**ОТЧЁТ ПО УЧЕБНОЙ ПРАКТИКЕ**

«Практика по получению профессиональных умений и опыта профессиональной деятельности»

приказ Университета о направлении на практику от «09 » февраля 20 23 г. № 733-C

Отчёт представлен к рассмотрению:

Студент группы ККСО­01­20 «14» июня 20 23 г. / Семин В.В.

(подпись и расшифровка подписи)

Отчёт утверждён.

Допущен к защите:

Руководитель практики

от кафедры «14» июня 20 23 г. / Маматов Д.Э.

(подпись и расшифровка подписи)

Москва 2023

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ**

«Разработка системы кластеризация файлов с применением статистических методов»

**Студенту** 3 **курса учебной группы** ККСО­01­20

Семину Владиславу Владимировичу

(фамилия, имя и отчество)

**Место и время проведения практики:** базовая кафедра №252 «Информационная безопасность»

1. **СОДЕРЖАНИЕ ПРАКТИКИ:**
   1. Изучить: методы сбора статистических данных с файлов разных типов
   2. Практически выполнить: систему кластеризации файлов статистическими методами
   3. Ознакомиться: с методами сбора статистических данных с файлов разных типов

**2. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ**:

1. **ОРГАНИЗАЦИОННО­МЕТОДИЧЕСКИЕ УКАЗАНИЯ:**

Руководитель практики от кафедры

«11» февраля 20 23 г.

(подпись)

( Маматов Д.Э. )

(ФИО)

Задание получил:

«11 » февраля 20 23 г.

(подпись)

( Семин В.В. )

(ФИО)

**СОГЛАСОВАНО:**

Заведующий БК № 252:

«11 » февраля 20 23 г.

(подпись)

(Корольков А.В.)

(ФИО)

( )

(Расшифровка, должность)



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

**РАБОЧИЙ ГРАФИК ПРОВЕДЕНИЯ УЧЕБНОЙ ПРАКТИКИ**

студента 3 курса группы ККСО­01­20 очной формы обучения,

обучающегося по направлению подготовки «Компьютерная безопасность», профиль «Анализ безопасности компьютерных систем»

|  |  |  |  |
| --- | --- | --- | --- |
| **Неделя** | **Сроки**  **выполнения** | **Этап** | **Отметка о выполнении** |
| 1-9 | 9 недель | Изучение методов сбора статистических данных |  |
| 10-11 | 1 неделя | Изучение методов оценки результатов кластеризации |  |
| 12-14 | 2 недели | Написание программы |  |
| 15 | 1 неделя | Оценка результатов |  |
| 16 | 1 неделя | Написание и предоставление отсчёта |  |

Руководитель практики от кафедры

(подпись)

Маматов Д.Э.

(ФИО, ученая степень, ученое звание)

Обучающийся

(подпись)

Семин В.В.

(ФИО, ученая степень, ученое звание)

**Согласовано:**

Заведующий БК №252

(подпись)

Корольков А.В., с.н.с.

(ФИО, ученая степень, ученое звание)

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 6](#_Toc137007455)

[1 ТЕКСТОВЫЕ ФАЙЛЫ 8](#_Toc137007456)

[1.1 ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ТЕКСТА 8](#_Toc137007457)

[1.1.1 Токенизация текста 8](#_Toc137007458)

[1.1.2 Стоп-слова 9](#_Toc137007459)

[1.1.3 Лемматизация 10](#_Toc137007460)

[1.2 TF-IDF 10](#_Toc137007461)

[2 ФАЙЛЫ ИЗОБРАЖЕНИЙ 12](#_Toc137007462)

[2.1 ЦВЕТОВАЯ ГИСТОГРАММА 12](#_Toc137007463)

[2.2 ТЕКСТУРНЫЕ ХАРАКТЕРИСТИКИ 13](#_Toc137007464)

[2.3 ОПРЕДЕЛЕНИЕ КОНТУРОВ ОБЪЕКТОВ НА ИЗОБРАЖЕНИИ 14](#_Toc137007465)

[2.3.1 Сглаживание 14](#_Toc137007466)

[2.3.2 Поиск градиентов 14](#_Toc137007467)

[2.3.3 Подавление не-максимумов 15](#_Toc137007468)

[2.3.4 Двойная пороговая фильтрация 15](#_Toc137007469)

[2.3.5 Трассировка области неоднозначности 15](#_Toc137007470)

[2.4 ГИСТОГРАММА НАПРАВЛЕННЫХ ГРАДИЕНТОВ 15](#_Toc137007471)

[2.4.1 Предварительная обработка 16](#_Toc137007472)

[2.4.2 Поиск градиентов 16](#_Toc137007473)

[2.4.3 Вычислить гистограмму градиентов в ячейках 8 × 8 16](#_Toc137007474)

[2.4.4 Нормализация блока 16 × 16 17](#_Toc137007475)

[2.4.5 Получение HOG 17](#_Toc137007476)

[3 АУДИО ФАЙЛЫ 18](#_Toc137007477)

[3.1 МЕЛ-КЕПСТРАЛЬНЫЕ КОЭФФИЦИЕНТЫ (MFCC) 18](#_Toc137007478)

[3.2 ХРОМАТИЧЕСКИЕ ПРИЗНАКИ 20](#_Toc137007479)

[3.3 ТЕМП И РИТМ 21](#_Toc137007480)

[3.4 СПЕКТРАЛЬНЫЙ КОНТРАСТ 21](#_Toc137007481)

[3.5 ZERO-CROSSING RATE 21](#_Toc137007482)

[3.6 ДРУГИЕ ХАРАКТЕРИСТЕКИ 22](#_Toc137007483)

[4 ОЦЕНКА КАЧЕСТВА КЛАСТЕРИЗАЦИИ 23](#_Toc137007484)

[4.1 МЕТРИКИ И ИХ СВОЙСТВА 23](#_Toc137007485)

[4.2 МЕТРИКИ, ОСНОВАННЫЕ НА ПОДСЧЁТЕ ПАР (КЛАССИФИКАЦИОННЫЕ) 23](#_Toc137007486)

[4.3 МЕТРИКИ, ОСНОВАННЫЕ НА СОПОСТАВЛЕНИИ МНОЖЕСТВ 24](#_Toc137007487)

[5 РАЗРАБОТКА ПРОГРАММЫ И ОЦЕНКА РЕЗУЛЬТАТОВ 25](#_Toc137007488)

[ЗАКЛЮЧЕНИЕ 29](#_Toc137007489)

[ЛИТЕРАТУРА 30](#_Toc137007490)

[ПРИЛОЖЕНИЕ А 33](#_Toc137007491)

# ВВЕДЕНИЕ

В первой части данной работы был рассмотрен такой статистический метод, как кластер-анализ с точки зрения возможности применения его к задаче о кластеризации данных различных типов. Результатом стало ознакомление с алгоритмами кластерами (k-means, means-shift, DBSCAN), их параметрами, характеристиками, эффективностью, ограничениями в применении. Была дана теоретическая оценка пригодности данных алгоритмов к поставленной задаче.

В этой части работы полученные данные будут использованы с практической точки зрения – для разработки системы кластеризация файлов с применением статистических методов.

Для разработки данной системы кроме уже проведённого изучения алгоритмов кластеризации также понадобится изучить методы сбора статистических данных из файлов различных типов. В этой работе мы ограничимся тремя типами файлов: текстовыми, изображениями и аудио. Возможность кластеризации данных типов файлов является достаточной, так как задачу о кластеризации большинства других видов файлов можно свети к задаче о кластеризации трёх обозначенных.

Например, для кластеризации видеофайла можно будет разделить его на отдельные кадры, извлечь аудиодорожку и субтитры (если имеются) и обработать их методами кластеризации изображений, аудио и текста соответственно. Кластеризацию исполняемых файлов можно провести с помощью извлечения из них строковых параметров, меток функций, информацию об используемых библиотеках и тому подобное, и, после этого, использовать на этой информации методы кластеризации текста.

