

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Кубанский государственный технологический университет»
(ФГБОУ ВО «КубГТУ»)
Институт компьютерных систем и информационной безопасности
Кафедра информационных систем и программирования

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по направлению: 09.03.04 Программная инженерия


(код и наименование направления)


на тему: Разработка web-приложения «Литература». Разработка алгоритма кластеризации данных. Комплексный проект.

(наименование темы)

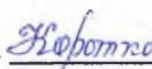
ИКСИБ.КИСП.09.03.04.014.ПП


(обозначение документа)

Автор  16.06.2020 / Н.О. Шелудько /
(подпись, дата, расшифровка подписи)

Руководитель  18.06.2020 / д-р техн. наук, проф. В.Н. Марков /
(подпись, дата, расшифровка подписи)

Консультанты:

Раздел безопасности и
экологичности проекта  20.06.2020 / д-р техн. наук., проф. Т.Г. Короткова /
(подпись, дата, расшифровка подписи)

Нормоконтролер  22.06.2020 / ст. преп. А.А. Ковтун /
(подпись, дата, расшифровка подписи)

Выпускная квалификационная работа
допущена к защите

23.06.2020
(дата)

Заведующая кафедрой  / канд. техн. наук, доц. М.В. Янаева /
(подпись)

Краснодар
2020

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Кубанский государственный технологический университет»
(ФГБОУ ВО «КубГТУ»)


Институт компьютерных систем и информационной безопасности
Кафедра информационных систем и программирования

УТВЕРЖДАЮ

Заведующая кафедрой ИСП

канд. техн. наук, доц.

 М.В. Янаева

« 14 »  мая 2020 г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

по направлению 09.03.04 Программная инженерия
(шифр и наименование)

студенту Шелудько Никите Олеговичу
(фамилия, имя, отчество)

Тема выпускной квалификационной работы: Разработка web-приложения «Литература». Разработка алгоритма кластеризации данных. Комплексный проект.

Утверждена приказом ректора университета от 13.05.2020 г. № 650-Ст

Руководитель д-р техн. наук, проф. В.Н. Марков
(должность, фамилия, инициалы)

Консультанты:

Раздел безопасности и
экологичности проекта д-р техн. наук, проф. Т.Г. Короткова
(должность, фамилия, инициалы)

Нормоконтролер ст. преп. А.А. Ковтун
(должность, фамилия, инициалы)

Срок сдачи выпускной квалификационной работы на кафедру 23.06.2020 г.

Содержание выпускной квалификационной работы

Введение

1 Нормативные ссылки

2 Термины, определения и сокращения

3 Анализ предметной области

4 Постановка задачи и техническое задание

5 Обзор используемых технологий и сред разработки

6 Кластеризация

7 Обработка текстовых данных

8 Программная реализация

9 Руководство пользователя

10 Безопасность и экологичность проекта

Заключение

Список использованных источников

Общее количество листов ПЗ 65

Объем иллюстративной части

1. Основная концепция проекта.

2. Кластеризация.

3. Обработка текстовых данных.

4. Использование системы.

Общее количество слайдов иллюстративной части 15

Список основной и рекомендуемой литературы

1. Чамберс Дж., Пэкетт Д. Тиммс С. ASP.NET Core. Разработка приложений – Пер. с англ. – СПб : Питер, 2018 – 464 с.

Календарный план выполнения выпускной квалификационной работы

Месяц	Числа месяца																														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Март																Описание требований к ПП и постановка задачи ВКР							Анализ предметной области ВКР. Техническое задание на ПП								
Апрель	Анализ предметной области ВКР. Техническое задание на ПП					Обзор технологий и средств программирования						Построение модели ПП. Проектирование и реализация алгоритмов кластеризации данных																			
Май	Реализация ПО																						Безопасность и экологичность проекта								
Июнь	Оформление пояснительной записки															Нормоконтроль, антиплагиат. Сдача ВКР на кафедру							Защита ВКР								

Студент Шелудько Н.О. 16.03.20
(подпись, дата)

Руководитель Марков В.Н. 16.03.20 д-р техн. наук, проф. В.Н. Марков
(подпись, дата)

Реферат

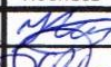


Выпускная квалификационная работа содержит 65 страниц, 18 рисунков, 3 таблицы, 7 формул, 14 листингов, 7 используемых источников.

ЯЗЫК С#, ASP.NET Core, MVC, HTML, CSS, ML.NET, АЛГОРИТМ, КЛАСТЕРИЗАЦИЯ, ТОКЕНИЗАЦИЯ, СТЕММИНГ, ВЕКТОРИЗАЦИЯ, ВЕБ-САЙТ, МЕТРИКА, МАШИННОЕ ОБУЧЕНИЕ, K-MEANS.

Объектом исследования является алгоритм кластеризации, способный работать с наборами текстовых данных, в роли которых выступают различные произведения русской литературы.

Цель выпускной квалификационной работы состоит в разработке, тестировании и имплементации алгоритмов кластеризации данных, способных работать с текстовыми наборами информации на веб-сайте «Литература».

К полученным результатам относятся: сгруппированные наборы тестовых данных с разнообразным количеством книг и авторов, а также приложение, полностью оптимизированное для работы на сайте «Литература», способное обеспечить кластеризацию данных с достаточно высокой точностью и предоставить итоговые результаты в удобном для восприятия виде.

					ИКСИБ.КИСП.09.03.04.014.ПП			
Изм.	Лист	№ докум.	Подпись	Дата	Пояснительная записка выпускной квалификационной работы	Лит.	Лист	Листов
Разраб.		Шелудько Н.О.		16.06.20				
Провер.		Марков В.Н.		18.06.20			5	65
Консульт.		Короткова Т.Г.		22.06.20		КуДГУ		
Н. Конт		Ковтун А.А.		22.06.20				
Утверд.		Янаева М.В.		23.06.20				

Содержание

Введение.....	8
1 Нормативные ссылки.....	9
2 Термины определения и сокращения.....	10
3 Анализ предметной области	12
4 Постановка задачи и техническое задание.....	14
4.1 Общие требования.....	14
4.2 Платформа.....	15
4.3 Пользовательский интерфейс	15
4.4 Книги	16
4.5 Кластеризация	17
5 Обзор используемых технологий и средств разработки.....	18
5.1 Язык программирования С#.....	18
5.2 Visual Studio 2019	20
5.3 ASP.NET Core	21
5.4 MS SQL Server	22
5.5 ML.NET	23
6 Кластеризация	25
6.1 Определение кластеризации	25
6.2 Метрики сходства	27
6.4 Классификация алгоритмов	30
6.4.1 Иерархические алгоритмы	30
6.4.2 Плоские алгоритмы	34
7 Обработка текстовых данных	38
7.1 Нежелательные символы.....	38
7.2 Токенизация	39
7.3 Стемминг	39
7.4 Векторизация	40
8 Программная реализация	42
8.1 Реализация алгоритма k-means	42
8.2 Применение библиотеки ML.NET.....	48
9 Руководство пользователя.....	50

10 Оценка эффективности.....	55
11 Безопасность и экологичность проекта	57
11.1 Значение и задачи безопасности жизнедеятельности	57
11.2 Анализ условий труда и мероприятия по защите от воздействия вредных производственных факторов	58
11.3 Обеспечение электробезопасности	61
11.4 Пожарная безопасность	62
Заключение	64
Список используемой литературы	65

Введение

На сегодняшний день информационные технологии активно растут и развиваются. Сеть Интернет достигла невероятной популярности и используется практически во всех сферах человеческой деятельности. Ежедневно люди читают, изменяют, обрабатывают, загружают и удаляют миллионы различной информации. Данные, циркулирующие по сети Интернет могут быть представлены в самой разнообразной форме, это могут быть видеоролики, аудиофайлы и изображения. Однако текстовые данные всегда были в большинстве. За сотни лет существования человечество создало огромное количество текстового материала. Такая тенденция прослеживается и сегодня – в Интернете было создано множество форумов, статей, публикаций и электронных книг. Из-за всего этого появилась необходимость в средствах фильтрации и группировки огромных массивов текстовых данных.

Именно по этой причине целью данной выпускной квалификационной работы является создание приложения, способного обеспечить кластеризацию наборов текстовых данных на сайте-библиотеке. Основным языком приложения является язык программирования C#, а также фреймворк ASP.NET Core. Использование объектно-ориентированного подхода к разработке позволяет в полной мере раскрыть возможности алгоритмов кластеризации с использованием различных модификаций и метрик расстояния. Кроме того, приложение будет обладать красивым и надёжным интерфейсом, созданным с использованием языка разметки HTML и таблиц стилей CSS.

Выполнение поставленных задач поспособствует приобретению дополнительных практических навыков в области программирования, разработки веб-технологий, алгоритмов анализа и обработки данных, а также организации удобного и интуитивно понятного пользовательского интерфейса.

1 Нормативные ссылки

В настоящей выпускной квалификационной работе использованы ссылки на следующие нормативные документы:

1. ГОСТ Р 1.12-2004. Стандартизация в Российской Федерации;
2. ГОСТ 20886-85 Организация данных в системах обработки данных.

Термины и определения.

3. ГОСТ 34.321-96 Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными.

4. ГОСТ 15971-90 Системы обработки информации. Термины и определения.

5. СанПиН 2.2.2/2.4.1340 03 Гигиенические требования к персональным электронно-вычислительным машинам и организации работы.

6. Трудовой кодекс Российской Федерации от 30.12.2001 № 197-ФЗ (ред. от 10.01.2016).

7. ТОИ Р 45 084 01 Типовая инструкция по охране труда при работе на персональном компьютере.

8. ГОСТ 12.0.003 2015 Система стандартов безопасности труда. Опасные и вредные производственные факторы. Классификация.

9. ГОСТ Р 2.2.2006 05 Руководство по гигиенической оценке факторов рабочей среды и трудового процесса. Критерии и классификация условий труда.

10. ГОСТ 12.1.019-2017 Электробезопасность.

2 Термины определения и сокращения

В настоящей выпускной квалификационной работе применяются термины с соответствующими определениями и сокращениями, установленные следующими нормативными документами:

- ГОСТ Р 1.12-2004. Стандартизация в Российской Федерации;
 - ГОСТ 34.321-96 Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными;
 - ГОСТ 15971-90 Системы обработки информации. Термины и определения;
 - ГОСТ 20886-85 Организация данных в системах обработки данных.
- Термины, определения и сокращения:

1 Язык программирования (ЯП) – формальный язык, предназначенный для записи компьютерных программ.

2 Электронно-вычислительная машина (ЭВМ) – комплекс технических, аппаратных и программных средств, предназначенных для автоматической обработки информации, вычислений, автоматического управления. При этом основные функциональные элементы (логические, запоминающие, индикационные и др.) выполнены на электронных элементах.

3 Персональная электронно-вычислительная машина (ПЭВМ) – настольная микро-ЭВМ, имеющая эксплуатационные характеристики бытового прибора и универсальные функциональные возможности.

4 База данных (БД) – совокупность взаимосвязанных данных, организованных в соответствии со схемой базы данных таким образом, чтобы с ними мог работать пользователь.

5 Система управления базами данных (СУБД) – совокупность программных и языковых средств, обеспечивающих управление базами данных.

6 Structured Query Language (SQL) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

7 Integrated Development Environment (IDE) – система программных средств, используемая программистами для разработки программного обеспечения.

8 Фреймворк – заготовки, шаблоны для программной платформы, определяющие архитектуру программной системы; программное обеспечение, облегчающее разработку и объединение разных модулей программного проекта.

9 Машинное обучение (machine learning) — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач.

10 Метрика сходства (также мера сходства, индекс сходства) — безразмерный показатель сходства сравниваемых объектов. Также известен под названиями «мера ассоциации», «мера подобия» и др.

3 Анализ предметной области

Сайт «Литература» – это электронная библиотека, интернет ресурс, обладающий обширной, постоянно обновляющейся, коллекцией электронных документов со средствами поиска и навигации. Каждый электронный документ представляет собой реально существующее художественное произведение русской литературы – книгу. Чтобы предоставить читателю избыточную информацию о том или ином произведении, каждая книга обладает определённым набором параметров:

- полное имя автора;
- название;
- год издания;
- обложка;
- жанр.

Данные характеристики в той или иной мере применяются для поиска книг на сайте. Особое внимание уделяется параметру жанр, потому что у каждого произведения может быть сразу несколько жанров.

Иногда возникают ситуации, когда сложно определить жанры или автора при загрузке той или иной книги. В результате некоторые характеристики остаются незаполненными и значительно ухудшают поиск материалов в электронной библиотеке. Поэтому, как и в любом другом хранилище данных, вся информация, хранящаяся на сайте «Литература» регулярно группируется, анализируется и дополняется.

Группировка данных производится по тем или иным признакам, характеризующим их близость. Алгоритмы, обрабатывающие информацию подобным образом, называются классификацией.

