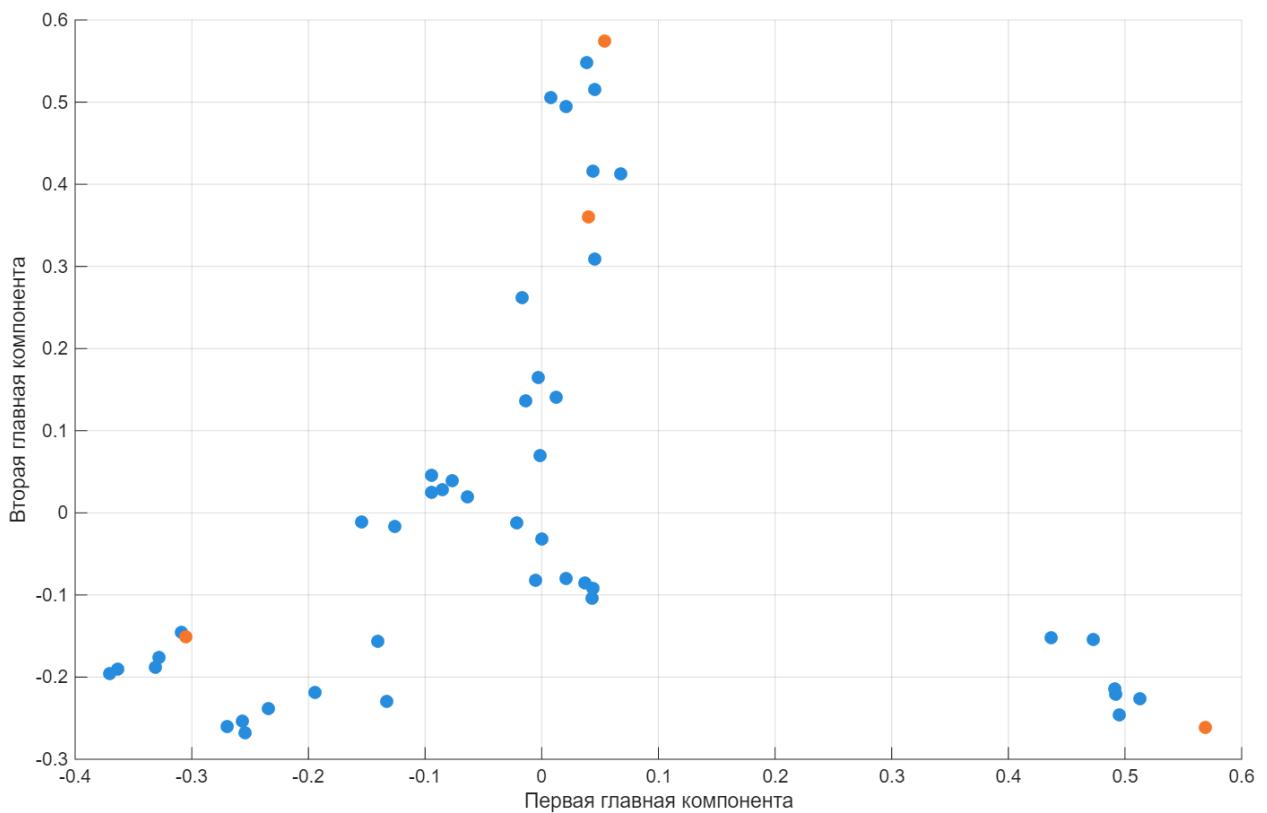


Рисунок 2.29 – Результат кластеризации методом Гат-Гевы при стандартный размерностях векторов, при $K=7$