**Объект исследования:** методы кластеризации данных различных типов.

**Предмет исследования:** система кластеризации данных различных типов.

**Цель:** разработать систему кластеризации данных различных типов.

**Задачи исследования:**

1. Изучение методов сбора статистических данных текстовых файлов
2. Изучение методов сбора статистических данных файлов изображения
3. Изучение методов сбора статистических данных аудио файлов
4. Изучение методов оценки результатов кластеризации
5. Разработка программы, которая будет определять тип файла, применять соответствующие методы сбора статистических данных на нём, и, после этого, использовать полученную информацию для кластеризации данных. Далее программа должна оценивать результаты кластеризации.

**Гипотеза исследования:** DBSCAN является наилучшим методом кластеризации файлов различных типов из рассмотренных.

**Методы исследования:** изучение отечественной и зарубежной литературы по теме, сравнение рассмотренных алгоритмов кластеризации, анализ полученных результатов.

# 1 ТЕКСТОВЫЕ ФАЙЛЫ

## ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ТЕКСТА

### Токенизация текста

Токенизация — это процесс разбиения текста на маленькие единицы, называемые токенами. Токеном может быть слово, часть слова или просто символы, такие как пунктуация.

На первый взгляд это может показаться довольно глупым. У нас есть некоторый текст, и мы хотим, чтобы компьютер работал с ним, так зачем нам разбивать текст на маленькие токены?

Языки программирования работают, разбивая исходный код на токены, а затем объединяя их с помощью некоторой логики (грамматики программы) в обработке естественного языка.

Разбивая текст на маленькие, известные фрагменты, мы можем применить некоторый набор правил для их комбинирования и создания более крупного смысла. В языках программирования токены связываются с помощью формальных грамматик.

Хотя в обработке естественного языка развились различные способы комбинирования токенов на протяжении многих лет, вместе с разнообразием методов токенизации. Однако мотивация токенизации осталась прежней - представить компьютеру некоторый конечный набор символов, которые можно объединить для получения желаемого результата [1].

Одной из больших проблем является возможность разбиения слов, когда пробелы или знаки пунктуации не определяют границы слова. Это особенно распространено для символьных языков, таких как китайский, японский, корейский и тайский.

Еще одной проблемой являются символы, которые значительно изменяют значение слова. Мы интуитивно понимаем, что знак "$" с прикрепленным к нему числом ($100) означает нечто отличное от самого числа (100). Знаки пунктуации, особенно в менее распространенных ситуациях, могут вызвать проблемы для компьютеров, пытающихся изолировать их значение как часть строки данных.

Сокращения, такие как "you're" и "I'm", также должны быть правильно разделены на соответствующие части. Неправильное токенизирование каждой части предложения может привести к непониманию на следующих этапах обработки естественного языка (NLP) [2].

### Стоп-слова

Стоп-слова представляют собой набор общеупотребительных слов в любом языке.

Причина, по которой стоп-слова критически важны для многих приложений, заключается в том, что, удаляя слова, которые часто используются в данном языке, мы можем сосредоточиться на важных словах. Например, в контексте поисковой системы, если ваш запрос для поиска звучит как "как разрабатывать приложения для информационного поиска", то если поисковая система попытается найти веб-страницы, содержащие термины "как", "разрабатывать", "приложения", "информационный", "поиск", то она найдет намного больше страниц, содержащих термины "как", "разрабатывать", чем страниц, содержащих информацию о разработке приложений для информационного поиска, потому что термины "как" и "разрабатывать" настолько общеупотребительны. Если мы проигнорируем эти два термина, поисковая система сможет фокусироваться на поиске страниц, содержащих ключевые слова: "разработка", "информационный", "поиск", "приложения" - что позволит вывести страницы, действительно представляющие интерес. Это лишь базовое понимание использования стоп-слов.

Примеры списков стоп-слов, которые можно использовать:

1. Определители (Determiners) - Определители обычно указывают на существительные, за которыми обычно следует определитель. Примеры: the, a, an, another.
2. Союзы-соединители (Coordinating conjunctions) - Союзы-соединители связывают слова, фразы и предложения. Примеры: for, an, nor, but, or, yet, so.
3. Предлоги (Prepositions) - Предлоги выражают временные или пространственные отношения. Примеры: in, under, towards, before [3].

### 1.1.3 Лемматизация

Лемматизация - это процесс объединения различных словоформ одного и того же слова. Цель лемматизации состоит в том, чтобы свести слово к его основной форме, также называемой леммой. Например, глагол "running" будет определен как "run".

Лемматизация берет слово и разбивает его на его лемму. Например, глагол "walk" может появляться в формах "walking", "walks" или "walked". Окончания, такие как "s", "ed" и "ing", удаляются. Лемматизация группирует эти слова в качестве своей леммы, "walk".

Слово "saw" может иметь различное толкование в зависимости от контекста предложения. Например, "saw" может быть разбито на леммы "see" или "saw". В таких случаях лемматизация пытается выбрать правильную лемму в зависимости от контекста слова, окружающих слов и предложения. Другие слова, такие как "better", могут быть разложены на лемму, такую как "good".

Простой способ выполнить лемматизацию - использовать алгоритм на основе поиска в словаре. Для этого процесса требуется подробный словарь, чтобы алгоритм мог найти конкретное слово и связать его с леммой слова. Более сложные словоформы или языки могут требовать использования системы лемматизации на основе правил [4].

## TF-IDF

TF-IDF (Term frequency–inverse document frequency) — статистический показатель, применяемый для оценки важности слова в контексте категории, документа или коллекции документов. Используется при анализе текстовых данных.

Как правило, TF-IDF определяется для каждого слова. Чем выше значение данного показателя, тем значимее слово в контексте категории, документа, коллекции. При этом данный показатель также позволяет учесть и широко употребляемые слова, понизив их значимость в контексте объекта для анализа.

Формула для определения показателя имеет вид (1.1).

|  |  |
| --- | --- |
|  | (1.1) |

Здесь — частота слова в конкретной категории/документе/коллекции (в зависимости от того, какие данные анализируются), — обратная частота документа (популярность слова).

Частота слова в категории определяется по формуле (1.2).

|  |  |
| --- | --- |
|  | (1.2) |

Здесь - количество отдельных слов в категории/документе/коллекции, - общее количество всех слов в категории/документе/коллекции.

Обратная частота документа (также часто называют инверсией частоты) определяется по формуле (1.3).

|  |  |
| --- | --- |
|  | (1.3) |

Здесь — количество категорий/документов/коллекций всего, - количество категорий/документов/коллекций, в которых содержится интересующее слово.

При анализе текстовых данных метрику TF-IDF лучше всего рассчитывать после проведения процессов токенизации, а также лемматизации или стемминга [5].

# ФАЙЛЫ ИЗОБРАЖЕНИЙ

## ЦВЕТОВАЯ ГИСТОГРАММА

Мы можем определить гистограмму изображения как двумерную столбчатую диаграмму. Горизонтальная ось представляет интенсивность пикселей, а вертикальная ось обозначает частоту каждой интенсивности.

Уточним, что мы будем считать изображение матрицей или матрицами пикселей. В случае изображения в оттенках серого эта матрица будет состоять из чисел от 0 до 255. Для изображений RGB у нас будет три матрицы, по одной для каждого цветового канала.

Цветовые гистограммы оказываются полезны в анализе изображений. Например, мы можем использовать гистограммы для определения порога сегментации изображения для выделения объекта на фоне.

Например, если мы хотим выделить розу на изображении (рис. 2.1 а) от фона, мы можем начать с анализа его гистограммы (рис. 2.1 б). Таким образом, мы можем видеть, что большинство пикселей фона белые или белесоватые. Это означает, что большинство пикселей фона близки к значению 255 [6].