Классификация обладает высокой точностью и осуществляет разбиение данных по заранее известным классам. Когда речь идёт о библиотеке в качестве классов могут выступать литературные жанры и

авторы. Такое разбиение является наиболее надёжным, однако его не всегда можно использовать. Если требуется обработать набор книг, жанры или авторы которых не известны определить изначальные классы, по которым будет осуществляться группировка данных, будет невозможно.

Именно поэтому целью данной выпускной квалификационной работы является разработка алгоритмов, способных оценивать характеристики больших наборов текстовых данных и разбивать их на группы даже без информации о жанрах и авторах. Такой процесс называется кластеризацией. Все алгоритмы, относящиеся к этой категории не требуют предварительного обучения на многочисленных наборах тестовых данных для корректной работы. На сегодняшний день можно выделить два основных подвида кластеризации:

- иерархические алгоритмы – формируют иерархическое дерево, состоящее из подгрупп объектов (кластеров).

- плоские – обладают высокой скоростью и точность, но требуют ввода изначального количества кластеров.

Кроме того, кластеризацию также можно разделить на чёткую и нечёткую:

- чёткая – каждый объект состоит ровно в одном кластере;

- нечёткая – каждый объект может состоять сразу в нескольких кластерах.

4 Постановка задачи и техническое задание

Для того, чтобы правильно обозначить основные цели того или иного программного продукта прежде всего необходимо сформулировать постановку задачи. Постановка задачи позволяет оценить входные и выходные данные, с которыми работает программа, а также содержит техническое задание.

Техническое задание описывает все требования, предъявляемые, к программному продукту, платформе и технологиям разработки, рабочему интерфейсу пользователя, алгоритмам кластеризации, а также объектам обрабатываемых данных.

4.1 Общие требования

Программное обеспечение предназначено для кластеризации данных, хранимых на сайте «Литература», и должно соответствовать следующим требованиям:

- для написания основной логики приложения должен использоваться объектно-ориентированный язык программирования C#;
- доступ к алгоритмам кластеризации должен осуществляться через сайт «Литература»;
- разрабатываемый алгоритм кластеризации должен корректно обрабатывать любые текстовые данные;
- любые текстовые файлы, загружаемые в библиотеку на сайте должны проходить через набор алгоритмов преобразования;
- доступ к алгоритмам кластеризации должен быть ограничен для пользователей, не являющихся администраторами сайта;

4.2 Платформа

Программный продукт должен быть разработан с использованием ASP.NET Core – удобного и надёжного инструментария для разработки веб-приложений и сервисов. Текстовые данные обрабатываемые алгоритмами кластеризации должны храниться в реляционной базе данных MS SQL Server. Кроме того, в проекте используются следующие технологии:

- объектно-ориентированный язык программирования C# 8.0.0;
- Visual Studio 2019;
- ML.NET 1.5.0;
- Entity Framework 3.1.4;
- Identity 2.2.0;
- Razor Pages 3.2.7;
- язык гипертекстовой разметки HTML 5.2.0;
- каскадные таблицы стилей CSS;
- мультипарадигменный язык программирования Java Script;
- bootstrap 4.5.0;

4.3 Пользовательский интерфейс

Кластеризация данных на сайте «Литература» доступна только для администраторов.

Для того, чтобы наглядно обозначить основные возможности пользовательского интерфейса алгоритма кластеризации была использована UML диаграмма вариантов использования. Данная диаграмма состоит из перечня операций, доступных пользователю при работе с приложением, каждая из которых заключена в эллипс, а также самого пользователя, изображённого в виде нарисованного человечка (рис. 4.1).

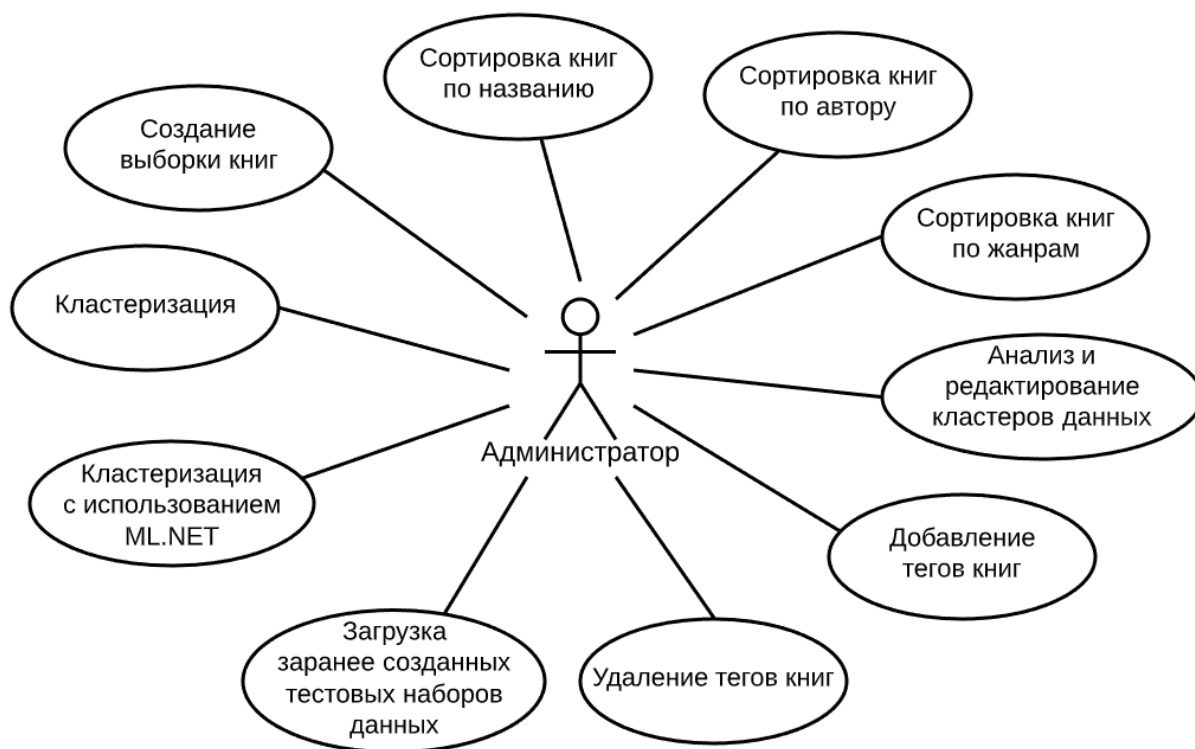


Рисунок 4.1 – Диаграмма вариантов использования

Кроме того, интерфейс должен соответствовать следующим требованиям:

- интерфейс должен быть простым и понятным;
- интерфейс должен корректно взаимодействовать с базой данных;
- списки книг должны корректно отображаться на странице сайта пользователя, в понятном для него виде;
- интерфейс должен гармонично сочетаться с внешним видом сайта.

4.4 Книги

Каждая единица текстовых данных, представленных на сайте «Литература», является книгой. Чтобы произвести кластеризацию и анализ книг администратору необходимо получить информацию о тех или иных характеристиках исследуемых литературных произведений. Кроме того, для корректной работы алгоритмов кластеризации, каждая книга должна

проходить первичную обработку. Таким образом на каждую книгу накладываются следующие требования:

- книга должна содержать название, автора и жанры;
- книга должна содержать основной и обработанный текст;
- обработанный текст книги не должен содержать нежелательные символы;
- обработанный текст должен пройти процедуру токенизации и стемминга;
- обработанный текст должен быть преобразован в векторную форму при помощи «Bag of words» или TF-IDF.

4.5 Кластеризация

Алгоритмы кластеризации используются для проведения анализа книжной коллекции путём разбиения её на группы. Алгоритмы кластеризации, применяемые в данном проекте должны соответствовать следующим требованиям:

- алгоритм кластеризации данных должен обладать высокой точностью и скоростью выполнения;
- алгоритм кластеризации должен быть плоским;
- по итогам выполнения алгоритма каждая книга должна принадлежать только к одному кластеру;
- алгоритм кластеризации данных должен быть приспособлен к работе с разнообразными метриками расстояния, такими как Евклидова величина, мера косинуса угла и т.д.
- алгоритм кластеризации должен обладать методами поиска оптимального количества кластеров, такими как метод локтя и метод силуэта;
- код, описывающий алгоритм кластеризации должен соответствовать основным положениям объектно-ориентированного программирования и его паттернам.

5 Обзор используемых технологий и средств разработки

Перед разработкой любого ПО необходимо определиться с выбором технологий и средств разработки. Исходя из постановки задачи и технического задания в качестве основного языка программирования был выбран C# и среда разработки Visual Studio 2019. Для работы с веб-технологиями был использован фреймворк ASP.NET Core, а также технологии Razor Pages, Entity Framework и Identity. В качестве базы данных для хранения обрабатываемых наборов книг был выбран MS SQL Server. Для работы с алгоритмами машинного обучения была использована библиотека ML.NET.

5.1 Язык программирования C#

C# – это объектно-ориентированный высокоуровневый язык программирования с широким спектром возможностей и задач. «Синтаксис C# очень богат, но при этом прост и удобен в изучении. Характерные фигурные скобки C# мгновенно узнаются всеми, кто знаком с C, C++ или Java». [4] Преимуществом C# является обширный набор инструментов, предоставляемых фреймворком .NET, который постоянно поддерживается и обновляется компанией Microsoft. Язык. Данный язык применяется для:

- веб-программирования;
- создания многофункциональных десктопных приложений;
- тестирования и отладки;
- написания скриптов;
- разработки игр с использованием среды Unity;

C# обладает простым и понятным синтаксисом. Благодаря объектно-ориентированному подходу структура программного кода представляет собой совокупность различных объектов данных, называемых классами, и взаимодействий между ними. Весь код написанный, на языке C# является

управляемым, т.е. при компиляции приложения он преобразуется в общий язык CIL, позволяющий значительно снизить затраты процессорной памяти компьютера. Остальные особенности ЯП C#:

- каждый объект данных является классом;
- есть интерфейсы и абстрактные классы;
- статические методы и классы, не требующие создания объектов;
- инкапсуляция, полиморфизм и наследование;
- перегрузка методов и конструкторов класса;
- JIT-компиляция, повышающая производительность;
- сборщик мусора или деструктор, позволяющий очищать память от всех, не используемых, данных;
- строгая типизация данных;
- единая система типов, благодаря которой абсолютно все типы данных можно свести к одному типу object;
- обработка исключений;
- типобезопасность, которая предостерегает пользователя от ошибок, связанных с чтением неинициализированных переменных;
- система управления версиями;
- NuGet пакеты, дающие доступ к обширной коллекции различных библиотек;
- IntelliSense, технология автодополнения, значительно упрощающая процесс написания кода

«C# является основным языком разработки программ на платформе .NET корпорации Microsoft. В нем удачно сочетаются испытанные средства программирования с самыми последними новшествами и предоставляется возможность для эффективного и очень практичного написания программ, предназначенных для вычислительной среды современных предприятий. Это, без сомнения, один из самых важных языков программирования XXI века». [1, 31]

5.2 Visual Studio 2019

Visual Studio 2019 – это IDE, включающая обширный спектр средств для работы с различными языками программирования. «Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки». [5] Существует три различных версии данной платформы:

- Visual Studio Community, бесплатная версия, предназначенная для школьников и студентов;
- Visual Studio Professional, версия для полноценных разработчиков или небольших команд;
- Visual Studio Professional, самая комплексная версия, обладающая средствами тестирования для больших и сложных командных проектов.

Во время выполнения данной выпускной квалификационной работы была использована версия Community. Несмотря на ограниченный функционал её более чем достаточно для разработки небольших веб-приложений. Несмотря на то, что язык C# считается основным при работе в Visual Studio, данная платформа обладает средствами для использования других объектно-ориентированных и функциональных языков, среди которых: Visual Basic, C++, а также F#.

Visual Studio содержит следующие средства для повышения производительности:

- рефакторинг, включающий изменение имени переменной во всех частях кода, преобразование строк кода в отдельные методы, изменение очередности параметров и много другое;
- набор функций, вносящий автокорректировки и исправления в написанный код;
- технология Live Share, позволяющая осуществлять совместное редактирование и отладку кода по сети Интернет;

- CodeLens, который помогает находить ссылки использования того или иного метода, класса или свойства;

- всплывающие окна, описывающие, варианты возможных действий.

На сегодняшний день существует множество других IDE со своими плюсами и недостатками, однако Visual Studio обладает рядом преимуществ, значительно выделяющих её на фоне конкурентов. К ним относятся:

- встроенный web-сервер. Для работы с веб-программированием в Visual Studio не требуется устанавливать какое-либо стороннее ПО. Заранее интегрированный web-сервер повышает безопасность приложения и позволяет работать, не входя из привычной среды проектирования;

- разнообразие поддерживаемых языков программирования позволяет писать код, не меняя среды разработки;

- большинство средств разработки, доступных в Visual Studio, содержат заранее сгенерированный код, позволяющий значительно сэкономить время на начальном этапе работы;

- интуитивный стиль кодирования. Visual Studio автоматически форматирует код по мере его написания. Различные цветовые обозначения и отступы делают любую программу гораздо более удобной для чтения и редактирования.