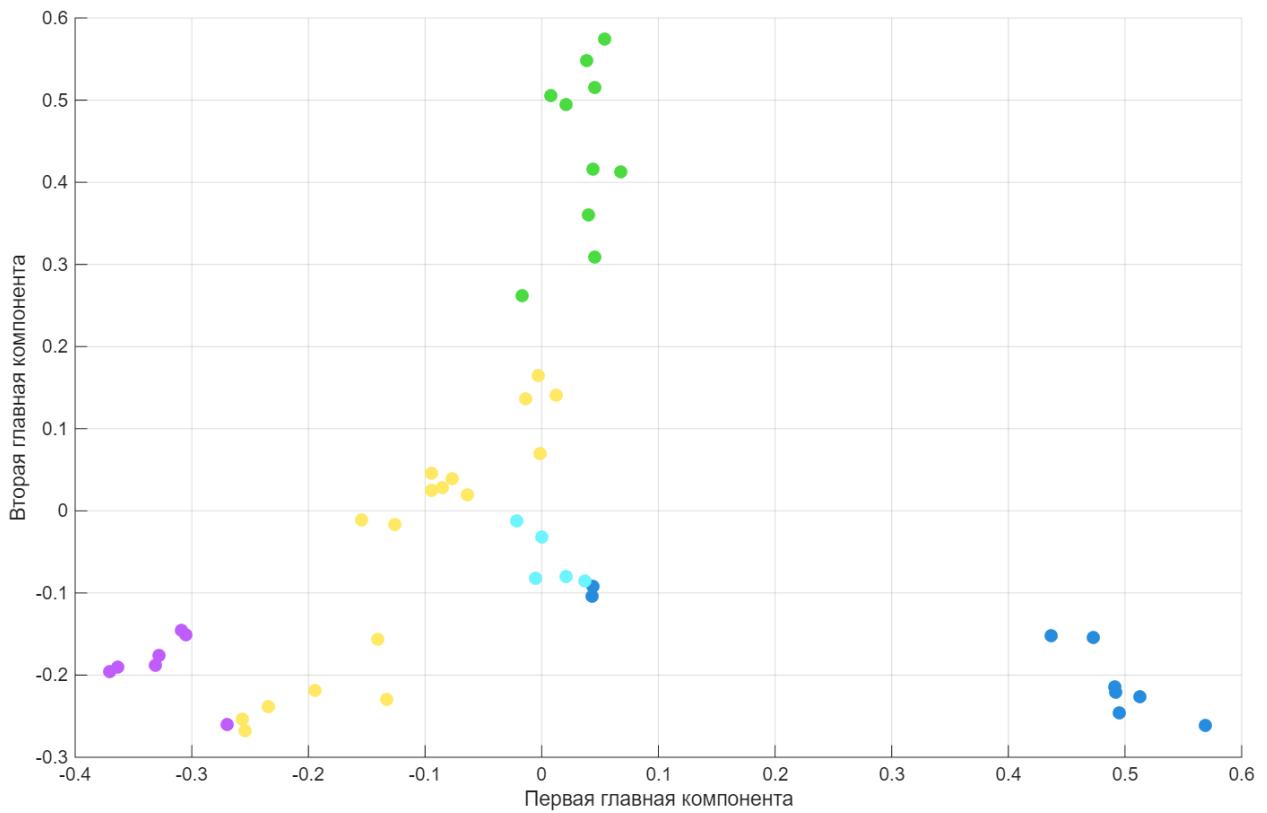


Рисунок 2.30 – Результат кластеризации методом Гат-Гевы при преобразованных векторах и лемматизации, при $K=7$

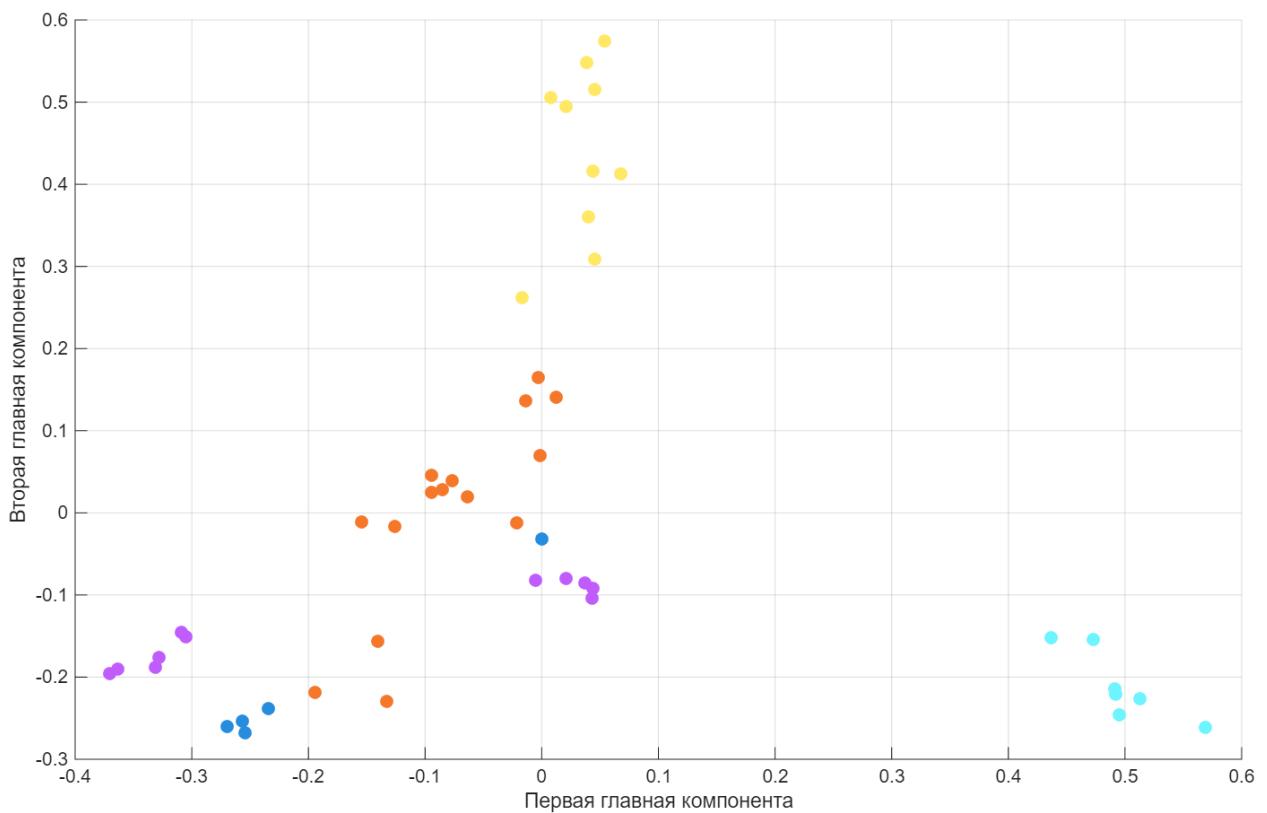


Рисунок 2.31 – Результат кластеризации методом Гат-Гевы при преобразованных векторах и лемматизации, при $K=10$