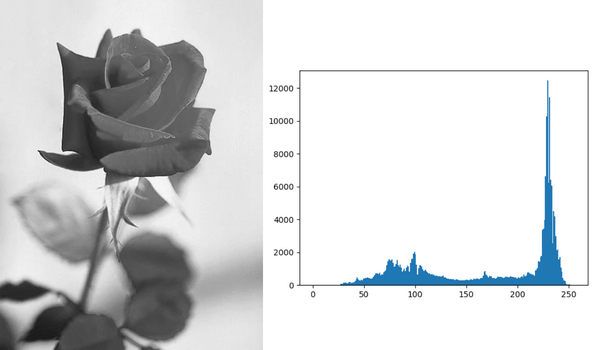


Рис 2.1 – Изображение (а) и его цветовая гистограмма (б)

## ТЕКСТУРНЫЕ ХАРАКТЕРИСТИКИ

Для определения текстуры (естественных рисунков на материале) можно использовать Gray-Level Co-Occurrence Matrix или GLCM.

Для расчёта GLCM мы рассматриваем два пикселя одновременно - опорный и соседний пиксели. Перед расчетом GLCM мы определяем определенное пространственное отношение между опорным и соседним пикселями. Например, соседом может быть пиксель, находящийся справа от текущего пикселя, или это может быть пиксель, находящийся выше на 3 пикселя, или пиксель, находящийся на расстоянии 2 пикселей по диагонали от опорного пикселя.

После определения пространственного отношения мы создаем матрицу GLCM заданного размера (диапазон интенсивностей x диапазон интенсивностей). Например, для 8-битного изображения с одним каналом у нас будет матрица GLCM размером 256x256. Затем мы проходим по изображению и для каждой пары интенсивностей, найденных для заданного пространственного отношения, увеличиваем значение соответствующей ячейки матрицы [7].

Используя GLCM, можно получить следующие параметры изображения [8]:

1. Контраст (contrast) — это мера разности интенсивности между пикселем и его соседом по всему изображению. Для "постоянного" изображения (без изменений) контраст равен нулю. Контраст вычисляется по формуле (2.1).

|  |  |
| --- | --- |
|  | (2.1) |

1. Местная однородность (homogeneity) измеряет близость распределения элементов в GLCM к его диагонали. Для диагонального GLCM однородность равна 1.

|  |  |
| --- | --- |
|  | (2.2) |

1. Корреляция GLCM также может быть полезной для анализа изображений, так как она показывает наличие или отсутствие взаимосвязи в оттенках серого между пикселями изображения.
2. Диссимиларность (dissimilarity) является мерой расстояния между парами объектов (пикселей).

|  |  |
| --- | --- |
|  | (2.3) |

1. Энергия (Energy) представляет сумму квадратов элементов в GLCM. Ее значения находятся в диапазоне от 0 до 1.

|  |  |
| --- | --- |
|  | (2.4) |

## 2.3 ОПРЕДЕЛЕНИЕ КОНТУРОВ ОБЪЕКТОВ НА ИЗОБРАЖЕНИИ

Для определения контуров объектов на изображении можно воспользоваться алгоритмом, разработанным Джоном Канни (John F. Canny) в 1986 г.

Алгоритм состоит из пяти отдельных шагов.

### 2.3.1 Сглаживание

Для подавления шума пользуются размытием изображения фильтром Гаусса.

### 2.3.2 Поиск градиентов

Оператор Собеля часто применяют в алгоритмах выделения границ. По сути, это дискретный дифференциальный оператор, вычисляющий приближенное значение градиента яркости изображения. Результатом применения оператора Собеля в каждой точке изображения является либо вектор градиента яркости в этой точке, либо его норма. Оператор Собеля основан на свёртке изображения небольшими целочисленными фильтрами в вертикальном и горизонтальном направлениях, поэтому его относительно легко вычислять. С другой стороны, используемая им аппроксимация градиента достаточно грубая, особенно это сказывается на высокочастотных колебаниях изображения.

### 2.3.3 Подавление не-максимумов

Пикселями границ объявляются пиксели, в которых достигается локальный максимум градиента в направлении вектора градиента. Значение направления должно быть кратно 45°.

### 2.3.4 Двойная пороговая фильтрация

Следующий шаг — применение порога, чтобы определить находится или нет граница в данной точке изображения. Чем меньше порог, тем больше границ будет находиться, но тем более восприимчивым к шуму станет результат, выделяя лишние данные изображения. Наоборот, высокий порог может проигнорировать слабые края или получить границу фрагментами.

Выделение границ Канни использует два порога фильтрации: если значение пикселя выше верхней границы – он принимает максимальное значение (граница считается достоверной), если ниже – пиксель подавляется, точки со значением, попадающим в диапазон между порогов, принимают фиксированное среднее значение (они будут уточнены на следующем этапе).

### 2.3.5 Трассировка области неоднозначности

Задача сводится к выделению групп пикселей, получивших на предыдущем этапе промежуточное значение, и отнесению их к границе (если они соединены с одной из установленных границ) или их подавлению (в противном случае). Пиксель добавляется к группе, если он соприкасается с ней по одному из 8-ми направлений [9].

## ГИСТОГРАММА НАПРАВЛЕННЫХ ГРАДИЕНТОВ

Дескриптор признаков — это представление изображения или фрагмента изображения, которое упрощает изображение, извлекая полезную и отбрасывая бесполезную информацию.

Обычно дескриптор признака преобразует изображение размером ширина x высота x 3 (каналы) в вектор/массив признаков длиной n. В случае дескриптора признака HOG входное изображение имеет размер 64×128×3, а выходной вектор признака имеет длину 3 780.

В дескрипторе признаков HOG в качестве признаков используются распределения (гистограммы) направлений градиентов (ориентированные градиенты). Градиенты (производные по y) изображения полезны, потому что величина градиентов велика по краям и углам (области резкого изменения интенсивности), и края и углы содержат гораздо больше информации о форме объекта, чем плоские области.

Вычисление HOG состоит из нескольких шагов.

### Предварительная обработка

Как упоминалось ранее, дескриптор функции HOG рассчитывается на участке изображения размером 64×128 пикселей. Конечно, изображение может быть любого размера. Единственное ограничение заключается в том, что анализируемые фрагменты имеют фиксированное соотношение сторон 1:2. Фрагмент вырезается из изображения и изменяется до 64×128. Теперь мы готовы вычислить дескриптор HOG для этого фрагмента изображения.

### Поиск градиентов

В выбранной части изображения ищутся градиенты с помощью оператора Собеля. О том, как это происходит, было написано в пункте 2.3.2.

### 2.4.3 Вычислить гистограмму градиентов в ячейках 8 × 8

На этом этапе изображение делится на ячейки 8 × 8 и гистограмма градиентов вычисляется для каждой ячейки 8 × 8.

Сначала вычисляют направления и модули градиентов каждого пикселя. Углы составляют от 0 до 180 градусов вместо 0 - 360 (беззнаковые градиенты). Следующим шагом будет создание гистограммы градиентов. Гистограмма содержит 9 интервалов, соответствующих углам 0, 20, 40 … 160. Если угол больше 160 градусов, он вносит вклад и в первую ячейку, и в последнюю.

### 2.4.4 Нормализация блока 16 × 16

В идеале, мы хотим, чтобы наш дескриптор не зависел от вариаций освещения. Другими словами, мы хотели бы «нормализовать» гистограммы, чтобы на них не влияли изменения освещения.

Блок 16×16 содержит 4 гистограммы 9×1. Мы можем объединить их в один вектор 36×1. После этого мы нормализуем его (делим каждую координату на модуль вектора),

### 2.4.5 Получение HOG

На финальном шаге все вектора 36×1 объединяются в один гигантский вектор для получения дескриптора HOG данного фрагмента [9].

# АУДИО ФАЙЛЫ

## 3.1 МЕЛ-КЕПСТРАЛЬНЫЕ КОЭФФИЦИЕНТЫ (MFCC)

Для того, чтобы понять, что такое MFCC разберём по-отдельности каждое слово в данном термине.

Мел – единица высоты звука, основанная на восприятии этого звука нашими органами слуха. Как известно, АЧХ человеческого уха даже отдаленно не напоминает прямую, и амплитуда – не совсем точная мера громкости звука. Поэтому, и ввели эмпирически подобранные единицы громкости, например, фон. Аналогично, воспринимаемая человеческим слухом высота звука не совсем линейно зависит от его частоты. Зависимость на рис 3.1 не претендует на большую точность, но зато описывается простой формулой (3.2).