5.3 ASP.NET Core

ASP.NET Core – это платформа для разработки веб приложений, которая была разработана компанией Microsoft как полностью обновлённая версия более старого фреймворка ASP.NET. «Миллионы разработчиков используют или использовали ASP.NET 4.x для создания веб-приложений. ASP.NET Core – это редизайн ASP.NET 4.x, включающий архитектурные изменения, которые приводят к более компактной и модульной структуре». [6]
Данный фреймворк обладает рядом преимуществ:

- ASP.NET Core является свободно распространяемым фреймворком, это означает, что весь исходный код находится в открытом доступе на сайте GitHub;
- Razor Pages позволяет легко и просто осуществлять вставки кода C# на HTML страницы веб приложений;
- Blazor позволяет использовать C# на равне с JavaScript для создания красивого и динамичного интерфейса;
- кроссплатформенность позволяет разрабатывать приложения доступные на Windows, MacOS и Linux;
- модель разработки MVC, которая объединяет старые технологии Web Pages и Web API;
- модульность ASP.NET Core позволяет загружать сотни различных библиотек и компонентов через менеджер Nuget;
- тэг-хелперы (tag helper), предлагающие более удобный способ взаимодействия между контроллером и представлением;

5.4 MS SQL Server

MS SQL Server – это одна из наиболее развитых реляционных СУБД в мире, которая была разработана компанией Microsoft. Как и для любой другой реляционной базы данных SQL Server свойственны следующие аспекты:

- целостность данных, характеризующая точность единообразие и полноту всех данных;
- транзакции, неделимые действия, состоящие из множества небольших операций, написанных на реляционном языке;
- соответствие требованиям ACID, которые подразумевают, что хранимые данных являются атомарными, единообразными, изолированными и надёжными.

В MS SQL Server написания запросов используется декларативный язык T-SQL. «T-SQL — это диалект стандартного языка SQL, который, в свою

очередь, является одновременно стандартом Международной организации по стандартизации (International Organization for Standards, ISO) и Американского национального института стандартов (American National Standards Institute, ANSI)». [3, 10] Главным преимуществом SQL Server перед конкурентами является полная интеграция с фреймворком ASP.NET Core, а также Entity. Благодаря этому появляется возможность создавать объекты данных и отношения между ними, не отвлекаясь от программирования на языке C# в IDE Visual Studio. При этом уже созданная БД может быть с лёгкостью изменена при помощи миграций, которые позволяют сохранять состояния моделей данных подобно коммитам на веб-сервисе GitHub.

5.5 ML.NET

ML.NET – это библиотека, предоставляющая обширный функционал для проведения машинного обучения на данных. Данная технология позволяет находить закономерности в загруженных наборах данных для проведения дальнейшего анализа.

«В основе ML.NET лежит модель машинного обучения. Эта модель определяет шаги, которые необходимо выполнить для получения прогнозов на основе входных данных. С помощью ML.NET вы можете обучить пользовательскую модель, указав соответствующий алгоритм, а также импортировать предварительно обученные модели TensorFlow и ONNX.» [7]

ML.NET обладает высокой универсальностью, с данной библиотекой можно работать на Windows, MacOS и Linux. Данные также можно загружать из самых разнообразных источников, таких как файлы JSON, TXT, различные реляционные базы данных или коллекции объектов созданные в процессе выполнения кода.

ML.NET позволяет использовать широкий спектр алгоритмов прогнозирования, от нейронных сетей Кохонена до k-means++. При помощи этой технологии можно производить следующие прогнозы:

- классификация, разбивающая информация по классам;
- регрессия, благодаря которой можно оценить, например, цены на мороженное отталкиваясь от места жительства и времени года;
- обнаружения аномалий, позволяющее выявить ошибки в расчётах;
- рекомендации, выявление такой информации, которую покупатель захочет увидеть прежде всего;
- кластеризация, которая разбивает данные на группы без каких-либо изначальных установок.

6 Кластеризация

6.1 Определение кластеризации

Кластеризация – это процесс разбиения определённого набора данных на группы, называемые кластерами. Каждая единица данных называется объектом и может являться практически чем угодно, например, можно проводить кластеризацию цветочных растений, машин, животных или даже комментариев на каком-либо сайте. Однако, у каждого объекта должен быть определённый набор характеристик. Каждая характеристика должна пройти процесс нормализации, по итогам которого её значение, обычно, представляется в виде числа с плавающей точкой. Совокупность всех характеристик объекта формирует вектор или точку в многомерном пространстве. Основываясь на расстояниях между точками объектов и происходит разбиение на кластеры. Кластеризацию можно ошибочно спутать с ещё одной очень известной процедурой, называемой классификацией. Классификация – это процесс распределения данных по заранее известным признакам (классам). Для того, чтобы объяснить разницу между этими двумя алгоритмами обработки данных необходимо обратиться к теории машинного обучения. «Имеется ряд категорий машинного обучения: контролируемое обучение или «обучение с учителем» (supervised learning), неконтролируемое обучение (unsupervised learning) (в частности, кластеризация), обучение с подкреплением (reinforcement learning)». [2, 18]

– обучение с учителем – подход, подразумевающий обучение на данных, составленных таким образом, что итоговый результат заранее известен. Классификация относится к этому виду обучения, поскольку ещё до начала обработки наборов данных у исследователя уже есть информация о количестве групп (классов), которое должно получиться по итогам работы алгоритма, а также о том какие характеристики объектов оказывают

наибольшее влияние на итоговый результат разбиения. Алгоритмы, работающие по такому принципу, обладают крайне низкой эффективностью на старте обучения, однако становятся всё лучше и лучше с течением времени. Это происходит из-за того, что исследователю заранее известен итоговый результат и, основываясь на разности между ним и текущим результатом алгоритма, можно внести соответствующие правки. Рано или поздно алгоритм научится различать данные по тем признакам, которые нужны исследователю для правильной классификации. Как заключение, можно сказать, что алгоритмы, обученные таким способом, обладают невероятно высокой точностью и скоростью вычислений, однако требуют значительных затрат времени и сил на подготовку оптимальных наборов тестовых данных;

– обучение без учителя – подход, подразумевающий вычисления на данных, результаты которых заранее не известны. Кластеризация является отличным примером данного метода, поскольку исследователю заранее не известно ни о том какими характеристиками обладают исследуемые объекты, ни о том на какое количество групп (кластеров они должны быть разбиты). Такие алгоритмы оценивают данные, основываясь на неких абстрактных характеристиках, которые можно найти практически у каждого объекта во Вселенной. Кластеризация в отличие от классификации, может разбивать объекты на группы с достаточно высокой точностью без надобности в обучении. Однако результаты, получаемые таким образом очень нестабильны, они могут меняться от раза к разу, а также обладают достаточно высокой погрешностью.

Всё вышесказанное, говорит о том, что кластеризация обычно производится для первичного обследования тех или иных данных, которые сложно разбить на классы вручную. Затем, полученные кластеры данных подвергаются анализу, выявляются взаимосвязи между их объектами и формируются классы данных. Когда итоговое разбиение набора данных становится известно, наступает финальный этап исследования –

классификация, распределяющая объекты по группам с наибольшей точностью.

6.2 Метрики сходства

Каждый объект в наборе данных, подготовленных для кластеризации, выражается определённым, уникальным набором характеристик. Для определения характеристик могут быть выбраны как количественные (вес, рост, скорость) так и качественные величины (цвет, вкус, запах).

Чаще всего выбираются количественные характеристики, т.к. их проще представить в евклидовом пространстве, однако в случае крайней необходимости существуют особые алгоритмы нормализации, сводящие качественные характеристики к количественным.

Набор характеристик формирует вектор, величина которого соответствует количеству измерений, исследуемого пространства. Данные, хранимые в каждом из векторов, для удобства вычислений, находятся в определённых числовых диапазонах обычно [0; 1] или [-1; 1].

Для того, чтобы распределить вектора данных по кластерам необходимо найти расстояния между ними. Для этих целей существует определённый набор метрик, эффективность каждой из которых на прямую зависит от изначального набора данных. Наиболее часто используемыми метриками являются:

– евклидово расстояние – классическая мера расстояния, возникшая в геометрии и вычисляемая по теореме Пифагора, формула (6.1).

$$\sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (6.1)$$

На рисунке 6.1 показано графическое представление Евклидова расстояния между двумя точками А и В;

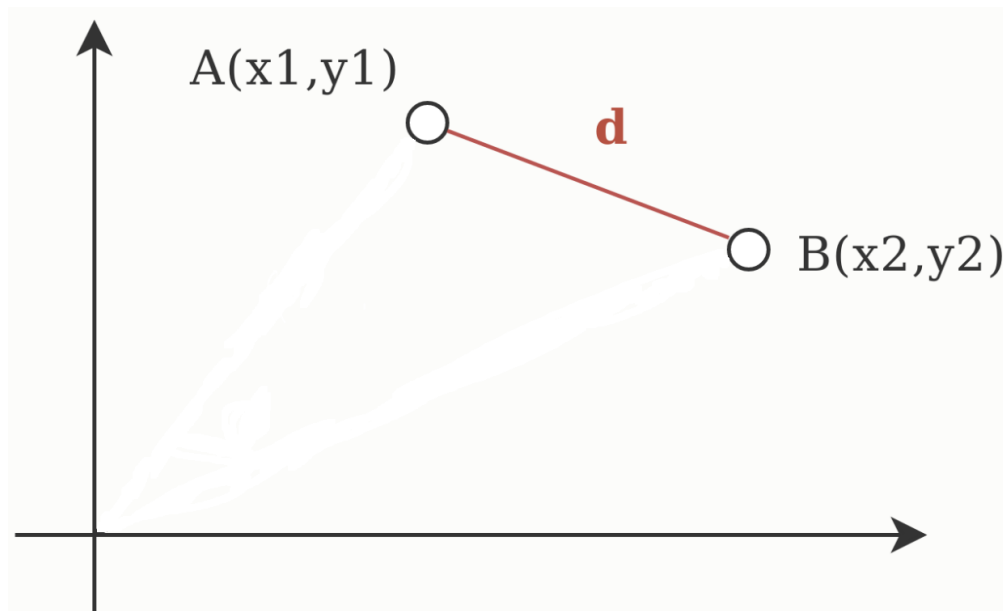


Рисунок 6.1 – Евклидово расстояние

– коэффициент Отиаи, также называемый мерой косинуса угла – мера, которая оценивает схожесть векторов данных при помощи косинуса угла между ними, формула (6.2).

$$\frac{\sum_{i=1}^n x_i * y_i}{\sqrt{\sum_{i=1}^n x_i^2} * \sqrt{\sum_{i=1}^n y_i^2}} \quad (6.2)$$

При значении косинуса близком к нулю, расстояние между двумя векторами очень велико, и, следовательно, они сильно различаются. Когда косинус угла по значению близок к единице, два сравниваемых вектора практически идентичны и с большой вероятностью находятся в одном и том же кластере. Обычно данная метрика используется для работы с информацией, представленной в виде текста. На рисунке 6.2 показано каким образом рассчитывается косинус угла для двух точек;

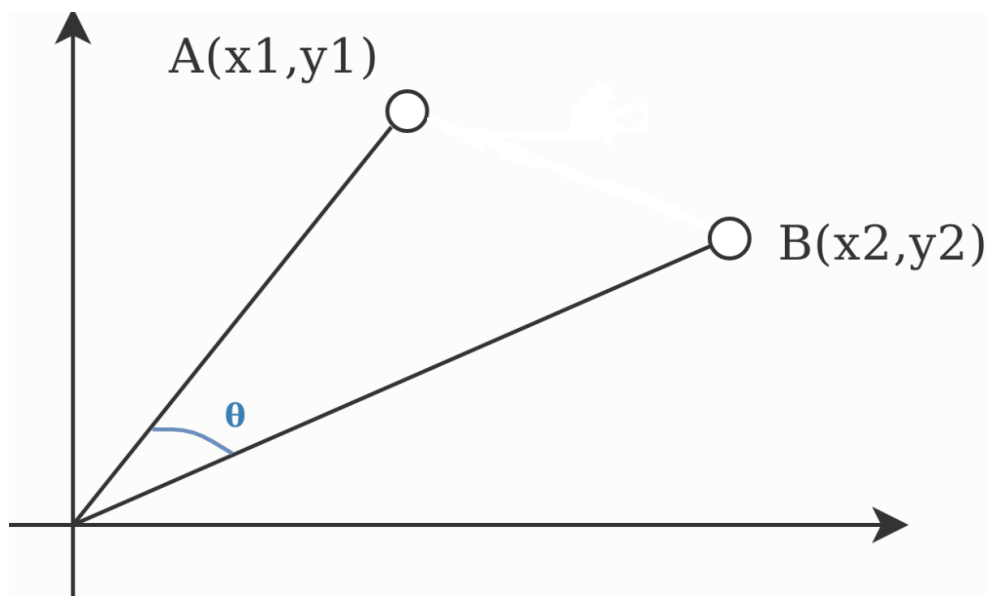


Рисунок 6.2 – Коэффициент Отиаи

– расстояние городских кварталов – метрика, разработанная Германом Минковским. Вычисляется как средняя разность по координатам и чаще всего приводит к результатам, схожим с расстоянием Евклида, формула (6.3).

$$\sum_{i=1}^n |x_i - y_i|; \quad (6.3)$$

– расстояние Чебышева – мера, применяемая тогда, когда требуется определить два объекта как различные при какой-то одной отличной координате. Расстояние Чебышева вычисляется по формуле (6.4).

$$\max(|u_i - v_i|) \quad (6.4)$$

6.4 Классификация алгоритмов

6.4.1 Иерархические алгоритмы

Иерархические алгоритмы кластеризации характеризуются тем, что после обработки набора данных они формируют иерархическое кластерное дерево. В вершине этого дерева всегда находится один большой кластер, содержащий все остальные кластеры и объекты из набора данных. На самом нижнем уровне дерева, содержащем больше всего ветвлений, количество кластеров соответствует общему количеству объектов, т.к. на данном этапе каждый объект формирует свой собственный кластер. Существует два подхода к формированию иерархического дерева: нисходящий и восходящий.

