МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО “СЕВЕРО-КАВКАЗСКИЙ ГОРНО-МЕТАЛЛУРГИЧЕСКИЙ ИНСТИТУТ   
(ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)

**Факультет** Информационных технологий и электронной техники

**Кафедра** Информатика и вычислительная техника

**Направление подготовки** Информатика и вычислительная техника

**Профиль** АСУ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

на тему: Разработка системы кластеризации документов на базе методов машинного обучения.

**Студент**  Кастуев Хетаг Асланович

**Руководитель проекта** Будаева А.А.

Проект рассмотрен кафедрой и допущен к защите в ГЭК

**Заведующий кафедрой** проф. Гроппен В. О.

г. Владикавказ 2021 г.

# Реферат

Пояснительная записка 100 с., 13 рис., 12 табл., 8 ист., 2 прил.

Кластеризация текстовых документов; мешок слов.

**Объект разработки:** системы кластеризации документов на базе методов машинного обучения.

**Цель выпускной работы:** создание ПП, способного кластеризовывать документы.

**Использованное прикладное и системное программное обеспечение:**

Операционная система Windows 10 PRO – 64 bit.

Среда разработки IntelliJ IDEA.

**Характеристики вычислительной системы:**

AMD Ryzen 5 3600 6-Core Processor 3.6 (4,2) ГГц   
NVIDIA GeForce RTX 2060 6GB  
ОЗУ 16Гб

**Полученные результаты:**

**<…>**

Оглавление

[Реферат 2](#_Toc73287916)

[Введение 5](#_Toc73287917)

[Глава 1. Аналитический обзор 9](#_Toc73287918)

[1.1. Цели кластеризации 9](#_Toc73287919)

[1.2. Применение 10](#_Toc73287920)

[1.2.1. Биология и биоинформатика 10](#_Toc73287921)

[1.2.2. Медицина 11](#_Toc73287922)

[1.2.3. Маркетинг 11](#_Toc73287923)

[1.2.4. Интернет 11](#_Toc73287924)

[1.2.5. Компьютерные науки 11](#_Toc73287925)

[1.3. Алгоритмы классификации 12](#_Toc73287926)

[1.3.1. Алгоритмы иерархической кластеризации; 12](#_Toc73287927)

[1.3.2. Алгоритмы квадратичной ошибки 12](#_Toc73287928)

[1.3.3. Нечеткие алгоритмы 13](#_Toc73287929)

[1.3.4. Алгоритмы, основанные на теории графов 14](#_Toc73287930)

[1.3.5. Алгоритм выделения связных компонент 15](#_Toc73287931)

[1.3.6. Алгоритм минимального покрывающего дерева 15](#_Toc73287932)

[1.3.7. Послойная кластеризация 16](#_Toc73287933)

[1.4. Сравнение алгоритмов 17](#_Toc73287934)

[1.4.1. Вычислительная сложность алгоритмов 17](#_Toc73287935)

[1.4.2. Сравнительная таблица алгоритмов 17](#_Toc73287936)

[1.5. Мера расстояний 18](#_Toc73287937)

[1.5.1. Евклидово расстояние 19](#_Toc73287938)

[1.5.2. Квадрат евклидова расстояния 19](#_Toc73287939)

[1.5.3. Расстояние городских кварталов 19](#_Toc73287940)

[1.5.4. Расстояние Чебышева 20](#_Toc73287941)

[1.5.5. Степенное расстояние 20](#_Toc73287942)

[1.6. Существующие программные средства 20](#_Toc73287943)

[1.7. Заключение 21](#_Toc73287944)

[Глава 2. Выбор и обоснование алгоритма 22](#_Toc73287945)

[Глава 3. ПРОГРАММНЫЙ КОМПЛЕКС 24](#_Toc73287946)

[3.1. Архитектура программного обеспечения 24](#_Toc73287947)

[3.2. Язык программирования Java 24](#_Toc73287948)

[3.3. Работа программного комплекса 25](#_Toc73287949)

[3.3.1. Кластеризация 25](#_Toc73287950)

[3.3.2. Классификация 28](#_Toc73287951)

[Глава 4. Экспериментальная часть 31](#_Toc73287952)

[4.1. Оценка производительности алгоритма кластеризации 32](#_Toc73287953)

[4.1.1. Зависимость от количества слов в файле 32](#_Toc73287954)

[4.1.2. Зависимость от количества файлов 33](#_Toc73287955)

[4.2. Оценка производительности алгоритма классификации 34](#_Toc73287956)

[4.2.1. Зависимость от количества слов в файле 34](#_Toc73287957)

[4.2.2. Зависимость от количества файлов 35](#_Toc73287958)

[4.2.3. Зависимость от количества классов 36](#_Toc73287959)

[ГЛАВА 5. Заключение 37](#_Toc73287960)

[Список использованных источников 38](#_Toc73287961)

# Введение

Трудно представить себе жизнь современного человека без машинного обучения и искусственного интеллекта. Данные технологии создают новые возможности. Например, системы безопасности в метро используют машинное обучение для распознавания и поиска лиц, которые находятся в розыске. Компьютерное зрение используется в беспилотных автомобилях и летательных аппаратах для коммерческих и военных целей.

Машинный интеллект - это направление технологических исследований. Оно связано с созданием системы, которая способна обучаться самостоятельно.

Искусственный интеллект широко применяется в самых разных областях, как:

1. lT-сфера: разработка приложений;

2. Рекламные компании;

3. Маркетинговые исследования;

4. Медицинская диагностика;

5. Техническая диагностика;

6. Автопилотирование;

7. Биоинформатика и в другие сферы;

8. Обработка текста (в том числи кластеризация).

Все это и многое другое работает при помощи машинного интеллекта.

С одной стороны, такой технический прогресс приносит человеку огромную пользу, взять хотя бы медицинскую диагностику: многие исследования стали быстрыми и доступными, а с другой стороны такой технический прогресс бросает вызов каждому из нас: ведь совсем скоро не нужны будут машинисты в электропоездах, многий обслуживающий персонал заменят компьютеры, такcистов заменят беспилотные автомобили. Поэтому тему машинного обучения я считаю актуальной в настоящее время с научной и практической точки зрения. Нужно не только знать, что существует так называемое машинное обучение, но и понимать, как это работает.

Машинное обучение избавляет программиста от необходимости писать большой код, т.е объяснять компьютеру, как нужно решить какую-нибудь проблему. В процессе машинного обучения компьютер учат самостоятельно находить правильное решение. То есть у нас есть какие-либо известные данные и на основе их мы собираем статистику и уже на новых данных мы учимся что-то понимать (находим новые, ещё не описанные закономерности).