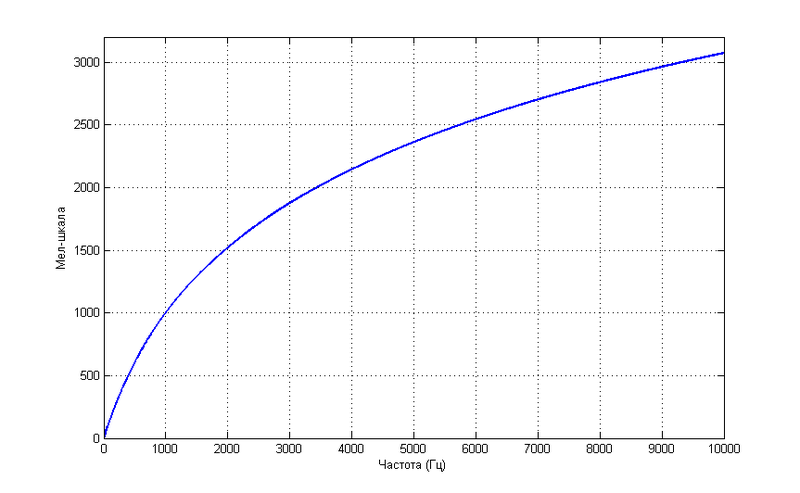


Рис. 3.1. Мел-частотная зависимость

|  |  |
| --- | --- |
|  | (3.1) |

В соответствии с теорией речеобразования речь представляет собой акустическую волну, которая излучается системой органов: легкими, бронхами и трахеей, а затем преобразуется в голосовом тракте. Если предположить, что источники возбуждения и форма голосового тракта относительно независимы, речевой аппарат человека можно представить в виде совокупности генераторов тоновых сигналов и шумов, а также фильтров. Схематичное изображение данной системы представлено на рис 3.2.

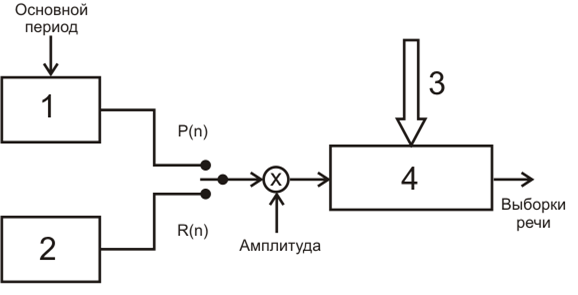


Рис 3.2. Модель речевого аппарата человека.

Здесь:

1. Генератор импульсной последовательности (тонов)

2. Генератор случайных чисел (шумов)

3. Коэффициенты цифрового фильтра (параметры голосового тракта)

4. Нестационарный цифровой фильтр

Сигнал на выходе фильтра (4) можно представить в виде свертки (3.2).

|  |  |
| --- | --- |
|  | (3.2) |

— изначальный вид акустической волны, а — характеристика фильтра (зависит от параметров голосового тракта).

В частотной области это выглядит так:

|  |  |
| --- | --- |
|  | (3.3) |

Произведение можно прологарифмировать, чтобы получить вместо него сумму:

|  |  |
| --- | --- |
|  | (3.3) |

Теперь нам нужно преобразовать эту сумму так, чтобы получить непересекающиеся наборы характеристик исходного сигнала и фильтра. Для этого есть несколько вариантов, например обратное преобразование Фурье даст нам:

|  |  |
| --- | --- |
|  | (3.3) |

Полученная величина и называется кепстр.

Для получения MFCC мы из изначального сигнала извлекаем спектр с помощью преобразования Фурье. Далее мы располагаем полученный спектр на мел-шкале. Для этого мы используем окна, равномерно расположенные на мел-оси. Простым перемножением векторов спектра сигнала и оконной функции найдем энергию сигнала, которая попадает в каждое из окон анализа. Нам осталось только получить из них кепстральные, или «спектр спектра». Далее ещё раз применяем преобразование Фурье, но в этот раз дискретное косинусное преобразование [11].

## 3.2 ХРОМАТИЧЕСКИЕ ПРИЗНАКИ

Анализ хроматических признаков (Chroma Feature) представляют интересную и мощную методику представления аудиофайлов музыки, в которой весь спектр звука проецируется на 12 диапазонов, представляющих 12 различных полутонов (или хроматических нот) в музыкальном октаве. Поскольку в музыке ноты, расположенные на расстоянии одной октавы друг от друга, воспринимаются как особенно схожие, знание распределения хромы, даже без абсолютной частоты (т.е. оригинальной октавы), может дать полезную музыкальную информацию о звуке, и даже может раскрыть воспринимаемую музыкальную схожесть, которая не является очевидной в оригинальном спектре [12].

Существует множество способов вычисления хроматических признаков. Например, преобразование звукозаписи в хроматическое представление (или хромаграмму) может быть выполнено с использованием кратковременного преобразования Фурье в сочетании с стратегиями биннинга или с помощью подходящих мультирежимных фильтровых банков. Кроме того, свойства хроматических признаков могут быть изменены путем введения подходящих пост- и предварительных обработок, изменяющих спектральные, временные и динамические аспекты [13].

## 3.3 ТЕМП И РИТМ

В общем случае алгоритмы определения темпа и ритма состоят из двух этапов: первый этап, который генерирует управляющую функцию путем прямой обработки аудиосигнала, и второй этап, который обнаруживает периодичности в этой управляющей функции для получения оценок темпа и/или такты [14].

## 3.4 СПЕКТРАЛЬНЫЙ КОНТРАСТ

Спектральный контраст определяется как разница в децибелах между пиками и впадинами в спектре [15].

Вычислить спектральный контраст можно следующим способом. Каждый кадр спектрограммы разделяется на поддиапазоны. Для каждого поддиапазона контраст энергии оценивается путем сравнения средней энергии в верхнем квантиле (пиковая энергия) с энергией в нижнем квантиле (долинная энергия). Высокие значения контраста обычно соответствуют четким сигналам с узким полосовым шумом, тогда как низкие значения контраста соответствуют широкополосному шуму [16].

## 3.5 ZERO-CROSSING RATE

Переход через ноль (Zero-Crossing) - это мгновенная точка, в которой знак математической функции меняется (например, с положительного на отрицательный). Он представлен пересечением оси (нулевым значением) на графике функции.

Скорость переходов через ноль (Zero-Crossing Rate, ZCR) - это скорость, с которой сигнал переходит от положительного к нулевому, затем к отрицательному или отрицательного к нулевому, затем к положительному. Значение ZCR широко используется как в распознавании речи, так и в извлечении информации о музыке для классификации. ZCR вычисляется по формуле (3.4) [17].

|  |  |
| --- | --- |
|  | (3.4) |

Здесь T – период спектра, – спектр.

## 3.6 ДРУГИЕ ХАРАКТЕРИСТЕКИ

Кроме этого для оценки аудиофайлов могут быть использованы другие характеристики: амплитуды спектрограммы, продолжительность, среднее значение амплитуды, энергия (как её найти описывалось в пункте 2.2), среднее квадратическое (оно может показать среднюю громкость).

# ОЦЕНКА КАЧЕСТВА КЛАСТЕРИЗАЦИИ

## 4.1 МЕТРИКИ И ИХ СВОЙСТВА

Обычно задано образцовое разбиение на кластеры и разбиение, построенное алгоритмически. Метрика качества должна оценить степень соответствия между ними.