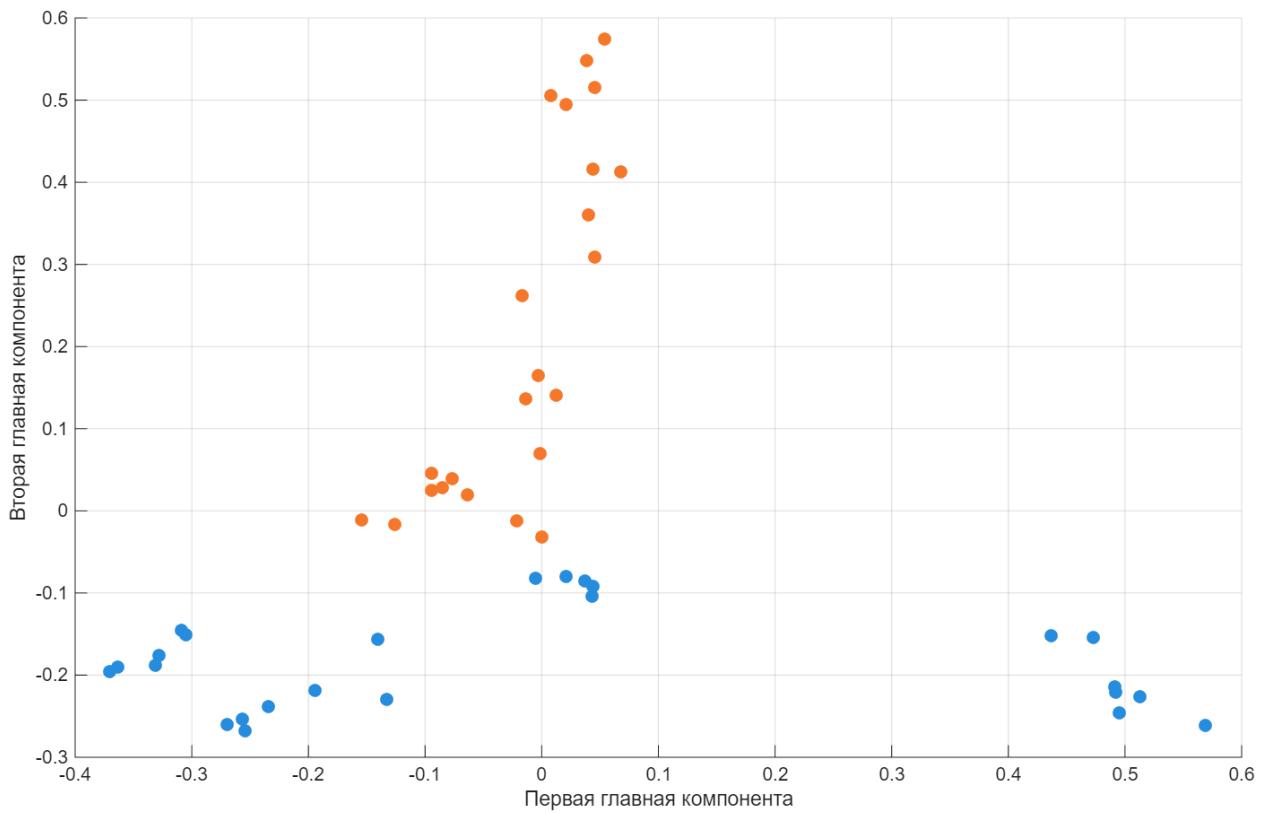


Рисунок 2.32 – Результат кластеризации методом Гат-Гевы при преобразованных векторах и лемматизации, при $K=2$

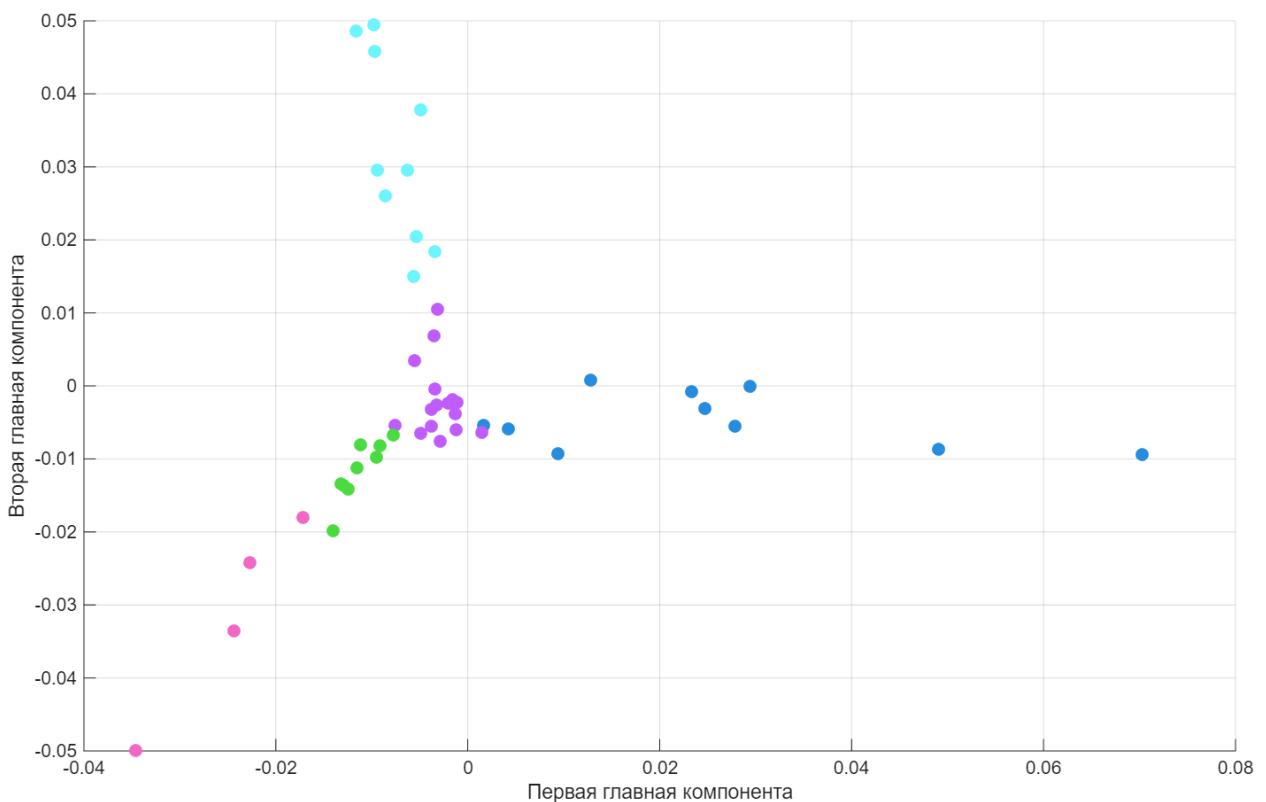


Рисунок 2.33 – Результат кластеризации методом Гат-Гевы при преобразованных векторах и без лемматизации, при $K=10$