Таким образом: Основная идея машинного обучения состоит в том, чтобы научить компьютер "учиться", т.е. вычленять из каких-либо данных полезные знания.

Надо отметить, что обучение может происходить как до работы программы, так и в процессе выполнения программы. Например, программа для распознавания лиц сначала учится, а затем работает на настоящих данных; программа распознавания спама учится во время исполнения.

На сегодняшний день выделяют два основных типа машинного обучения: дедуктивное обучение (аналитическое) и индуктивное ( статистическое)обучение.

Дедуктивное обучение относится к области экспертных систем. Экспертные системы не могут решать универсальные задачи как нейронные сети.

Экспертные системы - это большой обьем знаний экспертов формализация этих знаний инженерами по знаниям + грамотный код программиста. В итоге мы получаем хорошо составленную базу знаний. Минусом этой системы можно назвать то, что без поддержки разработчиков базы знаний быстро становятся невостребованными. Дедуктивное обучение (поскольку относится к экспертным системам) решает только узконаправленные специализированные задачи. Имеются знания экспертов и перенос их в компьютер в виде базы знаний. Во всех задачах дедуктивного обучения имеются знания, каким-либо образом отобранные и формализованные. Требуется вывести из них правило, применительно к каждому конкретному случаю. Что же касается второго типа, то здесь лежит выявление закономерностей в эмпирической информации. Обучающийся программе мы предлагаем большое количество примеров данных с закономерностями и на основе полученного опыта программа сама учится строить закономерности. Индуктивное обучение подразделяется на:

– Обучение с учителем - также называют "обучение по прецедентам".

– Обучение без учителя - обучение, в котором нет правильных ответов, только данные;

– Обучение с подкрепление, также называют "стимулируемое обучение" - обучение, в котором агент учится посредством собственных проб и ошибок;

– Активное обучение - очень похоже на обучение с учителем. Разница состоит в том, что ответы изначально неизвестны. Идея в том, что алгоритм сам может обучаться на малых выборках, если он сам выбирает какие данные ему нужны. То есть алгоритм составляет запросы, ответы на которые ему помогают обучаться.

– Частичное обучение - в этом случае большая часть ответов неизвестна.

В настоящее время крупные компании вкладывают большие средства в машинное обучение, потому что данная технология по-настоящему окупается. С каждым годом большие данные становятся все более сложнее и человеку становится уже недостаточно своих аналитических способностей. Некоторые задачи перейдут к искусственному интеллекту, который справится с ними лучше и быстрее, чем человек. Прогнозируется, что в ближайшее десятилетие искусственный интеллект займет около 7% рабочих мест в России. Часть процессов в организациях автоматизируют и управлять этими процессами поручат самообучающимся алгоритмам. Огромно место займет ИИ в интернете вещей. Интернет вещей требует обработки большого потока информации в реальном времени. Устройства, подключенные к сети генерируют гигантские массивы данных, которые необходимо будет обрабатывать, анализировать и хранить. Как пример, зубные щетки с функцией блютуз, которые будут отправлять информацию о состоянии зубов стоматологу или дроны, которые станут незаменимы в сельском хозяйстве. Они будут собирать информацию о зрелости урожая, о состоянии почвы, о вредителях и болезнях растений. И во всех технологиях будет использовать машинное обучение. Поэтому машинное обучение готовит для нас перспективное будущее и массу интересных инноваций.

Целью данной дипломной работы является разработка системы кластеризации документов на базе методов машинного обучения.

# Глава 1. Аналитический обзор

Задача кластеризации – частный случай задачи машинного обучения, в частности, обучения без учителя, которая сводится к разбиению имеющегося множества объектов данных на подмножества таким образом, что элементы одного подмножества существенно отличались по некоторому набору свойств от элементов всех других подмножеств. Объект данных обычно рассматривается как точка в многомерном метрическом пространстве, каждому измерению которого соответствует некоторое свойство (атрибут) объекта, а метрика – есть функция от значений данных свойств. От типов измерений этого пространства, которые могут быть как числовыми, так и категориальными, зависит выбор алгоритма кластеризации данных и используемая метрика. Этот выбор продиктован различиями в природе разных типов атрибутов.

* 1. Цели кластеризации
* Понимание данных путём выявления кластерной структуры. Разбиение выборки на группы схожих объектов позволяет упростить дальнейшую обработку данных и принятия решений, применяя к каждому кластеру свой метод анализа (стратегия «разделяй и властвуй»).
* Сжатие данных. Если исходная выборка избыточно большая, то можно сократить её, оставив по одному наиболее типичному представителю от каждого кластера.
* Обнаружение новизны. Выделяются нетипичные объекты, которые не удаётся присоединить ни к одному из кластеров.

В первом случае число кластеров стараются сделать поменьше. Во втором случае важнее обеспечить высокую степень сходства объектов внутри каждого кластера, а кластеров может быть сколько угодно. В третьем случае наибольший интерес представляют отдельные объекты, не вписывающиеся ни в один из кластеров.

Во всех этих случаях может применяться иерархическая кластеризация, когда крупные кластеры дробятся на более мелкие, те в свою очередь дробятся ещё мельче, и т. д. Такие задачи называются задачами таксономии.

Результатом таксономии является древообразная иерархическая структура. При этом каждый объект характеризуется перечислением всех кластеров, которым он принадлежит, обычно от крупного к мелкому.

Классическим примером таксономии на основе сходства является биноминальная номенклатура живых существ, предложенная Карлом Линнеем в середине XVIII века. Аналогичные систематизации строятся во многих областях знания, чтобы упорядочить информацию о большом количестве объектов.

* 1. Применение

### Биология и биоинформатика

* В области экологии кластеризация используется для выделения пространственных и временных сообществ организмов в однородных условиях;
* Кластерный анализ используется для группировки схожих геномных последовательностей в семейство генов, которые являются консервативными структурами для многих организмов и могут выполнять схожие функции;
* Кластеризация помогает автоматически определять генотипы по различным частям хромосом;
* Алгоритмы применяются для выделения небольшого числа групп генетических вариации человеческого генома.

### Медицина

* Используется в позитронно-эмиссионной томографии для автоматического выделения различных типов тканей на трехмерном изображении;
* Применяется для выявления шаблонов устойчивости к антибиотикам; для классификации антибиотиков по типу антибактериальной активности.

### Маркетинг

* Кластеризация широко используется при изучении рынка для обработки данных, полученных из различных опросов. Может применяться для выделения типичных групп покупателей, разделения рынка для создания персонализированных предложений, разработки новых линий продукции.

### Интернет

* Выделение групп людей на основе графа связей в социальных сетях;
* Повышение релевантности ответов на поисковые запросы путем группировки веб-сайтов по смысловым значениям поискового запроса.