Метрики должны обладать следующими свойствами:

1. Однородность кластеров. Значение метрики качества должно уменьшаться при объединении в один кластер двух эталонных.
2. Полнота кластеров. Это свойство, двойственное свойству однородности. Значение метрики качества должно уменьшаться при разделении эталонного кластера на части.
3. «Лоскутный мешок». Пусть есть два кластера: чистый, содержащий элементы из одного эталонного кластера, и шумный («лоскутный»), где собраны элементы из большого числа различных эталонных кластеров. Тогда значение метрики качества должно быть выше у той версии кластеризации, которая помещает новый нерелевантный обоим кластерам элемент в шумный кластер, по сравнению с версией, которая помещает этот элемент в чистый кластер.
4. Размер против количества. Значительное ухудшение кластеризации большого числа небольших кластеров должно обходиться дороже небольшого ухудшения кластеризации в крупном кластере.

## 4.2 МЕТРИКИ, ОСНОВАННЫЕ НА ПОДСЧЁТЕ ПАР (КЛАССИФИКАЦИОННЫЕ)

В этом подходе задача кластеризации сводится к бинарной классификации на множестве пар объектов. Пара объектов считается относящейся к положительному классу тогда и только тогда, когда оба объекта относятся к одному и тому же эталонному кластеру. Предсказание считается положительным тогда и только тогда, когда эти объекты относятся к одному и тому же алгоритмическому кластеру.

Высчитываются 4 значения:

— TP - пары определены как положительные и при этом являются положительными;

— FP - пары определены как положительные, но ими не являются;

— TN - пары определены как отрицательные и при этом являются отрицательными;

— FN - пары определены как отрицательные, но ими не являются.

Далее на основе этих параметров можно вычислить оценку кластеризации по формулам (4.1).

|  |  |
| --- | --- |
|  | (4.1) |

## 4.3 МЕТРИКИ, ОСНОВАННЫЕ НА СОПОСТАВЛЕНИИ МНОЖЕСТВ

Для определения метрик этого класса вначале вводятся метрики точности и полноты соответствия между кластером и эталоном:

|  |  |
| --- | --- |
|  | (4.2) |

Здесь – вычисленный кластер, – эталон.

Далее определяется F-мера [18]:

|  |  |
| --- | --- |
|  | (4.3) |

# РАЗРАБОТКА ПРОГРАММЫ И ОЦЕНКА РЕЗУЛЬТАТОВ

Была разработана программа на языке Python 3.10.5, которая является реализацией системы кластеризации файлов различных типов.

Данная программа определят тип файла, используя его сигнатуру. Узнав тип файла, она собирает соответствующие статистические параметры для него (из тех, что описаны выше). Признаки, которые являются векторами произвольной длины, заменяются средним значением, стандартным отклонением, гистограммой с 10 промежутками и наиболее вероятным распределением (вероятность распределения признака по определённому закону вычисляется по критерию Колмогорова-Смирнова). Далее она проводит кластеризацию файлов с использованием алгоритмов KMeans, DBSCAN и MeanShift по полученным данным. После этого она оценивает результаты кластеризации.

С кодом программы можно ознакомиться в приложении А.

Для оценки работы программы были выбраны файлы трёх тематик: автомобили, птицы, собаки. Результаты работы программы представлены в таблицах ниже. Однако, стоит учесть, что в процесс кластеризации есть элемент случайности. Поэтому результаты от случая к случаю могут не совпадать.

В таблице 5.1 представлен результат кластеризации текстовых файлов.

Таблица 5.1 – Результат кластеризации текстовых файлов

| Название файла | Кластеры KMeans | Кластеры DBSCAN  (-1 – выбросы) | Кластеры MeanShift |
| --- | --- | --- | --- |
| auto(1).txt | 2 | -1 | 0 |
| auto(2).txt | 2 | -1 | 3 |
| auto(3).txt | 2 | -1 | 2 |
| bird(1).txt | 0 | -1 | 8 |
| bird(2).txt | 0 | -1 | 5 |
| bird(3).txt | 0 | -1 | 4 |
| dog(1).txt | 1 | -1 | 7 |
| dog(2).txt | 1 | -1 | 6 |
| dog(3).txt | 1 | -1 | 1 |

В таблице 5.2 представлен результат кластеризации файлов изображений.

Таблица 5.2 – Результат кластеризации файлов изображений

| Название файла | Кластеры KMeans | Кластеры DBSCAN  (-1 – выбросы) | Кластеры MeanShift |
| --- | --- | --- | --- |
| auto(1).png | 0 | 0 | 7 |
| auto(2).png | 0 | 0 | 5 |
| auto(3).png | 0 | 0 | 6 |
| bird(1).png | 0 | 0 | 1 |
| bird(2).png | 2 | 0 | 8 |
| bird(3).png | 2 | 0 | 3 |
| dog(1).png | 1 | 0 | 0 |
| dog(2).png | 1 | 0 | 2 |
| dog(3).png | 1 | 0 | 4 |

В таблице 5.3 представлен результат кластеризации аудио файлов.

Таблица 5.3 – Результат кластеризации аудио файлов

| Название файла | Кластеры KMeans | Кластеры DBSCAN  (-1 – выбросы) | Кластеры MeanShift |
| --- | --- | --- | --- |
| auto(1).mp3 | 2 | 0 | 5 |
| auto(2).mp3 | 2 | 0 | 7 |
| auto(3).mp3 | 2 | -1 | 8 |
| bird(1).mp3 | 1 | -1 | 0 |
| bird(2).mp3 | 1 | -1 | 2 |
| bird(3).mp3 | 0 | -1 | 6 |
| dog(1).wav | 0 | -1 | 3 |
| dog(2).mp3 | 2 | -1 | 1 |
| dog(3).mp3 | 0 | -1 | 4 |

В таблице 5.4 представлена оценка результатов кластеризации.

Таблица 5.4 - Оценка результатов кластеризации

|  |  |  |  |
| --- | --- | --- | --- |
| Текст | | | |
| Метод оценки | Kmeans | DBSCAN | MeanShift |
| Подсчёт пар | 1 | 0 | 0,5 |
| Сопоставление множеств | 3 | 0 | 1,5 |
| Изображение | | | |
| Метод оценки | Kmeans | DBSCAN | MeanShift |
| Подсчёт пар | 0,821428571 | 0,5 | 0,5 |
| Сопоставление множеств | 2,657142857 | 1,5 | 1,5 |
| Аудио | | | |
| Метод оценки | Kmeans | DBSCAN | MeanShift |
| Подсчёт пар | 0,678571429 | 0,258064516 | 0,5 |
| Сопоставление множеств | 2,323809524 | 0,8 | 1,5 |

Как можно увидеть по данным таблицам, наилучший результат показал алгоритм KMeans.

# ЗАКЛЮЧЕНИЕ

Данная работа была посвящена разработке системы кластеризации файлов различных типов статистическими методами. Для разработки системы изучены различные алгоритмы кластеризации (в первой части работы), а также рассмотрены методы сбора статистической информации текстовых файлов, файлов изображений и аудио файлов. Кроме этого, были изучены методы оценки результатов кластеризации.

Данные типы файлов были выбраны, так как задачу о кластеризации большинства других типов файлов можно свести к задаче о кластеризации трёх обозначенных.

В результате работы система кластеризации файлов была разработана. Она показала отличные результаты при обработке текстовых файлов и достаточно неплохие при обработке файлов изображений и аудио файлов.

При оценке результатов работы разработанной системы гипотеза о пригодности алгоритма DBSCAN для поставленной задачи была опровергнута. Однако удалось установить, что хорошие результаты показывает алгоритм KMeans.

В дальнейшем данную систему можно будет значительно улучшить. Для этого нужно подобрать более подходящие параметры для алгоритмов, использующихся для сбора статистической информации файлов. Также можно будет применить заранее обученные нейросети глубоко анализа для сбора дополнительной информации об изображениях и аудио. Кроме этого, можно будет применить алгоритмы машинного обучения, которые с помощью анализа больших выборок смогут установить более подходящие параметры для самих алгоритмов кластеризации.