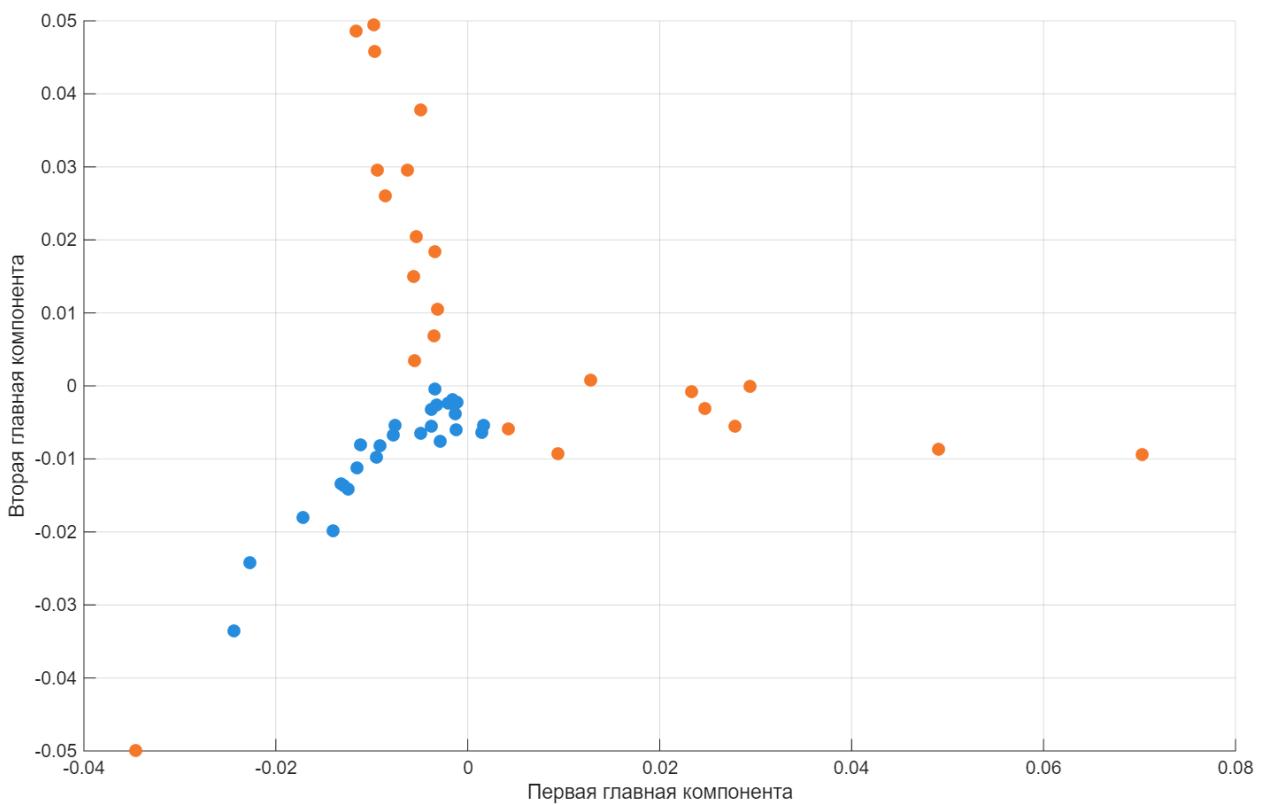


Рисунок 2.34 – Результат кластеризации методом Гат-Гевы при преобразованных векторах и без лемматизации, при $K=2$

Таблица 2.1 – Сравнение результатов кластеризации при лемматизации и различных количествах кластеров К

Критерии	K	K-средних	C-средних	Гат-Гева
Среднее межклusterное расстояние	2	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39
Среднее межклusterное расстояние	3	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39
Среднее межклusterное расстояние	4	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39
Среднее межклusterное расстояние	5	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39
Среднее межклusterное расстояние	6	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39
Среднее межклusterное расстояние	7	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39
Среднее межклusterное расстояние	10	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39

Таблица 2.2 – Сравнение результатов кластеризации без лемматизации и различных количествах кластеров К

Критерии	K	K-средних	C-средних	Гат-Гева
Среднее межклusterное расстояние	2	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39
Среднее межклusterное расстояние	3	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39
Среднее межклusterное расстояние	4	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39
Среднее межклusterное расстояние	5	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39
Среднее межклusterное расстояние	6	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39
Среднее межклusterное расстояние	7	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39
Среднее межклusterное расстояние	10	1,27	1,27	1,27
Среднее внутриклusterное расстояние		1,39	1,39	1,39

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы была решена задача кластеризации текстовых документов с использованием трёх методов: К-средних, С-средних и Гат-Гевы.

Основные выводы:

1. Предобработка данных и векторизация являются критически важными этапами. Применение лемматизации позволило снизить размерность пространства признаков за счёт приведения слов к их начальным формам, что улучшило качество кластеризации;
2. Метод К-средних продемонстрировал стабильность и предсказуемость результатов как при использовании лемматизации, так и без неё;
3. Метод С-средних показал менее стабильные результаты – результаты кластеризации при некоторых количествах кластеров были непредсказуемы. К нестабильности работы стоит также добавить требование настройки параметра нечёткости m ;
4. Проведенные расчеты выявили существование неочевидного порога количества кластеров = 5 при реализации метода Гат-Гевы. При значениях > 5 процедура кластеризации завершается без ошибок, но не производит необходимого разбиения. Неясно, является ли данное ограничение свойством самого алгоритма или особенностью конкретной программной реализации;
5. Сравнение с экспертной разметкой и анализ метрик (среднее внутрекластерное и межкластерное расстояние) позволили оценить, насколько автоматически полученные кластеры соответствуют тематической структуре данных. Наилучшие результаты были достигнуты при числе кластеров, равном к эксперному $K = 7$.