### Компьютерные науки

* Кластеризация используется в сегментации изображений для определения границ и распознавания объектов;
* Кластерный анализ применяется для определения образовавшихся популяционных ниш в ходе работы эволюционных алгоритмов для улучшения параметров эволюции;
* Подбор рекомендаций для пользователя на основе предпочтений других пользователей в данном кластере;
* Определение аномалий путем построения кластеров и выявления неклассифицированных объектов.

## Алгоритмы классификации

### Алгоритмы иерархической кластеризации;

Среди алгоритмов иерархической кластеризации выделяются два основных типа: восходящие и нисходящие алгоритмы. Нисходящие алгоритмы работают по принципу «сверху-вниз»: в начале все объекты помещаются в один кластер, который затем разбивается на все более мелкие кластеры. Более распространены восходящие алгоритмы, которые в начале работы помещают каждый объект в отдельный кластер, а затем объединяют кластеры во все более крупные, пока все объекты выборки не будут содержаться в одном кластере. Таким образом строится система вложенных разбиений. Результаты таких алгоритмов обычно представляют в виде дерева – дендрограммы. Классический пример такого дерева – классификация животных и растений.

Для вычисления расстояний между кластерами чаще все пользуются двумя расстояниями: одиночной связью или полной связью (см. обзор мер расстояний между кластерами).

К недостатку иерархических алгоритмов можно отнести систему полных разбиений, которая может являться излишней в контексте решаемой задачи.

### Алгоритмы квадратичной ошибки

Задачу кластеризации можно рассматривать как построение оптимального разбиения объектов на группы. При этом оптимальность может быть определена как требование минимизации среднеквадратической ошибки разбиения:

где *cj* — «центр масс» кластера *j* (точка со средними значениями характеристик для данного кластера).

Алгоритмы квадратичной ошибки относятся к типу плоских алгоритмов. Самым распространенным алгоритмом этой категории является метод k-средних. Этот алгоритм строит заданное число кластеров, расположенных как можно дальше друг от друга. Работа алгоритма делится на несколько этапов:

1. Случайно выбрать *k* точек, являющихся начальными «центрами масс» кластеров.
2. Отнести каждый объект к кластеру с ближайшим «центром масс».
3. Пересчитать «центры масс» кластеров согласно их текущему составу.
4. Если критерий остановки алгоритма не удовлетворен, вернуться к п. 2.

В качестве критерия остановки работы алгоритма обычно выбирают минимальное изменение среднеквадратической ошибки. Так же возможно останавливать работу алгоритма, если на шаге 2 не было объектов, переместившихся из кластера в кластер.

К недостаткам данного алгоритма можно отнести необходимость задавать количество кластеров для разбиения.

### Нечеткие алгоритмы

Наиболее популярным алгоритмом нечеткой кластеризации является алгоритм c-средних (c-means). Он представляет собой модификацию метода k-средних. Шаги работы алгоритма:

1. Выбрать начальное нечеткое разбиение *n* объектов на *k* кластеров путем выбора матрицы принадлежности *U* размера *n x k*.
2. Используя матрицу U, найти значение критерия нечеткой ошибки:  
   https://habrastorage.org/getpro/habr/post_images/21c/c5d/72b/21cc5d72b0c9bee8ad0476798eb4093c.jpg,  
   где *ck* — «центр масс» нечеткого кластера *k*:  
   https://habrastorage.org/getpro/habr/post_images/5fb/811/641/5fb811641b4fa3c84ba2490811ff53ea.jpg.
3. Перегруппировать объекты с целью уменьшения этого значения критерия нечеткой ошибки.
4. Возвращаться в п. 2 до тех пор, пока изменения матрицы *U* не станут незначительными.

Этот алгоритм может не подойти, если заранее неизвестно число кластеров, либо необходимо однозначно отнести каждый объект к одному кластеру.

### Алгоритмы, основанные на теории графов

Суть таких алгоритмов заключается в том, что выборка объектов представляется в виде графа G=(V, E), вершинам которого соответствуют объекты, а ребра имеют вес, равный «расстоянию» между объектами. Достоинством графовых алгоритмов кластеризации являются наглядность, относительная простота реализации и возможность вносения различных усовершенствований, основанные на геометрических соображениях. Основными алгоритмам являются алгоритм выделения связных компонент, алгоритм построения минимального покрывающего (остовного) дерева и алгоритм послойной кластеризации.

### Алгоритм выделения связных компонент

В алгоритме выделения связных компонент задается входной параметр R и в графе удаляются все ребра, для которых «расстояния» больше R. Соединенными остаются только наиболее близкие пары объектов. Смысл алгоритма заключается в том, чтобы подобрать такое значение R, лежащее в диапазон всех «расстояний», при котором граф «развалится» на несколько связных компонент. Полученные компоненты и есть кластеры.

Для подбора параметра R обычно строится гистограмма распределений попарных расстояний. В задачах с хорошо выраженной кластерной структурой данных на гистограмме будет два пика – один соответствует внутрикластерным расстояниям, второй – межкластерным расстояния. Параметр R подбирается из зоны минимума между этими пиками. При этом управлять количеством кластеров при помощи порога расстояния довольно затруднительно.

### Алгоритм минимального покрывающего дерева

Алгоритм минимального покрывающего дерева сначала строит на графе минимальное покрывающее дерево, а затем последовательно удаляет ребра с наибольшим весом. На рисунке изображено минимальное покрывающее дерево, полученное для девяти объектов.



Путём удаления связи, помеченной CD, с длиной равной 6 единицам (ребро с максимальным расстоянием), получаем два кластера: {A, B, C} и {D, E, F, G, H, I}. Второй кластер в дальнейшем может быть разделён ещё на два кластера путём удаления ребра EF, которое имеет длину, равную 4,5 единицам.

### Послойная кластеризация

Алгоритм послойной кластеризации основан на выделении связных компонент графа на некотором уровне расстояний между объектами (вершинами). Уровень расстояния задается порогом расстояния *c*. Например, если расстояние между объектами https://habrastorage.org/getpro/habr/post_images/f67/e6a/707/f67e6a707dcac300ff90658e523b135f.jpg, то https://habrastorage.org/getpro/habr/post_images/e91/fe1/65a/e91fe165a15d6fa1981cbb51f0e7fff3.jpg.