# ЛИТЕРАТУРА

1. What is Tokenization in Natural Language Processing (NLP)? [Электронный ресурс] Режим доступа: https://www.machinelearningplus.com/nlp/what-is-tokenization-in-natural-language-processing/ (Дата обращения 06.06.2023).
2. What is NLP (Natural Language Processing) Tokenization? [Электронный ресурс] Режим доступа: https://www.tokenex.com/blog/ab-what-is-nlp-natural-language-processing-tokenization/ (Дата обращения 06.06.2023).
3. What are Stop Words? [Электронный ресурс] Режим доступа: https://kavita-ganesan.com/what-are-stop-words/#:~:text=Stop%20words%20are%20a%20set,on%20the%20important%20words%20instead (Дата обращения 06.06.2023).
4. lemmatization [Электронный ресурс] Режим доступа: https://www.techtarget.com/searchenterpriseai/definition/lemmatization#:~:text=Lemmatization%20takes%20a%20word%20and,its%20lemma%2C%20%22walk.%22 (Дата обращения 06.06.2023).
5. Метрика TF-IDF (Term frequency–inverse document frequency) [Электронный ресурс] Режим доступа: https://wiki.loginom.ru/articles/tf-idf.html (Дата обращения 06.06.2023).
6. What Are Image Histograms? [Электронный ресурс] Режим доступа: https://www.baeldung.com/cs/image-histograms (Дата обращения 06.06.2023).
7. GLCM [Электронный ресурс] Режим доступа: https://juliaimages.org/ImageFeatures.jl/stable/tutorials/glcm/ (Дата обращения 06.06.2023).
8. Aborisade D.O., Ojo J.A., Amole A.O., Durodola A.O. Comparative Analysis of Textural Features Derived from GLCM for Ultrasound Liver ImageClassification // International Journal of Computer Trends and Technology (IJCTT) – volume 11 number 6 – May 2014.
9. Детектор границ Канни. [Электронный ресурс] Режим доступа: https://habr.com/ru/articles/114589/ (Дата обращения 06.06.2023).
10. Урок 2. Гистограмма направленных градиентов с использованием OpenCV. [Электронный ресурс] Режим доступа: https://waksoft.susu.ru/2021/11/01/histogram-of-oriented-gradients/ (Дата обращения 06.06.2023).
11. Мел-кепстральные коэффициенты (MFCC) и распознавание речи. [Электронный ресурс] Режим доступа: <https://habr.com/ru/articles/140828/> (Дата обращения 06.06.2023).
12. Chroma Feature Analysis and Synthesis. [Электронный ресурс] Режим доступа: https://www.ee.columbia.edu/~dpwe/resources/matlab/chroma-ansyn/ (Дата обращения 06.06.2023).
13. Nanzhu Jiang, Peter Grosche, Verena Konz and Meinard Muller. Analyzing Chroma Feature Types for Automated Chord Recognition // Saarland University and MPI Informatik, Campus E1.4, 66123 Saarbrucken, Germany
14. McKinney, Martin & Moelants, Dirk & Davies, Matthew & Klapuri, Anssi. (2007). Evaluation of Audio Beat Tracking and Music Tempo Extraction Algorithms. Journal of New Music Research. 36. 1-16. 10.1080/09298210701653252.
15. Jun Yang, Fa-Long Luo, Arye Nehorai. Spectral contrast enhancement: Algorithms and comparisons. // Fortemedia Inc., 20111 Stevens Creek Boulevard, Suite 150, Cupertino, CA 95014, USA, Quicksilver Technology, 6640 Via Del Oro, San Jose, CA 95119, USA, ECE Department, University of Illinois at Chicago, 851 S. Morgan Street, 1120 SEO, Chicago, IL 60607, USA
16. librosa.feature.spectral\_contrast. [Электронный ресурс] Режим доступа: https://librosa.org/doc/main/generated/librosa.feature.spectral\_contrast.html#id2 (Дата обращения 06.06.2023).
17. Analysis of Zero Crossing Rates of Different Music Genre Tracks [Электронный ресурс] Режим доступа: <https://www.analyticsvidhya.com/blog/2022/01/analysis-of-zero-crossing-rates-of-different-music-genre-tracks/> (Дата обращения 06.06.2023).
18. Оценка качества кластеризации: свойства, метрики, код на GitHub. [Электронный ресурс] Режим доступа: <https://habr.com/ru/companies/yandex/articles/500742/> (Дата обращения 06.06.2023).

# ПРИЛОЖЕНИЕ А

Код разработанной системы кластеризации файлов различных типов статистическими методами на языке Python 3.10.5.

1. **from** sklearn.feature\_extraction.text **import** TfidfVectorizer
2. **import** os
3. **import** librosa
4. **import** magic
5. **import** sys
6. **from** sklearn.cluster **import** KMeans, DBSCAN, MeanShift
7. **from** nltk.corpus **import** stopwords
8. **from** nltk.tokenize **import** word\_tokenize
9. **from** nltk.stem **import** WordNetLemmatizer
10. **from** pymorphy3 **import** MorphAnalyzer
11. **import** string
12. **from** langdetect **import** detect
13. **import** iso639
14. **import** cv2
15. **import** numpy as np
16. **from** skimage.feature **import** graycomatrix, graycoprops
17. **from** scipy.stats **import** kstest
18. **from** scipy.stats **import** uniform, norm, expon
19. **from** PIL **import** Image


23. **def** calculate\_dist(sample):
25. dist\_list **=** [uniform, norm, expon]
27. hist, bins **=** np.histogram(sample, bins**=**10)
29. dist\_prop **=** [np.mean(sample), np.std(sample)]
30. dist\_prop.extend(hist)
31. dist\_prop.extend(bins)
33. choosed\_dist **=** 0
34. max\_p\_value **=** 0
35. **for** i, dist **in** enumerate(dist\_list):
36. \_, p\_value **=** kstest(sample, dist.cdf)
37. **if** p\_value > max\_p\_value:
38. max\_p\_value **=** p\_value
39. choosed\_dist **=** i
40. dist\_prop.append(choosed\_dist)
42. **return** dist\_prop
44. # Текст
45. **def** preprocess\_text(text):
46. # Определение языка текста
47. lang\_code **=** detect(text)
48. lang\_name **=** iso639.to\_name(lang\_code).lower()
50. # Токенизация текста
51. tokens **=** word\_tokenize(text.lower(), lang\_name)
53. # Удаление знаков пунктуации
54. tokens **=** [token **for** token **in** tokens **if** token **not** **in** string.punctuation]

57. # Лемматизация слов
58. **if** lang\_name **==** 'russian':
59. lemmatizer **=** MorphAnalyzer()
60. tokens **=** [lemmatizer.parse(token)[0].normal\_form **for** token **in** tokens]
61. **elif** lang\_name **==** 'english':
62. lemmatizer **=** WordNetLemmatizer()
63. tokens **=** [lemmatizer.lemmatize(token) **for** token **in** tokens]
64. **else**:
65. print("Текущая версия программы не поддерживает лемматизацию для языка " **+** lang\_name **+** ". Дальнейшая обработка данного текста будет проведена без лемматизации.")
67. # Выбор набора стоп-слов на основе языка
68. stop\_words **=** set(stopwords.words(lang\_name))
69. tokens **=** [token **for** token **in** tokens **if** token **not** **in** stop\_words]
71. # Объединение токенов обратно в текстовую строку
72. processed\_text **=** ' '.join(tokens)
74. **return** processed\_text


78. # Изображение
79. **def** get\_color\_histogram(file\_path):
80. # Открываем картиику
81. image **=** Image.open(file\_path)
82. # Убеждаемся, что все картинки будут в одном режиме цветности
83. rgba **=** image.convert("RGBA")
84. # Получаем гистограмму
85. histogram **=** rgba.histogram()
86. **return** histogram

89. **def** calculate\_hog\_features(image):
91. # Парметры HOG
92. win\_size **=** (64, 128)  # Размер окна
93. block\_size **=** (16, 16)  # Размер блока
94. block\_stride **=** (8, 8)  # Шаг блока
95. cell\_size **=** (8, 8)  # Размер ячейки
96. nbins **=** 9  # Количество направлений градиента
98. # Приведение изображения к размеру окна
99. resized\_image **=** cv2.resize(image, win\_size)
101. # Вычисление гистограммы распределения направленных градиентов (HOG)
102. hog **=** cv2.HOGDescriptor(win\_size, block\_size, block\_stride, cell\_size, nbins)
103. hist **=** hog.compute(resized\_image)
105. # Нормализация гистограммы
106. hist **=** hist.flatten()
107. hist **=** hist **/** np.linalg.norm(hist)
109. **return** hist