ПРИЛОЖЕНИЕ А

Исходный код программы

Листинг А.1 – Исходный код программы

```
1 import parser
2 import vectorization
3 import numpy as np
4 import cluster
5 import visualizer
6 import dist
7
8
9
10 def dict_to_matrix(data_dict):
11     filenames = list(data_dict.keys())
12     X = np.vstack([data_dict[name] for name in filenames])
13     return X, filenames
14
15 def main():
16     folder_path = "тексты/"
17     results = parser.process_folder_simple(folder_path,
18                                              "результаты.txt")
19     # print(results)
20
21     # с лемматизацией
22     vectors_lemmatized =
23         vectorization.create_text_vectors(results,
24                                           method='tfidf', use_lemmatization=True)
25     # print(vectors_lemmatized)
26     for filename, vector in vectors_lemmatized.items():
27         print(f"\n{filename}:")
28         print(f"    Размер вектора: {len(vector)}")
29         print(f"    Ненулевые элементы: {np.sum(vector > 0)}")
30         print(f"    Пример первых 10 значений: {vector[:10]}")
31
32     # без лемматизации
33     print("\n" + "="*50)
34     print("Векторизация без лемматизации (словоформы, TF):")
35     vectors_wordforms =
36         vectorization.create_text_vectors(results, method='tf',
37                                           use_lemmatization=False)
```

```

33     for filename, vector in vectors_wordforms.items():
34         print(f"\n{filename}:")
35         print(f"    Размер вектора: {len(vector)}")
36         print(f"    Ненулевые элементы: {np.sum(vector > 0)}")
37
38 # Кластеризация методом K-means
39 result_kmeans = cluster.cluster_files(vectors_wordforms,
40                                       method='kmeans', n_clusters=10)
41 print(f"Метод: {result_kmeans['method']}")
42 print(f"Количество кластеров:
43       {result_kmeans['n_clusters']}")
44
45 # print(result_kmeans)
46
47 # Нечеткая кластеризация C-means
48 print("Кластеризация методом Fuzzy C-means:")
49 result_fcm = cluster.cluster_files(vectors_wordforms,
50                                     method='fcm', n_clusters=10, m=45)
51 print(f"Метод: {result_fcm['method']}")
52 print(f"Количество кластеров: {result_fcm['n_clusters']}")
53 print(f"Silhouette Score:
54       {result_fcm['silhouette_score']:.3f}")
55
56
57
58
59 # visualizer.visualize_clustering_result(result_kmeans,
60 #                                         plot_type='2d_pca',
61 #                                         show_filenames=True,
62 #                                         filename_limit=None)
63 visualizer.visualize_clustering_result(result_fcm,
64                                         plot_type='2d_pca',
65                                         show_filenames=True,
66                                         filename_limit=None
67 )

```

```
66
67     avg_inter, avg_intra =
68         dist.calculate_cluster_distances(vectors_lemmatized,
69             result_kmeans, metric='euclidean')
70     avg_inter, avg_intra =
71         dist.calculate_cluster_distances(vectors_lemmatized,
72             result_fcm, metric='euclidean')
73
74     print("K-Means: Average Inter Cluster Distance = ", avg_inter)
75     print("K-Means: Average Intra Cluster Distance = ", avg_intra)
76
77     print("Fuzzy C-Means: Average Inter Cluster Distance = ", avg_inter)
78     print("Fuzzy C-Means: Average Intra Cluster Distance = ", avg_intra)
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
```

ПРИЛОЖЕНИЕ Б

Код программы для кластеризации

Листинг Б.1 – Код программы для кластеризации

```
1 import numpy as np
2 from sklearn.cluster import KMeans
3 from sklearn.metrics import silhouette_score
4 from scipy.spatial.distance import cdist
5 import warnings
6 from typing import Dict, Tuple, Any
7
8 class VectorClustering:
9     def __init__(self):
10         self.models = {}
11
12     def k_means_clustering(self, vectors: np.ndarray,
13                           n_clusters: int = 3,
14                           random_state: int = 10, n_init: int
15                           = 10) -> Dict[str, Any]:
16
17         if len(vectors) < n_clusters:
18             n_clusters = max(2, len(vectors) // 2)
19
20         kmeans = KMeans(n_clusters=n_clusters,
21                         random_state=random_state, n_init=n_init)
22         labels = kmeans.fit_predict(vectors)
23
24         if len(np.unique(labels)) > 1:
25             try:
26                 silhouette = silhouette_score(vectors, labels)
27             except:
28                 silhouette = -1
29         else:
30             silhouette = -1
31
32     return {
33         'method': 'K-means',
34         'labels': labels,
35         'centers': kmeans.cluster_centers_,
36         'inertia': kmeans.inertia_,
37         'silhouette_score': silhouette,
```

```

35         'n_clusters': n_clusters
36     }
37
38     def fuzzy_c_means_clustering(self, vectors: np.ndarray,
39                                 n_clusters: int = 3,
40                                         m: float = 2.0, max_iter: int
41                                         = 100,
42                                         error: float = 1e-5,
43                                         random_state: int = 42) ->
44                                         Dict[str, Any]:
45
46         if len(vectors) < n_clusters:
47             n_clusters = max(2, len(vectors) // 2)
48
49         np.random.seed(random_state)
50         n_samples, n_features = vectors.shape
51
52         U = np.random.rand(n_samples, n_clusters)
53         U = U / np.sum(U, axis=1, keepdims=True)
54
55         for iteration in range(max_iter):
56             U_old = U.copy()
57
58             centers = np.zeros((n_clusters, n_features))
59             for j in range(n_clusters):
60                 numerator = np.sum((U[:, j] ** m)[:, ,
61                                         np.newaxis] * vectors, axis=0)
62                 denominator = np.sum(U[:, j] ** m)
63                 centers[j] = numerator / denominator
64
65             distances = cdist(vectors, centers,
66                                metric='euclidean')
67             distances = np.fmax(distances,
68                                 np.finfo(np.float64).eps)
69
70             temp = distances ** (-2/(m-1))
71             U = temp / np.sum(temp, axis=1, keepdims=True)
72
73             if np.linalg.norm(U - U_old) < error:
74                 break
75
76         labels = np.argmax(U, axis=1)