Алгоритм послойной кластеризации формирует последовательность подграфов графа *G*, которые отражают иерархические связи между кластерами:

https://habrastorage.org/getpro/habr/post_images/5ec/cd0/7cc/5eccd07cc4e56b227092cbe66501a67f.jpg,  
  
где *Gt = (V, Et)* — граф на уровне *сt*,  
https://habrastorage.org/getpro/habr/post_images/35f/249/c62/35f249c626b8d0e118efc7f472beee48.jpg,  
*сt* – t-ый порог расстояния,  
m – количество уровней иерархии,  
*G0 = (V, o)*, o – пустое множество ребер графа, получаемое при *t0* = 1,  
*Gm = G*, то есть граф объектов без ограничений на расстояние (длину ребер графа), поскольку *tm* = 1.  
  
Посредством изменения порогов расстояния {*с0, …, сm*},   
где 0 = *с0* < *с1* < …< *сm* = 1, возможно контролировать глубину иерархии получаемых кластеров. Таким образом, алгоритм послойной кластеризации способен создавать как плоское разбиение данных, так и иерархическое.

## Сравнение алгоритмов

### Вычислительная сложность алгоритмов

| **Алгоритм кластеризации** | **Вычислительная сложность** |
| --- | --- |
| Иерархический | O(n2) |
| k-средних c-средних | O(nkl),  где k – число кластеров, l – число итераций |
| Выделение связных компонент | зависит от алгоритма |
| O(n2 log n) | Минимальное покрывающее дерево |
| Послойная кластеризация | O(max(n, m)), где m |

### Сравнительная таблица алгоритмов

| **Алгоритм кластеризации** | **Форма кластеров** | **Входные данные** | **Результаты** |
| --- | --- | --- | --- |
| Иерархический | Произвольная | Число кластеров или порог расстояния для усечения иерархии | Бинарное дерево кластеров |
| k-средних | Гиперсфера | Число кластеров | Центры кластеров |
| c-средних | Гиперсфера | Число кластеров, степень нечеткости | Центры кластеров, матрица принадлежности |
| Выделение связных компонент | Произвольная | Порог расстояния R | Древовидная структура кластеров |
| Минимальное покрывающее дерево | Произвольная | Число кластеров или порог расстояния для удаления ребер | Древовидная структура кластеров |
| Послойная кластеризация | Произвольная | Последовательность порогов расстояния | Древовидная структура кластеров с разными уровнями иерархии |

## Мера расстояний

Итак, как же определять «похожесть» объектов? Для начала нужно составить вектор характеристик для каждого объекта — как правило, это набор числовых значений, например, рост-вес человека. Однако существуют также алгоритмы, работающие с качественными (т.н. категорийными) характеристиками.

После того, как мы определили вектор характеристик, можно провести нормализацию, чтобы все компоненты давали одинаковый вклад при расчете «расстояния». В процессе нормализации все значения приводятся к некоторому диапазону, например, [-1, -1] или [0, 1].

Наконец, для каждой пары объектов измеряется «расстояние» между ними — степень похожести.

### Евклидово расстояние

Наиболее распространенная функция расстояния. Представляет собой геометрическим расстоянием в многомерном пространстве:

### Квадрат евклидова расстояния

Применяется для придания большего веса более отдаленным друг от друга объектам. Это расстояние вычисляется следующим образом:

### Расстояние городских кварталов

Это расстояние является средним разностей по координатам. В большинстве случаев эта мера расстояния приводит к таким же результатам, как и для обычного расстояния Евклида. Однако для этой меры влияние отдельных больших разностей (выбросов) уменьшается (т.к. они не возводятся в квадрат). Формула для расчета манхэттенского расстояния:

### Расстояние Чебышева

Это расстояние может оказаться полезным, когда нужно определить два объекта как «различные», если они различаются по какой-либо одной координате. Расстояние Чебышева вычисляется по формуле:

### Степенное расстояние

Применяется в случае, когда необходимо увеличить или уменьшить вес, относящийся к размерности, для которой соответствующие объекты сильно отличаются. Степенное расстояние вычисляется по следующей формуле:

где *r* и *p* – параметры, определяемые пользователем. Параметр *p* ответственен за постепенное взвешивание разностей по отдельным координатам, параметр *r* ответственен за прогрессивное взвешивание больших расстояний между объектами. Если оба параметра – *r* и *p* — равны двум, то это расстояние совпадает с расстоянием Евклида.

## Существующие программные средства

На данный момент существует не одно программное средство, реализующее различные алгоритмы кластеризации. Примером такого проекта является Apache Mahout. Apache Mahout ― это открытый проект Apache Software Foundation (ASF), основной целью которого является создание масштабируемых алгоритмов машинного обучения, которые предлагаются для бесплатного использования по лицензии Apache. Mahout содержит реализации кластеризации, категоризации, CF и эволюционного программирования. Более того, там, где это разумно, он использует библиотеку Apache Hadoop, что позволяет Mahout эффективно масштабироваться в облаке.

## Заключение

В данной главе были описаны алгоритмы кластеризации и показано их сравнение.

Из рассмотренных в первой главе алгоритмов был выбран алгоритм минимального покрывающего дерева, так как он имеет произвольную форму кластера, а в качестве критерия схожести используется мера TF-IDF, так как этот алгоритм был предложен в задании к выпускной работе.

В третьей главе будет более подробно описан алгоритм кластеризации с помощью минимального покрывающего дерева.

# Глава 2. Выбор и обоснование алгоритма

*Содержательная постановка задачи*

Имеется фиксированное множество документов. Требуется разбить заданное множество документов на группы в соответствии с их семантической схожестью.

Предусмотреть 2 режима работы:

1. автоматическое выявление групп документов
2. выбор одной группы документа (из заранее определенных групп) на основании его содержания.

В качестве критериев схожести документов при кластеризации используется мера TF-IDF.

**Обозначения и формальная постановка задачи**

Обозначения:

*D* – исходное множество документов коллекции,

*di* ∈ *D* – *i*-й документ множества *D*,

*tj* ∈ *di* – *j*-е слово документа *di*,

 – число вхождений *tj*-го слова в документ *di*,

 – общее число слов в документе *di*, 

 – частота *tj*-го слова (term frequency) в документе *di*,

 – обратная частота документа (inverse document frequency) – инверсия частоты, с которой *tj*-ое слово встречается в документах коллекции

– число документов *D*, в которых встречается слово *tj* (когда )

 – булева переменная, равная 1 – если *di*-й документ относится к таксону *pk* и 0 – в противном случае,

 – оценка важности слова в контексте документа (мера TF-IDF)

- евклидово расстояние между *i*-м и *l*-м документами на основании меры TF-IDF



**Цель** – суммарное расстояния между документами, попавшими в один таксон, должно быть минимальным.

# Глава 3. ПРОГРАММНЫЙ КОМПЛЕКС

## 3.1. Архитектура программного обеспечения

Разработанное программное средство реализовано на языке Java, что в свою очередь позволяет запускать данный ПП под управлением любой ОС, на которой заранее установлена соответствующая виртуальная машина.