При нисходящем подходе алгоритм начинается с вершины дерева. Первый кластер, содержащий все объекты из набора данных делится на два новых кластера, таким образом, чтобы их центры находились на максимальном расстоянии друг от друга.

Второй подход, восходящий, является более распространённым. Теперь построение иерархического дерева начинается с самого нижнего уровня. Самые близкие друг к другу кластеры начинают объединяться до тех пор, пока не сольются в один единственный кластер.

Плюсы иерархических алгоритмов:

- при работе с иерархическими алгоритмами не нужно указывать результирующее количество кластеров;
- алгоритмы такого типа просты в реализации;
- иерархическое дерево, формируемое в процессе работы алгоритма, является крайне полезным для исследования наборов данных.

Минусы:

- время, затрачиваемое на создание иерархического дерева, значительно замедляет работу алгоритма в сравнение с плоскими алгоритмами кластеризации, такими как k-means;

– при кластеризации больших наборов информации получаются сложные для восприятия человеком иерархические диаграммы, из-за чего определить оптимальное количество кластеров практически невозможно.

Для более наглядной демонстрации работы иерархического алгоритма кластеризации данных, были сделаны графические изображения описывающие восходящий подход. На рисунке 6.3 изображён набор данных, каждая единица которого описывается вектором или точкой в двухмерном пространстве.

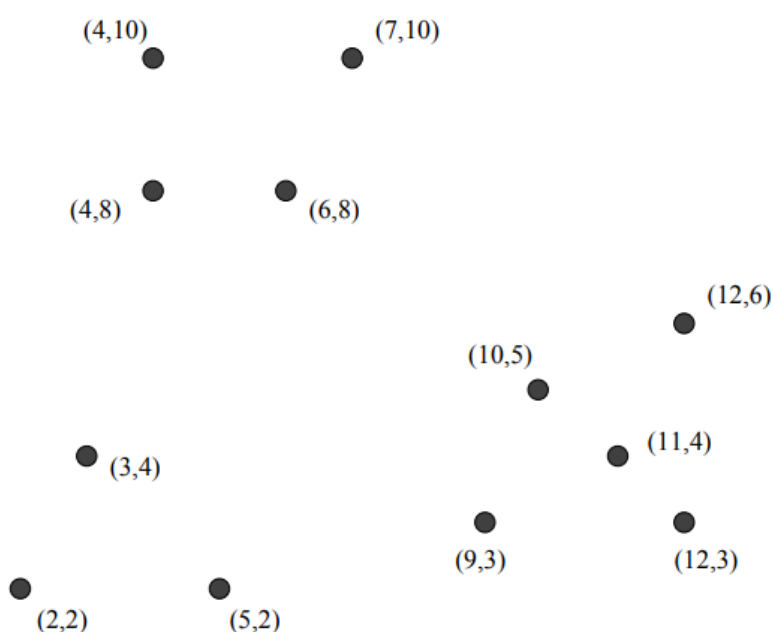


Рисунок 6.3 – Набор данных для иерархической кластеризации

На первом шаге иерархической кластеризации осуществляется поиск двух точек с минимальным расстоянием. Даже без проведения каких-либо расчётов отчётливо видно, что точки (11, 4) и (12, 3) находятся на минимальном расстоянии друг от друга. Следовательно, они формируют новый кластер (рис. 6.4).

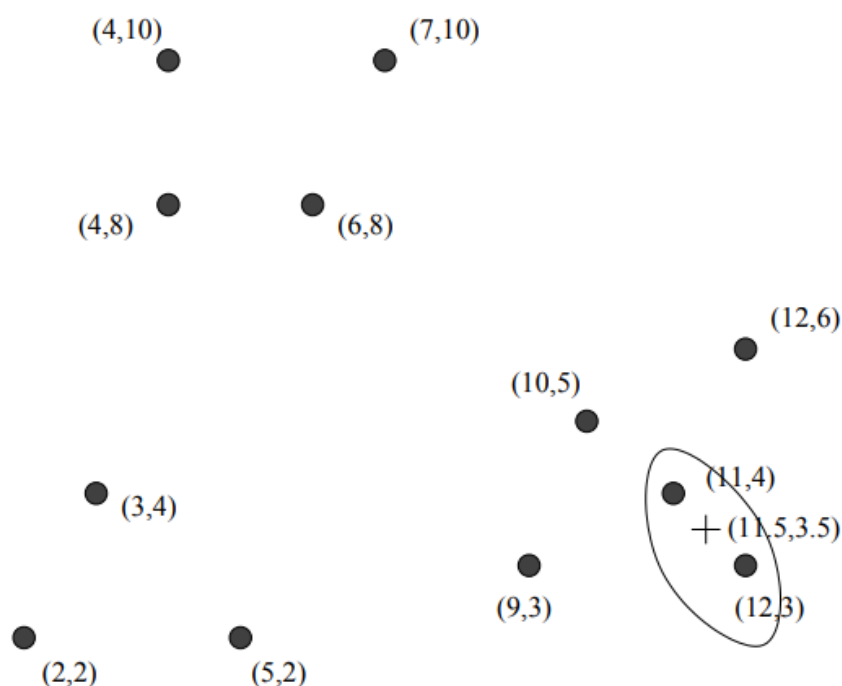


Рисунок 6.4 – Объединение двух точек в кластер

На следующем шаге осуществляется поиск двух кластеров с минимальным расстоянием между их центрами. Стоит отметить, что изначально каждая точка представляет собой кластер и, следовательно, является его центром. Однако после появления кластеров, состоящих из двух и более точек, их центр смещается и определяется как некая средняя величина от координат всех точек, входящих в данный кластер. Таким образом, точки (11, 4) и (12, 3) больше не являются центрами кластеров и измерять расстояние до них нужно учитывая расположение их нового центра, отмеченного крестиком на рисунке 6.4.

На третьем шаге алгоритма было сформировано три кластера, один из которых содержит кластер, созданный на первом этапе (рис. 6.5).

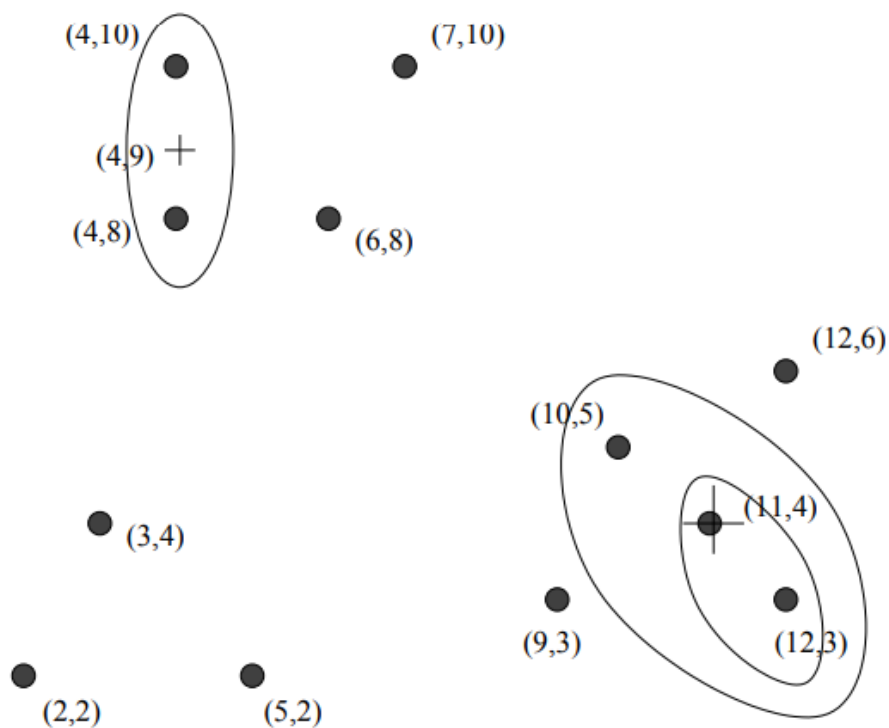


Рисунок 6.5 – Набор данных после формирования ещё двух кластеров

Если продолжить формирование кластеров, то рано или поздно все они объединятся в один. На рисунке 6.6 изображено итоговое иерархическое дерево.

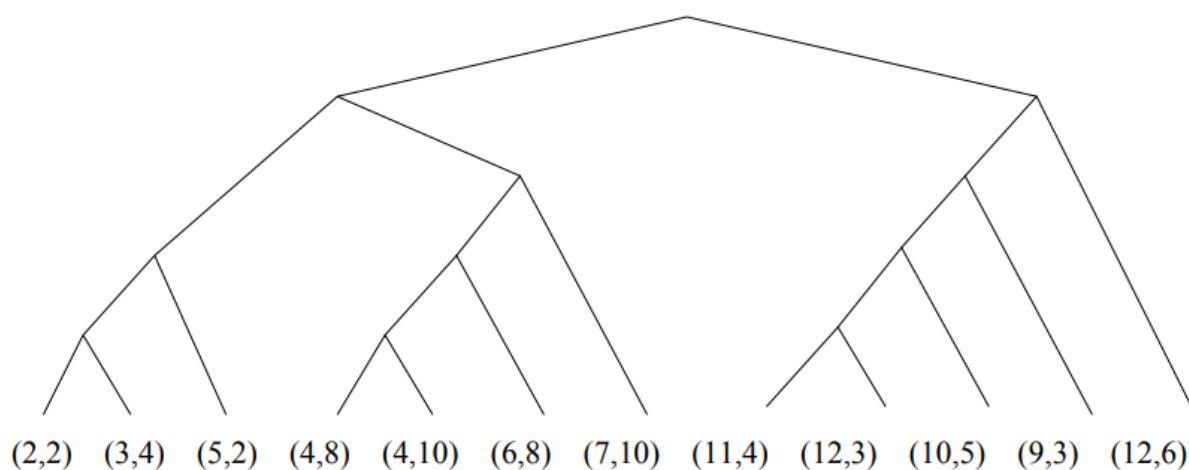


Рисунок 6.6 – Итоговое иерархическое дерево

6.4.2 Плоские алгоритмы

Плоские алгоритмы кластеризации данных характеризуются тем, что для их работы необходимо определить количество кластеров, на которые будет разбита исследуемая информация. Полученное таким образом разбиение является единственным и не может быть разбито на иерархию кластеров, чтобы получить новое разбиение необходимо запустить алгоритм снова, используя обновлённые данные. К данной категории относится один из самых широко известных и популярных алгоритмов – k-means.

Плюсы плоских алгоритмов:

- обладают высокой точностью и скоростью выполнения;
- пользуются наибольшей популярностью (алгоритм k-means);
- способны работать с кластерами самых различных форм и размеров;
- обладают обширным набором модификаций, значительно улучшающих изначальный алгоритм (k-means++, x-means).

Минусы плоских алгоритмов:

- требуют обязательного ввода изначального количества кластеров;
- эффективность сильно зависит от изначальной выборки данных;
- несмотря на относительно высокую точность, итоговые вычисления не лишены погрешности.

В качестве примера плоского алгоритма рассмотрим k-means. Данный алгоритм работает без надобности в предварительном этапе обучения на специально подготовленных тестовых наборах данных. Он обладает наибольшей популярностью среди всех остальных алгоритмов кластеризации. Такую известность k-means получил благодаря простоте реализации, а также точности и скорости выполнения.

Сердцем алгоритма k-means является циклический процесс, в ходе которого оцениваются все точки из набора данных, а затем каждая из них соотносится с ближайшим к ней кластерным центром – центроидом. Стоит отметить, что центроиды каждого кластера постоянно перемещаются с

каждым новым циклом алгоритма, точно также, как и точки постоянно переходят от одного кластера к другому. Когда все перемещения прекращаются или становятся пренебрежительно незначимыми алгоритм останавливается и выдаёт результат в виде данных разделённых на k кластеров.