```

```

69
70     if len(np.unique(labels)) > 1:
71         try:
72             silhouette = silhouette_score(vectors, labels)
73         except:
74             silhouette = -1
75     else:
76         silhouette = -1
77
78     partition_coefficient = np.sum(U ** 2) / n_samples
79     entropy_coefficient = -np.sum(U * np.log(U)) /
80             n_samples
81
82     return {
83         'method': 'Fuzzy C-means',
84         'labels': labels,
85         'centers': centers,
86         'membership_matrix': U,
87         'silhouette_score': silhouette,
88         'partition_coefficient': partition_coefficient,
89         'entropy_coefficient': entropy_coefficient,
90         'n_clusters': n_clusters,
91         'n_iterations': iteration + 1
92     }
93
94     def gustafson_kessel_clustering(self, vectors: np.ndarray,
95                                     n_clusters: int = 3,
96                                     m: float = 2.0, max_iter:
97                                         int = 100,
98                                         error: float = 1e-5,
99                                         random_state: int = 42)
100                                         -> Dict[str, Any]:
101
102     if len(vectors) < n_clusters:
103         n_clusters = max(2, len(vectors) // 2)
104
105         np.random.seed(random_state)
106         n_samples, n_features = vectors.shape
107
108         U = np.random.rand(n_samples, n_clusters)
109         U = U / np.sum(U, axis=1, keepdims=True)

```

```

105 cov_matrices = [np.eye(n_features) for _ in
106     range(n_clusters)]
107
108 for iteration in range(max_iter):
109     U_old = U.copy()
110
111     centers = np.zeros((n_clusters, n_features))
112     for j in range(n_clusters):
113         numerator = np.sum((U[:, j] ** m)[:, np.newaxis] * vectors, axis=0)
114         denominator = np.sum(U[:, j] ** m)
115         centers[j] = numerator / denominator
116
117     distances = np.zeros((n_samples, n_clusters))
118
119     for j in range(n_clusters):
120         diff = vectors - centers[j]
121         weighted_diff = (U[:, j] ** m)[:, np.newaxis] * diff
122         F_j = np.dot(weighted_diff.T, diff) /
123             np.sum(U[:, j] ** m)
124
125         F_j = F_j + np.eye(n_features) * 1e-6
126
127         rho = 1.0
128         det_F = np.linalg.det(F_j)
129         if det_F <= 0:
130             det_F = 1e-6
131
132         A_j = (rho * det_F) ** (1/n_features) *
133             np.linalg.inv(F_j)
134
135         diff = vectors - centers[j]
136         distances[:, j] = np.sum(np.dot(diff, A_j) *
137             diff, axis=1)
138
139         cov_matrices[j] = F_j
140
141     distances = np.fmax(distances,
142         np.finfo(np.float64).eps)

```

```

139
140         temp = distances ** (-1/(m-1))
141         U = temp / np.sum(temp, axis=1, keepdims=True)
142
143         if np.linalg.norm(U - U_old) < error:
144             break
145
146     labels = np.argmax(U, axis=1)
147
148     if len(np.unique(labels)) > 1:
149         try:
150             silhouette = silhouette_score(vectors, labels)
151         except:
152             silhouette = -1
153     else:
154         silhouette = -1
155
156     partition_coefficient = np.sum(U ** 2) / n_samples
157     entropy_coefficient = -np.sum(U * np.log(U)) /
158                             n_samples
159
160     return {
161         'method': 'Gustafson-Kessel',
162         'labels': labels,
163         'centers': centers,
164         'covariance_matrices': cov_matrices,
165         'membership_matrix': U,
166         'silhouette_score': silhouette,
167         'partition_coefficient': partition_coefficient,
168         'entropy_coefficient': entropy_coefficient,
169         'n_clusters': n_clusters,
170         'n_iterations': iteration + 1
171     }
172
173     def cluster_vectors(self, vectors_dict: Dict[str,
174                                         np.ndarray],
175                         method: str = 'kmeans', n_clusters: int
176                         = 3,
177                         **kwargs) -> Dict[str, Any]:
178         """
179         Основная функция для кластеризации векторов

```

```

177
178     Args:
179         vectors_dict: Словарь {filename: numpy.array}
180         method: Метод кластеризации ('kmeans', 'fcm', 'gk')
181         n_clusters: Количество кластеров
182         **kwargs: Дополнительные параметры для конкретных
183             методов
184
185     Returns:
186         Словарь с результатами кластеризации и
187             соответсвием файлов кластерам
188
189     """
190     if not vectors_dict:
191         raise ValueError("Словарь векторов не должен быть
192                         пустым")
193
194     filenames = list(vectors_dict.keys())
195     vectors = np.array(list(vectors_dict.values()))
196
197     if vectors.ndim != 2:
198         raise ValueError("Векторы должны быть двумерным
199                         массивом")
200
201     if len(vectors) < 2:
202         raise ValueError("Для кластеризации необходимо как
203                         минимум 2 вектора")
204
205     if n_clusters is None or n_clusters <= 0:
206         n_clusters = min(5, max(2, len(vectors) // 3))
207
208     method = method.lower()
209
210     with warnings.catch_warnings():
211         warnings.simplefilter("ignore")
212
213         if method == 'kmeans' or method == 'k-means':
214             result = self.k_means_clustering(vectors,
215                 n_clusters, **kwargs)
216         elif method == 'fcm' or method == 'fuzzy' or
217             method == 'c-means':
218             result =