В качестве среды для разработки программного продукта выбрана интегрированная среда разработки IntelliJ IDEA. Данная среда зарекомендовала себя, как надежный инструмент разработки, и имеет следующие преимущества:

1. Поддержка множества языков программирования;
2. Кроссплатформенность;
3. Наличие бесплатной версии (Community);
4. Удобный инструмент отладки;
5. Поддержка системы контроля версиями;

Из минусов можно выделить разве что высокое потребление ресурсов, но при текущих мощностях современных ЭВМ данный пункт нивелируется.

## 3.2. Язык программирования Java

Язык программирования в значительной степени влияет на качество и надежность программного обеспечения. Для реализации поставленной задачи был выбран язык программирования Java по следующим причинам:

1. Объектно-ориентированный подход к разработке. Это означает, что у разработчика появляется возможность описывать абстрактные конструкции на основе предметной области, а потом реализовывать между ними взаимодействие;
2. Большое количество готовых библиотек, ускоряющих разработку;
3. Удобная среда разработки (IntelliJ IDEA);
4. Безопасный код;
5. Кроссплатформенность.

Java — объектно-ориентированный язык программирования, разрабатываемый компанией Sun Microsystems и официально выпущенный 23 мая 1995 года.

## 3.3. Работа программного комплекса

В программе предусмотрено 2 режима работы:

* 1. Кластеризация
  2. Классификация

Далее будут продемонстрированы оба режима работы.

### 3.3.1. Кластеризация

Изначально, при запуске программы, бывает выбрал режим работы кластеризации, как это показано на рисунке 3.1.

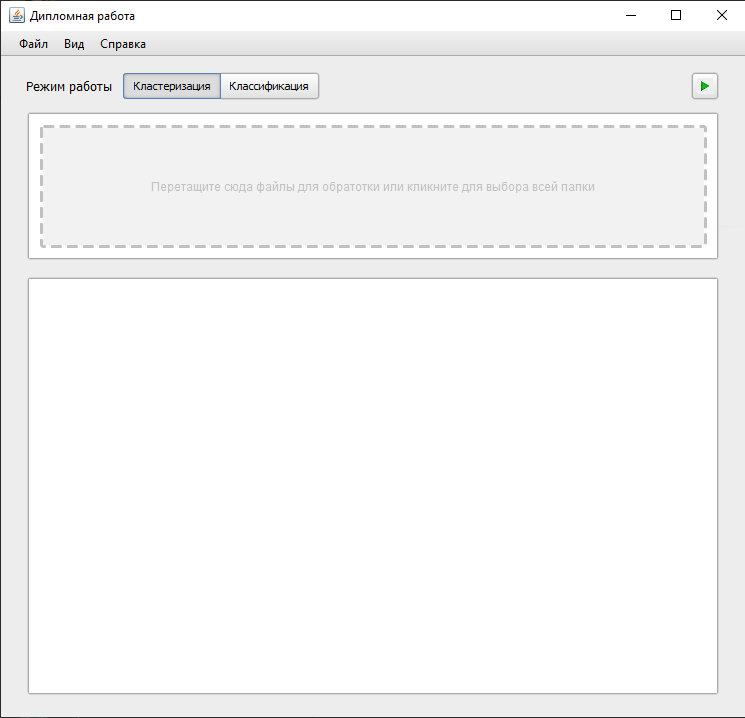
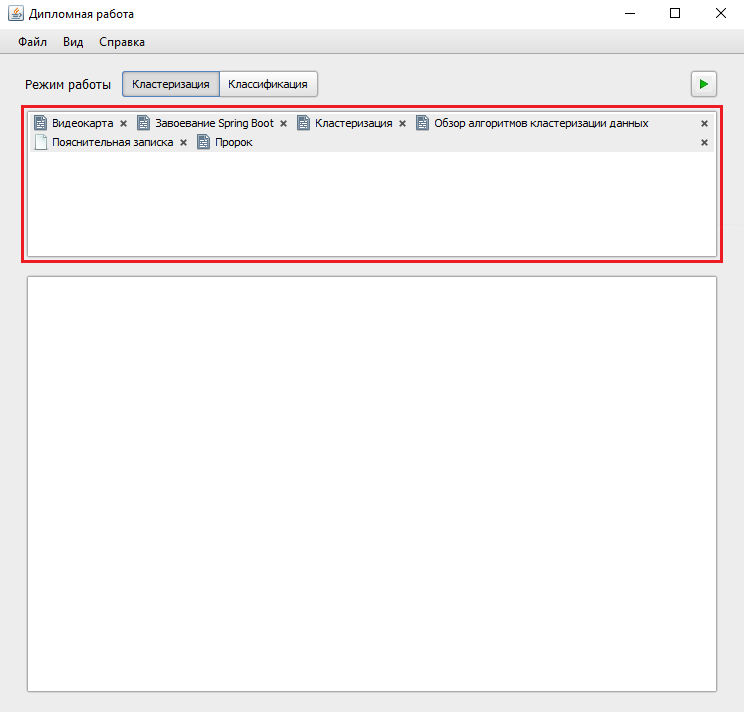


Рисунок 3.1.

Далее необходимо перетащить нужные файлы в соответствующую область или один раз кликнуть по той же области, для того чтобы в диалоговом окне выбрать папку, в которой мы хотим обработать файлы.

  
Рисунок 3.2.

Далее нужно запустить программу.

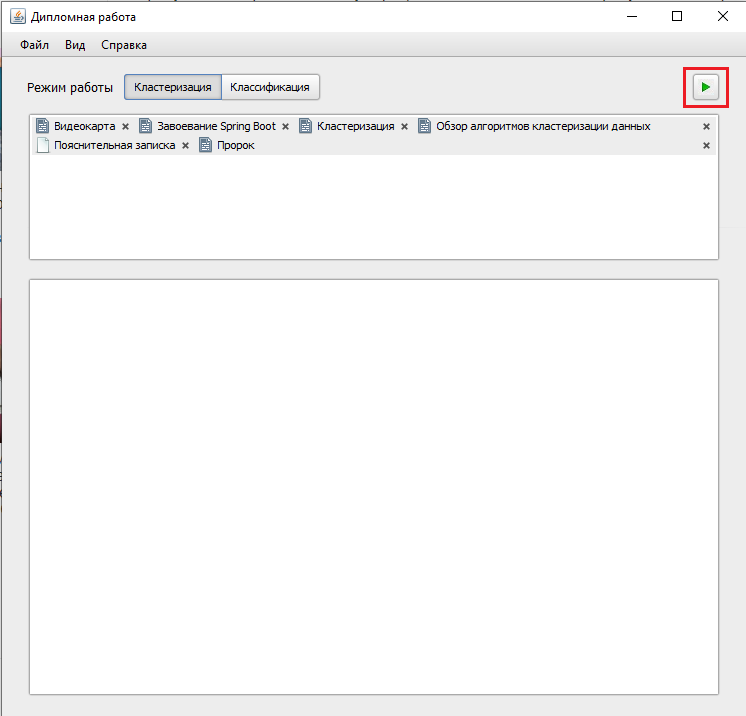


Рисунок 3.3.