Базовая реализация алгоритма может быть представлена последовательностью действий, описанной на рисунке 6.7.

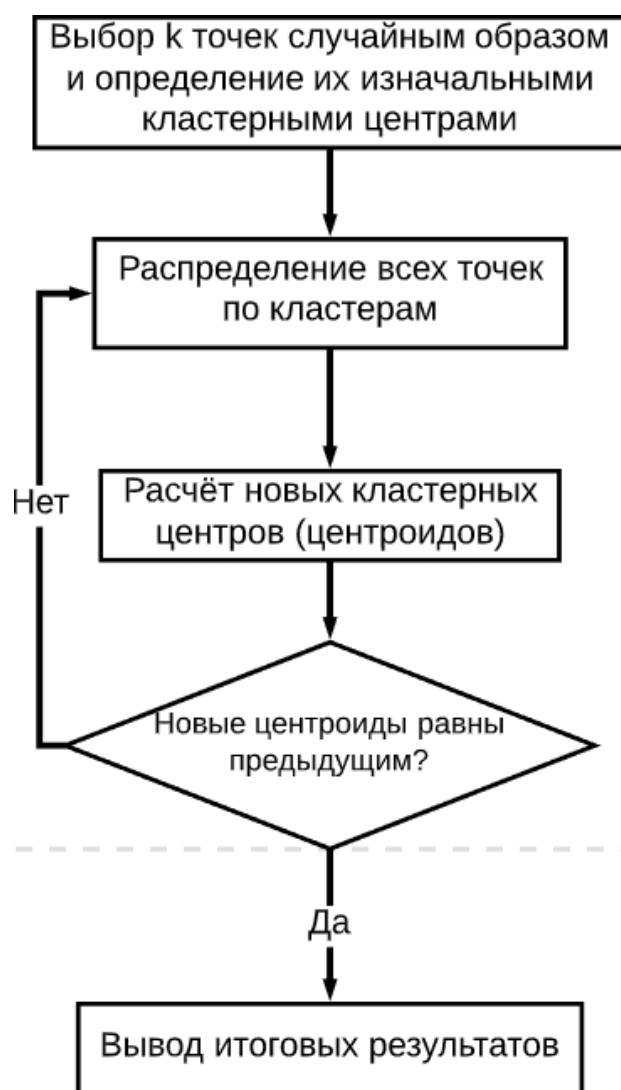


Рисунок 6.7 – Алгоритм работы k-means

На этапе инициализации алгоритм k-means случайным образом выбирает k точек и делает каждую из них кластером. Таким образом каждая

из этих точек является центром собственного кластера. Этот этап является слабым местом базовой реализации алгоритма k-means. Из-за того, что центры изначальных кластеров выбираются случайно конечный результат, станет нестабильным и будет обладать большой погрешностью. Эффективность такого алгоритма в большей степени зависит от удачи, что недопустимо, когда речь идёт о серьёзных научных исследованиях.

Однако алгоритм k-means практически никогда не используется в своём изначальном виде. На сегодняшний день абсолютно во всех областях науки, использующих плоские алгоритмы кластеризации, применяется модификация – k-means++. Данная модификация позволяет полностью избавиться от погрешностей, вызванных неопределённостью в выборе центров изначальных кластеров. Первый этап алгоритма k-means заменяется последовательностью действий, изображённой на рисунке 6.8.

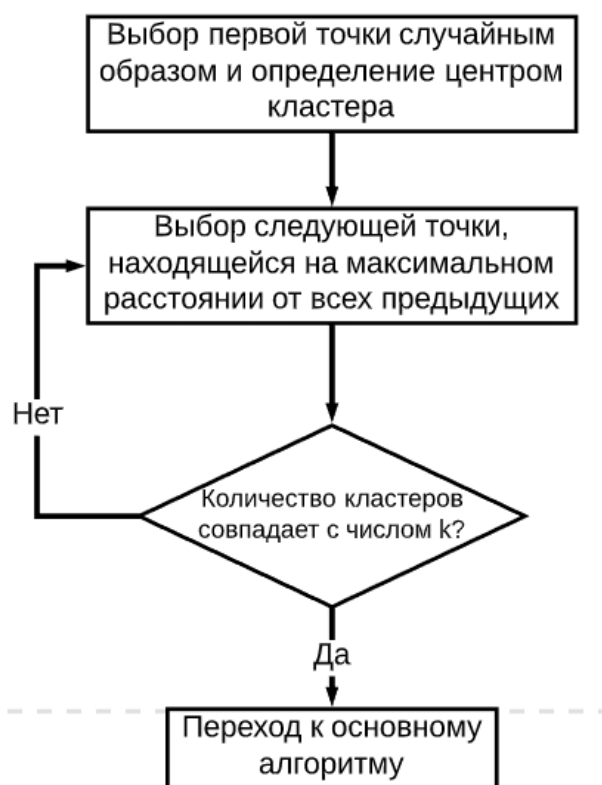


Рисунок 6.8 – Модификация k-means++

k-means++ осуществляет выбор изначальных кластеров, основываясь на дистанциях между торчками. Ведь если между двумя определёнными точками будет максимальная дистанция из всех возможных, то, вероятнее всего, они находятся в разных кластерах.

Ещё одним существенным недостатком k-means является необходимость в определении изначального количества кластеров. Она была решена в модификации x-means, которая использует метрики расчёта оптимальности количества кластеров.

Если число k неизвестно, то расчёты проводятся многократно для разных его значений (обычно от 2 до 10). Когда вычисления окончены наилучший результат определяется методом перебора. Для каждого числа кластеров рассчитывается определённая метрика, показывающая, то насколько удачно данный набор кластеров описывает выборку данных. Затем показатели сравниваются и выбирается число k , получившее наилучший результат. На рисунке 6.9 изображён один из подходов x-means – метод локтя. Метрики, полученные для каждого числа кластеров переносятся на координатную ось. В результате они выстраиваются в форме локтя и выбирается та из них, что находится в месте сгиба. В данном случае $k = 4$.

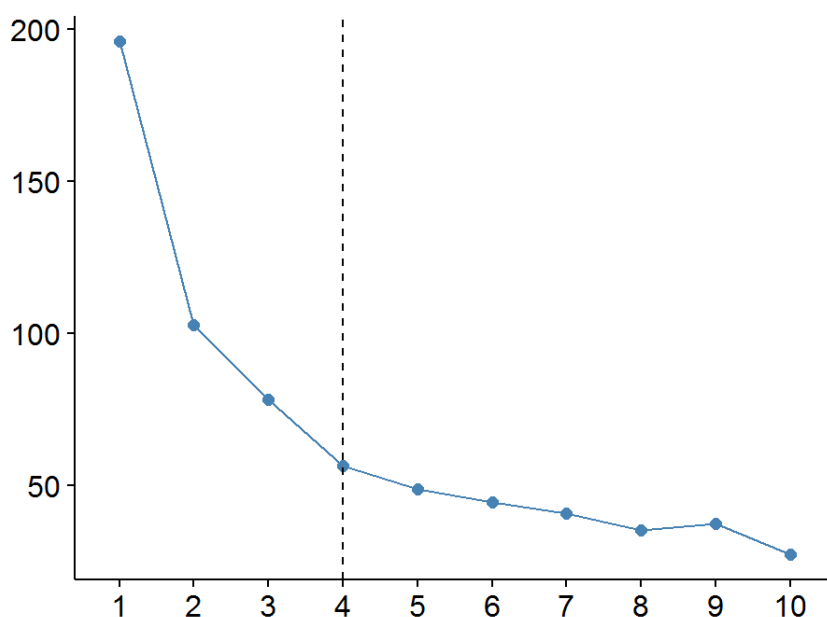


Рисунок 6.9 – Метод локтя

7 Обработка текстовых данных

Основной задачей данного отчёта являлось исследование и разработка алгоритмов для кластеризации данных, размещаемых на сайте. Так как объектом моего исследования был сайт, посвящённый литературным произведениям русских писателей, то наборы данных для кластеризации целиком состояли из текстовой информации. При этом каждый объект представлял собой текст, состоящий из более чем трёх тысяч символов.

Кластеризация текста – это не самая простая задача. Основные трудности заключаются в том, чтобы каким-то образом преобразовать письменные символы в числа, с которыми будет возможно работать в Евклидовом пространстве.

При этом каждое число должно в точности передавать изначальное значение, заложенное в том или ином слове или букве. Невозможно просто присвоить словам порядковый номер или какое-либо другое случайное значение, иначе кластеризация таких данных потеряет всякий смысл. Ситуация осложняется ещё и тем фактом, что кластеризацию необходимо проводить не просто на словах или фразах, содержащих сравнительно небольшое количество символов, а на полноценных литературных произведениях, состоящих из нескольких тысяч слов.

Для того чтобы алгоритм кластеризации смог работать с данными их необходимо преобразовать в векторную форму, числовые значения которой находятся в интервале от 0 до 1. Однако перед началом векторизации текста очень важно произвести предварительную обработку.

7.1 Нежелательные символы

Книга, загруженная из любого текстового файла помимо обычного русского текста, также содержит нежелательные особые символы, знаки

препинания и слова на других языках, которые могут значительно замедлить работу алгоритма, а также привести к снижению точности конечных результатов.

Поэтому первым этапом обработки текста является избавление от всех нежелательных символов. В листинге 7.1 приведён код программы, который приводит все символы текста к нижнему регистру, а также заменяет все символы кроме русских слов и пробелов на пустую строку.

Листинг 7.1 – Удаление нежелательных символов

```
var newText = Regex.Replace(text.ToLower(), @"[^ а-я]", string.Empty);
```

7.2 Токенизация

Второй этап – токенизация текста. В качестве токенов в данном случае выступают слова. Например, имеется предложение: “Каждое утро я пью чай”. После выполнения процедуры токенизации данное предложение будет разбито на набор токенов: каждое, утро, я, пью, чай. Для реализации данного алгоритма на языке C# достаточно использовать метод `Split`, который разделяет строку на подстроки по определённому символу-разделителю.

7.3 Стемминг

Третий этап подразумевает использование Алгоритма стемминга, разработанного Мартином Портером, который позволяет сократить количество слов путём отсекаания их окончаний и суффиксов. Этот этап не является обязательным и в некоторых ситуациях может даже ухудшить точность алгоритма, однако он незаменим при работе с нейронными сетями, поскольку благодаря ему можно уменьшить количество уникальных слов в наборе тестовом данных и, следовательно, уменьшить размер входного нейронного слоя

7.4 Векторизация

После первичной обработки текста можно приступить к его векторизации. Существует несколько способов преобразовать слова в числа. При разработке программного продукта изначально был использован самый популярный и простой в реализации метод – «Bag of words» (непрерывный мешок со словами). Данный метод подразумевает создание словаря, в котором содержатся все уникальные слова, встречаемые во всех книгах. Величина этого словаря будет соответствовать величине вектора, описывающего каждый текстовый документ. Каждое слово – это параметр, задающий ту или иную координату вектора. В своей изначальной реализации «Bag of words» предлагает считать, как много раз то или иное слово встречается в текстовом документе. Полученные таким образом значения будут формировать числовой вектор, который может быть с лёгкостью обработан любым алгоритмом кластеризации.

Однако, при таком подходе теряется часть информации, хранимой в словах. Поэтому наилучшим решением будет использование характеристики, называемой TF-IDF.

TF-IDF – это метрика наиболее точно характеризует значимость того или иного слова в тексте. Как видно из формулы (7.1), она представляет собой произведение двух составных компонентов TF и IDF.

$$TFIDF = TF * IDF \quad (7.1)$$

TF (term frequency) – это величина, характеризующая частоту появления того или иного термина (слова) в документе. Определяется как отношение числа повторов одного и того же слова в документе к общему числу слов данного документа, формула (7.2).

$$\text{tf}(k, t) = \frac{n_t}{N_k}, \quad (7.2)$$

где n_t – количество повторений слова t в документе k ;

N_k – это общее количество слов в документе k .

IDF (inverse document frequency) – обратная частота документа. Эта метрика оценивает слова относительно всей коллекции документов. Величина определяется как отношение общего числа документов к числу документов, содержащих текущий термин (слово), формула (7.3).

$$\text{idf}(t, D) = \log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|}, \quad (7.3)$$

где $|D|$ – общее число документов из набора данных;

$|\{d_i \in D \mid t \in d_i\}|$ – это число документов, в которых встречается слово t .

Результирующее значение TF-IDF рассчитывается для каждого слова в каждом документе. В итоге текстовые файл будут представлять собой вектора значений в интервале от 0 до 1. Такие данные можно использовать для кластеризации практически любым из известных алгоритмов.