```

```

                self.fuzzy_c_means_clustering(vectors ,
11             n_clusters , **kwargs)
12     elif method == 'gk' or method == 'gustafson' or
13         method == 'gustafson-kessel':
14         result =
15             self.gustafson_kessel_clustering(vectors ,
16                 n_clusters , **kwargs)
17     else:
18         raise ValueError(f"Неизвестный метод:
19                         {method}. Доступные методы: 'kmeans',
20                         'fcm', 'gk'")
21
22     file_clusters = {filenames[i]: int(result['labels'][i])
23                      for i in range(len(filenames))}
24
25
26     result['file_clusters'] = file_clusters
27     result['filenames'] = filenames
28     result['vectors'] = vectors
29
30
31     self.models[method] = result
32
33     return result
34
35
36 def find_optimal_clusters(self , vectors_dict: Dict[str ,
37 np.ndarray] ,
38                             method: str = 'kmeans' ,
39                             max_clusters: int = 10 ,
40                             **kwargs) -> Dict[str , Any]:
41
42     vectors = np.array(list(vectors_dict.values()))
43     max_clusters = min(max_clusters , len(vectors) - 1)
44
45
46     best_score = -1
47     best_result = None
48     best_n = 2
49
50
51     for n in range(2 , max_clusters + 1):
52         try:
53             result = self.cluster_vectors(vectors_dict ,
54                 method , n , **kwargs)
55             score = result['silhouette_score']
56
57             if score > best_score:
58                 best_score = score
59                 best_result = result
60                 best_n = n
61
62     return best_result

```

```

242         if score > best_score:
243             best_score = score
244             best_result = result
245             best_n = n
246     except:
247         continue
248
249     if best_result is None:
250         best_result = self.cluster_vectors(vectors_dict,
251                                             method, 2, **kwargs)
251     best_n = 2
252
253     best_result['optimal_n_clusters'] = best_n
254     best_result['best_silhouette_score'] = best_score
255
256     return best_result
257
258
259 def cluster_files(vectors_dict: Dict[str, np.ndarray],
260                    method: str = 'kmeans', n_clusters: int =
261                    None,
262                    find_optimal: bool = False, **kwargs) ->
263     Dict[str, Any]:
264
265     clusterer = VectorClustering()
266
267     if find_optimal and n_clusters is None:
268         result = clusterer.find_optimal_clusters(vectors_dict,
269                                                 method, **kwargs)
270     else:
271         if n_clusters is None:
272             n_clusters = min(5, max(2, len(vectors_dict) // 3))
273         result = clusterer.cluster_vectors(vectors_dict,
274                                             method, n_clusters, **kwargs)
275
276     return result
277
278
279
280 if __name__ == "__main__":
281     np.random.seed(42)
282     n_files = 20

```

```

278     vector_dim = 10
279
280     vectors_dict = {
281         f'file_{i}.txt': np.random.randn(vector_dim) for i in
282             range(n_files)
283     }
284
285     for i in range(n_files):
286         if i < 7:
287             vectors_dict[f'file_{i}.txt'] += np.array([2, 2,
288                 2] + [0] * (vector_dim - 3))
289         elif i < 14:
290             vectors_dict[f'file_{i}.txt'] += np.array([-2, -2,
291                 -2] + [0] * (vector_dim - 3))
292
293     print("Кластеризация методом K-means:")
294     result_kmeans = cluster_files(vectors_dict,
295         method='kmeans', n_clusters=3)
296     print(f"Метод: {result_kmeans['method']}")
297     print(f"Количество кластеров:
298           {result_kmeans['n_clusters']}")")
299     print(f"Silhouette Score:
300           {result_kmeans['silhouette_score']:.3f}")")
301     print(f"Метки кластеров: {result_kmeans['labels']}")")
302     print()
303
304     print("Кластеризация методом Fuzzy C-means:")
305     result_fcm = cluster_files(vectors_dict, method='fcm',
306         n_clusters=3, m=2.0)
307     print(f"Метод: {result_fcm['method']}")
308     print(f"Количество кластеров: {result_fcm['n_clusters']}")")
309     print(f"Silhouette Score:
310           {result_fcm['silhouette_score']:.3f}")")
311     print(f"Коэффициент разделения:
312           {result_fcm['partition_coefficient']:.3f}")")
313     print()
314
315     print("Распределение файлов по кластерам (K-means):")
316     for filename, cluster in
317         result_kmeans['file_clusters'].items():
318             print(f"  {filename}: кластер {cluster}")