И в текстовом поле появится результат работы

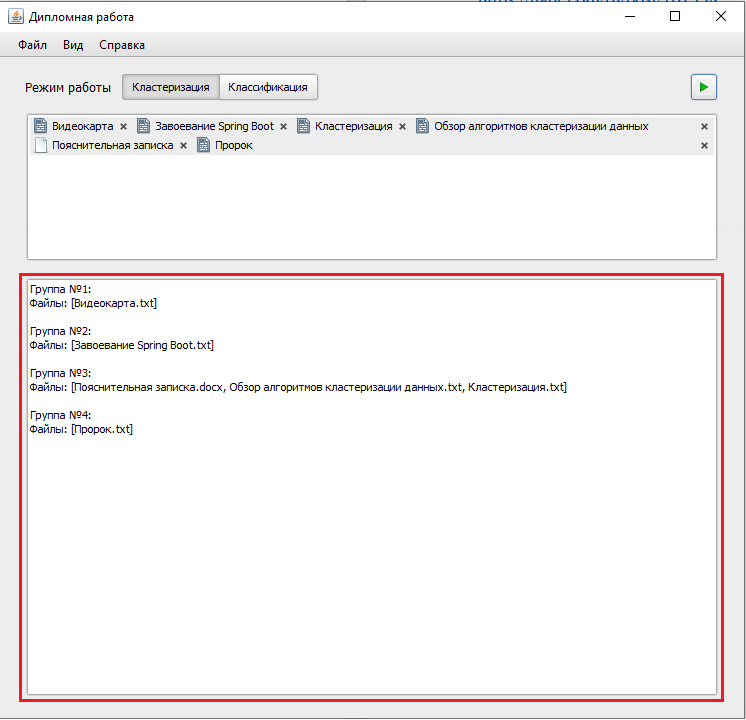
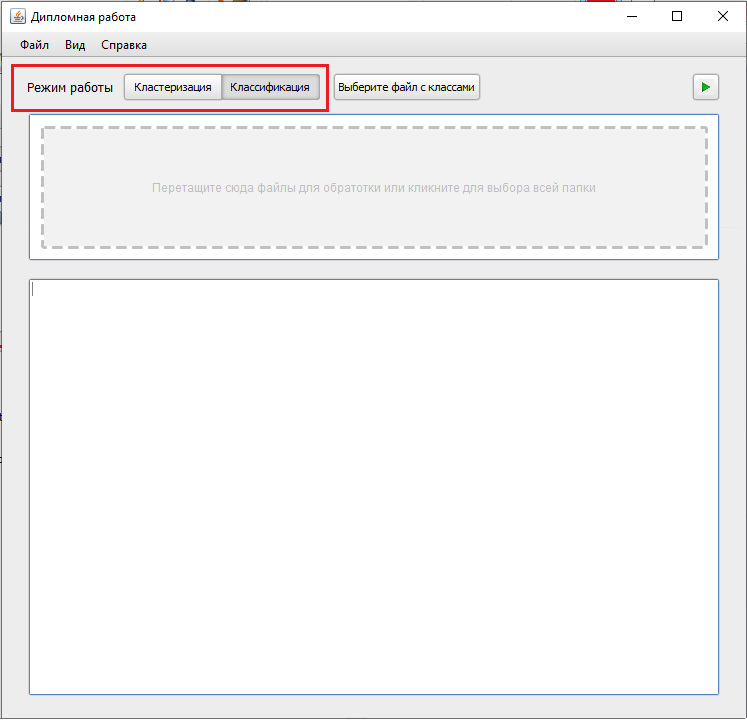


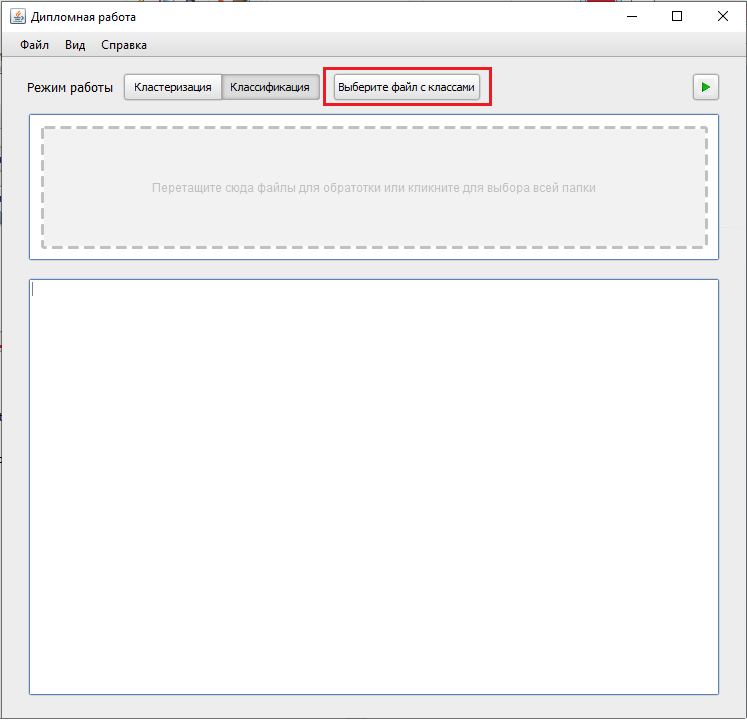
Рисунок 3.4.

### 3.3.2. Классификация

Для того, чтобы программа начала работать в режиме классификации необходимо выбрать соответствующий режим:

  
Рисунок 3.5.

Далее нужно выбрать файл с заранее заготовленными классами

  
Рисунок 3.6.

Далее необходимо перетащить нужные файлы в соответствующую область или один раз кликнуть по той же области, для того чтобы в диалоговом окне выбрать папку, в которой мы хотим обработать файлы.