8 Программная реализация

8.1 Реализация алгоритма k-means

Для разработки программы кластеризации данных был выбран наиболее популярный и надёжный алгоритм k-means. Работа приложения начинается ещё на этапе загрузки электронных документов в базу данных. Каждый документ проходит через процесс первичной обработки, который осуществляется при помощи метода TransformText, листинг 8.1.

Листинг 8.1 – Обработка текста

```
private (List<string>, List<string>) TransformText(string text)
{
    var processedText = new List<string>();
    var porterText = new List<string>();

    foreach (var term in _regex.Split(text))
    {
        var temp1 = Regex.Replace(term.ToLower(), @"[^а-я]",
string.Empty);

        var temp2 = Porter.TransformWord(temp1);

        if (temp1 != "")
        {
            processedText.Add(temp1);
            porterText.Add(temp2);
        }
    }

    return (processedText, porterText);
}
```

Данный метод принимает книжный текст и разделяет его на символы при помощи регулярных выражений Regex. Таким образом осуществляется токенизация текста. Далее каждый полученный токен очищается от всех нежелательных символов и в нём остаются только слова русского языка. Затем выполняется алгоритм стемминга, в процессе которого удаляются суффиксы

и окончания полученных слов. Так как этот этап применяется не всегда то результаты выполнения алгоритма стемминга хранятся в отдельной переменной. Далее полностью обработанный токен добавляется в итоговую коллекцию. Как результат, метод возвращает кортеж, содержащий сразу две коллекции данных с обработкой алгоритмом стемминга и без.

Для представления каждого документа используется класс `DocumentVector` (листинг 8.2), который содержит `id` книги, коллекцию обработанных токенов, а также вектор данных, необходимый для кластеризации.

Листинг 8.2 – Документ

```
public class DocumentVector
{
    public int Id { get; set; }
    public List<string> Content { get; set; }
    public double[] Vector { get; set; }
}
```

Чтобы получить вектор данных для каждого документа сначала необходимо найти словарь, содержащий все токены (слова) в единственном экземпляре. Для этого используется метод `GetVocabulary` (листинг 8.3).

Листинг 8.3 – Составление словаря

```
public void GetVocabulary()
{
    foreach (var document in DocumentVectorCollection)
    {
        foreach (var term in document.Content)
        {
            Vocabulary.Add(term);
        }
    }
}
```

Чтобы токены, занесённые в словарь `Vocabulary`, не дублировались он представлен коллекцией `HashSet`.

После составления словаря данных открывается возможность для расчёта векторов значений каждого документа при помощи метрики TF-IDF (листинг 8.4).

Листинг 8.4 – расчёт TF-IDF

```
private double FindTFIDF(List<string> content, string term)
{
    double tf = FindTermFrequency(content, term);
    double idf = FindInverseDocumentFrequency(term);

    return tf * idf;
}

private double FindTermFrequency(List<string> content, string term)
{
    int count = content.Count(x => x == term);

    return (count / (double)content.Count());
}

private double FindInverseDocumentFrequency(string term)
{
    int count = DocumentVectorCollection.Count(x =>
x.Content.Contains(term));

    return Math.Log(DocumentVectorCollection.Count / (double)count);
}
```

В метод FindTFIDF вводится коллекция токенов текущего документа, а также один токен для которого ведётся расчёт. Сначала необходимо найти значение TF, которое характеризует частоту использования токена в данном документе (метод FindTermFrequency). Затем рассчитывается IDF, характеризующее частоту появления данного токена во всех документах (метод FindInverseDocumentFrequency). После этого обе метрики перемножаются и получается итоговый результат TF-IDF – число, характеризующее значимость того или иного токена (слова) в тексте.

После формирования векторов данных коллекция книг полностью готова к кластеризации. Метод Cluster, приведённый в листинге 8.5, является сердцем алгоритма кластеризации k-means.

Листинг 8.5 – Алгоритм кластеризации

```
public List<Cluster> Cluster(List<DocumentVector> DVCollection, int k)
{
    iterationCount = 0;
    vocabularyCount = DVCollection[0].Vector.Length;

    if (k > DVCollection.Count)
        k = DVCollection.Count;

    List<Cluster> clusterCollection = new List<Cluster>();
    List<Cluster> prevClusterCollection;

    HashSet<int> seed = CreateSeed(DVCollection, k);

    foreach (int index in seed)
    {
        Cluster cluster = new Cluster
        {
            Centroid = DVCollection[index].Vector,
            DocumentGroup = new List<DocumentVector>()
        };

        clusterCollection.Add(cluster);
    }

    do
    {
        prevClusterCollection = clusterCollection;

        AttachDocuments(ref clusterCollection, DVCollection);
        UpdateMeanPoints(ref clusterCollection);

        iterationCount++;

    } while (!StopCheck(prevClusterCollection, clusterCollection));

    return clusterCollection;
}
```

Входными данными этого метода являются коллекция документов, составленная ранее, а также изначальное число кластеров k. Первым этапом

алгоритма является выбор изначальных документов в качестве кластерных центров.

В данном проекте используется модифицированный алгоритм кластеризации – k-means++, поэтому выбор изначальных кластерных центров, также называемых центроидами, осуществляется не случайным образом, а при помощи метрик расстояния. Метод CreateSeed выбирает центроиды, основываясь на расстояниях между ними. Таким образом получается коллекция k-изначальных документов, каждый из которых с наибольшей вероятностью находится в разных кластерах. В листинге, приведённом выше, можно увидеть, как сначала определяются индексы документов (переменная seed), а затем уже создаётся набор кластеро. Каждый кластер представлен классом Cluster (листинг 8.6). Он содержит коллекцию, принадлежащих ему документов, а также центроид.

Листинг 8.6 – Кластер

```
public class Cluster
{
    public double[] Centroid { get; set; }
    public List<DocumentVector> DocumentGroup { get; set; }
}
```

Расстояние между кластерами может быть посчитано с использованием различных метрик сходства. В данной выпускной квалификационной работе используется Евклидова величина (листинг 8.7) и мера косинуса угла (листинг 8.8).

Листинг 8.7 – Евклидова величина

```
public class EuclideanDistance : SimilarityMetricBase
{
    public override double FindDistance(double[] vecA, double[] vecB)
    {
        double euclideanDistance = 0;
        for (var i = 0; i < vecA.Length; i++)
        {
```

```

        euclideanDistance += (vecA[i] - vecB[i]) * (vecA[i] -
vecB[i]);
    }

    return Math.Sqrt(euclideanDistance);
}
}

```

Листинг 8.8 – Мера косинуса угла

```

public class CosineSimilarity : SimilarityMetricBase
{
    public override double FindDistance(double[] vecA, double[] vecB)
    {
        var dotProduct = DotProduct(vecA, vecB);
        var magnitudeA = Magnitude(vecA);
        var magnitudeB = Magnitude(vecB);
        double result = dotProduct / (magnitudeA * magnitudeB);

        return NaNCheck(result);
    }
}

```

Далее начинается цикл while, в котором коллекция кластеров будет постоянно изменяться. Критерием остановки алгоритма является метод StopCheck, который определяет есть ли различия между текущим набором кластеров и предыдущим. Если различий нет, то цикл останавливается.

По ходу выполнения цикла каждый документ присваивается к ближайшему кластероному центру при помощи метода AttachDocuments. Затем, в методе UpdateMeanPoints для каждого кластера высчитывается среднее арифметическое от всех векторов его документов. Полученные значения определяются как новые центры кластеров.

С каждой новой итерацией алгоритма также растёт счётчик iterationCount. Он является ещё одним критерием остановки, благодаря которому алгоритм не будет выполняться бесконечно и остановится после 11000 итераций.

После завершения работы метода Cluster получается итоговый набор документов, распределённых по k кластерам.

8.2 Применение библиотеки ML.NET

При разработке программного продукта для кластеризации помимо собственной реализации алгоритма k-means также была использована библиотека ML.NET, которая обладает большей точностью и скоростью вычисления.

Для взаимодействия с ML.NET был создан класс MLBook (листинг 8.9), который содержит id книги и необработанный текст, а также класс MLClusterPrediction (листинг 8.10), который хранит данные о выбранном кластере для той или иной книги.

Листинг 8.9 – Класс MLBook

```
public class MLBook
{
    public int Id { get; set; }
    public string Text { get; set; }
}
```

Листинг 8.10 – Класс MLClusterPrediction

```
public class MLClusterPrediction
{
    public int Id { get; set; }
    public uint PredictedLabel;
}
```

Работа ML.NET начинается с загрузки и обработки набора тестовых данных. В листинге 8.11 показано как при помощи метода LoadFromEnumerable осуществляется загрузка книг, предварительно преобразованных в объекты класса MLBook. Затем метод FeaturizeText разбивает текст на токены, убирает нежелательные символы и преобразует его в вектор. Полученная таким образом коллекция transformedData будет содержать элементы с тремя полями: идентификационный номер книги, необработанный текст и вектор.

Листинг 8.11 – Загрузка и обработка текстовых данных

```
var dataView = mlContext.Data.  
    LoadFromEnumerable(mlBooks);  
var transformedData = mlContext.Transforms.Text.  
    FeaturizeText("Features", "Text").  
    Fit(dataView).  
    Transform(dataView);
```

Далее, как показано в листинге 8.12, создаётся тренер машинного обучения k-means, для которого необходимо указать количество кластеров. Затем модель проходит обучение на наборе данных transformedData при помощи метода Fit.

Листинг 8.12 – Создание и обучение модели k-means

```
var pipeline = mlContext.Clustering.Trainers.  
    KMeans(numberOfClusters: k);  
var model = pipeline.  
    Fit(transformedData);
```

Теперь алгоритм k-means готов к работе. В листинге 8.13 показан заключительный этап кластеризации. Обученная модель используется для разбиения набора данных на кластеры, при этом каждый кластер преобразуется в объект класса MLClusterPrediction. Полученная, таким образом коллекция кластеров сначала конвертируется в Enumerable при помощи метода CreateEnumerable, а затем в список. Изначальный набор данных разбит на кластеры и готов дальнейшему исследованию.

Листинг 8.12 – Получение итогового набора кластеров

```
var clusterPredictions = mlContext.Data.  
    CreateEnumerable<MLClusterPrediction>(model.  
        Transform(transformedData), false).  
        ToList();  
clusterPredictions.  
    Sort((x, y) => x.PredictedLabel.  
        CompareTo(y.PredictedLabel));
```

9 Руководство пользователя

Интерфейс приложения полностью внедрён в функционал веб-сайта «Литература». Доступ к алгоритмам кластеризации осуществляется через главное окно сайта, изображённое на рисунке 9.1.

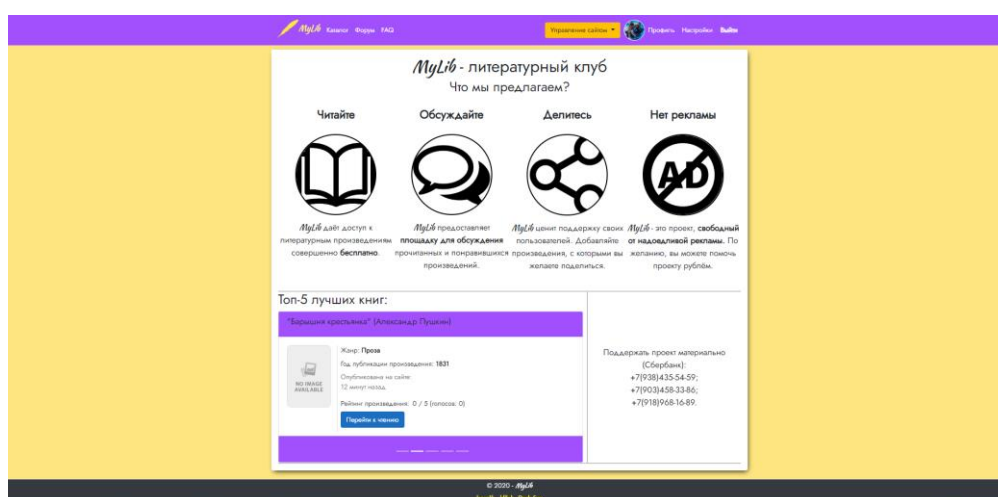


Рисунок 9.1 – Домашняя страница сайта «Литература»

Только в том случае если пользователь зарегистрирован и является администратором, в верхнем правом углу перед профилем пользователя появляется кнопка «Управление сайтом», которая открывает список возможных действий (рисунок 9.2).