112. **def** calculate\_texture\_features(image):
113. # Преобразование изображения в оттенки серого
114. gray\_image **=** cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)
116. # Вычисление матрицы смежности градиентов (GLCM)
117. distances **=** [1]  # расстояние между пикселями
118. angles **=** [0, np.pi**/**4, np.pi**/**2, 3**\***np.pi**/**4]  # углы для вычисления различных градиентов
119. glcm **=** graycomatrix(gray\_image, distances, angles, levels**=**256, symmetric**=**True, normed**=**True)
121. # Вычисление текстурных признаков на основе GLCM
122. contrast **=** graycoprops(glcm, 'contrast').flatten()
123. dissimilarity **=** graycoprops(glcm, 'dissimilarity').flatten()
124. homogeneity **=** graycoprops(glcm, 'homogeneity').flatten()
125. energy **=** graycoprops(glcm, 'energy').flatten()
126. correlation **=** graycoprops(glcm, 'correlation').flatten()
128. **return** np.concatenate((contrast, dissimilarity, homogeneity, energy, correlation))

131. **def** get\_image\_contours(image):
133. gray **=** cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)
135. # Применение алгоритма Кэнни
136. edges **=** cv2.Canny(gray, 100, 200)
137. # Контуры на изображении
138. contours, \_ **=** cv2.findContours(edges, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)
139. contours\_flattened **=** []
140. **for** contour **in** contours:
141. contours\_flattened.extend(contour.flatten())
143. **return** calculate\_dist(contours\_flattened)


147. # Аудио
148. **def** get\_audio\_features(file\_path):
149. # Загрузка аудиофайла
150. audio, sr **=** librosa.load(file\_path)
151. # Извлечение MFCC
152. mfcc **=** librosa.feature.mfcc(y**=**audio, sr**=**sr)
153. # Извлечение хроматических признаков
154. chroma **=** librosa.feature.chroma\_stft(y**=**audio, sr**=**sr)
155. # Извлечение темпа и ритмических признаков
156. \_, beat\_frames **=** librosa.beat.beat\_track(y**=**audio, sr**=**sr)
157. # Извлечение спектрального контраста
158. contrast **=** librosa.feature.spectral\_contrast(y**=**audio, sr**=**sr)
159. # Извлечение zero crossing rate
160. zcr **=** librosa.feature.zero\_crossing\_rate(y**=**audio)
161. # Извлечение спектральных коэффициентов
162. spectrogram **=** librosa.stft(y**=**audio)
163. spectral\_coefficients **=** librosa.amplitude\_to\_db(abs(spectrogram))
164. # Продолжительность аудиофайла в секундах
165. duration **=** librosa.get\_duration(y**=**audio, sr**=**sr)
166. # RMS (root mean square)
167. rms **=** librosa.feature.rms(y**=**audio)
169. **return** calculate\_dist(mfcc.flatten()), calculate\_dist(chroma.flatten()), calculate\_dist(beat\_frames.flatten()), calculate\_dist(contrast.flatten()), calculate\_dist(zcr.flatten()), calculate\_dist(spectral\_coefficients.flatten()), duration, calculate\_dist(rms.flatten())


173. # Метрики
174. **def** are\_elements\_in\_same\_row(matrix, element1, element2):
175. **for** row **in** matrix:
176. **if** element1 **in** row **and** element2 **in** row:
177. **return** True
178. **return** False

181. **def** get\_clusters(example\_labels, calculated\_labels):
182. max\_example\_label **=** max(label **for** label **in** example\_labels)
183. max\_calculated\_label **=** max(label **for** label **in** calculated\_labels)
184. example\_clusters **=** [[] **for** i **in** range(max\_example\_label**+**1)]
185. calculated\_clusters **=** [[] **for** i **in** range(max\_calculated\_label**+**1)]
186. **for** i, label **in** enumerate(example\_labels):
187. **if** label !**=** **-**1:
188. example\_clusters[label].append(i)
189. **for** i, label **in** enumerate(calculated\_labels):
190. **if** label !**=** **-**1:
191. calculated\_clusters[label].append(i)
192. **return** example\_clusters, calculated\_clusters

195. **def** classification\_metric(example\_labels, calculated\_labels):
196. example\_clusters, calculated\_clusters **=** get\_clusters(example\_labels, calculated\_labels)
197. TP **=** 0
198. FP **=** 0
199. TN **=** 0
200. FN **=** 0
201. **for** i **in** range(len(example\_labels)):
202. **for** j **in** range(len(example\_labels)):
203. in\_same\_example\_cluster **=** are\_elements\_in\_same\_row(example\_clusters, i, j)
204. in\_same\_calculated\_cluster **=** are\_elements\_in\_same\_row(calculated\_clusters, i, j)
205. **if** in\_same\_example\_cluster **and** in\_same\_calculated\_cluster: TP**+=**1
206. **elif** **not** in\_same\_example\_cluster **and** in\_same\_calculated\_cluster: FP**+=**1
207. **elif** **not** in\_same\_example\_cluster **and** **not** in\_same\_calculated\_cluster: TN**+=**1
208. **else**: FN**+=**1
209. **if** TP **==** 0: **return** 0
210. P **=** TP **/** (TP **+** FP)
211. R **=** TP **/** (TP **+** FN)
212. F **=** 2 **\*** P **\*** R **/** (P **+** R)
213. **return** F

216. **def** comparation\_of\_sets\_metric(example\_labels, calculated\_labels):
217. example\_clusters, calculated\_clusters **=** get\_clusters(example\_labels, calculated\_labels)
218. F\_matrix **=** [[] **for** i **in** range(len(example\_clusters))]
219. **for** i, example\_cluster **in** enumerate(example\_clusters):
220. **for** calculated\_cluster **in** calculated\_clusters:
221. intersection **=** set(example\_cluster) & set(calculated\_cluster)
222. P **=** len(intersection) **/** len (calculated\_cluster)
223. R **=** len(intersection) **/** len (example\_cluster)
224. F **=** 0
225. **if** **not** P **+** R **==** 0:
226. F **=** 2 **\*** P **\*** R **/** (P **+** R)
227. F\_matrix[i].append(F)
228. F **=** 0
229. **for** j **in** range(len(example\_clusters)):
230. F **+=** len(example\_clusters[j]) **/** len(example\_clusters) **\*** (max(F\_value **for** F\_value **in** F\_matrix[j]) **if** F\_matrix[j] **else** 0)
231. **return** F



236. **def** get\_file\_type(file):
237. # создаётся объект класса Magic, с помощью него читается информация о типе файла в виде тип/расширение (например 'text/plain', 'image/jpeg', 'audio/mp3')
238. **return** magic.Magic(mime**=**True, magic\_file**=**"magic.mgc").from\_file(file).split("/")

241. **def** is\_iterable(obj):
242. **try**:
243. iter(obj)
244. **return** True
245. **except** TypeError:
246. **return** False