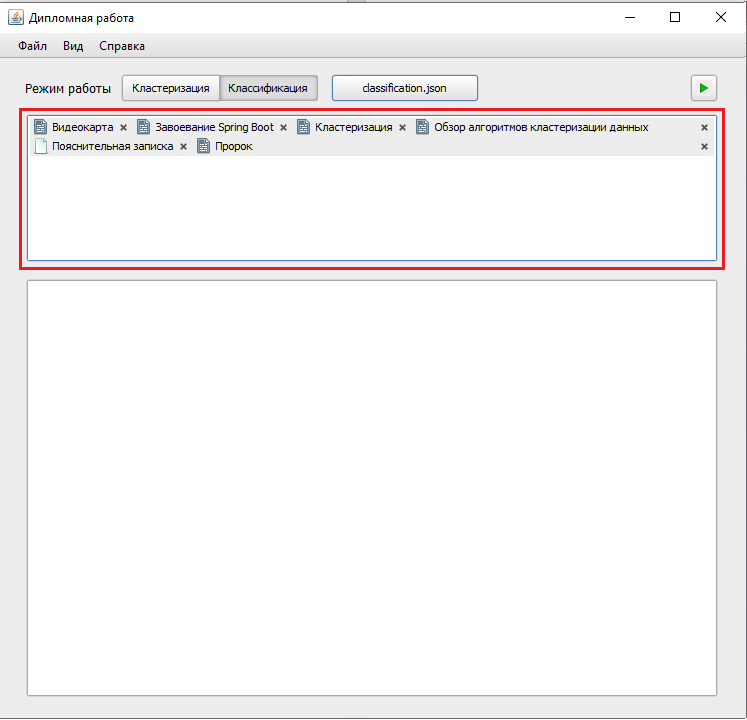
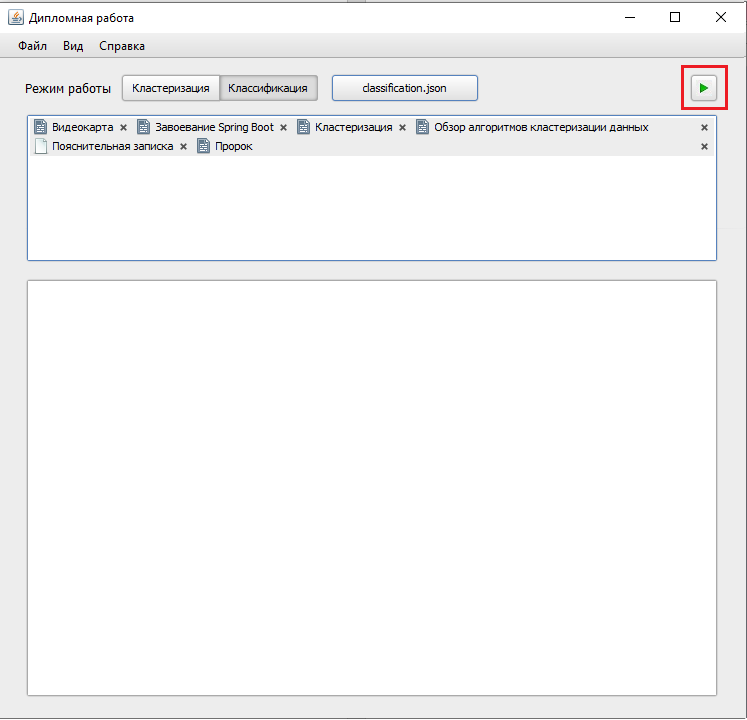


Рисунок 3.7.

И, наконец, запустить программу

  
Рисунок 3.8.

И в текстовое поле выведется результат работы программы

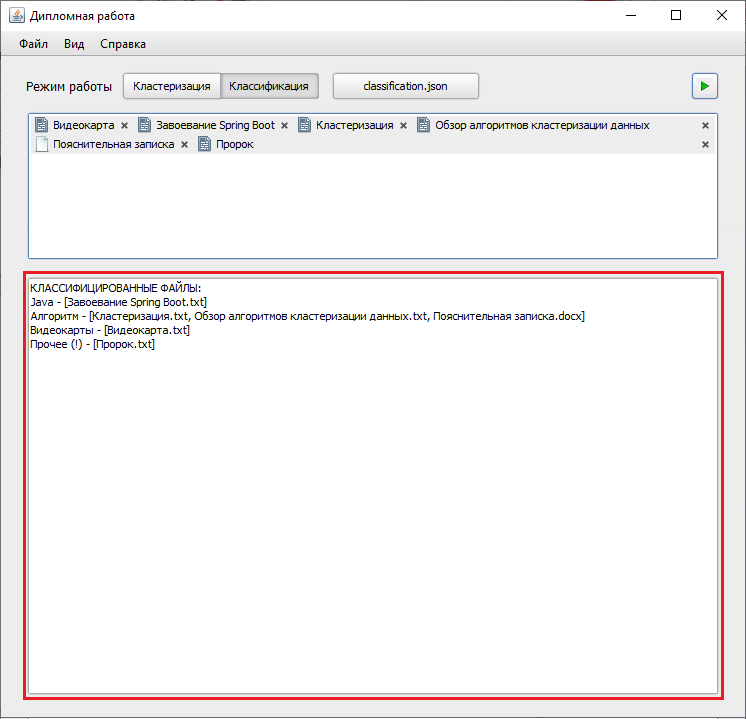
****

Рисунок 3.9.

# Глава 4. Экспериментальная часть

Для проведения анализа была проведена серия экспериментов.

Для проведения экспериментов использовался ПК со следующими характеристиками:

|  |  |
| --- | --- |
| Производитель процессора | AMD |
| Тип процессора | Ryzen 5 3600 |
| Количество ядер процессора | 6 |
| Баз. такт. частота | 3.59 GHz |
| Кэш память | 32 МБ |
| Оперативная память (RAM) | 16 Гб |
| Тип памяти | DDR4 |
| Производитель видеокарты | NVIDIA |
| Графический контроллер | GeForce RTX 2060 |
| Жесткий диск (SSD) | 256 ГБ |
| ОС | Windows 10 PRO (64 bit) |

Таблица 4.1. Характеристики ПК

Главной исследуемой характеристикой работы программного комплекса является время его работы.

## 4.1. Оценка производительности алгоритма кластеризации

### 4.1.1. Зависимость от количества слов в файле

**Цели эксперимента:**

* Построить графические зависимости, отражающие среднее время кластеризации в зависимости от количества слов в файле;
* Определить аналитическую зависимость, отражающие среднее время кластеризации в зависимости от количества слов в файле.