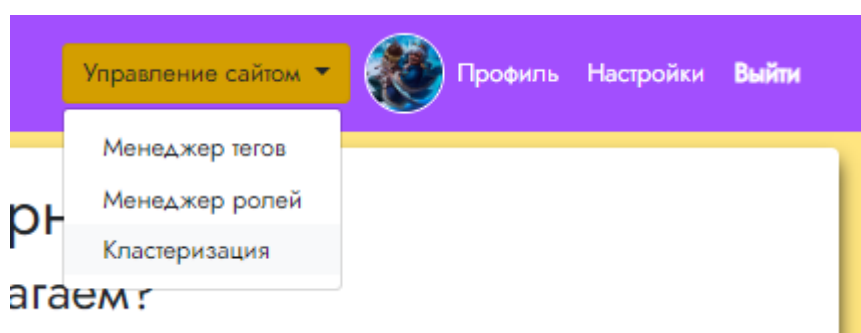


Рисунок 9.2 – Домашняя страница сайта «Литература»

Если перейти в раздел кластеризации, то пользователю откроется возможность анализировать отсортированные наборы книг при помощи разбиения на кластеры. Интерфейс страницы кластеризации изображён на рисунке 9.3.

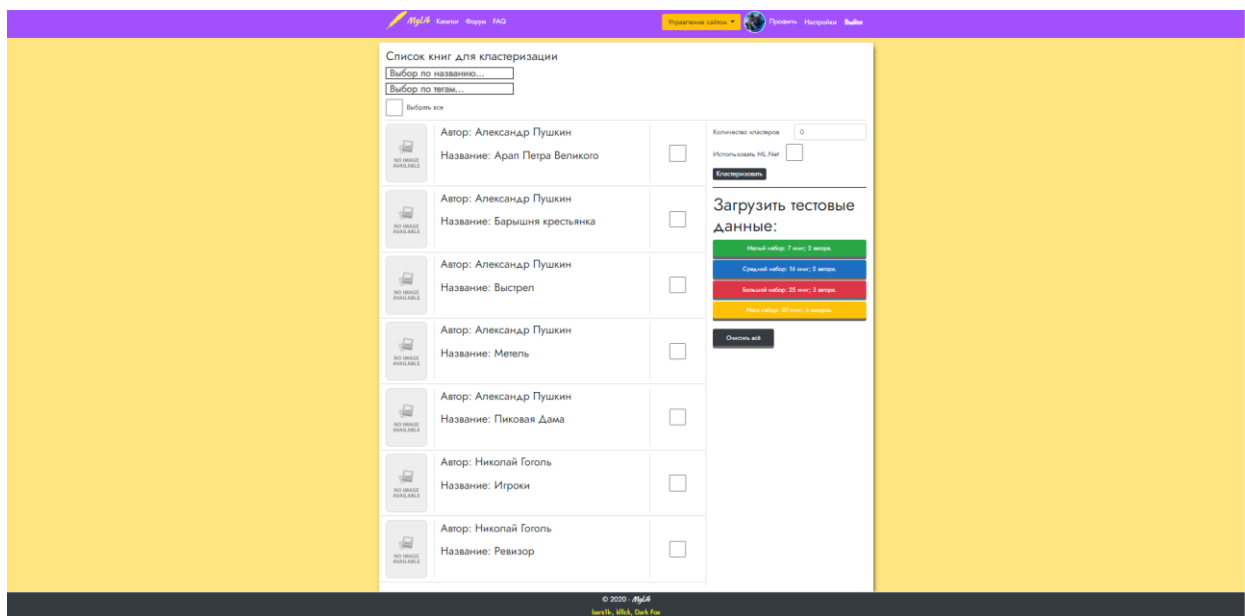


Рисунок 9.3 – Страница кластеризации данных

В середине окна располагается список всех книг, которые будут подвергнуты кластеризации. Существует возможность сортировки книг по названию и различным тега, характеризующим жанры и автора книги (рисунок 9.4).

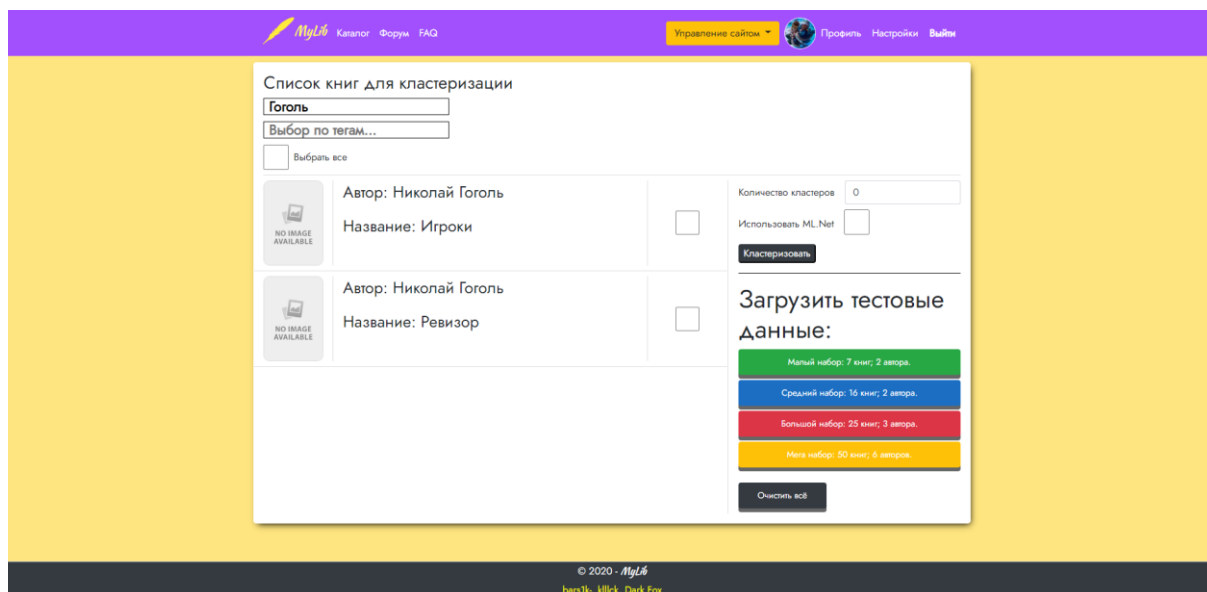


Рисунок 9.4 – Результаты сортировки книг по названию

В правом нижнем углу находятся кнопки (рисунок 9.5), позволяющие загружать тестовые наборы данных, различающиеся по количеству книг и авторов. Чем больше авторов будет присутствовать в тестовом наборе, тем на большее количество кластеров будет разбит изначальный набор книг и тем меньше будет точность итогового разбиения.

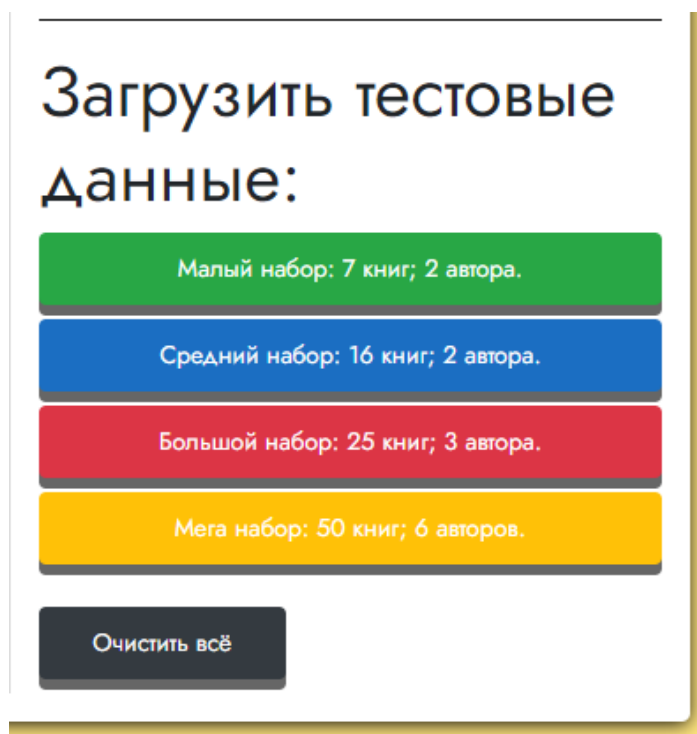


Рисунок 9.5 – Кнопки для загрузки тестовых наборов данных

В правом верхнем углу можно настроить работу алгоритма кластеризации (рисунок 9.6). Здесь пользователь может указать исходное число кластеров, а также выбрать использовать ли библиотеку ML.Net при проведении кластеризации или нет.

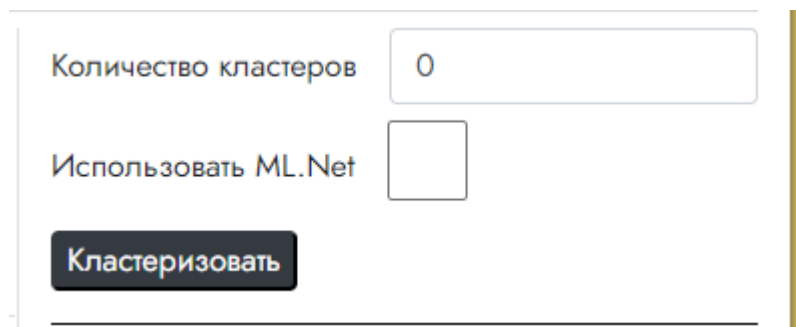


Рисунок 9.6 – Настройка алгоритма кластеризации

После того как набор книг для кластеризации определён и указано количество кластеров, которое при тестировании соответствует количеству авторов, пользователь может нажать на кнопку «Кластеризовать» и сайт автоматически перенаправит его на страницу с результатами. На рисунке 9.7 можно видеть окно с результатами кластеризации для малого набора данных, состоящего из семи книг и двух авторов, без использования библиотеки ML.Net.

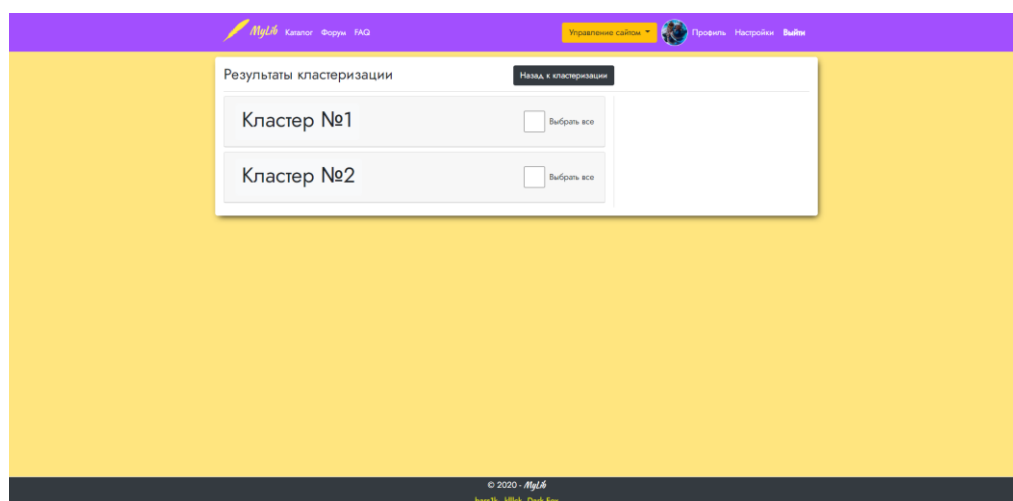







Рисунок 9.7 – Страница с результатами кластеризации данных

Кластеры данных представлены в виде аккордеонов, при нажатии на которые вниз раскрывается список, принадлежащих им книг (рисунок 9.8).

Кластер №1		
	Автор: Александр Пушкин Название: Арап Петра Великого	<input type="checkbox"/>
	Автор: Александр Пушкин Название: Барышня крестьянка	<input type="checkbox"/>
	Автор: Александр Пушкин Название: Выстрел	<input type="checkbox"/>
	Автор: Александр Пушкин Название: Метель	<input type="checkbox"/>
	Автор: Александр Пушкин Название: Пиковая Дама	<input type="checkbox"/>
<input type="checkbox"/> Выбрать все		



Кластер №2		
	Автор: Николай Гоголь Название: Игроки	<input type="checkbox"/>
	Автор: Николай Гоголь Название: Ревизор	<input type="checkbox"/>
<input type="checkbox"/> Выбрать все		

Рисунок 9.8 – Кластеры данных

Таким образом, основываясь на изображениях, приведённых выше, можно сказать, что алгоритм смог правильно разделить тестовый набор книг по авторам. Каждый кластер содержит только произведения одного автора, что говорит о высокой точности разработанного приложения.

10 Оценка эффективности

Для оценки эффективности алгоритмов кластеризации данных в реализованном программном продукте. Был проведён сравнительный анализ между алгоритмом, разработанным с нуля, и библиотекой ML.NET. В ходе тестирования использовались четыре различных по величине наборов данных. Учитывались такие параметры как скорость, точность, количество книг, а также количество различных авторов (кластеров данных). Итоговые результаты представлены в таблицах 10.1 и 10.2.