```

ПРИЛОЖЕНИЕ В

Код программы для Гат-Гевы

Листинг В.1 – Код программы для Гат-Гевы

```
1 clear; close all; clc;

2

3 filename = 'vectors_wordforms.json'; % файл по умолчанию

4

5 json_text = fileread(filename);
6 data_struct = jsondecode(json_text);

7

8 doc_names = fieldnames(data_struct);
9 num_docs = length(doc_names);

10

11 fprintf('Загружено %d документов\n', num_docs);

12

13 vectors = [];
14 for i = 1:num_docs
15     doc_name = doc_names{i};
16     doc_vector = data_struct.(doc_name);

17
18     if i == 1
19         vector_length = length(doc_vector);
20         vectors = zeros(num_docs, vector_length);
21     end

22
23     vectors(i, :) = doc_vector;
24 end
25 fprintf('Размерность векторов: %d\n', vector_length);

26

27 k = input('Введите число кластеров: ');
28 if isempty(k) || k < 2 || k > num_docs
29     k = min(3, floor(num_docs/2));
30     fprintf('Используем %d кластеров по умолчанию\n', k);
31 end

32

33 % Определение количества компонент (например, сохраняем 95%
34 % дисперсии)
35 desiredVariance = 0.10;
36 [coeff, score, latent, ~, explained] = pca(vectors);
```

```

37 % Находим количество компонент , объясняющих desiredVariance
38 % дисперсии
39 cumulativeVariance = cumsum(explained);
40 kComponents = find(cumulativeVariance >= desiredVariance *
41 100, 1);
42
43 fprintf('Количество главных компонент для сохранения %.1f %%\n'
44 % дисперсии: %d\n', ...
45 desiredVariance * 100, kComponents);
46
47 % Альтернативно: фиксированное количество компонент
48 % kComponents = min(50, size(vectors, 2)); % например ,
49 % максимум 50 компонент
50
51 % Применяем PCA
52 vectors_pca = score(:, 1:kComponents);
53
54 fprintf('Размерность данных до PCA: %d\n', size(vectors, 2));
55 fprintf('Размерность данных после PCA: %d\n', kComponents);
56
57 % 2. Выполняем FCM кластеризацию на данных после PCA
58 fprintf('\nВыполняем FCM кластеризацию на данных после
59 PCA...\n');
60
61 options = fcmOptions(...%
62 NumClusters=k, ...
63 DistanceMetric="fmle", ...
64 Exponent=2, ...
65 Verbose=true);
66
67 [centers_pca, U, ~] = fcm(vectors_pca, options);
68
69 % Присваиваем документы к кластерам
70 [~, cluster_idx] = max(U);
71
72 % % 3. Выполняем FCM кластеризацию
73 % fprintf('\nВыполняем FCM кластеризацию...\n');
74 % options = fcmOptions(NumClusters=k, DistanceMetric =
75 % "euclidean", Exponent = 10, Verbose = true);
76 % [centers, U, ~] = fcm(vectors, options);
77
78

```

```

72
73
74 fprintf(' \n==== РЕЗУЛЬТАТЫ КЛАСТЕРИЗАЦИИ ===\n');
75 fprintf(' Всего кластеров: %d\n', k);
76 fprintf(' Всего документов: %d\n\n', num_docs);
77
78 for i = 1:k
79     cluster_size = sum(cluster_idx == i);
80     fprintf(' Кластер %d: %d документов (%.1f%%)\n', ...
81             i, cluster_size, cluster_size/num_docs*100);
82 end
83
84 fprintf(' \nДокументы по кластерам:\n');
85 for i = 1:k
86     fprintf(' \n--- Кластер %d ---\n', i);
87     idx = find(cluster_idx == i);
88
89     num_to_show = min(20, length(idx));
90     for j = 1:num_to_show
91         fprintf('%s\n', doc_names{idx(j)} );
92     end
93
94     if length(idx) > num_to_show
95         fprintf(' ... и еще %d документов\n', length(idx) -
96                 num_to_show);
97     end
98 end
99 if vector_length == 2
100    figure(1);
101    colors = lines(k);
102    for i = 1:k
103        idx = find(cluster_idx == i);
104        scatter(vectors(idx, ...
105                      1), vectors(idx,2), 50, colors(i,:), 'filled');
106        hold on;
107    end
108    plot(centers(:,1), centers(:,2), 'kx', 'MarkerSize', 15,
109          'LineWidth', 2);
110    xlabel('Признак 1'); ylabel('Признак 2');
111    title(sprintf('Кластеризация документов (k=%d)', k));

```

```

111     grid on;
112     hold off;
113
114 elseif vector_length == 3
115     figure(1);
116     colors = lines(k);
117     for i = 1:k
118         idx = find(cluster_idx == i);
119         scatter3(vectors(idx,1), vectors(idx,2),
120                  vectors(idx,3), ...
121                  50, colors(i,:), 'filled');
122         hold on;
123     end
124     plot3(centers(:,1), centers(:,2), centers(:,3), ...
125            'kx', 'MarkerSize', 15, 'LineWidth', 2);
126     xlabel('Признак 1'); ylabel('Признак 2'); zlabel('Признак
127            3');
128     title(sprintf('Кластеризация документов (k=%d)', k));
129     grid on;
130     hold off;
131
132 else
133     fprintf('\nДля визуализации применяем РСА (данные
134             %d-мерные)\n', vector_length);
135     [~, score] = pca(vectors);
136
137     figure(1);
138     colors = lines(k);
139     for i = 1:k
140         idx = find(cluster_idx == i);
141         scatter(score(idx,1), score(idx,2), 50, colors(i,:),
142                 'filled');
143         hold on;
144     end
145     xlabel('Первая главная компонента');
146     ylabel('Вторая главная компонента');
147     % title(sprintf('Кластеризация (РСА проекция, k=%d)', k));
148     grid on;
149     hold off;
150 end

```

```

148 results_table = table();
149 results_table.Document = doc_names;
150 results_table.Cluster = cluster_idx';
151
152 for i = 1:k
153     results_table.(sprintf('Prob_Cluster_%d', i)) = U(i,:)';
154 end
155
156 % writetable(results_table, 'clustering_results.csv');
157
158 fid = fopen('clustering_report.txt', 'w');
159 fprintf(fid, 'Отчет по кластеризации документов\n');
160 fprintf(fid, '=====\\n');
161 fprintf(fid, 'Дата: %s\\n', datestr(now));
162 fprintf(fid, 'Файл данных: %s\\n', filename);
163 fprintf(fid, 'Число документов: %d\\n', num_docs);
164 fprintf(fid, 'Число кластеров: %d\\n\\n', k);
165
166 for i = 1:k
167     cluster_size = sum(cluster_idx == i);
168     fprintf(fid, 'Кластер %d (%d документов, %.1f%%):\n', ...
169             i, cluster_size, cluster_size/num_docs*100);
170
171     idx = find(cluster_idx == i);
172     for j = 1:min(10, length(idx))
173         fprintf(fid, ' - %s\\n', doc_names{idx(j)} );
174     end
175     if length(idx) > 10
176         fprintf(fid, ' ... и еще %d документов\\n',
177                 length(idx)-10);
178     end
179     fprintf(fid, '\\n');
180 end
180 fclose(fid);

```