**Ход эксперимента:**

Разместить N файлов, с количеством слов в диапазоне 5000 ≤ N ≤ 35000 c шагом 5000. При каждом фиксированном N алгоритм отрабатывает 100 раз.

|  |  |
| --- | --- |
| Кол-во слов | Время работы (мс) |
| 5000 | 2195,72 |
| 10000 | 2735,15 |
| 15000 | 3108,07 |
| 20000 | 3283,41 |
| 25000 | 3509,29 |
| 30000 | 3813,6 |

Таблица 4.2. Время работы кластеризации при фиксированном кол-ве файлов

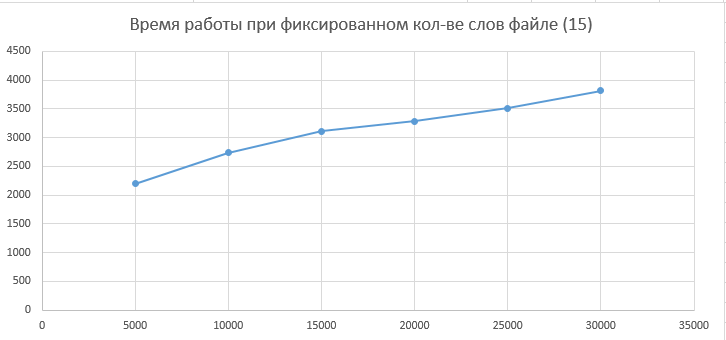


График 4.1. Время работы кластеризации при фиксированном кол-ве файлов

### 4.1.2. Зависимость от количества файлов

**Цели эксперимента:**

* Построить графические зависимости, отражающие среднее время кластеризации в зависимости от количества файлов;
* Определить аналитическую зависимость, отражающие среднее время кластеризации в зависимости от количества файлов.

**Ход эксперимента:**

Разместить N файлов в диапазоне 5 ≤ N ≤ 20 c шагом 5. При каждом фиксированном N алгоритм отрабатывает 100 раз.

|  |  |
| --- | --- |
| Кол-во файлов | Время работы (мс) |
| 5 | 426,88 |
| 10 | 1309,14 |
| 15 | 3031,75 |
| 20 | 5926,5 |

Таблица 4.3. Время работы кластеризации при фиксированном кол-ве файлов



График 4.2. Время работы кластеризации при фиксированном кол-ве файлов

## 4.2. Оценка производительности алгоритма классификации

### 4.2.1. Зависимость от количества слов в файле

**Цели эксперимента:**

* Построить графические зависимости, отражающие среднее время классификации в зависимости от количества слов в файле;
* Определить аналитическую зависимость, отражающие среднее время классификации в зависимости от количества слов в файле.

**Ход эксперимента:**

Разместить N файлов, с количеством слов в диапазоне 5000 ≤ N ≤ 35000 c шагом 5000. При каждом фиксированном N алгоритм отрабатывает 100 раз.

|  |  |
| --- | --- |
| Кол-во слов | Время работы (мс) |
| 5000 | 298,92 |
| 10000 | 576,34 |
| 15000 | 854,73 |
| 20000 | 1166,2 |
| 25000 | 1422,53 |
| 30000 | 1690,57 |

Таблица 4.4. Время классификации в зависимости от кол-ва слов

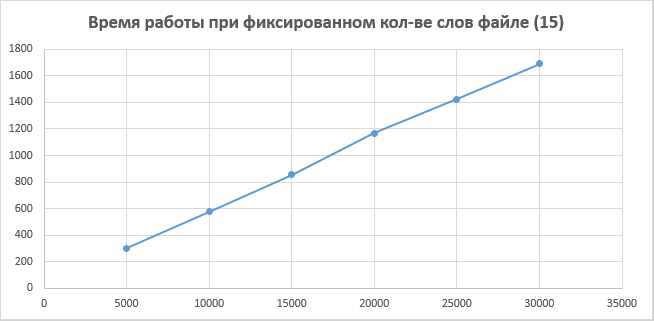


График 4.3. Время классификации в зависимости от кол-ва слов

### 4.2.2. Зависимость от количества файлов

**Цели эксперимента:**

* Построить графические зависимости, отражающие среднее время классификации в зависимости от количества файлов;
* Определить аналитическую зависимость, отражающие среднее время классификации в зависимости от количества файлов.

**Ход эксперимента:**

Разместить N файлов в диапазоне 5 ≤ N ≤ 20 c шагом 5. При каждом фиксированном N алгоритм отрабатывает 100 раз.

|  |  |
| --- | --- |
| Кол-во файлов | Время работы (мс) |
| 5 | 287,19 |
| 10 | 568,7 |
| 15 | 855,5 |
| 20 | 1139,44 |

Таблица 4.5. Время классификации в зависимости от кол-ва файлов

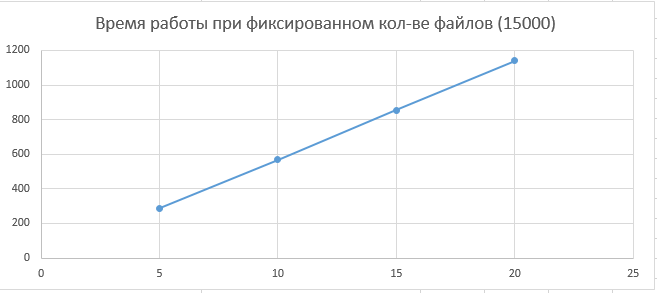


График 4.4. Время классификации в зависимости от кол-ва файлов

### 4.2.3. Зависимость от количества классов

**Цели эксперимента:**

* Построить графические зависимости, отражающие среднее время классификации в зависимости от количества классов;
* Определить аналитическую зависимость, отражающие среднее время классификации в зависимости от количества классов.

**Ход эксперимента:**

Разместить 10 файлов с 15000 слов в каждом и разбить на N классов, в диапазоне 5 ≤ N ≤ 10 c шагом 1. При каждом фиксированном N алгоритм отрабатывает 100 раз.

|  |  |
| --- | --- |
| Кол-во классов | Время работы (мс) |
| 5 | 569,83 |
| 6 | 571,18 |
| 7 | 564,03 |
| 8 | 575,43 |
| 9 | 579,3 |
| 10 | 577,84 |

Таблица 4.6. Время классификации в зависимости от кол-ва классов



График 4.5. Время классификации в зависимости от кол-ва классов

# ГЛАВА 5. Заключение

Данная выпускная работа посвящена программной реализации программного обеспечения, предназначенного для кластеризации и классификации документов.

Поставленную цель считаю достигнутой, а задачи выполненными.

# Список использованных источников

1. Гитис Л. Х*.* Кластерный анализ в задачах классификации, оптимизации и прогнозирования. 104 с.
2. Б. Дюран*.* Кластерный анализ. 128 с.
3. <https://habr.com/ru/post/101338>

# Приложение 1. исходный код программы