Таблица 10.1 – Оценка разработанного алгоритма k-means

Набор	Кол-во книг	Кол-во авторов	Время	Точность
Малый	7	2	34 сек.	100%
Средний	16	2	2 мин. 40 сек.	78%
Большой	25	3	8 мин. 50 сек.	76%
Очень большой	50	6	24 мин. 33 сек.	55%

Таблица 10.2 – Оценка кластеризации с использованием ML.NET

Набор	Кол-во книг	Кол-во авторов	Время	Точность
Малый	7	2	2 сек.	100%
Средний	16	2	7 сек.	81.25%
Большой	25	3	12 сек.	74%
Очень большой	50	6	48 сек.	41%

По результатам исследования видно, что точность обоих алгоритмов значительно снижается при использовании слишком большого количества кластеров (больше шести). Благодаря этому, можно сделать вывод, что для наилучшей работы кластеризации текстовые данные должны загружаться

порциями не больше 30 книг, а количество возможных авторов не должно превышать 3.

Кроме того, скорость работы библиотеки ML.NET значительно превышает алгоритм, разработанный специально для данной выпускной квалификационной работы. Это происходит по нескольким причинам:

- библиотека ML.NET использует другой подход для обработки, токенизации и векторизации текста, который не требует нахождения метрики TF-IDF для каждого токена (слова);

- алгоритм k-means в ML.NET использует другой критерий остановки, благодаря которому кластеризация прекращается если перемещения кластеров становятся не эффективными;

- алгоритмы ML.NET могут использовать более эффективные типы и методы работы с данными.

Однако, несмотря на значительное отставание в скорости, алгоритм k-means, реализованный в данной работе, немного превосходит ML.NET в точности на больших выборках данных. Таким образом, разработанный программный продукт, выгоднее использовать для оценки больших наборов текстовых данных. Если же требуется кластеризовать выборку из 10-15 книг, то наилучшим выбором будет использование библиотеки ML.NET.

11 Безопасность и экологичность проекта

11.1 Значение и задачи безопасности жизнедеятельности

На сегодняшний день компьютерные технологий развились до невероятных высот и проникли практически во все сферы человеческой деятельности. Электронно-вычислительные машины успешно применяются для загрузки, хранения и обработки огромных массивов информации в сети Интернет. Благодаря этому многократно увеличилась эффективность работы большинства современных компаний и учреждений, ежедневно использующих персональные компьютеры для выполнения самых разнообразных целей.

Однако, если не соблюдать определённых правил при обращении с электронно-вычислительными устройствами, они могут нанести значительный вред здоровью. Постоянная и продолжительная работа за компьютером в офисе, без правильно распределённой трудовой деятельности, приводит к быстрому переутомлению, ухудшению сердечно-сосудистой системы, нарушениям зрительного аппарата, а также возникновению головных болей, эмоционального стресса и психической истощённости.

Чтобы нивелировать всевозможные неблагоприятные воздействия при работе с персональным компьютером необходимо выполнять ряд требований техники безопасности БЖ, которые описаны в следующих нормативных документах:

- СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы»;
- «Трудовой кодекс Российской Федерации» от 30.12.2001 № 197-ФЗ (ред. от 10.01.2016);
- ТОО Р-45-084-01 «Типовая инструкция по охране труда при работе на персональном компьютере»;

– ГОСТ 12.0.003-2015 «Система стандартов безопасности труда (ССБТ). Опасные и вредные производственные факторы. Классификация».

11.2 Анализ условий труда и мероприятия по защите от воздействия вредных производственных факторов

«Литература» – это веб-сайт, предназначенный для хранения, чтения, и анализа различных произведений русской литературы при помощи алгоритмов сортировки и кластеризации. Администрация сайта ежедневно следит за порядком на форуме, редактирует книжные данные, вносит корректировки в текст, а также пользуется алгоритмами кластеризации для проверки книг на соответствие тем или иным жанрам. Такая напряжённая работа может крайне негативно сказаться на их физическом и эмоциональном самочувствии. Несомненно, использование электронно-вычислительных устройств и сети Интернет представляет огромную ценность для всего человечества, однако у всех людей длительно использующих ЭВМ без соответствующих правил предосторожности могут возникнуть заболевания различной степени тяжести.

Чтобы нивелировать всевозможные негативные последствия длительной работы с персональным компьютером в БЖ описаны основные правила по созданию комфортных условий труда в офисе, а также минимизации эмоционального перенапряжения, вызванного однообразием и монотонностью трудового процесса.

Проведём оценку работы администратора сайта при работе с сортировкой, редактированием и кластеризацией книжных наборов данных. Документ, по которому оценивается тяжесть – Р 2.2.2006-05 «Руководство по гигиенической оценке факторов рабочей среды и трудового процесса. Критерии и классификация условий труда». Анализ оценки напряжённости трудового процесса показан в таблице 11.1.

Таблица 11.1 – Оценка напряжённости трудового процесса

Показатели	Класс условий труда			
	1	2	3.1	3.2
1 Интеллектуальные нагрузки				
1.1 Содержание работы		+		
1.2 Восприятие информации и их оценка			+	
1.3 Распределение функций по степени сложности задания		+		
1.4 Характер выполняемой работы			+	
2 Сенсорные нагрузки				
2.1 Длительность сосредоточенного наблюдения (% времени смены)				+
2.2 Плотность сигналов (световых, звуковых) и сообщений в среднем за 1 час работы		+		
2.3 Число производственных объектов одновременного наблюдения	+			
2.4 Размер объекта различения (при расстоянии от глаз работающего до объекта различения не более 0,5 м) в мм при длительности сосредоточенного наблюдения (% времени смены)			+	
2.5 Работа с оптическими приборами при длительности сосредоточения наблюдения	+			
2.6 Наблюдение за экранами видеотерминалов (часов в смену)				+
2.7 Нагрузка на слуховой анализатор		+		
2.8 Нагрузка на голосовой аппарат	+			
3 Эмоциональные нагрузки				
3.1 Степень ответственности за результат собственной деятельности. Значимость ошибки			+	
3.2 Степень риска для собственной жизни	+			
3.3 Ответственность за безопасность других лиц	+			
3.4 Количество конфликтных ситуаций в течение смены – от 1 до 3.		+		

Окончание таблицы 11.1

4 Монотонность нагрузок				
4.1 Число элементов, необходимых для реализации простого задания или многократно повторяющихся операций		+		
4.2 Продолжительность выполнения простых заданий или повторяющихся операций			+	
4.3 Время активных действий		+		
4.4 Монотонность производственной обстановки (время пассивного наблюдения за ходом техпроцесса в % от времени смены)		+		
5 Режим работы				
5.1 Фактическая продолжительность рабочего дня		+		
5.2 Сменность работы	+			
5.3 Наличие регламентированных перерывов и их продолжительность	+			
Количество показателей в каждом классе	7	9	5	2
Общая оценка напряженности труда	3.1			

После проведённого анализа были получены следующие результаты: В 1 класс входят 7 показателей, во 2 класс – 9 показателей, в класс 3.1 – 5 показателя, а в класс 3.2 – 2 показателя. В итоге можно сделать вывод, что, по уровню тяжести и напряжённости трудового процесса, администраторов можно отнести к классу 3.1. Данный результат был получен в соответствии с методикой оценки напряжённости трудового процесса, в котором класс 3.1 может быть выбран если общее число показателей класса 3.1 не превышает шести, а количество показателей класса 3.2 находится в диапазоне от 1 до 3.

Нагрузки с самым высоким уровнем тяжести 2.1 и 2.6 оказывают негативное воздействие на органы зрения. Чтобы минимизировать их влияние были введены регулярные перерывы по 10-15 минут для снятия усталости и напряжения с глазных мышц. Благодаря правильной организации трудовой активности два показателя 3.2 были сведены во 2 класс условий труда. Таким

образом, работа администратора сайта «Литература» является допустимой и не наносит существенного вреда здоровью.

11.3 Обеспечение электробезопасности

Для обеспечения ежедневного функционирования веб-сайта «Литература» используются специальные рабочие помещения со множеством электронно-вычислительных устройств, в число которых входит серверный отдел и персональные компьютеры для работы администраторов. Чтобы минимизировать возникновение несчастных случаев, связанных с электротехникой были приняты соответствующие меры электробезопасности, описанные в следующих нормативных документах:

- приказ Минэнерго России от 13.01.2003 N 6, описывающий основные правила эксплуатации электроустановок потребителей (ПТЭЭП);
- приказ Минтруда России от 24.07.2013 N 328н, содержащий правила по охране труда при эксплуатации электроустановок;
- приказ Минэнерго России от 30 июня 2003 N 261, утверждающий правила по применению и испытанию средств защиты, используемых в электроустановках;
- ГОСТ 12.1.019-2017 «Электробезопасность».

В целях защиты сотрудников, отвечающих за работоспособность сайта, был принят комплекс технических и организационных мер по обеспечению электробезопасности:

- все высоковольтные провода были заземлены и помечены специальным маркером;
- все персональные компьютеры были оборудованы специальными устройствами бесперебойного питания, в целях предотвращения последствий высоковольтных скачков напряжения;

- корпуса электрогенераторов были оборудованы защитными экранами и помечены специальными предупреждающими знаками, чтобы уменьшить шанс прикосновения к источнику тока;
- относительная влажность рабочего места поддерживается в допустимых пределах и составляет 55%;
- температура офисных помещений поддерживается в районе 21-23°C;
- естественное и искусственное освещение равномерно распределены по всей площади рабочего помещения;
- была осуществлена закупка удобной и многофункциональной мебели для работников отдела;
- еженедельно производится очистка офисных помещений, чтобы предотвратить возникновение токопроводящей пыли.

11.4 Пожарная безопасность

Правила обеспечения пожарной безопасности в офисах необходимы для защиты от возникновения очагов возгорания, а также защиты сотрудников и имущества корпорации. На территории РФ существуют следующие нормативные акты, определяющие основные требования к пожарной безопасности:

- свод правил СП.12.13130.2009, благодаря которому оценивается взрывоопасность помещения;
- федеральный закон № 123-ФЗ от 22 июля 2008 г, содержащий требования пожарной безопасности для защиты от взрывов и возгораний административных зданий;
- постановление Правительства РФ от 25 июля 2012 г. № 390 «О противопожарном режиме», предназначенное для обеспечения защиты сотрудников от возгорания.

Для поддержания пожарной безопасности на высоком уровне были приняты следующие меры:

- на видных местах установлены таблички с номерами служб пожаротушения;
- разработаны и размещены эвакуационные планы для всех этажей;
- установлены всевозможные средства для ликвидации пожаров;
- установлены пожарные знаки безопасности, запрещающие курение;
- установлены системы сигнализации и пожаротушения.

Все вышеописанные мероприятия, при их выполнении, значительно уменьшают вероятность возникновения чрезвычайных ситуаций, способствуют снижению напряжённости трудового процесса, а также поддерживают пожарную безопасность на достаточно высоком уровне. Разработанный программный продукт соответствует всем нормам, требованиям и стандартам.

Заключение

В результате выполнения выпускной квалификационной работы был реализован программный продукт, обеспечивающий кластеризацию текстовых данных на сайте «Литература». В ходе разработки приложения применялся объектно-ориентированный язык C#, веб-фреймворк ASP.NET Core, библиотека машинного обучения ML.NET, язык строковой разметки веб-сайтов HTML, каскадные таблицы стилей CSS, а также реляционная база данных MS SQL Server.

Полученное приложение обладает красочным и интуитивно понятным дизайном, который делает работу с алгоритмами кластеризации текстовых наборов данных простой и удобной даже для людей не знакомых с теорией машинного обучения.

Благодаря детальному анализу предметной области с использованием различных графических изображений и формул с пояснениями были раскрыты всевозможные тонкости использования алгоритмов кластеризации, применения различных метрик определения расстояний, а также обработки и преобразования текста.

Были получены важные практические и теоретические знания, умения и навыки по написанию программ на объектно-ориентированных языках, предварительной обработке текстовых данных, включающей токенизацию, стемминг и векторизацию, созданию красивого, качественного и надёжного графического интерфейса при помощи языка разметки HTML и CSS, а также организации алгоритмов кластеризации данных.

Список используемой литературы

1. Шилдт Г. С# 4.0: полное руководство.: Пер. с англ. — М.: ООО "И.Д. Вильямс", 2011. — 1056 с.: ил. — Парал. тит. англ.
2. Вьюгин В.В. «Математические основы теории машинного обучения и прогнозирования» М.: 2013. — 387 с.
3. Бен-Ган И. Microsoft® SQL Server® 2012. Создание запросов. Учебный курс Microsoft: Пер. с англ. / И. Бен-Ган, Д. Сарка, Р. Талмейдж. — М.: Издательство «Русская редакция», 2014. — 720 с.: ил. + CD-ROM
4. Введение в язык С# — Документация по С# [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework> (дата обращения: 29.06.2019)
5. Обзор Visual Studio — Документация по Visual Studio [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019> (дата обращения: 31.06.2019)
6. Введение в ASP.NET Core — Документация по ASP.NET Core [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-3.1> (дата обращения: 02.05.2019)
7. Общие сведения об ML.NET — Документация по ML.NET [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/dotnet/machine-learning/how-does-mldotnet-work> (дата обращения: 04.05.2019)