249. **def** multi\_clust(features\_lists, num\_of\_clust):
250. features\_clusteraised\_kmeans **=** []
251. features\_clusteraised\_dbscan **=** []
252. features\_clusteraised\_meanshift **=** []
253. **for** feature\_list **in** features\_lists:
254. **if** is\_iterable(feature\_list[0]):
255. kmeans **=** KMeans(n\_clusters**=**num\_of\_clust, n\_init**=**10)
256. dbscan **=** DBSCAN(eps**=**0.5, min\_samples**=**2)
257. meanshift **=** MeanShift(bandwidth**=**0.5)
258. kmeans.fit(feature\_list)
259. dbscan.fit(feature\_list)
260. meanshift.fit(feature\_list)
261. features\_clusteraised\_kmeans.append(kmeans.labels\_)
262. features\_clusteraised\_dbscan.append(dbscan.labels\_)
263. features\_clusteraised\_meanshift.append(meanshift.labels\_)
264. **else**:
265. features\_clusteraised\_kmeans.append(feature\_list)
266. features\_clusteraised\_dbscan.append(feature\_list)
267. features\_clusteraised\_meanshift.append(feature\_list)
268. features\_clusteraised\_kmeans **=** [[raw[i] **for** raw **in** features\_clusteraised\_kmeans] **for** i **in** range(len(features\_clusteraised\_kmeans[0]))]
269. features\_clusteraised\_dbscan **=** [[raw[i] **for** raw **in** features\_clusteraised\_dbscan] **for** i **in** range(len(features\_clusteraised\_dbscan[0]))]
270. features\_clusteraised\_meanshift **=** [[raw[i] **for** raw **in** features\_clusteraised\_meanshift] **for** i **in** range(len(features\_clusteraised\_meanshift[0]))]
271. kmeans **=** KMeans(n\_clusters**=**num\_of\_clust, n\_init**=**10)
272. dbscan **=** DBSCAN(eps**=**0.5, min\_samples**=**2)
273. meanshift **=** MeanShift(bandwidth**=**0.5)
274. kmeans.fit(features\_clusteraised\_kmeans)
275. dbscan.fit(features\_clusteraised\_dbscan)
276. meanshift.fit(features\_clusteraised\_meanshift)
277. **return** kmeans.labels\_, dbscan.labels\_, meanshift.labels\_


281. **if** \_\_name\_\_ **==** "\_\_main\_\_":
283. **if** len(sys.argv) !**=** 3:
284. print("Запускать так:", sys.argv[0], "<директория с файлами> <количество кластеров>")
285. exit()
287. directory **=** sys.argv[1]
288. num\_of\_clust **=** int(sys.argv[2])
289. file\_names **=** os.listdir(directory)
291. text\_files **=** []
292. texts\_preprocessed **=** []
294. image\_files **=** []
295. images\_color\_histograms **=** []
296. images\_hog\_features **=** []
297. images\_texture\_features **=** []
298. images\_contours **=** []
300. audio\_files **=** []
301. audio\_mfccs **=** []
302. audio\_chromas **=** []
303. audio\_tempos **=** []
304. audio\_beat\_frames **=** []
305. audio\_contrast **=** []
306. audio\_zcr **=** []
307. audio\_spectral\_coefficients **=** []
308. audio\_duration **=** []
309. audio\_mean\_amplitude **=** []
310. audio\_energy **=** []
311. audio\_rms **=** []

314. **for** file\_name **in** file\_names:
315. print("Начата обработка файла", file\_name)
316. file\_path **=** os.path.join(directory, file\_name)
317. file\_type, \_ **=** get\_file\_type(file\_path)
318. match file\_type:
319. case "text":
320. text\_files.append(file\_name)
321. with open(file\_path, 'r', encoding**=**'utf-8') as f:
322. text **=** f.read()
323. processed\_text **=** preprocess\_text(text)
324. texts\_preprocessed.append(processed\_text)
325. case "image":
326. image\_files.append(file\_name)
327. image **=** cv2.imread(file\_path)
328. images\_color\_histograms.append(get\_color\_histogram(file\_path))
329. images\_hog\_features.append(calculate\_hog\_features(image))
330. images\_texture\_features.append(calculate\_texture\_features(image))
331. images\_contours.append(get\_image\_contours(image))
332. case "audio":
333. audio\_files.append(file\_name)
334. **for** features\_list, feature **in** zip([audio\_mfccs, audio\_chromas, audio\_beat\_frames, audio\_contrast, audio\_zcr, audio\_spectral\_coefficients, audio\_duration, audio\_rms], get\_audio\_features(file\_path)):
335. features\_list.append(feature)
336. case \_:
337. print(file\_type)
338. print("Файл", file\_name, "обработан")
339. print("\n")
341. # Вычисление TF-IDF
342. vectorizer **=** TfidfVectorizer()
343. tfidf\_matrix **=** vectorizer.fit\_transform(texts\_preprocessed)
345. # Кластеризация текста
346. kmeans\_text **=** KMeans(n\_clusters**=**num\_of\_clust, n\_init**=**10)
347. dbscan\_text **=** DBSCAN(eps**=**0.5, min\_samples**=**2)
348. meanshift\_text **=** MeanShift(bandwidth**=**0.5)
349. kmeans\_text.fit(tfidf\_matrix)
350. dbscan\_text.fit(tfidf\_matrix)
351. meanshift\_text.fit(tfidf\_matrix.toarray())
352. text\_labels\_kmeans **=** kmeans\_text.labels\_
353. text\_labels\_dbscan **=** dbscan\_text.labels\_
354. text\_labels\_meanshift **=** meanshift\_text.labels\_
355. **for** file, label\_kmeans, label\_dbscan, label\_meanshift **in** zip(text\_files, text\_labels\_kmeans, text\_labels\_dbscan, text\_labels\_meanshift):
356. print(file, label\_kmeans, label\_dbscan, label\_meanshift)
357. print("\n")
359. # Кластеризация картинок
360. image\_labels\_kmeans, image\_labels\_dbscan, image\_labels\_meanshift **=** multi\_clust([images\_color\_histograms, images\_hog\_features, images\_texture\_features, images\_contours], num\_of\_clust)
361. **for** file, label\_kmeans, label\_dbscan, label\_meanshift **in** zip(image\_files, image\_labels\_kmeans, image\_labels\_dbscan, image\_labels\_meanshift):
362. print(file, label\_kmeans, label\_dbscan, label\_meanshift)
363. print("\n")
365. # Кластеризация аудио
366. audio\_labels\_kmeans, audio\_labels\_dbscan, audio\_labels\_meanshift **=** multi\_clust([audio\_mfccs, audio\_chromas, audio\_beat\_frames, audio\_contrast, audio\_zcr, audio\_spectral\_coefficients, audio\_duration, audio\_rms], num\_of\_clust)
367. **for** file, label\_kmeans, label\_dbscan, label\_meanshift **in** zip(audio\_files, audio\_labels\_kmeans, audio\_labels\_dbscan, audio\_labels\_meanshift):
368. print(file, label\_kmeans, label\_dbscan, label\_meanshift)
369. print("\n")
371. # Оценка кластеризации
372. example\_labels **=** [0,0,0,1,1,1,2,2,2]
373. print("Текст:")
374. print("Подсчёт пар", classification\_metric(example\_labels, text\_labels\_kmeans), classification\_metric(example\_labels, text\_labels\_dbscan), classification\_metric(example\_labels, text\_labels\_meanshift))
375. print("Сопоставление множеств", comparation\_of\_sets\_metric(example\_labels, text\_labels\_kmeans), comparation\_of\_sets\_metric(example\_labels, text\_labels\_dbscan), comparation\_of\_sets\_metric(example\_labels, text\_labels\_meanshift))
376. print("Изображение:")
377. print("Подсчёт пар", classification\_metric(example\_labels, image\_labels\_kmeans), classification\_metric(example\_labels, image\_labels\_dbscan), classification\_metric(example\_labels, image\_labels\_meanshift))
378. print("Сопоставление множеств", comparation\_of\_sets\_metric(example\_labels, image\_labels\_kmeans), comparation\_of\_sets\_metric(example\_labels, image\_labels\_dbscan), comparation\_of\_sets\_metric(example\_labels, image\_labels\_meanshift))
379. print("Аудио:")
380. print("Подсчёт пар", classification\_metric(example\_labels, audio\_labels\_kmeans), classification\_metric(example\_labels, audio\_labels\_dbscan), classification\_metric(example\_labels, audio\_labels\_meanshift))
381. print("Сопоставление множеств", comparation\_of\_sets\_metric(example\_labels, audio\_labels\_kmeans), comparation\_of\_sets\_metric(example\_labels, audio\_labels\_dbscan), comparation\_of\_sets\_metric(example\_labels, audio\_labels\_meanshift))